Interaktionszentrierte Gestaltung von Informationssystemen als anwenderorientierte Vorgehensweise zur Entwicklung und Präzisierung von Anforderungen



Michael Schwarz

Interaktionszentrierte Gestaltung von Informationssystemen als anwenderorientierte Vorgehensweise zur Entwicklung und Präzisierung von Anforderungen

Dissertation
zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Informatik der Universität Ulm

Michael Schwarz
aus Mannheim

Universität Ulm
Fakultät für Informatik
Abteilung Programmiermethodik und Compilerbau
Leiter: Prof. Dr. H. A. Partsch

Dezember 2003

Amtierender Dekan:	Prof. Dr. Friedrich W. von Henke
1. Gutachter:	Prof. Dr. Helmuth A. Partsch
2. Gutachter:	Prof. Dr. Franz Schweiggert
3. Gutachter	Prof. Dr. Wolfgang Hesse
Tag der Promotion:	18. Dezember 2003

Für Verena, Benjamin und Janina

Vorwort

Mein Interesse für den Bereich Software Engineering wurde im Besonderen durch meine dreijährige Berufstätigkeit bei der sd&m GmbH in München im Anschluss an mein Informatikstudium peprägt. Erste Erfahrungen im Bereich der Entwicklung und Anwendung von Analysemethoden bei der Realisierung großer Softwaresysteme führten mich dort zu der Erkenntnis, dass die Softwarespezifikation kein sehr hilfreiches Medium für eine erfolgreiche Anforderungsklärung ist. Essentieller und herausfordernder als die formale Dokumentation des zukünftigen Softwaresystems schienen in diesen Projekten die einführenden kooperativen Lernprozesse beim Erwerb des notwendigen fachlichen Hintergrundwissens und die inhaltlichen Diskussionen aktueller Arbeitspraxis bzw. alternativer arbeitsorganisatorischer Gestaltungsoptionen zu sein.

Auch meine Erfahrungen bei der Entwicklung betrieblicher Informationssysteme für die Universität Ulm während meiner Zeit als wissenschaftlicher Mitarbeiter der Servicegruppe Informatik untermauerten diese Einschätzung. Auch hier war der Erfolg aller Entwicklungsprojekte stark von der ausreichenden Berücksichtigung des praktischen Anwendungskontexts abhängig.

Der Zwiespalt einer stark vereinfachenden naturwissenschaftlichen analytischen Denkweise der Informatik auf der einen und der lebendigen – formal nicht fassbaren – komplexen Alltagswelt auf der anderen Seite beschäftigte mich schon in meinem Studium an der Technischen Universität in München. Mit der vorliegenden Arbeit leiste ich nun einen kleinen Beitrag zur Anerkennung, Berücksichtigung und Überbrückung des fundamentalen Bruchs, der diese beiden Welten trennt.

Meine Dissertation wäre nicht möglich gewesen ohne die Hilfe und Unterstützung meiner Familie, zahlreicher Freunde und Kollegen.

An erster Stelle möchte ich hiermit dem Leiter der Servicegruppe Informatik, Dr. Klaus Murmann, für seine ausdauernde herzliche Unterstützung und Motivation danken, meinem Doktorvater, Prof. Dr. Helmuth Partsch, für seine zahlreichen Anregungen und Verbesserungsvorschläge und meinen beiden Gutachtern, Prof. Dr. Franz Schweiggert und Prof. Dr. Wolfgang Hesse, für ihre hilfreichen Anmerkungen.

Für den erfolgreichen Abschluss meines Promotionsvorhabens war auch die herzliche Atmosphäre innerhalb der Abteilung Programmiermethodik und Compilerbau und der Servicegruppe Informatik maßgeblich mit verantwortlich. Ich möchte hier besonders meinen Kollegen Klaus Achatz, Dietmar Ernst, Wolfgang Gehring, Tobias Häberlein, Friedemann Kiedaisch, Niko Kleiner, Stefan Sarstedt, Wolfram Schulte und Ton Vullinghs für ihre ständige Hilfs- und Gesprächsbereitschaft danken.

Zu guter Letzt möchte ich es jedoch natürlich auch nicht versäumen, meiner Frau Verena Schwarz zu danken, die mich unermüdlich und unbeirrbar in meinem Vorhaben bestärkt hat

und ohne die die vorliegende Arbeit wohl nicht existieren würde, meinen beiden Kindern Benjamin und Janina, die mir geholfen haben, das Wesentliche nicht aus den Augen zu verlieren und meinen Eltern Alfons und Helga Schwarz für ihr Vertrauen, ihre Geduld und liebevolle Unterstützung.

Seid bedankt!

Michael Schwarz
Ulm, Dezember 2003

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Motivation . . . 2
- 1.2. Ziel . . . 6
- 1.3. Randbedingungen . . . 7
- 1.4. Übersicht über die folgenden Kapitel . . . 10

2. Grundlegendes **13**
- 2.1. Software Engineering . . . 13
- 2.2. Grundlegende Probleme . . . 15
- 2.3. Requirements Engineering . . . 18
- 2.4. Die Systemtheorie als Lösungsansatz . . . 19
- 2.5. Die Aktivitätstheorie als Lösungsansatz . . . 24
- 2.6. Ein erweitertes Phasenmodell . . . 28

3. Einordnung von Koala in die Methodenwelt **33**
- 3.1. Gestaltungsebene Softwaresystem . . . 34
- 3.2. Gestaltungsebene Nutzungssystem . . . 38
- 3.3. Gestaltungsebene Handlungssystem . . . 41
- 3.4. Zusammenfassung . . . 49

4. Die Modellierungssprache von Koala **51**
- 4.1. Grundlagen der Modellierung . . . 53
- 4.2. Das Wissensmodell . . . 56
- 4.3. Das Organisationsmodell . . . 65
  - 4.3.1. Organisationsstruktur . . . 65
  - 4.3.2. Tätigkeitsprofile . . . 67
- 4.4. Das Kooperationsmodell . . . 73
  - 4.4.1. Kooperationsstruktur . . . 75
  - 4.4.2. Konversationsmuster . . . 79
  - 4.4.3. Aktivitätsmuster . . . 84
  - 4.4.4. Zusammenfassung . . . 90

5. Die Vorgehensweise von Koala **93**
- 5.1. Überblick . . . 94
- 5.2. Motivation . . . 96
- 5.3. Vision . . . 101
- 5.4. Kontextanalyse . . . 107
  - 5.4.1. Kooperationsbeziehungen . . . 110

5.4.2. Konversationen . . . 115
5.4.3. Begriffswelt . . . 118
5.4.4. Aktivitätsmuster . . . 121
5.4.5. Organisation . . . 124
5.4.6. Sonderfälle . . . 130
5.4.7. Zusammenfassung . . . 132
5.5. Systemgestaltung . . . 134
5.5.1. Gestaltungsparameter . . . 137
5.5.2. Systemkooperationen . . . 140
5.5.3. Systemkonversationen . . . 145
5.5.4. Systembegriffswelt . . . 147
5.5.5. Systemprofil . . . 150
5.5.6. Zusammenfassung . . . 157
5.6. Fiktion . . . 159
5.6.1. Nutzungskonzept . . . 162
5.6.2. Konversationsszenarien . . . 167
5.6.3. Aktivitätsszenarien . . . 169
5.6.4. Zusammenfassung . . . 173
5.7. Validierung . . . 174
5.8. Der Übergang zur Softwarespezifikation . . . 175

6. Zusammenfassung **181**
6.1. Das Vorgehen . . . 182
6.2. Die Modellierung . . . 183
6.3. Abgrenzung . . . 184
6.4. Kritik . . . 190
6.5. Ausblick . . . 193

A. Referenz der Modellierungssprache **199**
A.1. Modell . . . 201
A.2. Konzept . . . 202
A.3. Assoziation . . . 204
A.4. Akteur . . . 207
A.5. Bedingung . . . 208
A.6. Zustand . . . 210
A.7. Aktivität . . . 211
A.8. Interaktion . . . 213
A.9. Aktivitätsmuster . . . 215
A.10. Wechsel . . . 216

B. Beispiele für die semiformale textuelle Repräsentation **217**
B.1. Wissensmodell . . . 218
B.2. Organisationsmodell . . . 219
B.3. Kooperationsmodell . . . 222

Kurzfassung **243**

1. Einleitung

Die vorliegende Arbeit beschreibt eine neuartige Vorgehensweise zur systemischen Gestaltung betrieblicher Informationssysteme[1], deren Schwerpunkt in der kommunikationsorientierten Modellierung und partizipativen Umgestaltung realer Arbeitszusammenhänge liegt.

Die dargestellte Vorgehensweise trägt den Namen "Koala": **Ko**mmunikationsorientierte **A**nalyse und Gesta**l**tung von **A**rbeitskontexten.

Sie gibt Antwort auf eine Frage, die im Umfeld von Softwareentwicklungsprojekten oft gehört wird – zumal etablierte Analyse-Methoden sie in der Regel weitgehend vernachlässigen: "Wie kommt man eigentlich zu den Anforderungen, die man zu spezifizieren hat?".

KoA$_{\text{L}}$A begründet hierzu eine eigene Phase der Kontextgestaltung, die noch vor der Softwaregestaltung, d.h. der Spezifikation des zu realisierenden Softwaresystems angesiedelt ist, stellt angemessene neuartige Modellierungstechniken zur Verfügung und definiert eine spezifische zyklische Vorgehensweise zur Durchführung des notwendigen Gestaltungsprozesses. Endprodukt ist ein praxisorientiertes Modell des zukünftigen soziotechnischen Handlungssystems, das als Ausgangspunkt für den nahtlosen Übergang zur technischen Spezifikation des Softwaresystems verwendet werden kann.

KoA$_{\text{L}}$A fasst die Entwicklung von Anforderungen an Informationssysteme als kreativen Gestaltungsprozess auf, der nicht nur das Softwaresystem betrifft, sondern auch die Anwendungsumgebung miteinbezieht. KoA$_{\text{L}}$A bezeichnet diesen Prozess daher auch als "Kontext-Design".[2]

Der Schwerpunkt des Interesses liegt damit auf einer Phase der Softwareentwicklung, die noch vor der eigentlichen Spezifikation des zu entwickelnden Softwaresystems angesiedelt ist und von den meisten etablierten Ansätzen der Softwareentwicklung überwiegend stiefmütterlich oder nur implizit behandelt, wenn nicht sogar ignoriert wird.

Der Begriff "Informationssystem" wird in dieser Arbeit sowohl im allgemeineren Sinne eines soziotechnischen Handlungssystems als auch im Sinne eines konkreten technischen Softwaresystems benutzt, das integraler Bestandteil eines betrieblichen Handlungssystems ist. Sofern notwendig, wird im Text eine genauere Unterscheidung vorgenommen. Die Abkürzung IS bezieht sich im Folgenden immer auf das Softwaresystem.

[1] Ein *Informationssystem* wird dabei als ein soziotechnisches System aufgefasst, bei dem die Erfassung, Speicherung, Übertragung und/oder Transformation von Informationen durch den Einsatz technischer Systeme teilweise automatisiert ist.

[2] *Design*: formgerechte und funktionale Gestaltung, v. a. von Gebrauchsgegenständen [Brockhaus - Die Enzyklopädie]. Dieser Begriff wird im Folgenden generell zur Auszeichnung eines kreativen Gestaltungsprozesses verwendet, nicht nur zur Bezeichnung der Entwicklung von Grob- und Feinstruktur (auch "Architektur" genannt) eines technischen Softwaresystems!

1.1. Motivation

Das Gebiet des Software Engineering beschäftigt sich mit der systematischen, qualitätsorientierten und ressourcenbewussten Entwicklung von Softwaresystemen nach den Prinzipien der Ingenieurwissenschaften.

Angesichts der wachsenden Verbreitung und wirtschaftlichen Bedeutung bzw. Abhängigkeit von Softwaresystemen, die in den verschiedensten Branchen und Bereichen wie z.B. der Kommunikationsindustrie, den Finanzdienstleistern, der Versicherungswirtschaft, der allgemeinen Verwaltung, der Medizin, der industriellen Produktion, der Verkehrstechnik, der Haushalts- und Unterhaltungselektronik zu beobachten sind, nimmt es nicht wunder, wenn dem Gebiet des "Software Engineering" in einer Studie des BMBF aus dem Jahre 2000 eine entscheidende Bedeutung für die Wertschöpfung im Produktionsgüter- und Dienstleistungsbereich beschieden und als Produktionstechnik des 21. Jahrhunderts bezeichnet wird [Bea00].

Aufgabe des industriellen Engineering von Software ist die Entwicklung komplexer Softwaresysteme unter Berücksichtigung der Aspekte Kosten, Termine und Qualität. Von besonderer Bedeutung sind dabei neben den eher technisch formalen Qualitätsaspekten wie Korrektheit, Zuverlässigkeit, Performanz, Wartbarkeit und Erweiterbarkeit vor allem die nutzungsgerechte Handhabbarkeit der Systeme.

Aufgrund der immer weiter steigenden Komplexität sowohl der zu entwickelnden Systeme selbst, als auch ihres organisatorischen und technischen Umfelds, gehört die erfolgreiche Gestaltung großer Softwaresysteme heute zu den schwierigsten technischen Herausforderungen überhaupt [BR02]. Mit der wachsenden Durchdringung von Unternehmen mit Informationstechnologie wird insbesondere die Entwicklung betrieblicher Informationssysteme immer mehr zu einem interdisziplinären Unterfangen, das von ihrer Natur her viele andere Disziplinen wie die Betriebswirtschaft, Arbeitswissenschaft, Soziologie und Psychologie berührt und dementsprechend mittlerweile auch eigene disziplinübergreifende Lehrbücher wie z.B. [AF95] hervorgebracht hat.

Die Komplexität der Softwareentwicklung lässt sich auch an der relativ hohen Quote an Misserfolgen ablesen, der die Softwareindustrie gewahr werden muss. Laut einer Studie der Standish Group, die mehr als 8000 Softwareprojekte von über 350 Unternehmen untersuchte, werden nur ca. 9-16% der Projekte in der veranschlagten Zeitspanne fertig und halten ihren Kostenrahmen ein. Gladden spricht davon, dass 75 % aller Projekte entweder gar nicht fertig werden oder aber nicht genutzt werden, sobald sie fertig gestellt sind [LH87]. Über 50% der genannten Gründe für Misserfolge entfallen dabei auf Probleme mit der Erhebung, Definition und der Verwaltung der Softwareanforderungen ([Pfl98], S.139). Leffingwell geht sogar davon aus, dass bis zu 60 Prozent aller Fehler, die in Softwareprojekten gefunden werden, letztlich auf Fehler in der Anforderungsanalyse zurückzuführen sind [Lef97]. Endres und Rombach drücken diese Problematik darum auch in Form eines fundamentalen "Grundgesetzes" der Softwareentwicklungspraxis aus:

> *"Requirement deficiences are the prime source of project failures"* [ER03].

Den sogenannten "frühen" Phasen der Softwareentwicklung kommt damit eine Schlüsselrolle in den Bemühungen zu, die Erfolgsquote von Softwareprojekte zu steigern. Der Schwerpunkt existierender Softwareanalysemethoden – also der Methoden, die bei der Erhebung, Unter-

suchung und Dokumentation von Anforderungen an Softwaresysteme eingesetzt werden – besteht momentan im Allgemeinen in der vollständigen, konsistenten und korrekten Spezifikation aller Anforderungen an das zu realisierende Softwaresystem. Das Thema der darüber hinausgehenden Einbettung des Softwaresystems in seinen sozialen Kontext wird von diesen Methoden weitgehend ignoriert [DHP+99]. Sogar die Aufgabe der ergonomischen Gestaltung der Benutzerschnittstelle wird irrtümlicherweise oft als abgrenzbare und nachgeordnete Teilaufgabe angesehen, die keinen direkten Einfluss auf den Entwicklungsprozess nimmt (siehe [Pae00], S. 121).

Da sowohl Grundsatzfragen der Arbeitsaufteilung als auch Organisationsdetails in den üblichen Softwareentwicklungsmethoden kaum thematisiert werden, führt das im Letzten zu dem Umstand, dass fertige Softwaresysteme in der Praxis oft keine oder nur unzureichende Akzeptanz finden [Pae00]. Dies trägt viel zu dem Umstand bei, dass die Erfolgsquote von Softwareentwicklungsprojekten insgesamt relativ gering ist.

Bei der Analyse der Aufgabenstellung wird oft stillschweigend vorausgesetzt, dass bereits eine klare Problemstellung existiert, die es durch eine Analyse des Problembereichs aufzudecken bzw. zu analysieren und zu lösen gilt. Den Fragen nach den Wurzeln dieser Anforderungen, nach ihren Auswirkungen, nach ihrem Entstehungsprozess bzw. den Randbedingungen ihres Entstehens wird im Analyseprozess oft keine Aufmerksamkeit gewidmet.

Viele der aktuellen Methoden der Anforderungsanalyse entstammen den klassischen Softwareentwicklungsmethoden und besitzen ihren Ursprung letztendlich in Techniken der Programmierung, also in der Konstruktion von technischen Systemen. Demzufolge bieten sie im Allgemeinen ein mächtiges Konglomerat an unterschiedlichen Modellierungstechniken für die möglichst formale Darstellung des Softwaresystems aus verschiedenen Sichten wie z.B. Funktionen, Daten, Verhalten etc.

Von hier aus hielten die entwickelten Paradigmen und Techniken nach und nach Einzug in die früheren Phasen des Entwicklungsprozesses wie den Entwurf der Softwarearchitektur, die Analyse und schließlich auch in die Modellierung der Anwendungsumgebung ("Business Modelling", Domain Modelling etc.).

Viele dieser – eher technikorientierten – Methoden weisen folgende Gemeinsamkeiten auf:

- Im Allgemeinen ist die Vorgehensweise eher produktorientiert. Einer Studie der Umgebung bzw. der Befragung aller Interessengruppen folgt die textuelle bzw. grafische Aufbereitung, Strukturierung und Formalisierung aller Anforderungen in Form von Modellen, die meistens die Form einer vollständigen Spezifikation des zu realisierenden Softwaresystems annehmen und als direkte, möglichst formale Vorgabe für die sich anschließende Realisierung verwendet werden können. Der Abstimmungsprozess selbst wird nur sekundär betrachtet.

- Die Anforderungen an das Softwaresystem werden schrittweise aus den Modellen der Umgebung abgeleitet bzw. transformiert. Es finden normalerweise keine expliziten partizipativen Entwürfe der Problemwelt oder des potentiellen Lösungsmodells statt. Die Modellierung unterscheidet kaum zwischen den Gestaltungsbereichen Anwendungswelt und Systemwelt. Es findet kein Wechsel der Abstraktionsebenen statt, sondern nur ein Wechsel vom externen zum internen System bzw. eine Konkretisierung ihrer Schnittstellen (z.B. in Form der Automatisierung von Tätigkeiten der Anwendungswelt oder

der Festlegung von Benutzungsschnittstellen).

- Sofern auch der Anwendungskontext einer Modellierung unterzogen wird, wird diese Umgebung mit den gleichen Modellierungstechniken wie das Softwaresystem selbst modelliert (Objekte, Funktionen, Daten, Automaten etc.). Ziel ist eine vollständige, diskrete, deterministische Modellierung (trotz komplexer, kontinuierlicher, nichtdeterministischer realer Anwendungswelt).
- Das Softwaresystem selbst kommt in der Modellierung normalerweise nicht vor, obwohl Softwaresysteme in soziotechnischen Organisationsformen oft den eigenständigen Charakter einer "Institution" besitzen, die auf ihre Umgebung eine zentrale und prägende Wirkung ausübt und daher auch im Umgebungsmodell auftauchen sollte. Sinnvollerweise dann allerdings in ihrer "sozialen Rolle" und nicht in ihrer technischen Ausprägung bzw. Realisierung (siehe z.B.[Fre90]).

Zu einer genaueren Diskussion der Methodenlandschaft siehe Kap. 3.

In vielen Methoden wird auch die Anwendungsumgebung als technisches System modelliert. Dies bietet sich umso mehr an, als man Softwaresysteme möglichst formal definieren will, um diese Anforderungen gut überprüfen und aus ihnen Simulationen ableiten zu können oder sie schrittweise abgesichert letztendlich in ein Softwaresystem transformieren zu können. Insbesondere bei technischen und sicherheitskritischen Systemen – in der Regel sogenannte "eingebettete Systeme" – ist dieser Wunsch nach einer abgesicherten Entwicklung plausibel. Bei technischen Systemen ist dieser Ansatz (siehe z.B. [Ach00]) sehr erfolgversprechend, da die Umgebung solcher Systeme im Allgemeinen wieder aus technischen Systemen besteht (Motorsteuerung, Aufzug, Telekommunikationssysteme etc.), die als konstruiertes technisches Artefakt per se formal beschreibbar sind bzw. sein müssen (sonst könnte es auch kein Herstellungsverfahren geben).

Bei interaktiven Systemen sieht das jedoch meist anders aus, da die Komplexität von sozialen Systemen (bzw. lebenden Organismen) formal nicht vollständig fassbar ist.

> "*design methods appropriate for technical systems cannot simply be applied to socio-technical ones, and that consideration must be given equally to both human and technical issues, with success being seen as the construction of a relevant socio-technical system that meets the 'real' requirements of the organisation*". [DBCS94]

Erschwerend kommt – bei betrieblichen Informationssystemen in besonderem Maße – hinzu, dass sich die Umgebung des Systems – nicht zuletzt auch durch die Eigenschaften des Systems selbst – fortlaufend verändert. Der Einsatz eines IS im Unternehmen hat zwangsläufig eine Änderung des Unternehmens und damit der Systemumgebung zur Folge. Der Versuch einer technischen bzw. formalen Spezifikation einer solch komplexen Systemumgebung ist damit nur wenig erfolgversprechend und wird von (Kern-)Informatikern darum oft auch abgelehnt:

> "*Von einem formalistischen Standpunkt aus gesehen gehört die Erarbeitung einer fachlichen Spezifikation nicht zum eigenen (zu einem) Wissenschaftsgebiet, sondern sie wird als ein Bauvorhaben in einer Sumpflandschaft dargestellt, die man durch wissenschaftliche (methodische) Arbeit gar nicht trockenlegen kann*" [Ort02]

Neuere Sichtweisen begreifen moderne Betriebsorganisationen daher als soziotechnische Sys-

teme, in denen sowohl soziale wie auch technische Komponenten zu unterscheiden sind, die sich gegenseitig beeinflussen [WAP88, CdAS89, Jef96]. Die soziale Wirklichkeit stellt Ansprüche an die *Gestaltung* technischer Systeme, wie auch die Existenz technischer Systeme die soziale Wirklichkeit beeinflusst. Geschäftsprozesse und Arbeitszusammenhänge ändern sich, sobald neue Werkzeuge verfügbar werden oder sich neue begriffliche Konzepte durchsetzen. Die praktische Arbeitstätigkeit in arbeitsteiligen Organisationen ändert sich fortlaufend, nicht nur aufgrund dieses zyklischen Wirkungszusammenhangs von Tätigkeit und den zur Verfügung stehenden Werkzeugen. Die Organisationsentwicklung besitzt einen stark dynamischen Charakter, so dass die Aufgabe der "Konstruktion von Wirklichkeit" [Flo92] – und nichts anderes stellt der Entwurf und die Realisierung eines neuen Softwaresystems dar – zu einer wiederkehrenden, wenn nicht sogar zu einer kontinuierlichen Herausforderung wird.

Aus diesem Grund sind Anforderungen an ein IS nur aus dem Kontext der realen Arbeitspraxis heraus zu entwickeln bzw. zu verstehen. Will man eine stabile Basis für die Analyse von Anforderungen an Informationssysteme erreichen, so muss man bereits mit der Modellierung und Gestaltung der betrieblichen Realität beginnen und damit die Rolle des Softwaresystems festlegen, bevor man dieses aus technischer Sicht (Benutzerschnittstelle, Funktionen, Daten, Verhalten) formal spezifiziert.

Die Spezifikation eines Informationsystems, wie z.B. eines Systems zur Unterstützung der Organisation und Durchführung von Gremienwahlen einer Universität, ist wesentlich von der genauen Analyse der betroffenen Akteure, ihren Kooperationen und Arbeitsprozessen abhängig. Die Unterstützung des Auszählprozedere beispielsweise ist darauf abzustimmen, an welchem Ort (verteilt in den verschiedenen Wahllokalen oder zentral in einem eigenen Auszählraum?), durch wen (in Gruppen je nach Wahllokal, einzeln oder geordnet nach Gremien?) und wie (Rechnerunterstützung für jeden Zähler, jede Auszählgruppe oder nur ein Rechner für den Wahlleiter?) durchgeführt wird bzw. in Zukunft durchgeführt werden soll. Ohne eine genaue Analyse, Ausgestaltung bzw. Berücksichtigung der neuen Kooperationsmuster, wie sie nach Einführung eines neuen IS bestehen, ist die Akzeptanz und Gebrauchstauglichkeit nicht gesichert.

Mit dem Einsatz eines neuen IS geht implizit eine fundamentale Umgestaltung des Arbeitskontexts einher. Um diesen Gestaltungseffekt explizit zu machen, ist dieser Kontext strukturiert zu erfassen und *zusammen* mit dem Entwurf neuer Softwaresysteme auszugestalten. Der Gestaltungsprozess – und damit der neu entworfene Arbeitskontext – ist die Voraussetzung für eine Spezifikation des IS.

Der Softwareentwicklungsprozess beginnt also bereits mit der Entwicklung des zu lösenden Problems, noch bevor die eigentliche Problemlösung – das Softwaresystem – analysiert und spezifiziert wird. Die Entwicklung von Software wird damit als Gestaltungsaufgabe angesehen, die wesentlichen Einfluss auf die soziale Wirklichkeit nimmt. Die notwendigen Diskussions- und Abstimmungszyklen machen die Phase der Erhebung und Analyse von Anforderungen zu einem dialogischen Prozess, an dem alle Betroffenen teilhaben [Pas94, And95].

Nach [JP88] muss sich das Requirements Engineering darum auch von der Produktorientierung lösen und hin zur Prozessorientierung entwickeln. Die Designartefakte sind eher als Mittel für den kontinuierlichen Kommunikations- und Kooperationsvorgang anzusehen statt als formalisierte Zwischenprodukte eines schrittweisen Produktionsprozesses.

Obwohl sich diese Sichtweise mittlerweile durchzusetzen beginnt, sind entsprechende Software-

entwicklungsmethoden nur in Ansätzen zu entdecken. Insbesondere fehlen hier pragmatische Methoden, die einen Softwareingenieur in seiner Alltagspraxis unterstützen und sich nahtlos in bestehende Entwicklungsprozesse einfügen. Das ist die Motivation für die vorliegenden Arbeit.

1.2. Ziel

Es stellt sich die Frage, wie der Gestaltungsprozess beim Entwurf von Informationssystemen methodisch unterstützt und dem gestalterischen Aspekt im Entwicklungsprozess größere Bedeutung eingeräumt werden kann: Wie trägt man dem dynamischen und dialogischen Charakter dieses Prozesses Rechnung? Wie kann die "Passgenauigkeit" eines Informationssystems in der Arbeitswelt der Betroffenen sichergestellt werden? Wie kann ein Optimum an Praxistauglichkeit (Akzeptanz, Handhabbarkeit, Zufriedenheit, Gesamtnutzen) erzielt werden? Wie können die Aktivitäten zur Erhebung von Anforderungen an ein IS besser strukturiert und seine Entwurfsprinzipien (design rational) explizit gemacht werden, um eine bessere, d.h. stabilere und umfassendere Ausgangsbasis für die Realisierung zu gewinnen?

Gesucht ist eine anwenderorientierte Vorgehensweise zur Erhebung, Konstruktion und Ausgestaltung von Anforderungen an ein neues IS, die in den Gestaltungsprozess auch den Arbeitszusammenhang mit aufnimmt. Gesucht wird eine Antwort auf die Frage: "Wie komme ich eigentlich zu den Anforderungen, die ich dann im Rahmen des Softwareentwicklungsprozesses weiter analysieren und spezifizieren soll?". Die Frage "Wie spezifiziere ich Anforderungen an Softwaresysteme aus Entwicklersicht" ist hingegen bereits hinreichend gelöst und daher nicht Gegenstand der vorliegenden Arbeit.

Ein optimales Gestaltungsergebnis ist dann erzielt, wenn die intrinsischen – von "Innen" stammenden Motive, Ziele oder Wünsche – und extrinsischen – von "außen" stammenden bzw. auferlegten – Anforderungen aller Interessengruppen ausreichend identifiziert und in der Ausgestaltung des Informationssystems möglichst gut berücksichtigt wurden. Dabei spielen nicht nur die von den Betroffenen explizit formulierten Anforderungen eine Rolle, sondern auch die impliziten Anforderungen, die durch die betriebliche Arbeitspraxis gestellt werden. Erst der Erfolg und die Akzeptanz des IS in der betrieblichen Praxis entscheidet letztendlich über den Erfolg eines Entwicklungsvorhabens.

Für den Gestaltungsprozess sind in erster Linie Kommunikationsfähigkeit, Kreativität, Kompromissbereitschaft und die Fertigkeit zur Konsensfindung von Nöten. In zweiter Linie kann hier aber auch ein methodisches Vorgehen und entsprechende Techniken helfen, den Abstimmungsprozess so zu gestalten und zu strukturieren, dass man auf möglichst effizientem Weg zu vollständigen und "stimmigen" Anforderungen gelangt.

Die vorliegende Arbeit beschreibt eine solche Vorgehensweise, die "**Ko**mmunikationsorientierte **A**na**l**yse und Gestaltung von **A**rbeitskontexten" ("Koala"). KoA$_{\text{L}}$A dient der iterativen *Konstruktion* und *Ausgestaltung* von Anforderungen an Informationssysteme und besteht aus:

- einem Vorgehensmodell
- verschiedenartigen Modellen zur kontextreichen Darstellung der Arbeitspraxis
- modellunabhängigen Diagrammtypen und Modellierungskonzepten

KoA$_{\text{L}}$A ist eine modellbasierte Vorgehensweise zur anwendungsorientierten Gestaltung von Informationssystemen, die ihren Schwerpunkt in der systemischen Modellierung von Arbeitskontexten sieht und die Gestaltung als sozialen Konstruktionsprozess begreift. KoA$_{\text{L}}$A gibt dabei einen aus Einzelphasen bestehenden zyklischen Gestaltungsprozess an, der die dialogische Ausgestaltung und Einbettung des zu entwickelnden Informationssystems zum Ziel hat. KoA$_{\text{L}}$A definiert eine Reihe von Modellen, die zum Einen der Dokumentation des Anwendungsbereichs und zum Anderen der Beschreibung der Einbettung des Softwaresystems in diese Umgebung dienen. Die Modellierung dient dabei sowohl der Strukturierung der Projektarbeit als auch der Förderung eines partizipativen Gestaltungsprozesses.

Ausgehend von den aktuellen Kooperationsbeziehungen werden durch eine Analyse der typischen akteurübergreifenden Konversations- und Aktivitätsmuster die Tätigkeitsprofile, Motive und Aufgaben aller Akteure aufgedeckt und analysiert. Nach einer Bestimmung der zukünftigen Nutzergruppen und deren Anforderungen wird das IS als neuer technischer Akteur mit eigenem Tätigkeitsprofil und spezifischem Interaktionsverhalten entworfen und in diesen Arbeitskontext integriert. Durch Modellierung und Diskussion der sich neu ergebenden zentralen Aktivitätsmuster wird die gewählte Gestalt des IS validiert und iterativ fortentwickelt.

KoA$_{\text{L}}$A ist eine Vorgehensweise zur Problemkonstruktion und schließt damit nahtlos an existierende Methoden zur Problemlösung an, wie sie die etablierten Methoden der Softwareentwicklung darstellen, welche die Entwicklung von Softwaresystemen auf Grundlage einer möglichst formalen Spezifikation zum Ziel haben. Die vorliegende Arbeit beschreibt hierzu abschließend den nahtlosen Übergang vom Endergebnis des Gestaltungsprozesses, der Problemdefinition, hin zur Konkretisierung und Spezifikation des zukünftigen Softwaresystems – der Beschreibung einer konkreten Problemlösung (einer von potentiell vielen möglichen).

1.3. Randbedingungen

Im Folgenden soll nun der Einsatzbereich von KoA$_{\text{L}}$A näher eingegrenzt werden. Auf welchen Kontext bzw. Problembereich zielt die Vorgehensweise ab?

Welche Softwaresysteme? KoA$_{\text{L}}$A zielt allgemein auf den Bereich der Entwicklung interaktiver Softwaresysteme ab, d.h. Systeme mit menschlichen Nutzern, und hier im Besonderen auf die Unterklasse der betrieblichen bzw. "kooperativen" Informationssysteme. Diese Systeme zeichnen sich dadurch aus, dass sie starken Einfluss auf die "betriebliche" Arbeitsorganisation ausüben und die Zusammenarbeit vieler Akteure bestimmen. Sie besitzen meistens eine strategische Bedeutung für die Unternehmensführung und müssen im Laufe ihrer Lebenszeit oft an geänderte Anforderungsprofile angepasst werden.

Das KoA$_{\text{L}}$A zugrundeliegende Leitbild bei der Gestaltung eines IS ist ein von selbstgesteuerten autonomen Akteuren flexibel einsetzbares Werkzeug bzw. ein ihnen zugeordneter "intelligenter" Assistent, der durch seine Funktionalität bzw. Aufgabenbereiche die betriebliche Arbeitsorganisation zwar indirekt stark beeinflussen kann, diese menschlichen Tätigkeitsbereiche aber nicht direkt von außen steuert. Die Modellierung von Arbeitsabläufen dient also insofern in erster Linie nur der Absicherung bzw. Ableitung von Anforderungen an das IS,

nicht – wie im Falle von Workflow-Management-Systemen – dem Zweck der Realisierung einer systemkontrollierten Steuerung dieser Abläufe[3].

Reaktive Systeme mit starker Kopplung an einen technischen Prozess, wie z.B. eingebettete Systeme oder Echtzeitsysteme, besitzen eine eher technische Anwendungsumgebung und sollen hier nicht weiter betrachtet werden. Diese Softwaresysteme sind im Gegensatz zu Informationssystemen Teil eines weitgehend formalisierbaren Anwendungskontext, der daher auch gut mit entwicklungsnahen Methoden analysiert und spezifiziert werden kann (siehe z.B. [Ach00]).

Welche Problemstellung? In der Darstellung der Vorgehensweise wird von dem vereinfachten Fall ausgegangen, dass ein völlig neues und individuelles IS entwickelt und in einen gegebenen Arbeitskontext integriert werden soll. Es ist dabei kein abzulösendes, anzupassendes bzw. zu erweiterndes Altsystem zu berücksichtigen.

Der Raum für Gestaltungsmöglichkeiten ist hier sicherlich am größten. Das eigentliche Problem, welches das neue Softwaresystem lösen soll, ist hier noch weitgehend offen, so dass im Allgemeinen auch keine vordefinierten Anforderungen existieren, die es nur noch zu ermitteln gilt. Es ist vielmehr das eigentliche Problem "zu heben" bzw. zu konstruieren, das durch das noch zu entwickelnde Softwaresystem gelöst werden soll.

Dies schließt natürlich nicht aus, KoA$_{\text{L}}$A auch in eingeschränkteren Problemsituation einsetzen zu können, wie z.B. der Ablösung, Integration oder Erweiterung von gegebenen Altsystemen oder Standardsoftwarepaketen, da der Gestaltungsraum hier zwar beschränkt und teilweise bereits vordefiniert ist, aber nicht destotrotz immer vorhanden ist.

Desweiteren ist die Notwendigkeit einer methodischen Unterstützung der Exploration und der dialogischen Gestaltung der Anwendungsumgebung nebst "Einpassung" eines Softwaresystem in einen gegebenen Arbeitskontext natürlich in beiden Fällen gleich evident, unabhängig davon, welcher Art dieser Kontext nun konkret ist und in wieweit er bereits formalisiert bzw. vordefiniert ist.

Nicht zuletzt kann KoA$_{\text{L}}$A auch zur nachträglichen Evaluation der Passgenauigkeit eines gegebenen Softwaresystems verwendet werden, wenn eine vorgegebene Anwendungsumgebung mit KoA$_{\text{L}}$A systematisch modelliert und analysiert wird.

Wer ist beteiligt? Prinzipiell sind in den Gestaltungsprozess Vertreter aller Interessengruppen einzubinden. Besondere Bedeutung kommt dabei natürlich den direkten Systembenutzern

[3] Ein Workflowsystem kann natürlich auch als Werkzeug betrachtet werden, das Funktionen anbietet wie "Darstellung aller noch offener Arbeitsaufträge" oder "Automatischer Start einer passenden Applikation für den nächsten Arbeitsschritt". In erster Linie interessiert sich KoA$_{\text{L}}$A dabei aber dennoch für die – nichtformalisierbare – menschliche Arbeitssituation, nicht für eine Formalisierung der Arbeitspraxis in Form vordefinierter Prozesse.

KoA$_{\text{L}}$A ist natürlich auch für die Entwicklung von Workflowsystemen einsetzbar. Da sich, wie in [JBS97] dargestellt, der Entwicklungsprozess für Workflow-Management-Anwendungen erst im konkreten Entwurf der Benutzerschnittstelle und des Systemkerns von einem Entwicklungsprozess für ein betriebliches Informationssystem unterscheidet, werden diese Systeme nicht explizit ausgenommen, deren Spezifika werden aber auch nicht ausdrücklich berücksichtigt. KoA$_{\text{L}}$A ist hier also als flexibler und grundlegender Vorgehensrahmen zu sehen, der für spezialisiertere Anwendungsfälle noch erweitert werden kann.

und den indirekt in ihrer Arbeitspraxis durch das System betroffenen Akteuren zu. Diese sind zusammen mit dem Systemdesigner (Systemanalytiker) aktiv am Gestaltungsprozess beteiligt.

Aufgaben des Systemdesigners sind die Leitung, Moderation und aktive methodische Unterstützung des Abstimmungsprozesses. Er verwaltet, entwickelt, dokumentiert und validiert die Artefakte des Gestaltungsprozesses und ist Sachverständiger, was die technische Realisierbarkeit und deren Kosten anbelangt. Die Aufgabe der Interessengruppenvertreter hingegen ist neben der Darstellung der Arbeitspraxis, der Verbalisierung ihrer Anforderungen, dem kreativen Entwickeln und Durchdenken von (Um)Gestaltungsideen vor allem die modellbasierte praxisorientierte Evaluation und Bewertung der Ausgestaltungsvarianten.

Die Anwender sind also keine passiven Informationsquellen oder Validierer/Genehmiger, sondern vielmehr aktive Mitarbeiter im Gestaltungsprozess. Der Systemdesigner steuert den Gestaltungsprozess möglichst aus dem Hintergrund, so dass die eigentliche Gestaltung letztendlich durch die Betroffenen selbst ausgearbeitet und entschieden wird.

Zentrale Vorbedingung für den Einsatz von KoA$_{L}$A ist die Verfügbarkeit dieser Vertreter. Vor allem bei der Entwicklung von Standardprodukten "von der Stange" steht die mögliche Anwendung von KoA$_{L}$A unter dem Vorbehalt, dass entsprechend charakteristische Nutzergruppen bestimmt werden können und ein Vertreter solcher Gruppen verfügbar ist, der stellvertretend am Gestaltungsprozess mitwirken kann. KoA$_{L}$A spielt daher besonders bei spezialisierten Individual-Entwicklungen, bei denen die Benutzer und Betroffenen eindeutig definiert sind, seine Stärken aus.

Aus der engen Beteiligung der Anwender ergibt sich eine grundlegende Anforderung an die eingesetzten grafischen oder textuellen Notationen: Sie müssen intuitiv und einfach verständlich sein. Der Systemdesigner ist angehalten, die Hürde möglichst niedrig zu halten, indem er die Rolle der Dokumentenerstellung übernimmt oder zur Verständigung alternativ auch einfachere Medien bzw. Zugangsformen einsetzt (Aufbereitung der Dokumente oder Grafiken, Fragebögen, Interviews etc.).

Wie sieht das Ziel aus? KoA$_{L}$A dient der Problemdefinition. Ziel ist es, den Schritt zur nachfolgenden Softwarespezifikation optimal vorzubereiten, indem das Softwaresystem aus der anwendungsorientierten Sicht der Systembenutzer dargestellt und seine Eingliederung in die Arbeitspraxis kontextreich beschrieben wird. Voraussetzung hierzu ist ein ausreichendes Verständnis und die Beschreibung des Arbeitskontexts. Das Handlungssystem, d.h. die Arbeitswelt rund um das neue IS, und Teile des Nutzungssystems, d.h. die Systemnutzer und ihre Anwendungswelt, werden auf Grundlage eines systemischen Interaktionsparadigmas ausgestaltet und möglichst kontextreich dokumentiert. Indem das umfassende Gesamtsystem der neuen Arbeitswelt nach Einführung des IS antizipiert und beschrieben wird, werden implizit auch die Anforderungen an das in diese Arbeitspraxis eingebettete Softwaresystem spezifiziert.

Das System selbst wird hierzu als ein Akteur modelliert, der in ein Kommunikationsnetzwerk eingebunden ist. Der Systemagent wird dabei als weitere selbst- oder fremdgesteuerte organisatorische Einheit der betrieblichen Aufbauorganisation modelliert, die eine bestimmte Stellenbeschreibung besitzt, bestimmten Randbedingungen unterliegt, ein bestimmtes Fachwissen besitzt und über fest definierte Interaktionsbeziehungen bzw. -möglichkeiten verfügt. Die Rolle des Akteurs in der Ablauforganisation wird beispielhaft dargestellt, indem globa-

le Aktivitätsmuster und -szenarien dokumentiert werden. Weder die einzelnen Interaktionen, noch der Wortschatz oder die internen Aufgaben werden dabei weitergehend formalisiert. D.h. also, dass hier weder eine konkrete Benutzeroberfläche, noch Datenstrukturen oder technische Operationen weitergehend thematisiert oder konkretisiert werden. Dies geschieht erst in der folgenden Softwarespezifikation, die nicht mehr Gegenstand von KoA$_L$A ist.

KoA$_L$A betreibt keine explizite Modellierung, Analyse oder Optimierung von Geschäftsprozessen, wie sie etwa für die Einführung von Workflowsystemen sinnvoll wären. Prägen solche Prozesse allerdings die Arbeitsumgebung des zu entwickelnden IS, können sie als Tätigkeitsprofil von abstrakten organisatorischen Einheiten oder in kommunikationsorientierter Form als Konversationsmuster der beteiligten Akteure natürlich durchaus Eingang in die Modellierung finden.

Schwerpunkt von KoA$_L$A bleibt dabei aber nicht die Entwicklung der betrieblichen Organisation als solche, sondern die optimale "Einpassung" eines neuen IS in die aktuelle betriebliche Arbeitspraxis.

Zur Analyse von Geschäftsprozessen mit dem Ziel der Einführung von Workflow-Systemen sind sinnvollerweise formalere Methode wie die der klassischen Geschäftsprozessmodellierungsansätze (EPK, Petrinetze etc.) einzusetzen, weil diese die Simulation unterstützen, Metriken erlauben (Durchlaufzeiten, Kosten) und von vorne herein ihren Schwerpunkt auf die Modellierung der Arbeitstätigkeit in ihrer konkreten Ausprägung legt, statt sich wie KoA$_L$A mit dem abstrakteren Kontext dieser Tätigkeit zu beschäftigen.

Leitbild des Gestaltungsprozesses von KoA$_L$A ist es nicht, menschliche Arbeitstätigkeit durch Einführung eines Softwaresystems festzulegen oder von außen zu steuern bzw. zu überwachen, sondern die Akteure durch intelligente Assistenten oder Werkzeuge zu unterstützen bzw. in ihren Möglichkeiten zu stärken. Die sich ständig verändernde Arbeitspraxis bestimmt die Nutzung des Werkzeugs, nicht umgekehrt! Damit liegt der Schwerpunkt der Modellierung von vorne herein nicht auf einer Formalisierung und der rational-analytischen und vollständigen Modellierung der konkreten Arbeitsaktivität (wie in der Geschäftsprozessmodellierung, der activity-theory u.a.) , sondern vielmehr auf der umfassenden und damit zwangsläufig auch informelleren Beschreibung der Aktivitätssituation. Dies erreicht KoA$_L$A vor allem über die Berücksichtigung der menschlichen Konversationsmuster, die einhergeht mit einer Analyse der abstrakten Motive, Aufgaben und Ziele der beteiligten Akteure.

1.4. Übersicht über die folgenden Kapitel

Während in den vorhergehenden Abschnitten das Motiv für die vorliegende Arbeit erläutert und die eigentliche Problemstellung eingeführt wurde, findet sich in den folgenden Kapiteln nun die Beschreibung einer möglichen Lösung dieses Problems: die Vorgehensweise nach KoA$_L$A.

Kap. 2 führt zunächst grundlegende Konzeptionen des Software Engineering ein, die für ein Verständnis der dargestellten Vorgehensweise eine zentrale Rolle spielen und die gewählte Lösung plausibel machen. Nach einer Diskussion aktueller Herangehensweisen in Kap. 3 wird KoA$_L$A in den Kapiteln 4 und 5 ausführlich dargestellt und abschließend im Kapitel 6 zusammengefasst.

Im Anhang finden sich schließlich noch das Kap. 6.3 mit einer Diskussion von KoA$_L$A im Vergleich zu ausgewählten Ansätzen, das Kap. A mit einer Darstellung des den Diagrammen von KoA$_L$A zugrundeliegenden Metamodells und das Kap. B mit einer beispielhaften Ausarbeitung von Abschnitten einer textuellen Projektdatenbank.

KoA$_L$A hat sich in zahlreichen Entwicklungsprojekten für die Verwaltung der Universität Ulm bewährt, die von studentischen Hilfskräften, Diplomanden und Studenten im Studiengang Informatik (im Softwaregrundpraktikum arbeiten 30 bis 50 Teams parallel an der gleichen Aufgabenstellung) durchgeführt wurden.

Zur Erläuterung von KoA$_L$A dienen daher im Folgenden Projekte, die in den Jahren 1997 bis 2002 von der Fakultät Informatik für die Verwaltung der Universität realisiert wurden:

Hilve ("Hilfskraftverwaltung", 1997) ist ein IS zur Verwaltung und Budgetierung von Hilfskraftverträgen für die Personalabteilung der Universität Ulm. Neben der reinen Verwaltung der Verträge wurde vor allem eine Kostenberechnung unter Berücksichtigung der gesetzlichen Abgaben incl. der Überwachung der Haushaltsbudgets pro Kostenstelle bzw. Veranstalter realisiert.

Grauwal ("Gremien-Assistenzsystem zur Durchführung von Wahlen", 2000) ist ein IS zur Unterstützung der Organisation, Durchführung und Auszählung von Gremien- und Vertretungswahlen für das Dezernat V, Abteilung 2 "Innerer Dienst".

Apvel ("Ausbildungsplatzverteilung", 2001) ist schließlich ein IS, das für das Studiensekretariat der Universität Ulm entwickelt wurde und zur Verwaltung und Verteilung der Ärzte im Praktischen Jahr (dritter klinischer Ausbildungsabschnitt) dient. Das System unterstützt sowohl Haupt- wie Nachrückverfahren und berücksichtigt dabei überlappende Tertiale, Stellenkapazitäten und Zuteilungspräferenzen.

2. Grundlegendes

"Es sind nur die allzu naiven Menschen, welche glauben können,
dass die Natur des Menschen in eine rein logische verwandelt werden könne"

Nietzsche[1]

Die folgenden Abschnitte beleuchten einige zentrale Konzeptionen und grundlegende Probleme der Softwareentwicklung, die der Entwicklung von KoA$_{L}$A zugrundeliegen. Sie dienen dazu, den Anwendungskontext der Methode näher einzugrenzen und den gewählten Lösungsansatz plausibel zu machen.

2.1. Software Engineering

Das Software Engineering beschäftigt sich mit allgemeinen Problemlösungsmustern, die sich in der "industriellen" Entwicklung von Softwaresystemen unter Randbedingungen wie Wirtschaftlichkeit, Planbarkeit und Qualität bewährt haben bzw. sich dort zu bewähren haben. Dies betrifft dabei alle Tätigkeitsbereiche wie das Projektmanagement, die Softwareentwicklung, die Qualitätssicherung und die Verwaltung von Entwicklungsergebnissen, sowie auch alle Phasen des Entwicklungsprozesses von der Problemanalyse und -definition bis zur Softwarespezifikation, dem Softwareentwurf, der Realisierung und dem produktiven Einsatz.

Menschliches Handeln an sich lässt sich nicht vollständig formal fassen, da es immer kontextbezogen und damit zu komplex für eine lückenlose Analyse ist [Suc87]. Das betrifft naturgemäß auch die Entwicklung von Softwaresystemen selbst. Der praktische Softwareentwicklungsprozess lässt sich daher rein prinzipiell nicht a priori vollständig beschreiben oder festlegen. Er folgt in unterschiedlichen Projektkontexten unterschiedlichen Mustern und ist in der Realität darum oft sogar als irrational zu bezeichnen [PC86].

Eine formale Prozessbeschreibung – auch als "Vorgehensmodell" bezeichnet – kann jedoch durchaus helfen, die Entwicklungsarbeit näherungsweise zu steuern, anzuleiten, zu strukturieren und Entwicklungsprojekte somit effektiv und kontrollierbar zu machen. Von ihrer Natur her sind Prozess- oder Methodenbeschreibungen Handlungspläne, die erst a posteriori konstruiert werden, um Geschehenes und Geschehendes zu konzeptualisieren und es der intellektuellen Auseinandersetzung zugänglich machen zu können [Suc87]. Sie dienen dazu, bewährte Vorgehensweisen zu dokumentieren, um sie diskutieren, lehren und/oder verbessern zu können.

Bei der Entwicklung neuer Softwaresysteme, die für sich genommen stets Unikate darstellen, können Vorgehensmodelle nur eine partielle Hilfestellung bieten und müssen immer auch

[1] Friedrich Nietzsche: Sämtliche Werke. Herausgeber: Giorgio Colli und Mazzino Montinari. Kritische Studienausgabe in 15 Bänden: "Menschliches, Allzumenschliches I" , S. 31

durch ausreichende Praxiserfahrung der Projektbeteiligten ergänzt werden. Die Kontextabhängigkeit der eigentlichen Handlung verhindert die Möglichkeit ihrer Beschreibung a priori. Ein konkretes Vorgehen im Softwareentwicklungsprozess kann also niemals formal definiert, sondern nur idealtypisch und daher näherungsweise beschrieben werden.

Die eigentliche Qualifikation eines Softwareingenieurs besteht daher in seinem Erfahrungsschatz an Handlungsmustern, die sich in realen Entwicklungsprozessen bewährt haben und ihn dazu befähigen, diese Muster intuitiv erneut einzusetzen (siehe dazu z.B. [Kov98]). Diese Kenntnisse sind, wie in anderen Tätigkeitsbereichen von “Experten” auch, oft nicht immer direkt analytisch zugreifbar, sondern eher verstecktes unterbewusstes Praxiswissen (“tacit knowledge”, siehe [Sac95]).

Die Aufgabe des Software Engineering ist nun die Aufdeckung, Entwicklung, Verbesserung, Dokumentation und Evaluation solcher Handlungsmuster mit dem Ziel, Software-Ingenieure in diesen bewährten Vorgehensweisen schulen bzw. unterstützen zu können, um sie in ihrer täglichen Arbeit effizienter zu machen. Dies geht im Grunde nur über den Weg der praktischen Ausbildung, z.B. in Form eines Pilotprojekts bzw. einer ”Fallstudie”.

Eine Methodenbeschreibung selbst kann nur Grundkonzepte vermitteln bzw. das Handwerkszeug für die Arbeit zur Verfügung stellen. Die in der Praxis konkret durchzuführenden Tätigkeitsschritte selbst sind rein prinzipiell nicht lehrbar, sondern nur in der eigenen konkreten Entwicklungstätigkeit erfahrbar und damit nur indirekt auch a posteriori als “Erfahrungsschatz” wiederverwendbar. Software Engineering hat rein prinzipiell keine vollständig generischen Problemlösungsmethoden zu bieten, sondern nur eine Art Baukasten nebst ”Lehrplan” bzw. eine Anleitung für das ”Learning by Doing”.

Insofern ist der Einfluss von Software Engineering Methoden und Techniken in der Industrie auch nur sehr indirekt zu erreichen und besitzt eine große Verzögerung. Die Tatsache, dass Methoden ständig fortentwickelt und verändert bzw. neu entwickelt werden, verbessert diesen Zustand nicht. Neue Erkenntnisse finden keine nachhaltige Verbreitung, so dass sich viele Projektteams immer wieder neu um eine gemeinsame Arbeitsmethodik bemühen müssen.

Im Interesse einer weiten Verbreitung von Softwareentwicklungsmethoden in der Praxis stellt daher die Standardisierung der Methoden und Werkzeuge ein wichtiges Anliegen der Softwareindustrie dar. Dies kann man z.B. auch an der schnellen Akzeptanz und wirtschaftlichen Bedeutung der Unified Modelling Language (UML) ablesen, die eine Form der grafischen Beschreibung objektorientierter Softwaresysteme standardisiert [Rat97].

Zentrale Bedeutung kommt in allen Entwicklungsprozessen der Anforderungsspezifikation zu. Sie dient dazu, unter allen Beteiligten (Kunde, Management, Entwickler, Nutzer etc.) ein gemeinsames Verständnis für die Aufgabenstellung zu erzielen. Wesentliche Aufgabe dabei ist in erster Linie die *Verständigung* und nicht die formale Spezifikation. Zur Beschreibung der Umgebung oder des Softwaresystems haben sich vor allem grafische Modelle als Darstellungsform bewährt, da sie es erlauben, Informationen sehr kompakt darzustellen und durch Komplexitätsreduktion der rationalen Auseinandersetzung zugänglich zu machen.

Die Etablierung eines völlig rationalen Gestaltungsprozesses ist darum auch aus einem anderen Grund unrealistisch: Entwickler besitzen in ihrer Projektarbeit ein Gespür dafür, welche Wege der Kommunikation am effizientesten sind. Die Inhalte und Formen der Verständigung entwickeln sich in einem sozialen System wie z.B. dem Projektteam ständig fort. Verordnete

Standardsprachen haben in lebendigen Systemen oft keine Chance; es wird *immer* die effizienteste Kommunikationsform bevorzugt. Der Entwickler wird also stets *die* Modellierungs- oder Methodenkomponenten verwenden, die er oder die die meisten Beteiligten kennen. Eine bestimmte Entwicklungsmethode dient in einem konkreten Entwicklungsprojekt also immer nur als Mittel zum Zweck und nicht als vorgegebener starrer Rahmen der Aktivitäten. Die zur Verfügung gestellten Modellierungskonzepte und -Techniken müssen daher möglichst flexibel einsetzbar bzw. erweiterbar sein (siehe hierzu [Fit96, PC86, Web01]).

KoA$_{\text{L}}$A trägt diesen Sachverhalten Rechnung, indem es

- keine komplizierte starre und formale Methode definiert, sondern eine leicht lehrbare und lernbare offene Vorgehensweise beschreibt und
- seine Modellierungskonzepte zwar an UML anlehnt und formal definiert,
- sie jedoch flexibel sowohl modell- als auch diagrammübergreifend einsetzt und in ihrer Anzahl und Komplexität weitgehend minimiert.

2.2. Grundlegende Probleme

Im Unterschied zu anderen Ingenieursdisziplinen hat das Software Engineering im Wesentlichen mit folgenden Problemen zu kämpfen (siehe hierzu auch [BR02]):

Software ist immateriell Software besitzt keine materielle äußere Gestalt und ist damit außer in seiner textuellen "Programm"-Form prinzipiell nicht mit den üblichen Sinnen "greifbar". Der Übergang von programmiersprachlichen und anderen beschreibungstechnischen Konzepten zu den dadurch festgelegten Verhaltensweisen eines Softwaresystems erfordert darum einen nicht-trivialen gedanklichen Übergang, der den vielfältig verschachtelten Abhängigkeiten und vielschichtigen Strukturen von Softwaresystemen gerecht wird und daher nur von entsprechend geschulten Personen geleistet werden kann. Ein Softwaresystem und sein in der Regel komplexes Verhalten ist darum für den Endanwender oft nur indirekt erst in seiner konkreten Nutzung erfahrbar.

Aufgrund der abstrakten "Natur" von Software ist die Komplexität von Softwaresystemen weder physikalisch noch durch andere "natürliche" Randbedingungen eingeschränkt, sondern unterliegt im Grunde allein kognitiven Grenzen. Viele Softwaresysteme können darum zu den komplexesten von Menschen geschaffenen Artefakten gezählt werden.

Im etablierten klassischen Ingenieurswesen (Automobilbau, Brückenbau etc.) sind hingegen die grundlegenden Anforderungen weitgehend vordefiniert (wie z.B. der prinzipielle Aufbau eines Personenkraftfahrzeugs aus Motor, Fahrgastzelle, Ladefläche, zwei Achsen mit vier Räder etc.) bzw. eingeschränkt (wie z.B. durch die Fahrphysik, Materialeigenschaften oder einer Straßenverkehrsordnung, die die maximale Breite, Länge und Höhe festlegt).

Es existiert keine Unterscheidung zwischen Entwicklungs- und Produktionsprozess In den Ingenieursdisziplinen ist das Ergebnis der Entwicklung zunächst immer nur ein Prototyp, der erst im Rahmen der Ausarbeitung eines effizienten Produktionsprozesses sukkzessive zu

einem realen "Endprodukt" verbessert, umgestaltet und "neu erfunden" wird. Viele Fehler und Ungereimtheiten fallen erst bei der "Aufbereitung" eines Produkts für den Fertigungsprozess auf.

In der Softwareentwicklung ist das Ergebnis der Entwicklung hingegen auch gleichzeitig bereits das System, welches ausgeliefert und benutzt wird. Die hohen Aufwände für nachgelagerte Anpassungs- und Erweiterungsarbeiten erscheinen damit nicht verwunderlich. Der ökonomische Druck, Fehler, Auslassungen oder notwendige Anpassungen möglichst frühzeitig zu erkennen, ist darum in Softwareprojekten ungleich höher, nicht zuletzt auch deshalb, weil die Entwicklungskosten im Bau- oder Maschinenwesen nur ca. 10 Prozent der gesamten Produktkosten ausmachen. Gleichzeitig ist diese ökonomische Bedeutung der frühzeitigen Anforderungsklärung jedoch in der Praxis vielen Kunden und Softwareingenieuren nicht ausreichend bewusst.

Fehlende Standardisierbarkeit Während im Ingenieurswesen die Entwicklungs- und Herstellungsprozesse umfassend standardisiert sind (mit dem Brückenbau beschäftigt sich z.B. ein ganzer Berufszweig), wird die "Softwareproduktion" in der Regel mit komplexeren, vielfältigeren und immer wieder komplett neuen Problemstellungen konfrontiert, für die es im Allgemeinen keinen vordefinierten und bewährten Problemlösungs- bzw. Herstellungsprozess gibt.

Entwickler stehen oft vor der komplexen Herausforderung, sich in völlig neuartige Aufgabenstellungen und neue Anwendungsbereiche hineinzudenken bzw. diese Problemstellung selbst mit allen Betroffenen kooperativ zu entwickeln. Fehler in der Anforderungsdefinitionsphase verursachen häufig überproportional hohe Entwicklungskosten, so dass der Qualität der identifizierten und spezifizierten Anforderungen große Bedeutung zukommt. Diese entscheidet im Letzten über die Akzeptanz des Softwaresystems in der Anwendungsumgebung und damit über den Erfolg des ganzen Projekts.

Sowohl das Ziel als auch die letztendliche Gestalt bzw. Architektur der Lösung sind fast immer Neuland und Ergebnis eines kontinuierlichen Lernprozesses, der bis in die Realisierungsphase und auch darüber hinaus andauert. Daher ist die Idee der Einführung von standardisierten "Produktionsverfahren" in die Softwareentwicklung nur wenig erfolgversprechend. Der Entwicklungsprozess ist letztendlich ein kollaborativer Lernprozess, dessen Ökonomie eine andere ist als die eines Produktionsverfahrens [Kov01].

Softwareentwicklung bedeutet Change Management Im Vergleich zu den Problemstellungen des Ingenieurswesens ist die Umgebung des zu entwickelnden Systems bei Softwaresystemen in der Regel sehr viel komplexer bzw. in größerem Maß selbst Gegenstand eines zyklischen Gestaltungsprozesses.

Michelis et al. postulieren, dass in der Entwicklung von Informationssystemen drei wesentliche Aspekte einer Organisation zu berücksichtigen sind, die sich gegenseitig so beeinflussen, dass sie nur im Zusammenhang behandeln werden können, wenn ein Entwicklungsprojekt erfolgreich sein soll [MDJ+98]. Die Entwicklung ist danach unweigerlich mit dem Thema "Change Management" bzgl. der Arbeitsorganisation verbunden, das die Aspekte "Kooperation", "Organisation" und "System" gleichermaßen zu berücksichtigen und im Gleichgewicht zu halten

hat (vgl. mit Abb. 2.1).

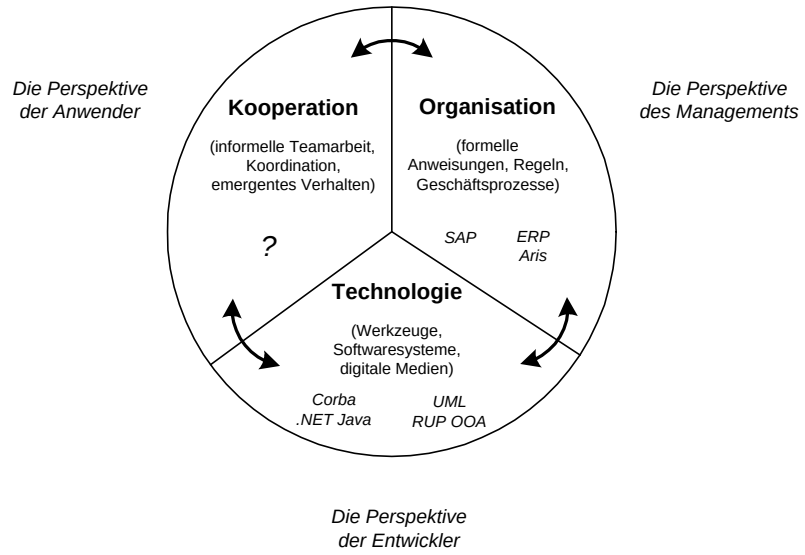


Abbildung 2.1.: Erfolgreiche soziotechnische Systeme halten die Aspekte Mensch, Technik und Organisation im Gleichgewicht

Auch in der allgemeinen Organisationtheorie [Fre90] findet sich eine entsprechende Unterscheidung. Ein Betrieb wird dort zwar als ein rationales Handlungssystem aufgefasst, dessen Handlungen zielorientiert und arbeitsteilig organisiert sind. Solche Handlungssysteme sind dabei jedoch zugleich natürliche Systeme, die informellen Normen und soziopsychologischen Randbedingungen unterliegen. Die Organisationsentwicklung befasst sich daher auch mit Überlegungen zu optimalen Leitungsstrukturen und der Mitarbeitermotivation.

In der Analyse von Anforderungen für Informationssysteme haben sich mittlerweile bei den Modellierungstechniken im Bereich "Organisation" als auch im Bereich "Technologie" einige Standards durchgesetzt. In der Softwareentwicklung ist das bei den Modellierungssprachen UML ("Unified Modelling Language"). Im Bereich "Enterprise Resource Planning" (ERP) sind hier z.B. die geschäftsprozessorientierten Methoden Aris oder Income zu nennen.

Im Bereich "Kooperation" hingegen besteht laut Jarke und Mayr noch erheblicher Forschungsbedarf:

> *"Für die Unterstützung eines arbeitspraxiszentrierten Requirements Engineering fehlen bis heute trotz zahlreicher Versuche standardisierte Ansätze auf einem Verbreitungsniveau, das dem auf der technischen und organisatorischen Modellierungsseite gleichkommt. Dementsprechend leiden auch heute noch viele Softwareentwicklungsprojekte unter unzureichend oder unvollständig erhobenen bzw. validierten Benutzeranforderungen, was später zu massiven Benutzbarkeits- oder Akzeptanzproblemen führt."* [JM02]

Die statische und dynamische Verwobenheit mit anderen Aspekten menschlicher Arbeitsorganisation und Kooperation macht die Entwicklung von Informationssystemen zu einer Heraus-

forderung. Aufgrund der hohen Volatilität des Anwendungskontexts (Wettbewerb, Strategie, gesetzliche Rahmenbedingungen) und der zur Verfügung stehenden Technik ist die Systementwicklung im Gegensatz zu einer klassischen Produktentwicklung kein einmalige oder diskrete Tätigkeit, sondern eine zyklische und fortlaufende Aktivität.

2.3. Requirements Engineering

Mit den vorausgehenden Erläuterungen und dargestellten Problemen der Softwareentwicklung wird deutlich, dass die Phase der Anforderungsermittlung eine zentrale Bedeutung für die Effektivität bzw. den Erfolg von Entwicklungsprozessen besitzt. Als spezielle Vorgehensweise für diese Phase trägt KoA$_{\text{L}}$A dieser Bedeutung Rechnung.

Der Teilbereich des Software Engineerings, der sich mit der systematischen Ermittlung, Beschreibung, Entwicklung und Verwaltung von Anforderungen an ein Softwaresystem befasst, ist das *Requirements Engineering*.

Zur Gewinnung bzw. Abstimmung von Anforderungen haben sich eine Reihe unterschiedlicher Techniken bewährt:

- Studien und Untersuchungen des Anwendungskontexts unter Berücksichtigung von Erkenntnissen der Wissenschaft (Soziologie, Psychologie, Arbeitswissenschaften) und der Anwendung bewährter Techniken wie z.B. der Ethnomethodologie bzw. der Ethnographie (aus dem Umfeld der soziologischen Feldforschung);
- Der direkte Dialog: Kooperative Erarbeitung der Anforderungen mit allen betroffenen Interessengruppen; punktuell (vereinzelte Treffen, Workshops, Interviews) oder kontinuierlich (Zusammenarbeit mit Betroffenen im Team, “participatory design”);
- Der indirekte Dialog: Analyse und Review von Anforderungsdokumenten, Auswertung von Umfragen, Fragebögen, Marktanalysen etc.;
- Grafische Modellbildung: Zur Darstellung von komplexen Sachverhalten in komprimierter und damit handhabbarerer Form, um sich z.B. über komplexe Systeme überhaupt unterhalten oder deren Eigenschaften untersuchen zu können;
- Prototypen bzw. Mock-Ups zur Exploration von Anforderungen oder zur iterativen und evolutionären Systementwicklung aufgrund einer kontinuierlichen Validierung des fertigen Systems durch die Anwender bzw. Betroffenen;
- Formale Analyse: Überprüfung von Anforderungen durch Simulation oder mathematischen Beweis. Nutzbar nur in formal fassbaren Kontexten wie z.B. dem technischen Umfeld eingebetteter Steuerungen, gesetzlich eindeutig festgelegten Regelungen (wie z.B. Auszählungsverfahren bei Wahlen), zu implementierenden standardisierten Kommunikationsprotokollen (wie z.B. ISDN, UMTS, TCP/IP) oder auch den Naturgesetzen beim Flugzeugbau (Fliehkräfte, Strömungstheorie etc.).

Das Konglomerat der eingesetzten Techniken ist natürlich stark von der Art des zu erstellenden Softwaresystems abhängig. Eine formale Analyse findet z.B. im Bereich der Informationssystementwicklung eher seltener statt, da die Arbeitsorganisation sich in der Regel nur

schwer formalisieren lässt. Die Entwicklung von Benutzerschnittstellen-Prototypen hingegen ist in diesem Bereich sehr weit verbreitet.

Besonders fundamental ist für alle Arten von Entwicklungsprojekten die "mündliche" Verhandlung der Anforderungen in Form eines direkten oder indirekten Dialogs mit dem Ziel der Konsensfindung. Als Diskussionsmittel wird hier auf die ein oder andere Weise immer mit modellartigen Wirklichkeitsvorstellungen gearbeitet.

Von ähnlicher fundamentaler Bedeutung ist der Einsatz von Prototypen, um eine Untersuchung der Handhabbarkeit und Nutzens in der realen Arbeitspraxis zu fördern. In Form von Prototypen wird das an sich immaterielle und abstrakte Softwaresystem "greifbar" und damit erfahrbar gemacht, um den Anwendern bzw. Interessengruppen Form, Arbeitsweise und Einfluss des neuen Systems im Kontext der realen Arbeitspraxis deutlich zu machen und eine Validierung der gewählten Systemgestaltung noch möglichst vor seiner endgültigen Realisierung zu ermöglichen. Im Grunde stellt jede initiale Version eines Informationssystems unweigerlich selbst einen Prototyp dar, der sich erst in der Arbeitspraxis bewähren muss und in der Folge oft umfangreiche Anpassungs- und Erweiterungszyklen erfahren muss, bevor seine optimale "Praxistauglichkeit" erzielt ist.

Die Idee von KoA$_{\text{L}}$A ist es nun, den teuren Umweg "Prototyp" zur Validierung der Praxistauglichkeit weitgehend zu vermeiden, indem die neue Arbeitspraxis selbst zum Gegenstand der Modellierung gemacht wird und damit entsprechende Diskussionen, Umgestaltungen und Verbesserungen auf Modellebene stattfinden können. KoA$_{\text{L}}$A erweitert die modellierte Systemebene hierzu über das konkrete technische Softwaresystem hinaus auf das abstrakte Handlungssystem und führt eine angemessene Modellierungssprache ein, die dem soziotechnischen Charakter desselben Rechnung trägt. An die Stelle einer analytischen technischen Modellierung tritt eine systemische Modellierung, die nun im folgenden Abschnitt erläutert wird.

2.4. Die Systemtheorie als Lösungsansatz

Um die Mitte des 20. Jahrhunderts entwickelten sich mit der Systemtheorie und der Ökologie zwei neue Wissenschaftsdisziplinen bzw. Denkweisen, die Zeichen eines Paradigmenwechsel in der wissenschaftlichen Welt darstellen. Der Glaube an eine objektive und wertfreie Wahrheit und den linearen Fortschritt der Wissenschaft wird durch wissenschaftliche Erkenntnisse wie der "Relativitätstheorie" [Albert Einstein] , der "Unschärferelation" [Werner Heisenberg] und der evolutionären Erkenntnistheorie [Konrad Lorenz, Erhard Oeser, Rupert Riedl] in Frage gestellt. Einem auf das messbare Einzelne reduziertem Denken wird ein Denken im Großen und Ganzen gegenübergestellt. Die wissenschaftliche Perspektive wendet sich ab von einer zergliedernden Suche nach den elementaren "Bausteinen" oder der einen "Weltformel" zu einer integrierenden Betrachtung des lebendigen Ganzen als einem Netzwerk von Prozessen[UP95].

Die allgemeine Systemtheorie versucht die Natur großer und komplexer Systeme zu verstehen. Sie beschäftigt sich dabei nicht mit der Struktur der Systembestandteile im Einzelnen, sondern mit den Interaktionsbeziehungen dieser Komponenten und dem Verhalten des Gesamtsystems. Das Verhalten eines komplexen Systems ist nicht vom isolierten Verhalten seiner Komponenten ableitbar, sondern nur durch eine integrierende Betrachtung des Gesamtsystems erschließbar: "Das Ganze ist mehr als die Summe seiner Teile" (frei nach Aristoteles). Im Ge-

gensatz zur verbreiteten rational-analytischen Handlungsmaxime "Devide and Conquer", die zur Bewältigung komplexer Probleme die Aufspaltung in mehrere handhabbare und isoliert beherrschbare Teile propagiert, behält die Systemtheorie das große Ganze im Auge. Während analytische Modelle bestimmte Komponenten einer gegebenen Wirklichkeit tendenziell als objektive und konstruierte Artefakte beschreiben, stellen systemische Modelle eher nur subjektive, d.h. vom Beobachter abhängige, Sichtweisen auf diese Realität dar.

In der Isolation von Systemkomponenten und deren analytischen Behandlung wird im Kontext der Modellierung soziotechnischer Systeme eine unzulässige Simplifizierung gesehen. Bezogen auf die Entwicklung von Informationssystemen fassen das Avison und Fitzgerald z.B. folgendermaßen zusammen:

> *"Decomposing complex structures is the accepted approach in a scientific discipline, but information systems concern peaple as well as technology, and the interactions are such that in these human activity systems it is important to see the whole picture. The human components in particular may react differently when examined singly as when they play a role in the whole system. Human activity systems are not predictable, they are open (interacting with the changing environment) and more complex."*[AF95].

Die grundlegenden Prinzipien einer systemischen Wirklichkeitsbeschreibung, die sich auch gut zur Modellierung soziotechnischer Systeme einsetzen lassen, sind die Folgenden [UP95]:

Das Ganze und seine Teile Ein System ist ein aus Teilen bestehendes dynamisches Ganzes, das nach außen abgrenzbar und von anderen "Dingen" unterscheidbar ist. Jedes Teil steht im Wirkungszusammenhang mit anderen Teilen und beeinflusst dadurch das Verhalten des Gesamtsystems.

Ein System hat nicht unbedingt eine direkte Entsprechung zu einer physischen oder organisatorischen Einheit, so dass die Systemgrenze oft erst mit ihrer Beschreibung bzw. Konstruktion durch einen Beobachter existent wird. Das Innere einer Systemkomponente, das selbst wieder ein System darstellen kann, wird als Black-Box betrachtet. Es hängt immer vom Beobachter ab, ob etwas als System oder als Teil eines Systems aufgefasst wird.

Bsp.: Soziotechnische Systeme sind aus menschlichen und technischen Akteuren aufgebaut, die in vielseitigen Interaktionsbeziehungen zueinander stehen.

Vernetztheit Die Systemkomponenten sind untereinander durch zirkuläre Beziehungen zu einem vielfältigen Netzwerk von Regelkreisen verbunden. Die Wirkungen der Komponenten aufeinander sind in der Regel nichtlinear, zeitlich variabel und rückgekoppelt. Die Wirkungsverläufe benötigen Zeit und führen in ihrem Gesamtbild zur Inkarnation variabler Prozesse.

Bsp.: Die Akteure soziotechnischer Systeme interagieren und kommunizieren miteinander, besitzen spezifische gegenseitige Erwartungshaltungen, gemeinsame Wertmaßstäbe und akzeptierten bestimmte soziale Übereinkünfte.

Offenheit Ein System ist niemals isoliert, sondern steht mit seiner Umgebung in Wechselwirkung. Das Verhalten eines Systems ist nur mit dem Wissen seiner Einbettung in diesen

umfassenden Kontext zu begreifen. Ein System steht mit seiner Umgebung im Austausch und kann sich an sie aktiv anpassen.

Bsp.: Ein Softwaresystem ist nur im Bezug zu seiner soziotechnischen Umgebung zu begreifen und muss zwangsweise fortlaufend an diese angepasst werden ("Wartungsphase").

Komplexität Ein System ist in der Regel nicht nur in seiner statischen Zusammensetzung *kompliziert*, sondern auch in seinem dynamischen Verhalten *komplex*, d.h. es kann innerhalb kurzer Zeit so viele verschiedene Zustände annehmen, dass sein eigendyamisches Verhalten nicht vollständig berechenbar bzw. vorhersagbar ist.

Bsp.: Das Verhalten eines soziotechnischen Systems lässt sich nicht vorhersagen, es ist inhärent emergent.

Ordnung Das Verhalten und die Strukturen komplexer Systeme lassen normalerweise immer Ordnungsmuster erkennen. Sein Verhalten ist nicht willkürlich und zufällig, sondern lässt eine Reihe von sich wiederholenden musterhaften Abfolgen und Szenarien erkennen, die sich aufgrund "eingebauter" Regeln, Prinzipien und standardisierten Abläufen realisieren.

Bsp.: In soziotechnischen Systemen lassen sich immer standardisierte Konversations- (z.B. Standard-Dialoge) und Aktivitätsmuster (z.B. "Geschäftsprozesse") identifizieren, die sich regelmäßig wiederholen.

Lenkung Ein System kann sich in seinem Verhalten selbst unter Kontrolle halten, indem es sich sowohl Steuerungsprozessen zur Erreichnung definierter Ziele unterwirft, als auch durch die Implementierung selbständiger Regelkreise möglichst gegen etwaige Abweichungen vom angestrebten Verhalten absichert. In komplexen Systemen ist die Lenkungsfunktion über das System verteilt.

Bsp.: Jeder menschliche Akteur in einem soziotechnischen System ist auf der einen Seite in seinem Verhalten autonom – er "lenkt" sich also selbst – auf der anderen Seite ist er jedoch durch gegenseitige Vereinbarungen, Aufträge und die Anforderungen seiner ausgeübten Rollen ("Stellenbeschreibung") entsprechend in seinem Verhalten eingeschränkt bzw. reglementiert.

Entwicklung Ein System ist nicht statisch, sondern entwickelt sich fortlaufend weiter, nicht zuletzt auch aufgrund einer sich selbst ständig verändernden Systemumgebung. Entwicklungsprozesse können als Lernprozesse des Systems aufgefasst werden.

Bsp.: Ein soziotechnisches System wie ein bestimmtes Unternehmen entwickelt sich fortlaufend weiter. Die Gründe dafür sind vielseitig: Anpassungen an neue Wettbewerbssituationen, Diversifizierung des Produktspektrums, Rationalisierung, sich wandelnde gesetzliche Rahmenbedingungen, Lernprozesse im Management wie bei den Mitarbeitern etc.

Systemebenen der Softwareentwicklung

Aus Sicht der Systemtheorie kann der Prozess der Softwareentwicklung als schrittweiser Verfeinerungsprozess gesehen werden, der auf sukzessive sich konkretisierenden Abstraktionsstufen bzw. Systemebenen die beiden folgenden Schritte durchläuft:

1. *Charakterisierung* des Systems als Einheit: Beschreibung der Funktion, des Verhaltens, der "Rolle" des Systems in seiner Umgebung, d.h. im umgebenden System. Festlegung seiner Bedeutung in Bezug zu seiner Umwelt. Beschreibung der "Semantik" des Systems.

2. *Konstruktion* des Systems aus kleineren Einheiten: Beschreibung des Aufbaus des Systems auf Grundlage kleinerer Systembestandteile, die in spezifischer Weise miteinander verknüpft sind. Festlegung seiner Struktur und seines Wirkungszusammenhangs. Beschreibung der "Syntax" des Systems.

Die Phase der Charakterisierung eines Systems kann als die Definition eines Gestaltungsproblems verstanden werden, welches in der anschließenden Konstruktionsphase durch den Entwurf einer bestimmten internen Systemstruktur gelöst wird.

Die Entwicklung betrieblicher Informationssysteme lässt sich beispielsweise in folgende systemische Abstraktionsstufen zergliedern:

1. Das *Handlungssystem* (ein soziotechnisches System wie z.B. ein Unternehmen, welches Informationstechnik einsetzt)

   Charakterisierung:
   : Geschäftsziele ("Business Goals") und Regeln, Identifikation von Kunden- und Geschäftsbeziehungen, zentralen Dienstleistungen, Anforderungen an das Unternehmen.

   Konstruktion:
   : Aufbau- und Ablauforganisation (z.B. in Form von Geschäftsprozessen). Netzwerk interagierender menschlicher und technischer Akteure (z.B. Informationssysteme).

2. Das *Softwaresystem*

   Charakterisierung = "Anforderungsanalyse":
   : Rolle des Systems in seiner Umgebung (menschliche Anwender oder technische Nachbarsysteme). Interaktionsverhalten ("essential use cases"), Nutzungskonzept, Aufgaben, Randbedingungen und andere Anforderungen.

   Konstruktion = "Softwarespezifikation":
   : Festlegung der Benutzungsschnittstellen ("concrete use cases"), des Systemkerns und aller anderer verwendeter Subsysteme.

3. Der *Systemkern*

   Charakterisierung:
   : Spezifikation von Systemoperationen, konzeptionellem Datenmodell und Verhalten.

Konstruktion = "Softwareentwurf":
Zerlegung in Komponenten (z.B. als objektorientierter Klassenentwurf).

4. Die *Programmkomponente*

   Charakterisierung:
   Imperative oder deklarative Spezifikation von Komponenten (Klassen, Module, Funktionen etc.)

   Konstruktion = "Programmierung":
   Datenstrukturen, Befehle, Nutzung von Programmbibliotheken.

Für jede dieser Systemebenen gibt es angemessene spezialisierte Beschreibungs- und Konstruktionstechniken (siehe hierzu auch Kap. 3). Zur Charakterisierung eines IS auf Ebene 2 (Softwaresystem) eignet sich z.B. die Beschreibung allgemeiner Interaktionsmöglichkeiten im Stile kommunikationsorientierter Dialoge – als sogenannte "essential use cases" [CL99], während eine Konkretisierung dieser Dialoge z.B. in Form des Entwurfs einer bestimmten grafischen Oberfläche plus Festlegung atomarer Interaktionsmöglichkeiten erst in einer späteren Verfeinerungsebene angesiedelt ist.

Ein besonders interessantes System ist das sogenannte *Nutzungssystem* [Pae00], das neben dem Softwaresystem selbst auch alle seine direkten Nutzer bzw. "Betroffenen" umfasst (siehe Abb. 2.2). Es stellt damit keine eigene systemische Abstraktionsebene, sondern nur einen bestimmten Ausschnitt des Handlungssystems dar. Das Nutzungssystem modelliert die konkrete Arbeitssituation aller direkt vom System betroffenen Akteure und beschreibt dabei die Verteilung der fachlichen Aufgaben auf das Softwaresystem und seine Umgebung. Es dokumentiert eine gemeinsam erschaffene zukünftige Arbeitswelt und ist damit nach [Nur98] im Kontext der Entwicklung von Informationssystemen die eigentliche Anforderungsspezifikation: "*Requirements specification abstractly describe a future real world which stakeholders and requirements engineers have agreed on*".

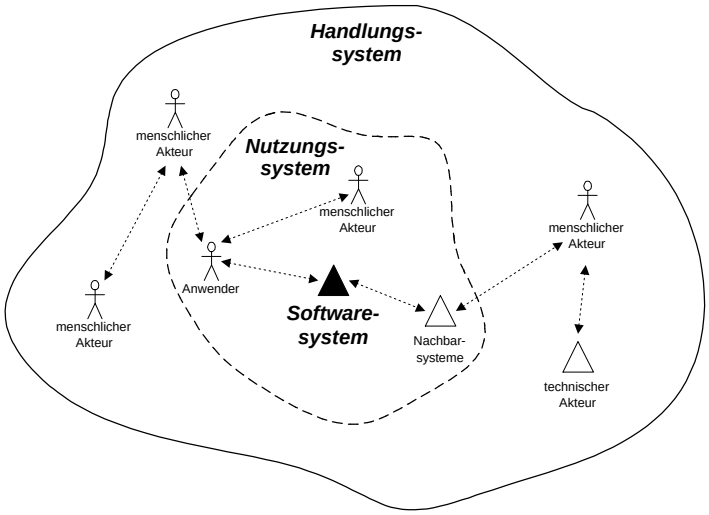


Abbildung 2.2.: Die Systemebenen

Menschliche Handlungssysteme sind im Gegensatz zu technischen Systemen nichtdeterministisch, eingebettet in eine sich stets verändernde Umgebung und nie vollständig analysierbar.

Sie eignen sich damit gut für einen systemischen Modellierungsansatz.

KoA$_{\text{L}}$A verwendet daher eine systemische Modellierungssprache zur Darstellung, Konstruktion bzw. Ausgestaltung von Nutzungssystemen (siehe Kap. 4).

2.5. Die Aktivitätstheorie als Lösungsansatz

Die Softwareentwicklung ist eine menschliche Aktivität. Nach Maßgabe der Aktivitätstheorie [KKB95][2] lassen sich bei jeder Aktivität folgende Aspekte unterscheiden:

1. Das Motiv: *Warum* wird man aktiv? Die Beweggründe, Probleme, Visionen und übergeordnete Ziele.
2. Das Ziel: *Was* will man erreichen? Der gewünschte Endzustand.
3. Der Plan: *Wie* will man dabei vorgehen? Die Festlegung des Vorgehens auf Basis von verfeinerten Einzelaktivitäten: Wie lassen sich Hauptziele auf Unterziele "herunterbrechen"?
4. Die Durchführung der Aktivität selbst: *Wodurch* wird das Ziel erreicht? Die real durchgeführten Einzelaktivitäten bzw. Tätigkeitsschritte.

Jede Aktivität lässt sich damit auf der einen Seite rekursiv in Unteraktivitäten zerlegen bzw. verfeinern und auf der anderen Seite rekursiv in übergeordnete Aktivitäten eingliedern.

Das Ziel einer übergeordneten Aktivität stellt das Motiv aller untergeordneten Aktivitäten dar. Das Ziel einer übergeordneten Aktivität wird erreicht (realisiert), indem Ziele für untergeordneten Aktivitäten festgelegt und diese realisiert werden. Diese Reduktion bzw. Lösung eines Problems durch eine Verfeinerung in konkrete Teilziele stellt den eigentlichen kreativen Akt einer Entwicklungsaktivität dar, der immer auch gleichzeitig einen gestalterischen Aspekt beinhaltet. Da das Hauptziel implizit das Motiv der Unterziele ist, lässt sich diese Frage alternativ auch so formulieren: "Wie komme ich vom Motiv zur Zieldefinition?". Die Problemlösung bzw. Gestaltung, d.h. die Zielsetzung, kann durch Methoden und Vorgehensmodelle unterstützt werden, die im Normalfall mehrere dieser Aktivitätsebenen umspannen.

Im Kontext der Softwareentwicklung lassen sich z.B. folgende Aktivitäts- bzw. Gestaltungsebenen unterscheiden. Für die Gestaltungstätigkeiten wird im Folgenden bewusst der Begriff "Design" verwendet[3]. Der Brockhaus definiert diesen Begriff als "Formgerechte und funktio-

[2] Die "Activity theory" von Alexey Leontiev begründet ein philosophisches bzw. psychologisches Rahmenwerk für die Analyse -gemeinsamer- menschlicher Aktivität. Sie versucht die Kluft zwischen Sozial- (gemeinschaftliches Verhalten) und Geisteswissenschaft (individuelles Verhalten) zu schließen und über Probleme des psychologischen Teilgebiets der "menschlichen Informationsverarbeitung" hinwegzukommen. Nach dieser Theorie entsteht menschlicher Geist erst mit praktischem Handeln in der Welt. Menschliche Aktivität wird damit zum Untersuchungsgegenstand psychologischer Analyse.

[3] An der Kunsthochschule Kassel ist neben dem "Produktdesign" auch das "Systemdesign" einer der Lehrschwerpunkte des Produktdesign-Studiums. Dabei steht nicht die Entwicklung eines Produkts im Vordergrund, sondern der Entwurf von ganzheitlichen Systemen und Abläufen in den Bereichen Organisation, gesellschaftlichen Situationen oder digitalen Räumen!

Der Begriff "Design" ist im Folgenden insbesondere nicht nur auf die Entwicklung der Grob- bzw. Feinarchitektur von Softwaresystemen beschränkt. Er bezeichnet allgemein einen kreativen Gestaltungsprozess, der auf mehreren Ebenen der Gestaltung stattfindet!

nale Gestaltung, vor allem von Gebrauchsgegenständen (-> Industriedesign)", was auch im Bezug zur Ausgestaltung von Informationssystemen eine treffende und tragfähige Bezeichnung darstellt. Leitbild für den Entwicklungsprozess ist das kreative Design, nicht die Produktion – im Gegensatz zur etablierten Sichtweise, (siehe z.B. [RP99], S. 781).

1. Unternehmensdesign

   Akteure: das Management des Unternehmens

   Ziel: Gewinn, Wachstum, Erfolg

   Methoden: der Betriebswirtschaft (z.B. BPR), der Marktforschung etc.

   Teilziele: Effizienzsteigerung, Rationalisierung, Problembeseitigung, Produktdiversifikation etc.

2. Organisations- bzw. Kontextdesign

   Akteure: die Organisationsabteilung, Stakeholder, IT-Berater, Moderatoren

   Ziel: Effizienz, Problembeseitigung, Zielvorgaben etc.

   Methoden: SSM ("soft systems methodology"), STD ("sociotechnical systems design"), PD ("participatory design"), "contextual design",KoA$_{\text{L}}$A etc.

   Teilziele: Stellen, Rollen, Aufgaben, Anforderungen, Abläufe, Interaktionen von menschlichen und technischen Akteuren wie z.B. Informationssysteme

3. Softwaredesign[4]

   Akteure: ein zusammengesetztes Projektteam: Auftraggeber, Stakeholder, Entwickler etc.

   Ziel: die Vision eines IS als neuer technischer Akteur in der Organisation

   Methoden: OOA, SA, HCI etc.

   Teilziele: werden durch eine Softwarespezifikation definiert (in Begriffen der vom System sichtbaren Außenwelt und der von außen sichtbaren Systemwelt), welche die Systemschnittstelle festlegt (Benutzeroberfläche, Funktionen, Informationseinheiten, Verhalten)

4. Komponentendesign

   Akteure: das Softwareentwicklungsteam

   Ziel: das spezifizierte Softwaresystem

   Methoden: Entwurfstechniken wie CRC oder OID etc.

   Teilziele: Softwarearchitektur, Komponentenspezifikation

[4]Bezeichnet den kreativen Entwurf des Softwaresystems in seiner *nach Außen* sichtbaren Gestalt. Nicht zu Verwechseln mit dem im Software Engineering weit verbreiteten Begriff "Design" bzw. "Entwurf", der dort hauptsächlich nur im Zusammenhang mit der Entwicklung bzw. Dokumentation der inneren Struktur eines Softwaresystems, der "System-Architektur" verwendet wird. Diese Gestaltungsebene wird im Folgenden durch den Begriff "Komponentendesign" präzisiert.

5. Programmdesign

   Akteur: der Programmierer

   Ziel: die spezifizierten Komponenten

   Methoden: OOP, strukturierte Programmierung etc.

   Teilziele: Algorithmen in Form realen Programmcodes

Aus einer rational-analytischen Perspektive heraus findet der Übergang von einer Ebene zur anderen dabei sequentiell durch eine sukkzessive Verfeinerung der Handlungspläne statt. Die Ebenen beschreiben dann den klassischen Weg einer Problemreduktion durch schrittweise Verfeinerung der Einzelaufgaben ("Divide and Conquer"). In einer realistischeren Sichtweise, die den selbstbezüglichen Charakter dieser Aktivitäten berücksichtigt, sind diese Tätigkeiten natürlich nicht streng sequentialisierbar. Teilaspekte der konkreten Gestaltung des Softwaresystems nehmen Einfluss auf die Organisationsgestaltung, wie auch entsprechende organisatorische Maßnahmen Einfluss auf die globale Unternehmensführung ausüben. Die einzelnen Gestaltungsaktivitäten spielen ineinander und beeinflussen sich gegenseitig, da sich die Handlungsstränge gegenseitig überlappen und im Normalfall iterativ durchlaufen werden müssen.

Anwendungsbereich von KoA$_{\text{I}}$A ist die von den etablierten Softwareentwicklungsmethoden bisher vernachlässigte Aktivitätsebene des "Kontextdesigns" und damit die Frage nach dem Weg zur Vision eines bestimmten IS als neuer technischer Akteur. Die Ebene der Softwareentwicklung, also der Weg von der Vision eines IS in Form einer praxisorientierten Rollendefinition hin zur formalen Softwarespezifikation bleibt etablierten Methoden überlassen.

Die Bedeutung von Informationssystemen im Hinblick auf die Arbeitsorganisation und ihre wechselseitige Abhängigkeiten bzw. Wirkungen wurden in den Standardmethoden des Software Engineering bislang eher stiefmütterlich behandelt. Der vorherrschende Ausgangspunkt ist die Annahme, dass die Anforderungen an das Softwaresystem bereits prinzipiell existent sind, also nur noch "eingesammelt" und "analysiert" werden müssten. Neuere Erkenntnisse gehen hingegen davon aus, das solche Anforderungen in der Regel erst mit dem Anwender zusammen "konstruiert" werden und mit diesem Design gleichzeitig eine Organisationsentwicklung einhergeht bzw. gehen muss ([Flo92], [Che89], [WAP88] etc.).

Analysiert man die Aktivität der Softwarerealisierung nach Maßgabe der Aktivitätstheorie, stellt man fest, dass ihre Aspekte wie "Motiv", "Ziel", "Plan" und "Durchführung" zum großen Teil – und unter den unterschiedlichsten Namen – Niederschlag in den verschiedenen Prozessphasen etablierter Standardvorgehensmodelle (wie z.B. dem Wasserfallmodell) gefunden haben. Die statischen inhaltlichen Aspekte einer Aktivität ergeben also auch eine natürliche Strukturierung der dynamischen Aspekte eines Entwicklungsvorhabens, nämlich des sich aus den Einzelaktivitäten konstituierenden Entwicklungsprozesses.

Dieser Prozess lässt sich allgemein in folgende Subphasen strukturieren:

1. **Motivierung** (Entwicklung der **Vision**): *Warum* wird das IS entwickelt? Wozu dient es, welche Rolle soll es spielen? Was sind die Aufgaben dieses neuen Akteurs? Welche Probleme werden durch seinen Einsatz adressiert, welche Verbesserungen werden beabsichtigt? Was soll es in der Anwendungsumgebung/Außenwelt leisten und bewirken?

Verwandte Begriffe**:** Anforderungsdesign, Problemidentifikation, User Requirements Specification, Vorstudie, Lastenheft, Produktskizze, Planung

Entspricht dem ***Kontextdesign***: Das IS und sein praktischer Arbeitskontext werden entworfen und fachlich beschrieben. Die Beschreibung umfasst eine Festlegung der konzeptionellen Begriffswelt des IS, sein Interaktionsverhalten mit der Außenwelt, die zu berücksichtigenden Randbedingungen (Anforderungen, Erwartungen, Aufgaben, Konsistenzbedingungen), seine Einbettung in die Arbeitspraxis anderer Akteure des Handlungssystems, eine Beschreibung seiner Rolle etc. Die Beschreibung ist in der Begriffswelt des Arbeitsumfelds abgefasst und definiert die beabsichtigten "Wirkungen", die der Einsatz des IS zeitigen soll bzw. das "Problem", welches durch das Entwicklungsprojekt gelöst werden soll:

> *"Inventing requirements is a matter of inventing a well-defined problem to solve" [Kov98]*

Inhalt dieser Phase ist auch eine Analyse dieser Problemdefinition, wozu eine Untersuchung der Praktikabilität ("Durchführbarkeitsstudie"), des Kosten/Nutzen-Verhältnisses (Kosten aufgrund vorläufiger Planung), möglicher Alternativen und der *Risiken* gehört.

2. **Zielsetzung**: *Was* ist das Ziel der Aktivität? Was soll gebaut werden? Was sind seine Eigenschaften? Was soll das System leisten? Wie soll es von Außen aussehen?

   Verwandte Begriffe**:** Softwarespezifikation, Software Requirements Specification, Pflichtenheft

   Entspricht dem ***Softwaredesign***: Nachdem bereits ein praxisorientiertes Leistungsprofil für ein IS im Kontext des Nutzungssystems und der Begrifflichkeit der Benutzungswelt entwickelt wurde, wird das Softwaresystem nun detailliert aus technischer Systementwicklungssicht spezifiziert, indem Daten, Funktionen, Verhalten und die Benutzerschnittstelle genau definiert werden. Die eigentliche Systemmodellierung kann beginnen.

   Hier ist die klassische Softwareanalyse am Zug, die nahtlos an die Ergebnisse von KoA$_{\text{L}}$A anschließt und eine Lösung des definierten Problems systematisiert, d.h als Softwaresystem beschreibt. Spezifiziert wird das "Was", das konkrete Projektziel.

   Das System wird als Blackbox beschrieben. Beschrieben werden also hauptsächlich seine "Schnittstellen" nach außen:

   - Statik: Dialoglayout, physikalische Schnittstellen, Dateiformate, Konzeptuelles Datenmodell
   - Funktionen: Leistungen, Kontrollflüsse, Informationsflüsse etc.
   - Dynamik bzw. Verhalten (per Zustandsautomat, Protokolldefinitionen etc.)

3. **Planung**: *Wie* will man vorgehen, um das Ziel zu erreichen? Wie kann man so ein System überhaupt bauen?

   Verwandte Begriffe**:** Architektur, Systementwurf, Projektplanung

   Entspricht dem ***Komponentendesign*** und der ***Projektplanung****:* Hierzu gehört eine Beschreibung der Architektur bzw. des Bauplans, der verwendeten (Fremd)Komponenten, der vorgesehenen Realisierungseinheiten und ihrer Abhängigkeiten (Projektplan), die

Berücksichtigung aller Einschränkungen (Systembeschränkungen, Randbedingungen, Urlaubsplan, etc.), eine Zuordnung von Zuständigkeiten an Entwickler, eine Zeitplanung der Projektarbeit etc.

Auch zu dieser Phase gehört wiederum eine Analyse dieser Spezifikation, wozu eine Untersuchung der Praktikabilität, des Kosten/Nutzen-Verhältnisses, möglicher Alternativen und der *Risiken* gehört.

4. **Durchführung**: *Wodurch* wird das Ziel erreicht? Welche konkreten Tätigkeiten müssen durchgeführt werden, um das Ziel zu erreichen?

   Verwandte Begriffe**:** Realisierung, Implementierung, Codierung

   Entspricht dem ***Programmdesign***: Das Softwaresystem wird realisiert bzw. ausprogrammiert.

In der allgemeinen SE-Literatur werden die beiden Phasen Motivierung und Zielsetzung irrtümlicherweise oft miteinander gleichgesetzt. Im Besonderen werden dabei dem Begriff "Anforderung" implizit zwei fundamental unterschiedliche Bedeutungen gegeben, diese aber nicht explizit diskutiert. Es wird nicht klar zwischen den *Anforderungen* an das Softwaresystem und der *Spezifikation* des Softwaresystems selbst unterschieden, das, wenn realisiert, diese Anforderungen erfüllt. Problem- und Lösungsbeschreibung werden damit also oft nicht unterschieden oder nur zum Teil berücksichtigt. Die Motivierung fristet in der Regel ein Schattendasein, während die Zieldefinition, d.h. die Softwarespezifikation, oft als Anforderungsdefinition bezeichnet wird.[5]

Schwerpunkt der KoA$_{\text{L}}$A-Methode ist die Motivierungsphase, d.h. das *Kontextdesign*, und zwar im konkreten Kontext der Entwicklung betrieblicher Informationssysteme und unter der Annahme, dass die Methode von Softwareentwicklern angewendet wird, die in den klassischen Methoden des Softwaredesigns bewandert sind und entweder die Akzeptanz und damit die Qualität ihres Entwicklungsprodukts verbessern wollen, eine konkrete Anleitung für die *Entwicklung* von Anforderungen suchen oder sich nur den Einstieg in das Softwaredesign erleichtern wollen. KoA$_{\text{L}}$A betrachtet damit nur einen kleinen Ausschnitt des sehr umfangreichen Themas der allgemeinen Organisationsentwicklung.

2.6. Ein erweitertes Phasenmodell

"*Requirements engineering from an organisational perspective needs to be viewed as social engineering*" [BCDS93]

Nach den Vorgaben des Software Engineering geschieht die Entwicklung technischer Artefakte planmäßig in Form einzelner zielorientierter Arbeitsschritte. Dabei werden diese Einzelschritte sowohl zeitlich bestimmten Phasen als auch inhaltlich bestimmten Tätigkeitsbereichen (RUP spricht hier von "workflows") zugeordnet. Da sich die Tätigkeitsbereiche oft einer bestimmten

[5] Insbesondere viele objektorientierte Softwareentwicklungsansätze verwechseln die Anforderungsanalyse mit einer Systemspezifikation. Siehe z.B. [CAB+93], [Jac94b], [Car94].

Eine schöne Unterscheidung dieser beiden Phasen findet sich jedoch z.B in [GGJZ00] oder [Pre94] während sich in [Ach00] sogar ein Vorgehensmodell für die Spezifikation von Anforderungen an Prozessautomatisierungssysteme findet, das zentral auf diese Unterscheidung aufbaut.

Phase zuordnen lassen, tragen sie häufig denselben Namen. Das Standard-Phasenmodell der Softwareentwicklung (siehe z.B. [RP99], S. 777 oder [Hes98]) kennt hier die Phasen Analyse & Definition, Entwurf, Implementierung, Test, Einsatz & Wartung und versteht sie auch als ebensolche Tätigkeitsbereiche. RUP hingegen teilt den Entwicklungsprozess in die zeitlichen Abschnitte Inception, Elaboration, Construction und Transition ein, während es als einzelne Tätigkeitsbereiche die workflows "Business Modelling" (Business-Use-Cases und Workflows), "Requirements" (vision, features, use cases), "Analysis & Design" (Architekturentwurf), "Implementation", "Test", "Deployment", "Configuration & Change Management", "Project Management" und "Environment" unterscheidet. Wie auch das V-Modell berücksichtigt RUP als Tätigkeitsbereiche eines Entwicklungsprojekts damit auch die über die reine Systemerstellung hinausgehenden Aktivitäten des Projektmanagements, der Qualitätssicherung und des Konfigurationsmanagements.

Phasen werden normalerweise ergebnisorientiert über ihre Endprodukte definiert. Diese sogenannten "Meilensteine" stellen die notwendigen Zwischenprodukte auf dem Weg zum fertigen Artefakt dar. Diese idealisierte Sicht auf den Entwicklungsprozess hat sich vor allem aus Managementsicht bewährt, da sie den Erstellungsprozess plan- und kontrollierbar macht. Dieses Idealbild der streng sequentiellen Entwicklung des Endprodukts über definierte "Produktionsstufen" mit fest definierten Zwischenergebnissen hinweg liegt auch dem klassischen "Wasserfall"-Prozessmodell zugrunde.[6] In der Realität beeinflussen und überlappen sich die Tätigkeitsbereiche jedoch oft gegenseitig (siehe Abb. 2.3), so dass sich in der Regel auch die Zwischenergebnisse dynamisch fortentwickeln. Die Softwareentwicklung wird damit zu einer Tätigkeit, die parallel auf unterschiedlichen Gestaltungsebenen bzw. Abstraktionsstufen tätig ist und die sich jeweils in eigenen Artefakten niederschlagen, den Zwischenergebnissen in Form von Dokumenten, Modellen oder Programmfragmenten. Oft werden mehrere dieser Entwicklungstätigkeiten in Personalunion wahrgenommen.

KoA$_{L}$A ergänzt bzw. konkretisiert die erste Phase des Standard-Phasenmodells um eine weitere Phase bzw. einen Tätigkeitsbereich: das Kontextdesign. Verglichen mit dem Phasenmodell von RUP erweitert und konkretisiert KoA$_{L}$A sowohl den Workflow "Business-Modelling" als auch die erste Hälfte des "Requirements"-Workflows, der sich mit der Definition der "User-Needs" in Form von "Features" beschäftigt. Im V-Modell [BD93] könnte man das Kontextdesign vergleichsweise den Aktivitäten SE 1 "Systemanforderungsanalyse" und SE 2 "System-Entwurf" zuordnen, das Softwaredesign der folgenden Aktivität SE 3 "SW-Anforderungsanalyse". In keiner dieser Vorgehensmodelle wird explizit auf den notwendigen Gestaltungsprozess eingegangen oder gar eine methodische Unterstützung desselben angegeben. KoA$_{L}$A identifiziert im Kontextdesign eine eigenständige Gestaltungsebene bzw. einen Tätigkeitsbereich und gibt hier eine konkretisiertere Vorgehensweise zur Entwicklung der Nutzer-Anforderungen an. Aktuelle Methoden sind hingegen überwiegend der kausalanalytischen Denkweise zuzurechnen, die die soziotechnische Welt allzu rational zu definieren sucht und den Übergang von der Ist-Kontextmodellierung zur Definition der Nutzer-Anforderungen weitgehend offen lässt bzw. zu sehr simplifiziert (siehe Kap. 2.4).

In Abb. 2.4 wird die Unterscheidung von Kontext- und Softwaredesign bzw. der Unterschied der Gestaltungswelten des soziotechnischen Systems und des technischen Softwaresystem noch einmal genauer ausgearbeitet.

[6] Für eine Kritik aktueller Vorgehensmodelle, die diese Idealisierung in der Regel beibehalten, siehe z.B. [Hes96, Hes97]

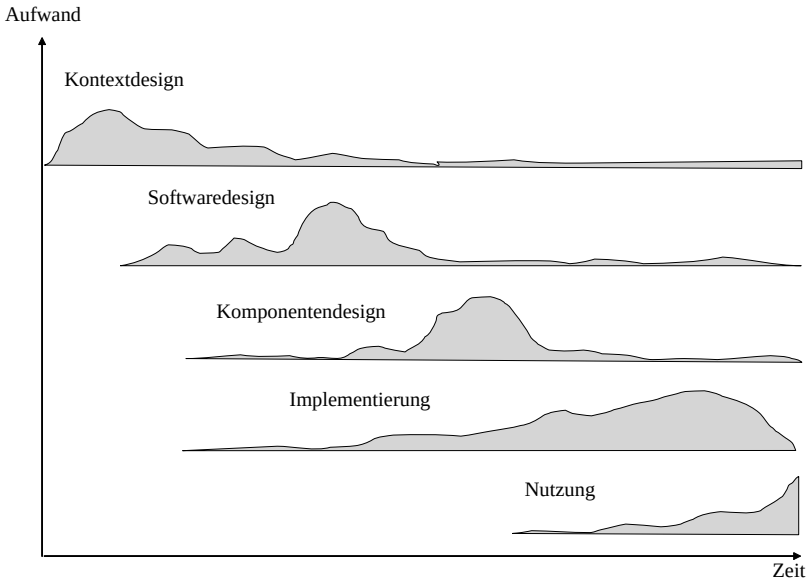


Abbildung 2.3.: Die überlappenden Tätigkeitsbereiche eines Softwareprojekts

Die Entwicklung betrieblicher Informationssysteme tangiert unweigerlich zwei fundamental unterschiedliche Welten: die der Technik und die der menschlichen Arbeitsorganisation. Jedes Entwicklungsvorhaben nimmt in beiden Bereichen Gestaltungsoptionen wahr, die Auswirkungen auf den jeweils anderen Bereich besitzen. Die Gestalt eines technischen Softwaresystems bedingt eine Anpassung von Arbeitsprozessen auf Anwenderseite und umgekehrt. Die beiden Gestaltungsebenen sind zwar eng verknüpft, jedoch durchaus trennbar. Mit einer anwenderorientierten Beschreibung der Rolle eines technischen Akteurs ist noch keine Aussage über seine konkrete Gestalt im Sinne einer Softwarespezifikation notwendig. Die Rolle eines IS lässt sich noch in der Begrifflichkeit der realen Anwendungswelt beschreiben. Erst die konkrete Ausgestaltung des Softwaresystems ist in einer formalen technischen Begrifflichkeit zu beschreiben.

KoA$_{\text{L}}$A bezeichnet den Gestaltungsprozess im Bereich des soziotechnischen Systems als "Kontextdesign", im Bereich des technischen Softwaresystems hingegen als "Softwaredesign". Während sich in der Beschreibung technischer Systeme formale Spezifikationsmodelle bewährt haben, setzt KoA$_{\text{L}}$A zur Darstellung von soziotechnischen Systemen systemische Modelle ein. Der Schritt von der einen "Welt" zur anderen ist ein eigener kreativer Schritt der Problemlösung: das "Softwaredesign". Das "Kontextdesign" jedoch ist allein in der Welt der realen Arbeitspraxis beheimatet und kann bzw. muss daher auch in der Begrifflichkeit der Anwendungswelt stattfinden, um z.B. eine Validierung durch die betroffenden Interessengruppen zu ermöglichen. Die Rolle eines Softwaresystems wird dabei in der Sprache ihrer Anwender, in ihrer Realitätsauffassung bzw. Denkwelt beschrieben ("Das IS 'Grauwal' soll das Wahlergebnis berechnen", "Das IS 'Apvel' soll eine Kostenabrechnung für jedes Krankenhaus erstellen"), und nicht in der formalisierten Denkwelt der Programmentwicklung.

Das Kontextdesign, die Bestimmung der Rolle des neuen Informationssystems und seine Einbettung in die Arbeitspraxis, ist eine eigenständige Phase, die logisch noch vor der konkreten Ausgestaltung des Softwaresystems kommt. Während durch das Ergebnis des Kontextdesigns

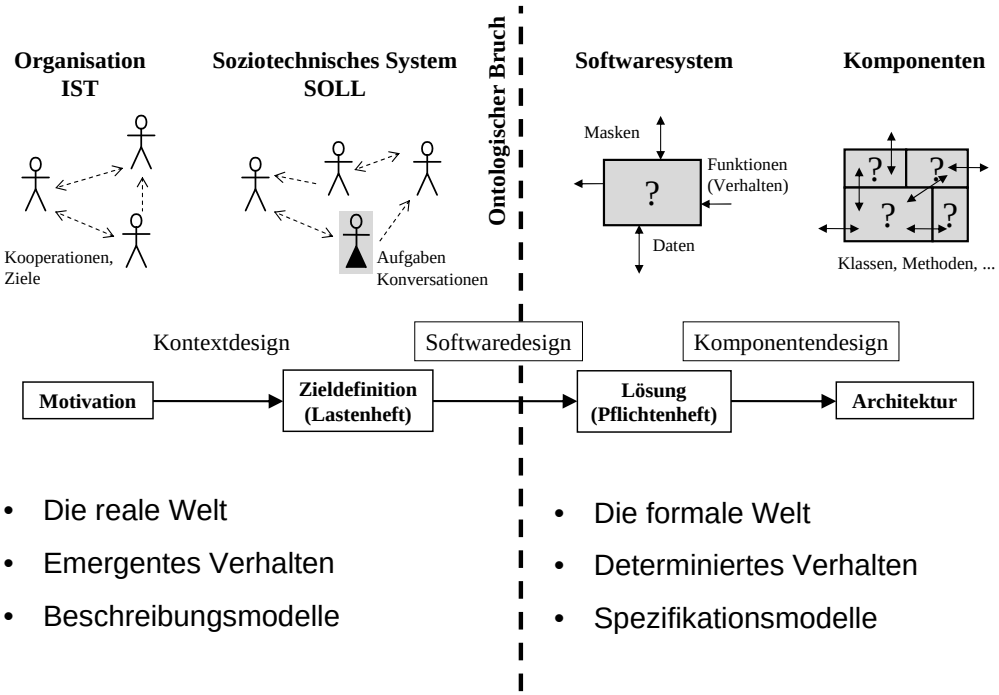


Abbildung 2.4.: Die zwei unterschiedlichen Modellierungswelten

das Problem, d.h. die Menge aller möglichen Lösungen definiert wird, spezifiziert und detailliert das Softwaredesign in Form des Pflichtenhefts bzw. der Softwarespezifikation daraus genau eine bestimmte Lösung, die dann im Komponentendesign weiter konkretisiert wird. In der Praxis finden die Gestaltungsprozesse notwendigerweise ineinander verzahnt bzw. parallel statt. Mit der Diskussion technischer Möglichkeiten ändert sich auch die ins Auge gefasste Rolle des IS und damit potentiell natürlich auch wieder Vorstellungen über mögliche Formen der Arbeitsunterstützung.

KoA$_{L}$A geht von der Prämisse aus, dass die Praxistauglichkeit eines Softwaresystems sein eigentliches Qualititätsmerkmal darstellt. Die Praxistauglichkeit eines IS bestimmt in hohem Maße die Qualität des sozialen Handlungssystems, in das es eingebettet ist, und bestimmt damit maßgeblich die Akzeptanz und damit den Erfolg des Entwicklungsprojekts. Da sich die Qualität eines Handlungssystems – und damit auch die Qualität des eingesetzten IS – gut von der Qualität der Kooperationsnetzwerke bzw. den statischen und dynamischen Interaktionsbeziehungen ableiten lässt, ist die Modellierung dieser Kooperationskontexte wesentlicher Inhalt von KoA$_{L}$A.

Um bei der Entwicklung eines IS zu einer stabilen und umfassenden Anforderungsdefinition zu kommen, muss vom praktischen Handeln selbst ausgegangen werden. Anforderungen an das System sind hierzu im Arbeitskontext zu begreifen, wie auch die Auswirkungen des Systems

auf das praktische Handeln zu antizipieren und in den Gestaltungsprozess mit einzubeziehen sind. Das Problem muss erst in enger Zusammenarbeit mit den Betroffenen entwickelt und spezifiziert werden, bevor es softwaretechnisch analysiert werden kann.

> *"Eines der Hauptprobleme traditioneller Software-Entwicklung liegt darin, dass alle bisher primär an Software-Entwicklungen beteiligten Personengruppen nicht wahrhaben wollen, dass Software-Entwicklung in den meisten Fällen primär Arbeits- und Organisationsentwicklung ist." [Rau92]*

Nach [Jac97] sind die Methoden des Software Engineering traditionell eher auf die formale und homogene Lösungswelt, als auf die im Allgemeinen inhomogene und informale Problemwelt fokussiert. Problemanalyse und -lösung in Form einer softwaretechnischen Spezifikation werden oft in einem Schritt durchgeführt. Der Nachteil dieses Vorgehens ist offensichtlich: Neben suboptimalen Lösungen, die ihren Schwerpunkt eher auf die akzidentellen denn auf die essentiellen Eigenschaften des Softwaresystems legen, werden damit auch Lösungsalternativen wie die Wiederverwendung oder Weiterentwicklung bereits existierender Lösungen potentiell ignoriert.

Jackson konstatiert es als Fehler vieler etablierter sogenannter "Problemanalyse"-Methoden, dass sie nicht das Problem beschreiben, sondern die Lösung des Problems, d.h. bereits die "Maschine" bzw. das Softwaresystem selbst. Das Problem ist nur auf Basis von Konzepten der System-*Umgebung* zu formulieren, nicht über eine Beschreibung des Softwaresystems.

Ein weiterer Grund für die explizite Unterscheidung einer echten Problem-Definitionsphase von der folgenden Lösungs-Konzeption sieht Jackson letztendlich auch in der Reduktion der Komplexität des Software Engineerings allgemein:

> *"Focusing on the problem and deferring consideration of putative solutions is in itself a contribution to mastering complexity in software engineering"* [Jac97]

3. Einordnung von Koala in die Methodenwelt

"Generalized problem-solving methods dont't work, at least not well enough to base a method of requirements-writing on them."

Benjamin L. Kovitz [Kov98]

Existierende Methoden im Bereich des Requirements Engineering, die zur Erhebung, Analyse und Definition von Anforderungen an Softwaresysteme herangezogen werden, lassen sich von ihrem Schwerpunkt bzw. Ursprung her normalerweise einer der drei Gestaltungsebenen "Handlungssystem", "Nutzungssystem" oder "Softwaresystem" zuordnen. Einige der Methoden überspannen dabei mehrere Gestaltungsebenen bzw. können potentiell ganz oder teilweise in mehreren dieser Ebenen eingesetzt werden.

Die konzeptionelle Datenmodellierung beispielsweise ist zwar eher den Methoden bzw. Techniken der Softwaresystemspezifikation zuzurechnen, sie ist jedoch auch aus der Modellierung der Anwendungsdomäne nicht mehr wegzudenken. Petrinetze lassen sich sowohl zur Spezifikation nebenläufiger Regelprozesse als auch zur Modellierung von Geschäftsprozessen einsetzen. Die Methodologie Multiview (siehe Abschnitt 3.3) ist desweiteren ein Beispiel für einen Ansatz, der alle drei Gestaltungsebenen umfasst.

Neben der Gestaltungsebene lassen sich zur Charakterisierung dieser Methoden im Wesentlichen noch zwei weitere grundlegende "Dimensionen" heranziehen: ihre epistemologische Ausrichtung (*analytisch* versus *systemisch*; siehe hierzu Abschnitt 2.4) und ihr Organisationstyp (produkt- versus dialogorientiert; siehe hierzu z.B. auch [Flo92]).

Die epistemologische Ausrichtung gibt an, ob die Methode eher einem mechanistischen und objektivistischen Weltbild verhaftet ist, in dem ein System nur durch seine vollständige Zerlegung und formale Beschreibung zu verstehen und berechenbar zu machen ist (= "analytische" Ausrichtung) oder ob die Methode eine ganzheitlichere systemische Herangehensweise propagiert, in der ein System nach Maßgabe der Systemtheorie eher als "Organismus" begriffen wird, der auf Grund seiner Komplexität nur qualitativ, unvollständig und in der Summe seiner Teile untersucht und verstanden werden kann (= "systemische" Ausrichtung).

Produktorientierte Methoden verstehen die Systementwicklung als einen Herstellungsprozess, der eine Reihe fest definierter Zwischenprodukte kennt (Artefakte wie z.B. Modelle, Diagramme, Spezifikationen). Die Vorgehensweise ist durch Vorgabe einer bestimmten – oft auch linearen – Erstellungsreihenfolge festgelegt und klar strukturiert. Der konkrete Weg zu diesen Produkten ist erst in zweiter Linie relevant, auch wenn hierfür durchaus Techniken und Vorgehensweisen beschrieben werden. Endergebnisse werden oft formal dokumentiert und aus anderen Zwischenergebnissen abgeleitet bzw. durch sie verifiziert. In den objektorientierten Methoden beispielsweise sind *architekturzentrierte* Ansätze auf der Basis von Objektmodellen vorherrschend (siehe z.B. RUP [BR97]). Der Entwicklungsprozess ist durch die vordefinierten Zwischenergebnisse stark strukturiert und darum gut beherrschbar.

Dialogorientierte Methoden verstehen die Systementwicklung hingegen primär als gemeinsamen sozialen Lern-, Gestaltungs- und Abstimmungsprozess. Zwischenergebnisse werden, wenn überhaupt, überwiegend informal bzw. natürlichsprachlich dokumentiert. Schwerpunkt liegt auf der generischen Beschreibung des konkreten Vorgehens. Eine Definition der Arbeitsmittel wie z.B. einzelner Modellierungstechniken erfolgt, wenn überhaupt, nur informell. Ein Erfolg des Methodeneinsatzes ist stark von der methodischen Erfahrung der Entwickler abhängig.

Die produktorientierten Methoden lassen sich desweiteren an Hand ihrer vorherrschenden Paradigmen bzw. Modellierungskonzepten einordnen (Funktionen, Objekte, Ziele, Sprechakte etc.) auf denen die erstellten Artefakte basieren bzw. die in der Erstellung der Artefakte eine ausgezeichnete Rolle spielen.

Die nachfolgende Beschreibung will nun auf Grundlage dieser Kategorien einen knappen Überblick über die Methodenlandschaft geben, um die Positionierung von KoA$_{\text{L}}$A als systemische produktorientierte Methode deutlich zu machen.

Bei der folgenden Auflistung und Einordnung der verschiedenen RE-Methoden ist zu berücksichtigen, dass der Einsatz von RE-Methoden in der realen Entwicklungspraxis in der Regel durchaus Neuland ist. Als erste und einzige RE-Aktivität im Entwicklungsprozess finden sich oft allein die Dokumentation der Anforderungsdefinition selbst in Form eines natürlichsprachlichen Dokuments, das zwar evtl. eine standardisierte Strukturierung aufweist, dabei jedoch nicht auf methodisch abgesicherte Weise erstellt, dokumentiert, strukturiert oder validiert wurde.

Die folgende Beschreibung konzentriert sich hauptsächlich auf die Darstellung und Einordnung von *durchgängigen* und *gestalterisch* wirksamen Methoden bzw. Methodenfamilien. Es soll dabei weniger auf die einzelnen *Techniken* (siehe z.B. [GL93], [Mag97]) wie z.B. das Brainstorming, Paper Prototyping, Szenario Building, Storyboarding, Wizard of Oz, Diary Methods, etc. eingegangen werden, die zwar in der Literatur oft auch als Methoden des Requirements Engineering bezeichnet werden, jedoch nur für die Durchführung einzelner methodischer Schritte Anwendung finden und damit – in unserem Sinne – eigentlich methodenunabhängig sind. Typische Einzelschritte vollwertiger Methoden sind dabei z.B. die Anforderungserhebung, die Durchführung kooperativer Designsitzungen, die Anforderungsdefinition, die Analyse bzw. Validierung der Anforderungen, der Entwurf und die Evaluation von Benutzerschnittstellen etc.

3.1. Gestaltungsebene Softwaresystem

Die verwendeten Konzepte und Methoden (siehe z.B. [Wie98]) dieser Gestaltungsebene entstammen überwiegend der Welt der Programmentwicklung und sind in der Regel den analytischen und produktorientierten Methoden zuzurechnen. Gestaltungsobjekt ist hier alleine das Softwaresystem selbst. Den Auswirkungen des Softwaresystems auf die Arbeitspraxis wird in diesen Methoden im Gegensatz zu KoA$_{\text{L}}$A wenig bis keine Beachtung geschenkt. Viele der in KoA$_{\text{L}}$A eingesetzten Modellierungskonzepte stammen allerdings von Methoden dieses Bereichs ab bzw. lehnen sich an deren Notation (UML) an, so dass sie auch für Entwickler einfach zu verstehen und einzusetzen sind.

Produktorientierte Methoden

Ziel dieser Methoden ist die möglichst formale Definition des zu entwickelnden zukünftigen Softwaresystems. Betrachtet man den Werdegang von Softwareentwicklungsmethoden, so fällt auf, das viele bewährte Modellierungskonzepte ihren Ursprung in Programmierkonzepten besitzen. Konzepte wie Funktion, Objekt, Zustand, Nachricht/Ereignis, Modul, Komponente etc. entstammen in der Regel realen Programmiersprachen. Nachdem sie sukzessive auch in Methoden des Programm- bzw. Architekturentwurfs Verwendung fanden, werden sie nun auch in der Analysephase verwendet, um hier Anforderungen an Softwaresysteme auf abstrakter und in möglichst formaler Art zu spezifizieren.

Produktorientierte analytische Methoden der Gestaltungsebene "Softwaresystem" lassen sich an Hand ihres zentralen Modellierungskonzepts weiter unterscheiden (siehe hierzu z.B. auch [Dav93] oder [UU93]):

Funktion

Methoden basierend auf der Structured Analysis (SA, SADT, MSA, SSADM) stellen die einzelnen Verarbeitungsfunktionen und deren Daten- bzw. Kontrollflüsse in den Vordergrund. Ein Softwaresystem wird als Transformation von Eingabe- in Ausgabedaten gesehen und in Form eines Datenflussdiagramms dargestellt, das zusätzlich zu den Funktionen auch Datenspeicher, Datenquellen und -senken dokumentiert. Erweiterungen dieser Methoden um Echtzeitaspekte wie die Realtime Extensions von Ward Mellor inkorporieren auch Kontrollflüsse und -prozesse (spezifiziert als Zustandsmaschine).

Die Methoden CORE [Mul79] und VOSE [FKN$^+$92] erweitern den Modellierungsbereich etwas in Richtung Nutzungskontext, indem sie die strukturierten Methoden noch um eine Berücksichtigung unterschiedlicher Standpunkte (viewpoints) ergänzen. Der analytische technikorientierte Charakter der Methoden bleibt davon aber unberührt. Der Kontext wird wie das Softwaresystem selbst als Netzwerk von Transformationsfunktionen modelliert.

Zustand

Hauptaugenmerk liegt hier auf dem Verhalten des Systems in Wechselspiel mit seiner Umgebung. Die Steuerungsfunktion des Systems steht im Vordergrund und wird im Allgemeinen an Hand virtueller zustandsbasierter Maschinen spezifiziert, die normalerweise auch nebenläufige Prozesse unterstützen (siehe z.B. State Charts [Har87], SREM [Alf85], PAISLey [Zav82], Petri Netze [Pet62] etc.). Da diese Ansätze vor allem auf dem Gebiet eingebetteter Systeme bzw. automatischer Steuerungen Anwendung finden, berücksichtigen sie im Allgemeinen auch die Modellierung von Echtzeitbedingungen, die Prozesssynchronisation bzw. -kommunikation, die Erkennung von Deadlocks etc. Hier entwickelten sich auch rein formale Ansätze wie die modellbasierten Spezifikationssprachen Z [Spi88]oder VDM [Jon90] und spezialisierte Prozessalgebren wie z.B. CSP [Hoa85] oder CCS [Mil89].

Die formale Verhaltensspezifikation scheint vor allem im technischen Kontext möglich bzw. sogar nötig und erfolgversprechend zu sein. Die Modellierung menschlicher Akteure als "Zu-

standsautomaten" würde hingegen ihrer sozial bestimmten Arbeitssituationen nicht gerecht werden.

Für die Analyse interaktiver Systeme erweitert aber beispielsweise die Viewpoint-basierte Methode VORD [KS96] den Modellierungsbereich, indem sie neben dem System selbst auch die "externen Anforderungen" in Form direkter und indirekter Betroffener und deren unterschiedliche spezifische Anforderungen an das System modelliert. Unterschieden wird dabei zwischen funktionalen (services), nichtfunktionalen (constraints) und verhaltensbezogenen Anforderungen (event scenarios in Form von state charts).

Objekt

In den objektorientierten Analysemethoden wird das Softwaresystem auf Grundlage von konzeptuellen Objekten, ihren Eigenschaften und ihren statischen bzw. dynamischen Beziehungen modelliert. Objekte spiegeln dabei in der Regel Konzepte der realen Anwendungswelt wieder. Zwar werden damit zur Strukturierung des Systems Außenweltkonzepte verwendet, letztenendes wird damit aber trotzdem oft bereits das System selbst entworfen, ohne vorher Eigenschaften oder Anforderungen der Außenwelt zu beschreiben. Bei Jacobson findet sich eine Definition, die diesen Sachverhalt sehr deutlich macht:

> "*The purpose of object-oriented analysis, as with all other analysis, is to obtain an understanding of the application: an understanding depending only on the system's functional requirements*" ([Jac94b], S.76).[1]

Neuere objektorientierte Methoden wie z.B. RUP oder Catalysis [DW99] kennen zwar mittlerweile auch Modelle zur Analyse und Darstellung der Anwendungsumgebung ("business modelling"). Diese Modelle sind jedoch optional und dienen in der Regel eher dem Verständnis bzw. der Einarbeitung in den Anwendungskontext. Sie werden nicht zur Anforderungsentwicklung, -definition oder -gestaltung eingesetzt.

Da die gleichen Konzepte und Techniken zum Einsatz kommen wie bei der Spezifikation des technischen Softwaresystems, ist ihre Ausdrucksmächtigkeit darüberhinaus auch weitgehend eingeschränkt und zu starr für eine weitergehende *offene* Modellierung der komplexen soziotechnischen Arbeitswelt.

Laut Avison sind Methoden, die eine zu starke technisch rationale Ausrichtung besitzen, eine ungeeignete Grundlage für die Entwicklung von Informationssystemen [AWHVW98]. Hirschheim bescheinigt darum diesen erweiterten analytischen Methoden – seien es funktions- oder objektorientierte Entwicklungsansätze – das gleiche ungenügende philosophische Fundament: ein funktionalistisches Paradigma des Objektivismus und der Gesellschaftsordnung [HKL95].

Der Einsatz von wissenschaftlichen reduktionistischen Ansätzen zur Modellierung und Analyse von Organisationen ist trotzdem weit verbreitet. Einen Überblick über die in der Literatur geäußerte fundamentale Kritik an dieser Entwicklung findet sich in [Fit94].

[1] Diese Sichtweise schlägt sich auch in seiner Weiterentwicklung wieder, dem Rational Unified Process (RUP). Hier beginnt die Anforderungsdefinition mit dem Aufschreiben von Use-Cases, die typische Systemnutzungsfälle dokumentieren, und damit auf einer Ebene, an der die konkrete Systemschnittstelle bereits festgelegt ist. Das ist jedoch bereits Teil der Spezifikation des Softwaresystems! Eine realisierungsunabhängige Beschreibung der Problemstellung findet hier erst gar nicht statt.

Das OO-Paradigma ist für die Gestaltungsebene des Nutzungs- oder Handlungssystems nur sehr eingeschränkt geeignet. Aus Anwendersicht sind Daten und Operationen darauf gleichwertige Modellierungskonzepte. Dies kann man z.B. am Erfolg der Geschäftsprozessmodellierung (siehe Abschnitt 3.3) bzw. der aufgabenorientierten Methoden des HCI (siehe Abschnitt 3.2) ablesen, die der Modellierung der Aktivitäten vorrangige Bedeutung beimessen und sie von der nachgeordneten Informationsmodellierung trennen.

Eine Kopplung von Daten und Operationen ähnlich zu einem abstrakten Datentyp erscheint erst aus technischer Entwurfs- bzw Realisierungssicht sinnvoll, wenn es um Qualitäten wie Modularisierung, Vererbung, Wiederverwendbarkeit und Wartbarkeit geht. Daher verwenden auch einige OO-Methoden bereits eigene Modellierungskonzepte wie z.B. "Systemoperationen" (Fusion [CAB+93], Larman [Lar98] etc.) zur Spezifikation funktionaler Leistungen und modellieren den Systemzustandsraum (die Domäne bzw. die Daten) getrennt davon in einem eigenen konzeptionellen Datenmodell (Klassenmodell ohne Methoden). Eine direkte Kopplung von Attributen und Operationen bzw. Verantwortlichkeiten findet sich hingegen z.B. bei der Coad/Yourdon-Methode [CY90]. Dies führt dort in der Analyse oft zu verqueren und unnatürlichen Modellierungen wie z.B. zu einem Buch, das sich selber ausleihen kann und seine Rückgabe vermerkt (zu bestaunen in [Car94], S. 170).[2]

Objektorientierte Analysemethoden tendieren dazu, in der Modellierung nicht klar zwischen den Objekten des Problembereichs – den physikalischen Objekten des Anwendungsbereichs, die wichtig für das Problemverständnis sind – und den internen Objekten des Softwaresystems – den für die Realisierung wichtigen Informationsobjekten – zu unterscheiden (siehe z.B. [Dav93], S.59). Sie unterlassen diese Unterscheidung zugunsten des einfachen Übergangs von der Spezifikation zum Systementwurf. Dabei vernachlässigen sie den Umstand, dass diese Vereinfachung in der Praxis im Gegenzug dazu führt, dass in der Analysephase oft bereits der Systementwurf durchgeführt wird. Die Vermeidung dieser verfrühten Einengung des Lösungsraums ist jedoch laut Davis eines der größten Probleme der Analysephase ([Dav93], S.55).

Dialogorientierte Methoden

Neben den etablierten produktorientierten Methoden lassen sich auch eher dialogorientierte Methoden identifizieren.

Hierzu gehören neben iterativen Ansätzen wie dem evolutionären oder explorativen Prototyping als neuester Vertreter das "Extreme Programming" (XP [Bec00]). Andere RAD-Ansätze ("rapid application development") sind z.B. DSDM [Sta97] oder SOMA [Gra98].

Diese Ansätze beschreiben Vorgehensweisen, die eine der Realisierung vorgeschaltete Phase der Anforderungsdefinition weitgehend vermeiden bzw. reduzieren und den zyklischen Entwicklungsprozess in den Vordergrund stellen. Die Verifikation des Endergebnisses aufgrund der Anforderungsspezifikation wird hier ersetzt durch die häufige Validierung bzw. Abstimmung der einzelnen Produktversionen durch den Anwender. Dies setzt neben einem kleinen

[2] Eine sehr viel verständlichere Modellierung würde man hier erhalten, wenn man sich auf die abstrakte Darstellung des Handlungssystem beschränken würde und die Ausleihe als Aktivität eines Lesers und die Rückgabebearbeitung als Aufgabe der Bücherei-Angestellten modellieren würde. Coad/Yourdon beginnt jedoch bereits mit der Modellierung des Softwaresystems.

Realisierungsteam natürlich auch die entsprechende "Verfügbarkeit" konkreter bzw. charakteristischer Anwender voraus.

3.2. Gestaltungsebene Nutzungssystem

Methoden dieser Ebene beziehen die zukünftigen Nutzer in den auszugestaltenden Systembereich mit ein. Die Methoden können dabei anhand ihrer Reichweite unterschieden werden.

Die *oberflächenorientierten* Methoden zielen vor allem auf die Qualität der Benutzerschnittstelle des Systems ab, die *kontextorientierten* Methoden hingegen umfassender auf die Nutzbarkeit und Passgenauigkeit des Systems im Rahmen der täglichen Arbeitspraxis seiner Nutzer. KoA$_{\text{L}}$A gehört zu diesen kontextorientierten Methoden der Gestaltung von Nutzungssystemen.

Die Einbettung von Software in die praktische Arbeitstätigkeit hat in den letzten Jahren eine interdisziplinäre Kooperation von Sozialwissenschaftlern und Informatikern stimuliert, die besonders im Bereich der "computer supported cooperative work" (CSCW) oder "human computer interaction" (HCI) augenfällig geworden ist. Erkenntnisse aus der activity theory, ethnografische Methoden, diskursive und systemische Vorgehensweisen tragen zum Verständnis menschlicher Kommunikation, kooperativer Zusammenarbeit und der Einflüsse bzw. Folgen von Technologieeinsatz und Arbeitsorganisationen bei [FDK].

In den aktuellen (Weiter-)Entwicklungen der Methoden in diesem Bereich meint [Ale99] sogar eine allgemeine Tendenz hin zum kooperativen Requirements Engineering ausmachen zu können, da die Nutzer und Betroffenen nun natürlicherweise auch oft aktiv in den Gestaltungsprozess mit einbezogen werden.

Oberflächenorientiert

Methoden aus dem Bereich HCI konzentrieren sich auf die Gestaltung der direkten Benutzerschnittstelle des Systems. Beim Dialogdesign stehen die Welt des einzelnen Benutzers in Form mentaler Modelle [Dut94] und ergonomische Fragestellungen im Vordergrund. Die Entwicklung der Benutzerschnittstelle wird dabei geleitet vom Verständnis des direkten Nutzungskontexts – der Nutzer, ihrer Aufgaben und den real zu beobachtenden Tätigkeitsfolgen. Während auf Softwaresystemebene noch die Effizienz und Korrektheit der Systemleistung qualitätsbestimmend sind ("system quality"), sind auf der Ebene des Nutzungssystems nun die Nutzerzufriedenheit bzw. Nutzerproduktivität ausschlaggebend ("quality in use").

Schwerpunkt von Ansätzen im Bereich HCI sind die Bereitstellung einzelner Techniken für die Analyse, Studie, Evaluierung und schrittweise Verbesserung von Nutzungssituationen. Dieser Unterbereich des HCI, der unter dem Begriff "Usability Engineering" [Nie94] subsumiert wird, stellt ein umfangreiches Instrumentarium zur Verfügung, das zur Durchführung von Studien (z.B. Contextual Inquiry, ethnografische Studien, Fragebögen, Journaled Sessions etc.), Inspektionen (z.B. Heuristic Evaluation, Cognitive Walk Throughs, Mock Ups, Feature Inspection etc.) und Tests (z.B. Thinking Aloud Protocol, Co-Discovery Method, Performance Measurement etc.) verwendet werden kann, um Anforderungen an interaktive Systeme zu

erheben oder die Gebrauchstauglichkeit von realen oder prototypischen Softwaresystemen zu studieren, zu verbessern bzw. sicherzustellen.

Viele dieser Techniken des Usability Engineering setzen dabei die Existenz eines – evtl. zu verbessernden – Softwaresystems bzw. einer entprechend bewertbaren Benutzerschnittstelle voraus. Schwerpunkt ist hier nicht die kreative Neuschöpfung eines Software- bzw. passenden Nutzungssystems, sondern deren Validierung bzw. Optimierung.

Dies schließt natürlich nicht aus, dass auch solche Techniken erfolgreich zur Entwicklung bzw. beim Neuentwurf soziotechnischer Handlungs- bzw. Nutzungssysteme eingesetzt werden können. Deutlich kommt dies z.B. in der dialogorientierten Methode des PARTICIPATORY DESIGN [NS92, GK91] zum Tragen, in der der Nutzer mit dem Entwickler zusammen eine Idee für das neue Softwaresystem entwirft und sie mittels Mock Ups und Prototypen aktiv im konkreten Arbeitskontext kooperativ entwickelt. Da die Möglichkeit der rein intellektuellen Erfassung und Gestaltung von Anforderungen generell angezweifelt wird, findet in diesem Ansatz keine explizite vorgeschaltete Analyse-, Konstruktions- oder Spezifikationsphase statt. Die Idee zum Produkt wird vielmehr von Nutzer und Entwickler zusammen in der konkreten Nutzungssituation erschaffen, validiert und zyklisch fortentwickelt.

Produktorientierte Methoden, die bei der Neuentwicklung von Benutzerschnittstellen angewendet werden, verwenden zur Anforderungserhebung meist aufgabenbasierte Ansätze ("task oriented approaches"), um Wissen über die Arbeitswelt der Benutzer zu sammeln und zu strukturieren. Die Systemschnittstelle wird so gestaltet, dass sie die Durchführung der Anwender-"Tasks", d.h. einzelner wohldefinierter elementarer Arbeitsschritte der Systembenutzer, optimal unterstützt. Dabei wird die elementare Aufgabenstruktur, auf die der Enwurf der Nutzerschnittstelle ausgerichtet wird, normalerweise als vorgegeben bzw. als vollständig analysierbar angenommen. In KoA$_{\text{L}}$A hingegen ist diese Aufgabenstruktur mit Gegenstand der Gestaltung, die neben den atomaren Tätigkeiten auch die Entwicklung abstrakterer Aufgabenbereiche beinhaltet.

Die "Task"-orientierten Methoden sind in der Regel den analytischen und produktorientierten Ansätzen zuzurechnen. Zu ihnen gehören HTA ("Hierarchical Task Analysis"; [CSA89]), TKS ("Task Knowledge Structures"; [CJS92], "Object-oriented Specification of User Interfaces" nach Jaaksi[Jaa95], die "Systemanalyse menschlichen Handelns" nach [DR97] oder aber auch die verschiedenen um Use-Cases erweiterten objektorientierten Methoden wie RUP, Fusion etc.

Die meisten in der Literatur beschriebenen Methoden zum Entwurf von Benutzerschnittstellen sind jedoch dialogorientiert (siehe z.B. [May99], [LR] oder [HH93]). Sie beschreiben, im Allgemeinen eher informell, bewährte Vorgehensweisen, wie beim Entwurf bzw. der Entwicklung von Benutzeroberflächen allgemein vorzugehen ist und greifen hierbei in der Regel auf eine spezifische Auswahl an Techniken des Usability-Engineering zurück.

Kontextorientiert

Eine Reihe von Methoden erweitern die Gestaltungsperspektive über die direkte Benutzerschnittstelle einzelner Nutzer hinaus zur Ergonomie des Softwaresystems im Kontext der globalen Arbeitsprozesse von Systemanwendern und aller anderen direkt und indirekt be-

troffenen Akteure. Diese Ausweitung des Gestaltungsbereichs interaktiver Softwaresysteme wird im Allgemeinen auch mit dem Schlagwort "User Centred Design" bezeichnet [ND86] und hat hier ihren Niederschlag in der Entwicklung zahlreicher Techniken gefunden (siehe hierzu [CL99]).

Neben diesen einzelnen Techniken haben sich eine Reihe vollständigerer Ansätze entwickelt, die in überwiegender Zahl sogenannte "Workshop-Methoden" darstellen. Wesentliche Gemeinsamkeit dieser ***dialogorientierten*** systemischen Methoden ist die kooperative und iterative Gestaltungsarbeit in enger Zusammenarbeit mit möglichst vielen direkten Nutzern oder anderen betroffenen Akteuren der zukünftigen Systemumgebung. Schwerpunkt ist die Festlegung und Beschreibung der Organisation der Workshops bzw. ihre Vorbereitung, Durchführung oder Abfolge. Eine zentrale Rolle kommt in diesen Methoden dem *facilitator* bzw. Moderator zu, der für das Aufdecken, Diagnostizieren und Lösen von Kommunikationsproblemen und anderen Konflikten der Gruppe verantwortlich ist.

In den dabei eingesetzten Techniken finden sich auch viele der im Usability Engineering entwickelten Techniken wieder. Schwerpunkt der Methoden liegt auf der strukturierten partizipativen Anforderungserhebung und gemeinsamen Konfliktlösung, weniger auf der Untersuchung von Umgestaltungsoptionen im Arbeitsumfeld. Viele dieser Methoden entwickelten sich aus dem Umfeld der Produktentwicklung heraus, die aufgrund der großen Zahl möglicher Benutzer des Softwaresystems der Analyse ihrer sozialen, organisatorischen und kognitiven Bedürfnisse eine mindestens ebenso große Bedeutung zumessen wie den rein technischen Entwurfsaufgaben.

Ein Beispiel für eine solche Methoden ist USTM ("User Skills and Task Match"[MFKH90, Mac97]), die wie KoA$_{\text{L}}$A auch in der Erstellung der Anforderungsspezifikation aus Kundensicht eine eigenständige Entwicklungsaufgabe sieht, die komplementär zur Aufgabe der formalen Spezifikation des Softwaresystems aus Entwicklersicht ist.

USTM sieht drei Workshops vor, die jeweils zwei bis drei Tage lang sein können. Im ersten wird die Projektmotivation in Form einer gemeinsamen Vision des Entwicklungsergebnisses entwickelt (Describing a Product Opportunity, DPO). Im zweiten Workshop (Identifying a High Value Solution, HVS) werden aus einer viewpoint-orientierten Analyse heraus die Bedürfnisse der verschiedenen beteiligten oder betroffenen Akteure dokumentiert, bevor dann im letzten Workshop (Delivering the Business Solution, DBS) in einem gemeinsamen Entscheidungsfindungsprozess und nach einer Validierung der Anforderungen durch Umfragen und Prototyping ein vorläufiger Systementwurf durchgeführt wird.

Weitere Methoden zur kooperativen Erarbeitung von Anforderungsdefinitionen in Workshop-Settings sind JAD [WS95] und IBRA (Inquiry-Based Requirements Analysis, [PTA94]).

Neben diesen eher dialogorientierten Vorgehensweisen bei der kontextorientierten Gestaltung von Nutzungssystemen haben sich jedoch auch ***produktorientierte*** analytische Methoden etabliert.

Die Einbeziehung der Nutzungssituation hatte in diesen Ansätzen die Entwicklung neuer Modellierungskonzepte zur Folge. Zusätzlich zu den softwaresystembezogenen halten nun auch nutzungsorientierte Entwurfsrepräsentationen Einzug in den Gestaltungsprozess. Eine besonders erfolgreiche Rolle spielen hierbei die Beschreibung realer Anwendertätigkeiten in Form sogenannter "Szenarien" oder direkter Systemnutzungssequenzen in Form von "use cases" (zu den

vielen unterschiedlichen Anwendungsformen dieses Modellierungskonzepts siehe z.B. [Car95]).

Ein Beispiel für einen solchen eher traditionellen Ansatz ist VOLERE [Rob97], der zur Erhebung von Anforderungen eine genaue Untersuchung des zukünftigen Einsatzgebietes vornimmt (apprenticing, video, use case szenarios etc.) und die praktische Arbeitstätigkeit studiert, bevor dann die funktionalen Anforderungen in Form von Use-Case-Beschreibungen dokumentiert werden, die Teile dieser Arbeit automatisieren sollen. VOLERE unterscheidet dabei deutlich zwischen der im ersten Schritt zu erstellenden "Requirements Specification" als Ergebnis des "Requirements Process" und der im Folgenden zu entwickelnden "Analysis Specification" als Ergebnis der "Systems Analysis" [SR99]. Das Ergebnis wird vom Kunden z.B. unter Einsatz von Prototyping oder textuellen Szenarienbeschreibungen validiert und in der Folge iterativ überarbeitet.

Die use-case-orientierten Ansätze modellieren in Form der direkten Systemnutzungssequenzen in der Regel nur einen sehr kleinen Bereich des Anwendungskontexts, und dies normalerweise auch in recht informeller Weise z.B. in Form strukturierter natürlichsprachlicher Beschreibungen. Eine globalere Einordnung dieser Tätigkeiten in übergeordnete Arbeitsvorgänge oder die Einbeziehung intentionaler Aspekte der Arbeitssituation, um z.B. Optionen für die Umgestaltung einem möglichen Diskurs zugänglich machen zu können, unterbleibt dabei überwiegend.

Neuere, umfassendere und formalere Ansätze wie z.B CONTEXTUAL DESIGN[BH98] beziehen in die Modellierung auch diesen übergreifenden Kontext mit ein. CONTEXTUAL DESIGN beschreibt ein dialogorientiertes Vorgehen zum Design von Softwaresystemen basierend auf einer genauen Analyse der Arbeitszusammenhänge. Über Einzelinterviews ("contextual inquiry") informieren sich die Entwickler über die reale Alltagspraxis und beschreiben sie in einer Reihe von Modellen, die die allgemeine Arbeitssituation verallgemeinern. Zur Darstellung der Arbeit aus fünf unterschiedlichen Perspektiven werden die Kommunikationsbeziehungen ("flow model"), Aktivitätsmuster ("sequence model"), Arbeitsmittel ("artifact model"), der kulturelle Kontext ("cultural model") und der physische Arbeitsplatz ("physical model") in jeweils eigenen Modellen beschrieben. Diese Modelle werden benutzt, um die anschließende partizipative Erfindung bzw. Definition eines erfolgreichen Softwareprodukts zu fördern.

Schwerpunkt des Ansatzes liegt dabei in der Beschreibung eines linearen Entwicklungsprozesses, weniger in der Modellierung (die weitgehend informell durchgeführt wird) oder der iterativen Gestaltung bzw. Anpassung des Nutzungssystems, die nicht nur die Ausgestaltung des Softwaresystem, sondern auch seiner soziotechnischen Umgebung zur Folge hat. Zur weiteren Diskussion von KoA$_{\text{L}}$A im Vergleich zu dieser Methode, die ebenso wie KoA$_{\text{L}}$A zu den produktorientierten systemischen Methoden gehört, siehe Abschnitt 6.3.

3.3. Gestaltungsebene Handlungssystem

Methoden dieser Ebene beziehen nun nicht nur die direkten Nutzer oder Betroffenen des Softwaresystems in die Gestaltung mit ein, sondern das gesamte soziotechnische System, dessen einzelne Bestandteile neben weiteren technischen und menschlichen Akteuren das zukünftige Softwaresystem und seine Nutzer sind. Die Gestaltungsperspektive weitet sich auf globalere Einheiten arbeitsteiliger Handlungssysteme aus: von einzelnen Arbeitsgruppen oder Abteilungen bis hin zu ganzen Unternehmen. Die Tätigkeitszusammenhänge in diesem alle mensch-

lichen oder technischen Akteure umfassenden soziotechnischen Systems werden modelliert, umgestaltet und untersucht, um dadurch implizit die optimale Gestalt eines technischen Akteurs – des neuen Softwaresystems – zu gewinnen oder diese dann in der Folge explizit ableiten bzw. konstruieren zu können.

Das Gestaltungsobjekt dieser Methoden ist damit ein sehr viel weitreichenderes als das von KoA$_{L}$A, das sich auf die direkte Nutzungssituation und Arbeitsumgebung der Softwaresystem-Anwender beschränkt. Einige der hier dargestellten Methoden können daher – zumindest teilweise – als komplementär zu KoA$_{L}$A angesehen werden (z.B. SSM, ORDIT oder BPR) genau wie auch KoA$_{L}$A als Komplement bereits etablierter Methoden der Softwaresystemgestaltung angesehen werden kann. Die drei Gestaltungsebenen Handlungs-, Nutzungs- und Softwaresystem unterscheiden sich sowohl von ihren Inhalten, als auch von den aktiven Gestaltungsakteuren bzw. vorherrschenden Gestaltungszielen so weit, dass sich für sie auch der Einsatz unterschiedlicher Methoden rechtfertigen lässt.

Die verschiedenen etablierten Methoden werden nun im Folgenden auf Grundlage ihrer zentralen Modellierungselemente in unterschiedliche Abschnitte gegliedert. Sind Methoden dabei nicht eindeutig einem zentralen Modellkonzept zuzuordnen, erfolgt die Zuordnung nach Maßgabe ihrer historischen Entstehungsgeschichte oder inhaltlicher Ähnlichkeit zu anderen Methoden des jeweiligen Abschnitts.

Funktion

Methoden, die die Informationsverarbeitung bzw. den Informationsfluss in den Vordergrund stellen sind z.B. die diversen analytischen und produktorientierten Kommunikationsanalysemethoden aus dem Umfeld der Büroautomatisierung (KSA, ADV/Orga, COMplan, CORAN, EPOS, FIBA, KOM-System, KSS, MOSAIK, PLAKOM, POKAL, SOPHO-PLAN, VERIKS etc., ein Überblick findet sich z.B. in [ST89]). Hier wird die Arbeitswelt als Informationsverarbeitungsprozess modelliert, werden Engpässe, Deadlocks oder Medienbrüche in den Kommunikations- bzw. Verarbeitungsketten identifiziert, das Datenvolumen, der Zeitbedarf etc. bestimmt, um z.B. den Durchsatz des Gesamtsystems durch entsprechende organisatorische Maßnahmen zu optimieren bzw. durch Einführung eines neuen Softwaresystems zu verbessern.

Die Sicht auf den Systemkontext ist daher eher technisch analytisch. Eine weiterführende Berücksichtigung sozialer Dimensionen bzw. qualitativer Eigenschaften der Arbeitssituation unterbleibt weitgehend. Ähnlich formale Ansätze zur Modellierung von Büroarbeit finden sich auch in der Office Procedure Specification Language (OPSL von [Zis77]) und in Information Control Net (ICN von [Ell79]). Vorherrschender Formalismus zur Beschreibung der Arbeitsprozesse ist das Petrinetz (siehe hierzu auch die Methoden OPAL, SOS, SCOOP etc. [ALL88]).

Vollständigere analytische Methoden, die nicht nur eine lokale Optimierung von Organisationseinheiten betreiben, sondern den globalen Top/Down-Entwurf ganzer Organisationen auf Grundlage einer Modellierung und Optimierung der informationsverarbeitenden Prozesse und ihrer Datenflüsse betreiben, findet sich z.B. im Information Engineering [Mar90] oder SDM [TLH^{+}88]. Zusätzlich zu formalen Modellen für die Darstellung von Kontroll- und Datenflüssen kennen diese Methoden auch weiterreichende infologische und organisatorische Modelle

zur Definition von Informations- und Organisationsstrukturen.

Umfassende Ansätze wie die Office Analysis Methodology (OAM; [SSK+84]) oder Office Support Analysis and Design (OSSAD [BBC+88]) modellieren zusätzlich noch weitere Aspekte der Aufbauorganisation (Leute, Rollen, Verantwortlichkeiten, Weisungsbefugnisse) und integrieren in die Modellierung der Ablauforganisation neben intentionalen Beschreibung von Zielen ("business goals") und sonstigen Randbedingungen auch eine Untersuchung von Ausnahmen bzw. Alternativen der Verarbeitung.

Gemeinsam ist all diesen analytischen und produktorientierten Methoden ein objektivistischer Blickwinkel, der eine Organisation aufgrund ihrer informationsverarbeitenden Funktionen begreift. Die technisch formale Modellierung ihrer Verarbeitungsprozeduren soll die fundierte Analyse und anschließende Optimierung bzw. Umgestaltung der Organisation ermöglichen – mit oder aber auch ohne Einsatz neuentwickelter technischer Systeme.

In KoA$_L$A dient die Kontextmodellierung hingegen nicht der abgesicherten Ableitung einer optimalen Systemgestalt, sondern der Förderung einer partizipativen Gestaltungsarbeit, die letztendlich von allen Beteiligten des Gestaltungsprozesses zu verantworten ist. KoA$_L$As Modelle sind Gestaltungsmedien. Sie sollen kontextreiche systemische Beschreibungen der zukünftigen Systemumgebung sein, keine vollständigen und analytischen Spezifikationen, die zur formalen Ableitung, Transformation oder Verifikation von Subsystemen benutzt werden können. Eine solche analytische Modellierung kann aufgrund der Komplexität sozialer Systeme immer nur Stückwerk bleiben. Selbst die Modellierungen von Verarbeitungsprozeduren stellen vereinfachende planbasierte Darstellungen dar, die sich niemals vollständig mit in der Realität durchgeführten Aktivitätssequenzen decken werden, da diese immer "situiert" sind (siehe [Suc87]), d.h. situationsbezogen durch subjektive Urteile und Entscheidungen beeinflusst werden, die den weiteren Fortgang der Handlung bestimmen.

Prozess

Die Orientierung an den beobachtbaren Bearbeitungsmustern bzw. der "Ablauforganisation" ist vielen Methoden zu eigen, die aus dem Gebiet der Betriebswirtschaft bzw. der Wirtschaftsinformatik stammen.

Die Wirtschaftsinformatik beschäftigt sich mit Methoden der Entwicklung von soziotechnischen Systemen nach Maßgabe von Unternehmenszielen. Im Vordergrund stehen damit systematische Verfahren für die objektivistische Konstruktion bzw. Optimierung soziotechnischer Systeme. Kommunikation wird in den entwickelten Methoden überwiegend aus Sicht der Datenübermittlung thematisiert und als Austausch von Nachrichten zwischen Menschen und/oder Maschinen gesehen, die dem Zweck der Informationsübermittlung dienen (siehe z.B. [Hei99]).

Ein Informationssystem wird definiert als: "Ein Mensch/Aufgabe/Technik-System zum Beschaffen, Herstellen, Bevorraten und Verwenden von Information, kurz: ein System zur Informationsproduktion und Kommunikation für die Deckung von Informationsnachfrage" [Hei99]. Informations- und Kommunikationstechnologie wird dabei als Mittel angesehen, die Produktion und Übertragung von Information zu unterstützen und den Produktions- und Kommunikationsprozess zu verstärken.

Diese Gestaltungsperspektive ist sehr reduktiv und schlägt sich dementsprechend auch in Methoden nieder, die die Rolle eines betrieblichen Informationssystems auf seine informationsverarbeitenden, -speichernden und -transportierenden Qualitäten reduzieren. Sie lassen die systemischen Interaktions- und Wirkungsbeziehungen zwischen menschlichen und technischen Akteuren der Anwendungsumgebung und ihrer Arbeitspraxis außer Acht. Ein IS unterstützt und begleitet menschliche Aktivität nicht nur, sondern es verändert sie wesentlich.

Methoden der Wirtschaftinformatik wie ARIS [Sch99a], PROMET [OBG+95] oder SOM [FS90] haben die Entwicklung ganzer Unternehmen im Hinblick auf ihre Informations- und Kommunikationsprozesse und eingesetzte Technologien zum Ziel [Sin99]. Sie wenden dabei für die Modellierung und Konstruktion der Aufbau- und Ablauforganisation in der Regel dieselben analytischen Modelle und Techniken ein, wie sie sie für die Spezifikation bzw. Konstruktion der technischen Softwaresysteme vorsehen.

Vorherrschendes Konzept in der Modellierung von Unternehmen sind dabei sogenannte "Geschäftsprozesse" (in Aris auch "Ereignisprozessketten" genannt), die den innerbetrieblichen Weg der Leistungserbringung aller Dienste darstellen, die das Unternehmen aus externer Kundensicht erbringt. Ein Prozess wird dabei als eine beliebig verfeinerbare Kette von Einzelaktivitäten modelliert, die von menschlichen oder technischen Akteuren bzw. Rollenvertretern durchgeführt werden und im Allgemeinen bestimmte Informationsobjekte verarbeiten, d.h. als "Ein/Ausgabe-Daten" besitzen. Der Übergang von einer Aktivität zur anderen ist fest definiert, regelbasiert (wenn/dann) oder "willkürlich" (entweder/oder, freier Wille). Prozesse beschreiben fest definierte bzw. vorgegebene Bearbeitungsreihenfolgen zur Erbringung der verschiedenen Leistungen des Unternehmens.

Ziel der Modellierung ist die Untersuchung und Verbesserung der Unternehmensorganisation (Stichwort "Business Process Reengineering"), eine Stärkung der Wettbewerbsfähigkeit durch Flexibilisierung und Kundenorientierung, und letztendlich die Unterstützung und Überwachung der praktischen Arbeitstätigkeit durch Workflow Management Systeme, die sowohl eine Steuerung der Abläufe erlauben, als auch Auskunft über den Bearbeitungszustand von bestimmten Prozessinstanzen geben können.[3]

In Vorgehensweisen zur Entwicklung von Softwaresystemen wird die Geschäftsprozessmodellierung auch als Technik für den "fachlichen" Entwurf bzw. Analyse des soziotechnischen Systems eingesetzt, um dann anschließend den "DV-technischen" Entwurf des eigentlichen Softwaresystems zu entwickeln bzw. abzuleiten (siehe z.B. [Win96] oder [Sch99a]). Hier hat sich diese Modellierung auch als Weg zu einem gemeinsamen Verständnis und als Basis für Reflexionen bewährt. Sie hilft Ziele und Verantwortlichkeiten zu klären und Mehrdeutigkeiten und Missverständnisse der Problemstellung aufzudecken, auch wenn diese Modellierung nicht ausreicht, um ein wirklich umfassendes Verständnis des Handlungssystems zu gewinnen:

"*Work is not encountered as a set of discrete tasks involving chunks of data and*

[3] Während früher das Hauptaugenmerk der Büroorganisation auf Effizienz und Kontrolle gerichtet war (Konzentration auf interne Bedürfnisse wie z.B. der Optimierung interner Herstellungsprozesse, Einführung von Kontrollmechanismen, Gestaltung der Organisationsstruktur), ist sie heute auf Innovation, Flexibilität, Qualität und vor allem Kundenzufriedenheit gerichtet (Konzentration auf die Befriedigung externer Kundenbedürfnisse; also auf die eigentlichen Ziele des Unternehmens). Oft geht damit die Zuordnung von Verantwortlichkeiten für solch einen Prozess auf einzelne Personen oder Arbeitsgruppen über, die sich nun im größeren Rahmen intern selbst organisieren/kontrollieren können/müssen (Demokratisierung; Kontrolle bleibt beim Ausführenden und wird nicht nach extern – wie z.B. das Management – gegeben).

> *distinct procedures, but rather as a free flowing gestalt contexture, with activities overlapping, and moving through foreground and background in response to (and constituting) the situation as it develops*" [PS93]

Wie in den Ansätzen der Büroautomatisierung wird die Arbeitswelt in diesen Ansätzen als ein analytisch und formal zu beschreibendes technisches System aufgefasst, das bestimmte "Programme" ausführt. Ähnlich zu den Methoden der Büroautomatisierung integrieren die Methoden der Geschäftsprozessanalyse bzw. des Business Process Reengineering jedoch auch intentionale Aspekte wie Ziele und sonstige Randbedingungen. Geschäftsprozesse werden auf Geschäftsziele des Unternehmens ausgerichtet, die wiederum beispielsweise durch eine Analyse der Kundenbeziehungen gewonnen werden; die Analyse von Zielhierarchien und realen Szenarien wird zur Analyse und Optimierung von Prozessen angewendet etc.(siehe hierzu z.B. [AMP94]).

Interaktion

Ansätze, die sich aus der "Speech act theory" [WF87] begründen, konzentrieren sich auf die in der Organisation stattfindende Kommunikation. Die einzelnen Interaktionen werden auf Basis sogenannter "Speech-Acts" bzw. Sprechakte modelliert, die neben dem Aspekt der reinen Datenübermittlung auch noch den illokutionären Funktions-Aspekt der Interaktion (die Intention des Sprechers) dokumentieren. Interaktionen werden nicht nur zur Übermittlung von Informationen durchgeführt, sondern kreieren oder verändern ein soziales Netzwerk gegenseitiger Verpflichtungen. Eine Interaktion kann einen Befehl übermitteln, ein Versprechen ausdrücken, vertragliche Vereinbarungen bestätigen und damit ins Leben rufen etc.

Ein Beispiel einer Methode, die diese kommunikativen Handlungen zur Modellierung der Büroarbeit benutzt, ist SAMPO (Speech-Act-Based Office Modelling Approach, [ALL88], [AHL92]). Als eine der wenigen Analysemethoden für Informationssysteme, die die Office-Kommunikation explizit studiert, analysiert sie die sich wiederholenden standardisierten Kommunikationsmuster ("conversations") mit dem Ziel, sie durch ein IS optimal zu unterstützen.

Die Kommunikationsanalyse kann dabei sowohl zur Anforderungserhebung für Informationssysteme, als auch für das Redesign der Büro-Kommunikationsprozesse eingesetzt werden. SAMPO modelliert die Diskurse vollständig und sehr formal ähnlich zu Geschäftsprozessen, die aus einzelnen Sprechakten aufgebaut sind. Modelliert wird das "management of commitments". Hier steht also nicht das Speech-Act-Konzept als solches im Vordergrund, sondern deren Bedeutung in Bezug auf die Installation und Verwaltung von gegenseitigen Verpflichtungen und Zusagen in der täglichen Arbeitspraxis.

SAMPO ist wiederum ein analytischer produktorientierter Ansatz, der sich allerdings diesmal auf die Kommunikationsstrukturen und -muster konzentriert. Wie in den traditionellen Ansätze der Büroautomatisierung und Geschäftsprozessmodellierung werden hier stark reduktiv nur Teilaspekte der Arbeitswirklichkeit erfasst. SAMPO will die Kommunikation selbst studieren und auf Kohärenz/Effektivität hin untersuchen bzw. gestalten.

In KoA$_{\text{L}}$A hingegen dient die Kommunikationsanalyse nur als Ausgangspunkt für die Modellierung der Systemumgebung und als Mittel zur Validierung der Gebrauchstauglichkeit von Softwaresystemen.

Eine Methode, die die Kommunikationsanalyse zur Vervollständigung bzw. "Untermauerung" einer eher traditionellen Vorgehensweise zur Analyse von Anforderungen an Informationssysteme benutzt ist COMMODIOUS [Hol94]. Diese Methode stellt eine Erweiterung des Ansatzes zur Unternehmensmodellierung dar, wie er im F3-Projekt ("From Fuzzy to Formal", [BRLD94]) entwickelt wurde. Sie verwendet dabei die Kommunikationsanalyse als Hilfsmittel bzw. Technik, um zu abgesicherteren Modellen der Organisation zu gelangen. Commodious setzt Discource-Sequence-Diagramme als Elicitation-Technik ein, um ein vollständigeres Verständnis der sozialen Wirklichkeit und ihrer Anforderungen zu erhalten, bevor aufbauend darauf Ziele (objectives model), Informationsobjekte (concept model), Ablauf- (activities and usage model) und Aufbauorganisation (actors model) modelliert und im letzten Schritt die Anforderungen an das IS abgeleitet werden. F3 stellt sehr ausgereifte und komplexe Modellierungstechniken bereit, die zur Entwicklung von Informationssystemen eingesetzt werden können. Die Vorgehensweise steht dabei jedoch in der Tradition der analytischen produktorientierten Methoden der Büroautomatisierung, die nicht den Gestaltungsprozess in den Vordergrund stellen, sondern die Problemanalyse und die Ableitung einer Problemlösung in Form des notwendigen Softwaresystems. Die Kommunikationsanalyse wird hauptsächlich dazu benutzt, die Qualität der klassischen Modellierung (Vollständigkeit, Verständlichkeit etc.) zu verbessern.

System

Zur Gestaltung von Handlungssystemen existieren auch eine Reihe systemorientierter Methoden, die den organisatorischen Kontext, in dem das IS später laufen soll, als eigenes System betrachten, das es mit dem IS zusammen oder ihm vorausgehend zu entwickeln gilt.

Viele dieser Methoden basieren auf Ideen des SOCIOTECHNICAL SYSTEMS DEVELOPMENT (STD, [BH77, WAP88, Eas88]). In diesen Methoden, die speziell auf die Entwicklung von Informationssystemen abzielen, werden das soziale und das technische System gleichberechtigt nebeneinander entwickelt. Zusätzlich zur Entwicklung des technischen Systems findet auch eine Entwicklung bzw. Ausgestaltung des organisatorischen Umfelds statt, in dem das Softwaresystem eingesetzt werden soll. Die Gestaltung beider Systeme erfolgt dabei in wechselseitiger Abhängigkeit. Die Methoden sind in der Regel eher traditionell und analytisch, da sie versuchen, das soziale Teilsystem in gleicher Weise wie das technische Teilsytem möglichst formal und vollständig zu erfassen bzw. zu konstruieren.

Ein Beispiel einer solchen Methode im Kontext der Entwicklung von Informationssystemen ist ETHICS ("Effective Technical and Human Implementation of Computer-based Systems", [Mum93, IH96]). ETHICS ist ein analytischer partizipativer Ansatz zur Ableitung von Softwaresystem und Arbeitsorganisation ausgehend von der Unternehmens-"mission" bzw. den "key objectives" über die "key tasks" hin zu den "information needs" und schließlich zu den möglichen arbeitsorganisatorischen und technischen Umsetzungsmaßnahmen. Neben dieser traditionell analytischen Modellierung der Organisation berücksichtigt die Methode jedoch auch "weiche" soziale Aspekte wie z.B. die Arbeitsqualität bzw. den Grad der "job satisfaction" oder den Selbstbestimmungsgrad der Arbeitsplätze.

Hauptaugenmerk von ETHICS liegt auf der partizipativen Organisationgestaltung und der systematischen Ableitung von Anforderungen an das IS. $\text{KoA}_{\text{L}}\text{A}$ hingegen konzentriert sich

auf die Ausgestaltung des IS in einem bereits global vorgegebenen organisatorischen Kontext.

Einen anderen Niederschlag hat der Systemgedanke in der einflussreichen SOFT SYSTEMS METHODOLOGY (SSM, [Che81, Che89]) gefunden. SSM war ursprünglich ein allgemeiner Ansatz zur Organisationsentwicklung, der sich erst später auch als Methode zur Entwicklung von Informationssystemen bewährt hat. Ziel ist ein iterativer Lernzyklus der betroffenen Anwender, die ihre Wirklichkeit an Hand systemischer Modelle reflektieren, debattieren und umgestalten bzw. weiterentwickeln.

SSM ist eine systemische dialogorientierte Methode, in der die eingesetzten Modelle recht informell gehalten sind und die Softwareentwicklung nur sehr oberflächlich berücksichtigt wird. Schwerpunkt im Vorgehen liegt auf der Organisation und Strukturierung des sozialen Abstimmungsprozesses, weniger in der Modellierung. Ziel ist die Gestaltung neuer Handlungssysteme. Die Organisation wird in einzelne "Aktivitätssysteme" ("systems for purposeful activity") zerlegt, die in Form sogenannter "Root Definitions" (natürlichsprachlicher Text, der nach dem "CATWOE"-Prinzip[4] strukturiert ist) deskriptiv attributiert und in Form konzeptioneller Modelle dokumentiert werden. Die Modelle setzen informell beschriebene "Aktivitäten" in logische Abhängigkeiten und werden zur Problemanalyse nebst partizipativer Lösungssuche bzw. Organisationsentwicklung eingesetzt.

Zwei Methoden, die Ideen dieser systemischen Modellierung in traditionelle analytische und produktorientierte Methoden der Entwicklung von Informationssystemen integrieren, sind MULTIVIEW [AWH90] und FAOR [Sch88].

MULTIVIEW ist weniger eine fest definierte Methode als vielmehr eine offene Methodologie bzw. ein generischer Rahmen zur situationsspezifischen Entwicklung oder Anwendung von Methoden der Informationssystementwicklung. Multiview berücksichtigt im Systementwicklungsprozess neben der traditionell technischen auch eine persönliche und eine organisatorische Modellierungsperspektive und deckt damit alle drei Gestaltungsebenen ab.

Vom Charakter her stellt Multiview einen traditionellen produktorientierten analytischen Ansatz dar, der um die Kontextmodellierung und -gestaltung ergänzt wurde. Zur Analyse und funktionalen Beschreibung der Organisation wird SSM propagiert. Ergänzt um eine konzeptionelle Modellierung der Informationseinheiten und der Verarbeitungsprozesse hat der Ansatz hier starke Ähnlichkeit mit dem INFORMATION ENGINEERING (s.o.). Für die Analyse und den Entwurf soziotechnischer Aspekte der persönlichen Arbeitssituation, also zur Einpassung des Softwaresystems in die Organisation, wird z.B. ETHICS (s.u.) eingesetzt. MULTVIEW legt seinen Schwerpunkt dabei eher auf die Gestaltungsebene des Nutzungssystems. Eine Analyse des Handlungssystems dient eher der Absicherung bzw. Ableitung der Anforderungen, weniger der Organisationsumgestaltung.

FAOR ("Functional Analysis of Office Requirements") verwendet SSM hingegen hauptsächlich zur Problemanalyse und Unterstützung partizipativer Lernprozesse. FAOR ist ein sehr umfassender und komplexer Ansatz der Organisationentwicklung, der zur Modellierung der Organisation stark auf analytische Konzepte wie Funktionen, Informationsflüsse, Petrinetze etc. setzt. Schwerpunkt ist die systematische Analyse und Umgestaltung der Organisation, weniger die partizipative Ausgestaltung konkreter Informationssysteme.

[4] C = Customer, A = Actors, T = Transformation Process, W = Weltanschauung, O = Owner, E = Environmental Factors

Ein Ansatz zur spezifischen formalen Beschreibung des organisationellen Systemkontexts ist die HUMAN SYSTEM ANALYSIS nach Jeffrey [Jef96]. Das "human system" wird in diesem Ansatz weitgehend formal mittels der zentralen Konzepte der "community" und der "intentional actions" beschrieben, die dem Gebiet der Psychologie entlehnt werden. Eine "community" wird beschrieben, indem ihre menschlichen oder technischen Mitglieder bestimmt werden und zusammen mit ihren Rollen, Pflichten, Aufgaben, Sprache, Prinzipien und Praktiken modelliert werden. Die Praktiken bzw. Aktivitätsmuster der community werden durch "intentional actions" dargestellt. Die Aktivitäten werden dabei an Hand ihrer Ziele, dem notwendigen Wissen, den benötigten Fähigkeiten, den durchzuführenden Prozeduren und dem zu erzielenden Endergebnis beschrieben. Die Methode bietet damit zwar ein umfangreiches Instrumentarium zur strukturierten Modellierung soziotechnischer Systeme, sie gibt dabei jedoch keine weitergehende Hilfestellung für ein konkretes Vorgehen bei der Umgestaltung bzw. der Konstruktion bzw. Ableitung von Anforderungen an Informationssysteme.

Anders die werkzeugunterstützte Methode ORDIT [DBCS94], die auch den gestalterischen Prozess mit berücksichtigt. Zentrale Aufgabe ist die methodische Erhebung, Analyse und Diskussion von nichtfunktionalen organisatorischen bzw. politischen Anforderungen, die auch Aspekte wie das Machtgefüge, Verpflichtungen, Verantwortlichkeiten, Kontrolle, Autonomie, Werte und Ethik thematisieren. ORDIT stellt hierzu eine sehr mächtige und komplexe Sprache zur umfangreichen Modellierung von Organisationen zur Verfügung, die sowohl textuelle als auch grafische Notationen kennt. Die Arbeitsorganisation bzw. das "enterprise model" wird auf Grundlage von Agenturen, Aktivitäten und Ressourcen modelliert. Agenturen stehen in strukturellen Beziehungen ("structural roles") und besitzen funktionelle Verantwortlichkeiten ("functional roles"), die die Zusammenarbeit ermöglichen bzw. festlegen. Zur Ableitung von Informationsmodellen und nichtfunktionalen Anforderungen werden die realen Konversationen modelliert. Ausgehend von den Verantwortlichkeiten der verschiedenen Agenturen dienen die erstellten Modelle der partizipativen Aufdeckung bzw. Diskussion der nutzerspezifischen Anforderungen bzgl. des Informationsbedarf, der durchzuführenden Aktivitäten und der Speicherung von historischen Daten.

ORDIT stellt ähnlich zu $\text{KOA}_\text{L}\text{A}$ eine gemeinsame Modellierungssprache zur Verfügung, die als gemeinsames konzeptuelles Instrumentarium und Kommunikationsmedium die Problem-Besitzer (Anwender) und Problemlöser (Entwickler) zu einer dialektischen Erforschung des Problem- und Lösungsraumes befähigt (eine neutrale Sprache unabhängig von Problem oder Lösungswelt). Allerdings entwickelt ORDIT hierzu völlig neue und komplizierte Notationen, während $\text{KOA}_\text{L}\text{A}$ sich hierbei möglichst an UML orientiert.

Sonstiges

Im Allgemeinen lassen sich Ansätze, die sich mit der Gestaltungsebene Handlungssystem befassen, auch dem Gebiet der "social informatics" zuordnen [IH96]. Hier finden sich viele Arbeiten, die den sozialen, politischen und kulturellen Aspekten und Auswirkungen des Einsatzes von Computersystemen Rechnung tragen. Ein einflussreiches Beispiel ist hier z.B. der "interactionist approach", der Informationssystemen nicht aus technischer Werkzeugsicht betrachtet, sondern sie in ihrer Rolle als soziale Institution untersucht [KS82][KI89], die in sozialen Wechselwirkungen mit anderen Organisationsmitgliedern steht.

Ansätze aus diesem Bereich dienen dabei im Wesentlichen der Untersuchung der Auswirkungen von Informationssystemen. Sie geben in der Regel keine konstruktiven Hilfestellungen für den Entwurf dieser Systeme an.

3.4. Zusammenfassung

Klassische Softwareentwicklungsmethoden (SA, OOA, Fusion, Z etc.) sind stark von der Entwicklungssicht beeinflusst und darum auf der Gestaltungsebene des Softwaresystems beheimatet. Sie konzentrieren sich auf die formale Modellierung des zu entwickelnden Softwaresystems und seiner Schnittstelle nach außen, um eine gesicherte Ausgangsposition für die Lösung des Entwicklungsproblems, d.h. die Softwarekonstruktion und Realisierung zu gewinnen.

Werden diese analytischen produktorientierten Methoden auch für die weitergehende Modellierung der Anwendungsumgebung eingesetzt (wie z.B. im Business Modelling des RUP), so finden auch hier die gleichen objektivistischen Modellierungskonzepte Anwendung. Verglichen mit der realen Komplexität eines soziotechnischen Systems kann dies nur zu sehr vereinfachenden Darstellungen der Anwendungswelt führen (RUP beispielsweise beschränkt sich hier auf die Geschäftsprozesse bzw. Datenflüsse).

Ein weitere Komplikation, die diese Ausdehnung des Einsatzbereichs von Softwaremodellierungssprachen fragwürdig erscheinen lässt, sind die unweigerlich entstehenden Kommunikationsprobleme. Auftraggeber bzw. Interessengruppen haben in der Regel große Probleme mit dem Verständnis und der Validierung formaler Spezifikationen. Es besteht ein “impedance mismatch” zwischen der nur informal fassbaren Problemwelt und dem zwingend zu formalisierenden Lösungsraum – der technischen Spezifikation eines Softwaresystems.

Mit dem Thema der Ausgestaltung des globalen Handlungssystems, also der eigentlichen Arbeitspraxis, beschäftigen sich neben den eher traditionellen analytischen Ansätzen aus dem betriebswirtschaftlichen Bereich (wie z.B. der Büroautomatisierung, dem Information Engineering und dem Business Process Reengineering) auch neue systemische Ansätze, die Erkenntnisse der Arbeitswissenschaft und der Soziologie in den Bereich der Softwareentwicklung integrieren. Diese basieren im Allgemeinen auf Ideen der Soft Systems Methodology oder des Sociotechnical Systems Development, die dem organisatorischen Kontext des neuen Softwaresystems nun ebensoviel Beachtung schenken wie den Eigenschaften des Softwaresystems. Alle Methoden zur Gestaltung der Handlungssystemebene zielen jedoch auf die Modellierung, Analyse und optimierende Umgestaltung des globalen Organisationsgefüges ab, in deren Kontext zwar auch IS eingesetzt werden, deren Ausgestaltung methodisch gesehen im Grunde jedoch keine ausgezeichnete Rolle spielen.

Genau diese Gestaltung von Informationssystemen im Kontext ihrer direkten Anwendungsumgebung ist jedoch Gegenstand von $\text{KoA}_{\text{L}}\text{A}$ und aller anderen Methoden, die auf die Gestaltungsebene des Nutzungssystems abzielen.

Während eine Reihe dieser Ansätze aus dem Gebiet des HCI abstammen und sich eher mit der direkten Bedienungsschnittstelle des Softwaresystems und ihrer Ergonomie befassen, verlagert sich der Gestaltungsbereich in neueren Ansätzen konzeptionell eher vom Computer weg zu den Systemanwendern und ihrer direkten Arbeitsumgebung. Die Ergonomie eines Softwaresystems wird hier insbesondere in Bezug auf seine Passgenauigkeit zum umgebenden sozialen

System beurteilt. Der Gestaltungsprozess zielt auf eine optimale Unterstützung der kooperativen Arbeitsprozesse von Systembenutzern und sonstigen Betroffenen über organisatorische und technische Grenzen hinweg ab[Gru90].

Methoden, die im Bereich der Gestaltung von Nutzungssystemen eingesetzt werden sind in der Regel stark dialogorientiert. Neben den "Workshop-Methoden" wie USTM oder JAD oder dem weiterentwickelten Contextual Design kommen aber auch traditionelle analytische produktorientierte Ansätze wie Volere zum Einsatz, die den Kontext zumindest in Form von Use-Cases in der Modellierung berücksichtigen.

Ein weitergehender produktorientierter Ansatz zur iterativen Ausgestaltung von Nutzungssystemen, der sowohl ein gut strukturiertes und beherrschbares Vorgehen beschreibt, als auch eine systemische und darum ganzheitlichere Modellierung des sozialen Arbeitskontexts ermöglicht, fehlt bislang.

Genau diese Lücke wird durch die systemische und produktorientierte Methode KoA$_{\text{L}}$A geschlossen, die in den folgenden Abschnitten beschrieben wird.

4. Die Modellierungssprache von Koala

"Informatik-Modelle stehen nicht als formale Gebilde für sich genommen, sondern werden in den von uns getragenen sozialen Prozessen konstituiert. Informatik-Modelle sind so zu gestalten, daß sie die sozialen Prozesse, in denen sie wirksam werden, als Werkzeuge ("Handgriffe") unterstützen und nicht behindern."

Christiane Floyd [FK98]

KoA$_L$As Ausgangspunkt bei der Gestaltung eines IS ist die konkrete Arbeitssituation der vom zukünftigen Softwaresytem Betroffenen, d.h. ihr Arbeitskontext, der durch das neue Softwaresystem umgestaltet werden wird. Zur Beschreibung dieses Handlungssystems stellt KoA$_L$A eine Modellierungssprache zur Verfügung, die nun in den folgenden Abschnitten vorgestellt wird. Die genaue Vorgehensweise bei der Ausgestaltung des Softwaresystems wird im nächsten Kapitel (5) beschrieben.

Da der Kontext des praktischen Handelns eines Organisationsmitglieds ganz wesentlich von seinen Interaktionsbeziehungen mit anderen Mitgliedern der Organisation bestimmt ist, spielen Kommunikationsstrukturen und deren Inhalte in KoA$_L$A eine wesentliche Rolle. Im Unterschied zu konventionellen Methoden der Kommunikationsanalyse [SN87], die ihren Schwerpunkt auf eine Analyse der Informationsflüsse legt, werden die Kommunikationsakte bzw. -prozesse selbst als eigenständige Aktivitäten bzw. Aktivitätsmuster und damit als integrierte Bestandteile des Handlungssystems betrachtet, die sich im Umfeld eines neuen IS konstituieren und welche es auszugestalten gilt.

Wesentliche gestalterische Mittel von KoA$_L$A sind dabei eine Reihe von Modellen (dargestellt in Form von grafischen Diagrammen und semiformalen Texten), die in enger Zusammenarbeit mit allen Betroffenen erstellt, abgestimmt und weiterentwickelt werden. KoA$_L$A sieht die Gestaltung hierbei als einen iterativen Prozess, der in engem Dialog mit den Betroffenen zusammen durchgeführt wird und ausgehend von einer realen Arbeitssituation diese iterativ – "auf dem Papier" – fortentwickelt, bevor mit der Softwarespezifikation die eigentliche Realisierungsphase beginnt.

KoA$_L$A modelliert arbeitsteilig organisierte Handlungssysteme, bestehend aus mehreren technischen oder natürlichen Einzelsystemen bzw. "Akteuren". Zur Darstellung des Gesamtsystems werden drei unterschiedliche Dimensionen betrachtet bzw. entsprechende Sub-Modelle erstellt: ein "Organisationsmodell", ein "Kooperationsmodell" und ein "Wissensmodell" (die Begriffswelt):

- Das Organisationsmodell dokumentiert die statische Struktur der Organisation, indem es alle technischen und menschlichen Akteure des modellierten Wirklichkeitsbereichs inklusive ihrer Struktur, Motivation und Aufgabenverteilung beschreibt.

- Das Kooperationsmodell dokumentiert die dynamischen Aspekte, indem sie Interaktionsbeziehungen zwischen den Organisationseinheiten und ihre geregelten bzw. spontanen Interaktions- bzw. Aktivitätsmuster modelliert.

- Das Wissensmodell schließlich dokumentiert Teile des in der Arbeitspraxis verwendeten Fachwissens.

Als Ausgangspunkt für die Identifikation der näheren Softwaresystemumgebung (Nutzungssystem) werden zunächst die globaleren Organisationszusammenhänge (das Handlungssystem) skizziert. Als Ergebnis der nachfolgenden Aus- bzw. Umgestaltung des Nutzungssystems tritt das IS dann als neuer technischer Akteur in Erscheinung, der in seiner organisatorischen Rolle und seinem Kommunikationsverhalten inhaltlich festgelegt ist. Aufgabe der dann folgenden Software-Spezifikation ist die konkrete technische Ausgestaltung des Akteurs als Softwaresystem.[1]

Zur Dokumentation der verschiedenen Modelle stehen fünf verschiedene Diagrammtypen zur Verfügung: das Konzept-, Kontext-, Profil-, Aktivitäts- und Interaktionsdiagramm.

Neben einer optionalen grafischen Darstellung werden alle Modellelemente auch in semiformaler Weise textuell dokumentiert. Dabei werden sowohl die Elemente selbst unter einem eindeutigen Namen und mit allen ihren Merkmalen notiert, als auch ihre Beziehungen zu anderen Modellelementen. Die textuellen Elementbeschreibungen zusammen genommen bilden die sogenannte *Projektdatenbank* ("repository", "data dictionary" etc.).

Die Projektdatenbank dient der vollständigen Dokumentation der in den Diagrammen verwendeten Modellelemente bzw. ihrer freisprachlichen Erläuterung und Ergänzung. Diese Diagramme dienen vor allem dem Überblick und der Verwendung in Design-Meetings bzw. Workshops. Sie stellen verschiedene Sichten auf die Projektdatenbank her bzw. greifen bestimmte Ausschnitte oder Aspekte zum Zwecke einer anschaulichen Darstellung heraus.

Die verwendeten Modellierungskonzepte und -notationen lehnen sich – sofern möglich – eng an UML an, um die Anwendung von KoA$_{\text{L}}$A für Entwickler (und evtl. auch Anwender) zu erleichtern, die diese Notation schon aus dem Einsatz im Bereich der Softwarespezifikation und -entwicklung her kennen.

Im Folgenden werden nun im Anschluss an eine Darlegung allgemeiner Modellierungsgrundsätze die einzelnen Modelle von KoA$_{\text{L}}$A vorgestellt. Dabei wird vor allem auf die Möglichkeiten zur grafischen Darstellung der Modelle in Form verschiedener Diagramme eingegangen. Die Möglichkeiten zur textuellen Repräsentation der Modellelemente werden hingegen nur exemplarisch im Anhang erläutert (siehe Kap. B).

Die Modelle und ihre Modellierungskonzepte werden eher informell vorgestellt. Eine formale und vollständige Darstellung bzw. Definition der Konzepte findet sich ebenso im Anhang (A).

[1] [CdAS89] unterscheidet diese unterschiedlichen Gegenstandsbereiche der Modellierung auch durch eine eigene Nomenklatur. Das "Abstract Model" stellt das normative Handlungssystem dar. Das "Descriptive Model" beschreibt das operative Nutzungssystem. Das "Specification Model" schließlich steht für die technische Softwarespezifikation.

4.1. Grundlagen der Modellierung

Modelle spielen in fast allen Methoden der Softwareentwicklung eine zentrale Rolle. Je nach aktueller Projektphase und durchzuführenden Entwicklungsschritten werden unterschiedliche Modelle der gleichen oder unterschiedlicher Wirklichkeiten (z.B Ist- bzw. Soll-Modelle) erstellt. Modelle sind zentrale "Artefakte" der Projektarbeit, die im Laufe der Entwicklung als Kommunikationsmittel eingesetzt und sukkzessive (fort-)entwickelt werden, bis schließlich ein Modell des zu realisierenden technischen Systems präzisiert ist und rechnerbasiert umgesetzt werden kann. Ein Entwicklungsprozess spezifiziert Art, Inhalt und Reihenfolge der zu erstellenden Artefakte. Eine Methode definiert ihre genaue Ausprägung und die bei der Erstellung der Artefakte durchzuführenden Schritte bzw. die dabei anzuwendenden Techniken.

Wesentliche Aufgabe der Modellierung ist die kompakte und übersichtliche Darstellung komplexer Sachverhalte, um sie kommunizieren bzw. diskutieren zu können. Während in der Analysephase Modelle noch eher der Validierung bzw. Gestaltung des Anforderungsprofils an das neue System dienen, werden Modelle in der Entwurf- und Realisierungsphase zur abstrakten, möglichst formalen Spezifikation des Softwaresystems eingesetzt, um sie in der Folge schrittweise zu konkretisieren bzw. in Code zu transformieren.

In der Informatik lassen sich unterschiedliche Typen von Modellen unterscheiden [WGKI98]:

"*Idealmodelle*" stellen Abbildungen bestimmter Sachverhalte des realen Anwendungsbereichs dar. Solche Modelle finden sich vor allem bei der Modellierung technischer Systeme, d.h. bei Systemen, die sich sowohl in ihrer Außenwirkung als auch in ihrer Binnenstruktur durch Zielfunktionen bestimmen lassen. Idealmodelle werden hier z.B. als Konstruktionsplan dargestellt oder als formale Spezifikation von Softwaresystemen notiert. Idealmodelle finden daher in der Softwareentwicklung vor allem in der Spezifikations-, Entwurfs- und Realisierungsphase Anwendung, in der das zu erstellende technische System festzulegen und sukzessive zu konkretisieren ist.

"*Simulationsmodelle*" stellen spezialisierte Idealmodelle dar, die zur Vermeidung teurer Realsimulationen am Rechner durchgespielt werden, um sich konkretisierte Sachverhalte vor Auge zu führen bzw. bestimmte Eigenschaften zu untersuchen. Ein Beispiel für ein solches Modell ist ein Oberflächenprototyp, der zur möglichst frühen Validierung eines noch zu entwickelnden Softwaresystems herangezogen wird.

"*Veranschaulichungsmodelle*" schließlich dienen der "metaphorischen" Darstellungen von Idealmodellen, die zu abstrakt oder zu komplex für die menschliche Vorstellungskraft sind. Um unverständliche Modelle der menschlichen Anschauung zugänglicher zu machen, werden sie beispielsweise durch eine passende grafische Notation vergegenständlicht, oder aber durch eine passende Analogie zu bereits existierenden und eingeführten Modellen in Beziehung gesetzt. Ein heute gängiger Weg zum Nachweis der strukturellen Ähnlichkeit von Veranschaulichungs- und Originalmodell ist dabei der Umweg über Supertheorien wie z.B. der allgemeinen Systemtheorie oder der Informationstheorie, die es ermöglichen, selbst "*komplexe biologische oder sozialwissenschaftliche Phänomene in den Horizont technischer Rationalität zu ziehen*" [WGKI98].

KoA$_{\text{L}}$As Modelle besitzen – so wie die meisten Modelle aller Softwareentwicklungsmethoden auch – die Systemtheorie als zugrundeliegende Supertheorie. Die Zielrichtung der Modellie-

rung liegt bei KoA$_L$A jedoch auf der *Veranschaulichung* komplexer Sachverhalte des sozialen Umfelds, während klassische Modelle von Entwicklungsmethoden vor allem auf eine *Idealisierung* des zu entwickelnden technischen Systems abzielen. Diese unterschiedlichen Ansätze der systemtheoretischen Modellierung spiegeln sich auch in der Literatur in unterschiedlichen Strömungen wieder, die dem analytischen einen ganzheitlichen Zugang zum Erkennen entgegenstellen, dem deskriptiven einen normativen Zugang, dem technikzentrierten einen menschenzentrierten usw. (siehe hierzu z.B. [Rap88]).

Eine sehr grundlegende und revolutionäre Kritik an den vorherrschenden rational-analytischen Vorgehensweisen findet sich z.B. auch in [Goo94]. Goorhuis kritisiert hier die in der Informatik übliche Subjekt-Objekt-Spaltung im Modellbildungsprozess, indem er den selbstbezüglichen Charakter jeder Wahrnehmung bzw. Darstellung von Realität untersucht und auf Grundlage der Allgemeinen Modelltheorie die dadurch unweigerlich auftretenden Zielkonflikte nachweist.[2] Eine Überwindung dieser Zielkonflikte gelingt Goorhuis dabei durch die Anwendung des erkenntnistheoretischen Modells des Konstruktivismus[3], nach dem alle Wahrnehmung bzw. alles Wissen über die Welt als konstruiert zu betrachten ist.

Modelle der Informatik können darum inhärent nie "korrekte" Abbildungen ojektiver Sachverhalte sein, sondern sind immer nur subjektive kognitive Konstruktionen, die einen bestimmten praktischen Zweck erfüllen. Der unweigerlich konstruktivistische Charakter jeder Modellbildung führt zu der Erkenntnis, dass jedes informatische Modell und jeder Modellbildungsprozess den *dynamischen* sozialen Kontext seiner Entstehung berücksichtigen muss, um nützlich bzw. erfolgreich zu sein.

KoA$_L$As Modelle stellen daher Veranschaulichungsmodelle dar, die die komplexe Arbeitswirklichkeit partiell, vereinfacht, systemisch, anschaulich und aus verschiedenen Perspektiven darstellen sollen. Diese Anschaulichkeit ist für den Prozess der Ausgestaltung von Handlungssystemen essentiell, wenn er partizipativ mit den Betroffenen zusammen durchgeführt werden soll. Wie viele im Bereich der Softwareentwicklung erstellten Modelle stellen auch KoA$_L$As

[2] Solche inhaltlichen Zielkonflikte der Modellbildung sind z.B die Folgenden: 1. Das Abbilden einer nichtformalen Welt in ein formales Modell. 2. Modelle der Informatik sollen situativ, gleichzeitig aber allgemeingültig sein. 3. Die Pluralität der Sichtweisen, d.h. der individuellen Perzeptionsmodelle, muss für die Informatik reduziert werden, ist aber letztlich nicht reduzierbar. 4. Die Ambiguitäten der Wirklichkeit sind nur umständlich formal abbildbar und treten daher für das Modellsubjekt als mangelhafte Merkmale auf. etc.

[3] Der radikale *Konstruktivismus* geht zurück auf Heinz v. Foerster, Ernst v. Glasersfeld, Paul Watzlawick u.a. (sieh z.B. [Wat81]). Entgegen der Auffassung der traditionellen philosophischen Erkenntnistheorie, die den Unterschied von Subjekt und Objekt als konstitutiv für die Philosophie ansieht und die Existenz einer der Erkenntnis zugänglichen objektiven Wirklichkeit postuliert, vertritt der Konstruktivismus den Standpunkt, dass es keine vom Beobachter unabhängige Wirklichkeit gibt und wir unsere Wirklichkeit selbst konstruieren. Von starkem Einfluss ist hier der Begriff des "autopoietischen Organismus", der von dem chilenischen Biologen und Erkenntnistheoretiker Humberto Maturana stammt und als Unterscheidungsmerkmal lebender und nicht-lebender Systeme dient [MV87]. Menschliche Individuen haben als autopoietische Systeme keinen informationellen Input und Output, sondern erzeugen ihre Informationen innerhalb des Systems. Selbst einfache Sinneswahrnehmungen wie Sehen und Hören stellen individuelle Konstruktionen dar, da die eigentliche Wahrnehmung nicht in den Sinnesorganen stattfindet, sondern Ergebnis kognitiver Prozesse in den mit den Sinnesorganen vernetzten Hirnregionen ist [GM95]. Dieses Konzept der "informationellen Geschlossenheit" steht in klarem Widerspruch zum Informationsverarbeitungsansatz, der von Kognitivisten bzw. Objektivisten vertreten wird. Durch die strukturelle Kopplung von Organismus und Umgebung werden Anpassungsprozesse ausgelöst, die die Voraussetzung für das Fortbestehen des Organismus bilden. Lebewesen sind autonom und strukturdeterminiert, d. h. sie können ihre eigenen Gesetzlichkeiten herausbilden. Es gibt keine Ursache-Wirkungs-Relation zwischen den Reizen der Umwelt und den individuell aufgebauten Wissensstrukturen [KC91].

Modelle damit primär keine Realitätsabbilder dar, sondern sie rekonstruieren eine Sprache, mit der Fachleute Sinnzusammenhänge im Anwendungskontext beschreiben. Sie definieren eine sozial konstruierte Realität, die Handlungen (an-)leiten, indem sie die Wahrnehmung kanalisieren, die Basis für Entscheidungsprozesse begründen etc. Nach [FK98] sind Informatik-Modelle darum so zu gestalten, dass sie die sozialen Prozesse, in denen sie wirksam werden, als Werkzeuge ("Handgriffe") unterstützen und nicht behindern.

Eine solche Forderung findet sich z.B. auch bei Schefe (siehe [Sch99b]): Aufgabe der Anforderungsanalyse ist es nicht, die Wirklichkeit mathematisch und von jedem Realitätsbezug befreit formal zu repräsentieren, sondern sie in Form von handlungsorientierten Konzeptualisierungen textuell zu beschreiben: *"Es gilt, das Paradigma der abbildenden Modellierung durch ein Paradigma der normierenden Beschreibung abzulösen"*. Nur so kann Sinn, Bedeutung und Intentionen erfasst und greifbar gemacht werden. Erst die informalen Anforderungen können den notwendigen Realitätsbezug liefern. Eine intentionale Bedeutungszuweisung ist grundsätzlich von der einer mathematischen Interpretation verschieden. Schefe nennt sie daher "Sinnzuweisung".

Alle Modelle bzw. Diagramme von KoA$_{\text{L}}$A sind implizit sowohl Gestaltungsergebnis als auch Diskussionsgegenstand und müssen daher als Arbeitsmittel für einen Abstimmungsprozess geeignet sein, an dem sowohl die Benutzer als auch das Management und die Softwareentwickler teilnehmen. Da kein technisches System vollständig beschrieben, sondern ein soziales System kontextbezogen entworfen werden soll, muss die Modellierung auf der einen Seite spezifische Kontextinformationen wie z.B. Beispiele, Sonderfälle, Randbedingungen, Interessenkonflikte einbeziehen, aber auf der anderen Seite auch Auslassungen erlauben.

KoA$_{\text{L}}$A unterstützt daher den beliebigen Wechsel der Abstraktionsebenen, auch innerhalb einer Darstellung. Unterstützt werden dabei auf Modell- wie auch auf Darstellungsebene folgende Arten der Abstraktion bzw. der vereinfachenden Darstellung:

- Selektion, Projektion (unvollständige Modellierung)
- Verfeinerung (z.B. mittels Aggregation und Expandierung)
- Klassifikation (z.B. mittels Spezialisierung, Konkretisierung, Rollenzuweisung)

Die Modellierung von KoA$_{\text{L}}$A folgt dabei folgenden Prinzipien:

- Die Klassifikationsebene kann in der Darstellung beliebig gemischt werden. Sowohl Muster als auch Szenarien, sowohl Typ und Instanzkonzepte können in Modellen als auch in Diagrammen nebeneinander stehen.
- Möglichst wenige unterschiedliche Modellierungskonzepte, die aber orthogonal in mehreren Modellen bzw. Diagrammarten für viele unterschiedliche Zwecke verwendet werden können.
  Beispielsweise könnte ein Kontextdiagramm zur Darstellung einer Kooperationsstruktur (siehe Kap. 4.4.1) durchaus auch einzelne Tätigkeitsprofile definieren und einzelne Organisationseinheiten durch Aggregationsbeziehungen verfeinern.
- Offene Modellierung: Der Detailierungsgrad ist frei wählbar, auch innerhalb eines Diagramms. Modelle können unvollständig sein, Diagrammabschnitte redundant. Die erstellten Modelle sind daher im Allgemeinen nicht simulier-, animier- oder formal trans-

formierbar. Die Modellierung ist im Allgemeinen "viewpoint"-orientiert, d.h. es werden punktuell nur solche Teilbereiche herausgegriffen, die gerade von Interesse sind.

- Einfache Notation (intuitiv verständlich, geeignet für Ad-Hoc-Modellierung). Auch leicht mit Papier und Bleistift anzuwenden (einfache Notationselemente)
- Alle grafischen Modellelemente werden auch freitextlich in einem semiformalen Dokument der Projektdatenbank beschrieben.
- Konzepte und Notation sind weitgehend an UML angelehnt (Wiedererkennungswert).

4.2. Das Wissensmodell

> "Der Sprachbildner war nicht so bescheiden, zu glauben, dass er den Dingen eben nur Bezeichnungen gebe, er drückte vielmehr, wie er wähnte, das höchste Wissen über die Dinge mit den Worten aus"
>
> Nietzsche[4]

Konzept: lat. concipere; zusammenfassen, begreifen, sich vorstellen

Im Wissensmodell von KoA$_{\text{L}}$A wird der konzeptuelle Wortschatz der Protagonisten, d.h. ihre Sprache bzw. die fachliche Begriffswelt (auch "Thesaurus", "Fachsprache", "Terminologie" oder "Ontologie" genannt) und das damit verbundene Hintergrundwissen dokumentiert. Dabei werden fachliche Konzeptionen nicht nur intrinsisch, d.h. abstrakt erläutert, sondern auch durch Hinzunahme konkreter Beispiele konkretisiert.

Zum einen sind adäquate und übereinstimmende mentale Konzeptionen des Anwendungsbereichs (d.h. eine gemeinsame Sprache) zwingende Voraussetzung für eine erfolgreiche Kommunikation der Interessengruppen, der Anwender und der Entwickler. Zum Anderen führt der Einsatz eines neuen IS immer auch zu einer weitergehenden Standardisierung bzw. Neubildung dieser Begriffswelt, so dass das Wissensmodell im Kontext der Entwicklung von Informationssystemen selbst immer auch einen eigenständigen Gestaltungsbereich darstellt. Diese Standardisierung findet z.B. zwangsläufig dadurch statt, dass ein Softwaresystem nur ganz bestimmte – u.U. auch neuartige – mentale Konzepte abbildet, verarbeitet, transportiert oder speichert.

Das Wissensmodell definiert eine Ontologie des Anwendungsbereichs, indem es die verwendeten Begriffe und ihre Beziehungen festlegt und ihre Bedeutung im Bezug zur realen Welt definiert.

In der Kommunikationstheorie bzw. der Semiotik wird die Beziehung zwischen den in einer Kommunikation verwendeten Symbolen (wie z.B. Worte, Zeichen, Laute), ihrer Bedeutung (das mentale Modell oder das Konzept, das die Kommunikationsteilnehmer unter dem Symbol verstehen) und den referenzierten Dingen der realen Welt durch das semiotische Dreieck ausgedrückt (siehe hierzu z.B. [Sta02]). Ein Symbol korrespondiert dabei mit einem Ding der

[4] Friedrich Nietzsche: Sämtliche Werke. Herausgeber: Giorgio Colli und Mazzino Montinari. Kritische Studienausgabe in 15 Bänden: "Menschliches, Allzumenschliches I" , S. 11

realen Welt, wobei die Referenz selbst durch den Sender bzw. Empfänger auf dem Umweg der subjektiven Begriffsbildung hergestellt wird.

Eine erfolgreiche Kommunikation setzt voraus, dass die in der Kommunikation verwendeten Symbole bzw. Begriffe bei Sender und Empfänger möglichst deckungsgleiche mentale Konzepte aktivieren, d.h. Konzepte, die sich auf die gleichen Dinge der realen Welt beziehen. Auf soziokultureller Seite ist es in einer arbeitsteiligen Organisation darum unabdingbar, sich auf eine gemeinschaftliche Begriffswelt bzw. Ontologie zu einigen, wenn man Kooperations- und Kommunikationsprozesse erklären, modellieren oder gestalten will. Eine vollständige Formalisierung dieser Ontologie ist nur dann notwendig, wenn z.B. eine automatische Auswertung bzw. Informationsverarbeitung in einer Maschine beschrieben werden muss (siehe hierzu das Objekt- bzw. Datenmodell der Spezifikationsphase, Abschnitt 5.8). Zur anwenderorientierten Beschreibung bzw. Gestaltung des Anwendungsbereichs reicht hingegen eine informellere bzw. abstraktere Ebene (siehe Abschnitt 5.4.3) der sprachlichen Begriffsfestlegung.

Da die verwendete Fachsprache wesentlich durch das praktische Handeln bedingt ist[5], nimmt in KoA$_L$A die Modellierung der alltäglichen Arbeitspraxis einen großen Raum ein (siehe die folgenden Abschnitte). Die im Wissensmodell von KoA$_L$A definierten Konzepte finden sich auch in den Interaktions- und Konversationsstrukturdiagrammen wieder und symbolisieren dort den Inhalt der modellierten Kommunikationsbeziehungen.

Gerade in den Anfangsphasen der Kontextgestaltung ist eine kognitiv adäquate Repräsentation dieses Fachwissens entscheidend für eine erfolgreiche gemeinsame Analyse und Definition der relevanten Fachkonzeptionen. Vor einer ähnlichen Herausforderung stehen auch Entwickler wissensbasierter Systeme, die in enger Zusammenarbeit mit den betroffenen Experten versuchen, deren Wissen zu erfassen und zu formalisieren. Auch hier ist die Qualität der Repräsentationsform von entscheidender Bedeutung für eine erfolgreiche Kooperation bzw. selbstständige Introspektion.

Von großem Einfluss auf dieses Teilgebiet der Informatik waren die in der Psychologie zur Modellierung des Gedächtnisses entwickelten und auf Konzepthierarchien basierenden begrifflichen Repräsentationformen [Gör93]. Im Fachgebiet der künstlichen Intelligenz führten sie zu eigenen – auch kognitionswissenschaftlich abgesicherten – Wissensrepräsentationsmodellen, die allgemein als "semantische Netze" bezeichnet werden. Eine andere Konzeption der kognitiven Psychologie, die auf informatischem Gebiet eine ebensolche Erfolgsgeschichte aufweisen kann, ist die Schemakonzeption bzw. der Frame-Ansatz, der seinen Schwerpunkt weniger auf die semantischen Beziehungen von Wissenseinheiten, sondern auf deren komplexe innere Strukturierung legt.

Ein ähnlicher Einfluss dieser in der psychologischen Forschung entwickelten Konzeptionen

[5]Sprache bzw. Bewusstsein treten nach Heidegger erst dann in Erscheinung, wenn im praktischen Arbeitshandeln ein Bruch ("Breakdown") eintritt, und zur Überwindung dieses Hindernisses Kommunikation mit Anderen oder wissensbasiertes Handeln (z.B. der Gebrauch eines Werkzeugs) notwendig wird. Begriffe bilden keine Wirklichkeit ab, sondern sind zweckorientierte und in der Praxis bewährte Mittel bei der Überbrückung solcher Zusammenbrüche [WF87]. Diese Ideen finden sich auch in Analysemethoden der Sozialwissenschaften wie z.B. der des "Symbolic Interactionism" [Blu86], der Sprachphilosophie nach Wittgenstein und den "Speech Actions" von Searle wieder.

Sprache tritt hier als Werkzeug in Erscheinung, das Handlungsmöglichkeiten schafft und darüber die Wirklichkeit verändert. Damit schließt sich der Hermeneutische Zirkel: Die Ausgestaltung von Handlungssystemen führt zur Um- bzw. Neugestaltung einer Sprache, die neue Handlungsmöglichkeiten schafft und daher wiederum neue Gestaltungsmöglichkeiten bzw. -Notwendigkeiten eröffnen.

lässt sich auch in anderen Gebieten der Informatik wie z.B. in den Modellierungstechniken erkennen, die im Bereich der Datenbankforschung entwickelt wurden und darauf aufbauend auch in Entwicklungsmethoden des Softwareengineerings Einzug hielten. Hier sind vor allem die semantische Datenmodellierung bzw. die darauf aufbauende objektorientierte Modellierung zu nennen, die sowohl Ideen der semantischen Netzwerke, als auch Ideen der framebasierte Wissensrepräsentation inkorporierten [CM90a].

KoA$_{\text{L}}$As Wissensmodell nun greift Ideen der in diesen Gebieten der Informatik entwickelten Daten- bzw. Wissensrepräsentationstechniken auf, um eine adäquate Beschreibungsform zur Verfügung zu stellen, die den Prozess der gemeinsamen Untersuchung und Dokumentation von fachlichem Hintergrundwissen möglichst gut unterstützt. Dabei steht im Gegensatz zu den entwickelten informatischen Modellen nicht die rechnergestützte Wissensverarbeitung, d.h. die Definition der zu verwaltenden Daten, Informationen oder gar Datenstrukturen im Vordergrund, sondern deren anwenderorientierte, d.h. kognitiv adäquate Darstellung.

KoA$_{\text{L}}$A bietet hier eine einfache, flexible und mächtige Notation an, die den Prozess der gemeinsamen Erhebung, Diskussion und Dokumentation von Anwendungswissen fördert bzw. erst ermöglicht. Nicht die informationstechnische Erfassung, Abbildung oder Formalisierung von Konzepten des Anwendungsbereichs steht im Vordergrund, sondern deren inhaltlich sprachliche Definition.

Zentrales Modellierungskonzept im Wissensmodell von KoA$_{\text{L}}$A ist daher der *Begriff*:

> **Begriff** = sprachliche Repräsentation eines mentalen Konzepts, das konkrete oder abstrakte Gegenstände oder Phänomene anhand ihrer Merkmale umschreibt.
> Beispiele: *Baum*, *Farbe*, *Verb*

Ein Begriff stellt eine gedankliche Einheit oder Erkenntnisform dar, die eine bestimmte Kategorie (oder auch "Familie", "Gattung", "Art", "Typ", "Klasse" oder "Sorte" genannt) von Phänomenen umschreibt (siehe hierzu auch [Rei91]). Zugrundeliegendes elementares Erkenntnisprinzip ist dabei die Analogiebildung. Ein Begriff kategorisiert ähnliche Erfahrungsinhalte und abstrahiert sie zu mentalen "Ideen". Eine solche Abstraktion ist unabänderliche Voraussetzung für jede Verständigung, jedes Erkennen oder jedes logische Schlussfolgern.

Die Bedeutung eines Begriffes wird entweder durch Angabe von Beispielen (Instanziierungen) verdeutlicht[6] oder durch Bezugnahme auf andere Begriffe eingegrenzt, indem alle im Arbeitskontext relevanten *Merkmale* (Aspekte) des zu definierenden Konzepts unter Zuhilfenahme bereits eingeführter Begriffe spezifiziert werden. Merkmale drücken auf diese Weise Beziehungen zu anderen Begriffen aus. Nach [KM96] spezifizieren sie sogenannte "Zusammenhänge". Es werden dabei nicht nur solche Aspekte aufgenommen, die evtl. später Bestandteil von zu realisierenden Datenstrukturen werden, sondern alle Aspekte, die in der anwendungsorientierten Modellierung des Wissensbereichs Relevanz besitzen. Das sind insbesondere alle Aspekte, die zum fachlichen Verständnis des Anwendungsbereichs hilfreich sind.

In der Wissensmodellierung von KoA$_{\text{L}}$A wird nicht wie in einer Datenmodellierung üblich zwischen komplexen und elementaren Konzepten wie z.B. bestimmte vorgegebene Datentypen unterschieden. Die Abstraktionsebene ist vielmehr im weitesten Sinne offen. Je nach Modellierungskontext könnte beispielsweise das Konzept "Adresse" weiter konkretisiert werden zu

[6] Was oft wesentlich einfacher ist als ein länglicher sprachlicher Definitionversuch, der nach Maßgabe des hermeneutischen Zirkels im Grunde eigentlich endlos ist; siehe z.B. [Fal98] S. 193

"Name", "Strasse", "Hausnummer" oder aber völlig offen bleiben, d.h. als gegeben bzw. vordefiniert vorausgesetzt werden. Es besteht kein Zwang zur vollständigen Modellierung. Tiefe und Umfang der Modellierung ist vom Kontext abhängig, d.h. von den an der Modellierung Teilnehmenden, von ihrer Sprache etc.

> **Merkmal** (eines Begriffs) = bestimmte Eigenschaft, die die Begriffskonzeption selbst oder alle Elemente seines Gegenstandsbereichs auszeichnet. Setzt ein Konzept in eine bestimmte semantische Beziehung zu anderen Konzepten.
> Beispiele: "Ein *Baum* besitzt einen *Stamm*" , "*Blumen* sind *Pflanzen*", "*Pflanzen* sind normalerweise *grün*", "die *Universität Ulm* bietet den Studiengang *Informatik* an", "*Ulm* ist eine *Stadt*" etc.

Konzepte können also durch Rückgriff auf bereits eingeführte Begriffe näher spezifiziert werden[7]. Diese Beziehungen zwischen Konzepten können dabei folgende ausgezeichnete Qualitäten besitzen:

- typisierend oder instanziierend: das zugeordnete Merkmalskonzept ist als Kategorie bzw. Typ (Bsp.: "Die *Universität Ulm* hat einen *Kanzler*") oder aber als Konzeptausprägung bzw. Wert (Bsp: "*Pflanzen* sind normalerweise *grün*") zu interpretieren
- deskriptiv oder definitorisch (bzw. Typ- oder Instanzmerkmal): das Merkmal ist dem Konzept selbst (Bsp.: "*Ulm* hat eine *Universität*") oder aber seinen Ausprägungen zugeordnet (Bsp.: "Ein *Baum* besitzt immer einen *Stamm*")[8]
- ein oder mehrwertig: einem Konzept können als Merkmal ein oder mehrere andere Konzepte zugeordnet werden (Bsp.: "Ein *Baum* hat genau einen *Stamm"*, "Ein *Baum* hat normalerweise mehrere Äste")
- parametrisiert oder nicht: ein Konzeptmerkmal kann funktional von u.U. mehreren anderen Konzepten abhängen. Bsp.: "Ein *Baum* hat (zu einer bestimmten *Jahreszeit*) *Blätter*", "Ein *Arzt* hat (an einem bestimmten *Tag* zu einer bestimmten *Zeit)* einen bestimmten *Bereitschaftsdienst"*)
- attributiert oder nicht: Beziehungen selbst lassen sich auch als Konzept auffassen, über das weitere Aussagen getroffen werden können. Bsp.: "Ein *Kunde* bestellt *Produkte* (zu einem bestimmten *Preis* und in einer bestimmten Menge)"
- optional oder obligatorisch:
  Ein definitorisches Merkmal ist optional, wenn es keine konstante Eigenschaft beschreibt, die für alle Konzeptausprägungen zwingend zutrifft. Dies kann z.B. für prototypische

[7] Merkmale dokumentieren genau die von uns Menschen *bemerkbaren* Eigenschaften, d.h. Eigenschaften, welche die Dinge in Bezug auf uns und unsere kontextabhängige Wahrnehmung besitzen. Erst dieser Bezug zu uns macht sie zu begreifbaren mentalen Konzepten. Es gibt für uns kein "unbedingtes" Ding an sich; erst der Bezug zu uns formt diese Begriffe.

[8] definitorische Merkmale könnten auch noch in notwendige oder hinreichende Merkmale unterschieden werden. Bsp: Die Tatsache, dass eine Lehrveranstaltung einen Dozenten hat, ist zwar notwendig (sonst wäre es keine Lehrveranstaltung); die Tatsache alleine, das einem Gegenstand ein Dozent zugeordnet ist, heißt allerdings noch lange nicht, das dieser Gegenstand eine Lehrveranstaltung sein muss. Es könnte sich z.B. auch um einen Lehrstuhl handeln. Diese verfeinerten Qualitäten bleiben im Wissensmodell von $\text{KOA}_{\text{L}}\text{A}$ jedoch zur Vereinfachung unberücksichtigt. Das Ziel in der Modellierung von Konzepten ist in der Regel das Finden von definitorischen und notwendigen Merkmalen, die in Kombination mit einer freisprachlichen textuellen Beschreibung eine hinreichende Begriffsdefinition ergeben.

Begriffsdefinitionen verwendet werden, in denen charakteristische Merkmale definiert werden, die zwar für die meisten Gegenstände der Extension erfüllt sind, jedoch nicht unbedingt für alle. Gegenstände der Extension können in solchen Merkmalen also von der intrinsischen Begriffsdefinition abweichen. Bsp.: "Ein *Vogel* kann normalerweise *fliegen*, ein *Strauß* aber nicht", "*Bäume* besitzen normalerweise *Blätter*, *Nadelbäume* jedoch nicht". Fast alle begrifflichen Konzeptionen natürlichen Ursprungs sind prototypischer Natur ("Baum", "Sommer", "Fluss"). Definitorische Merkmale sind hingegen obligatorisch, wenn sie statische Eigenschaften festlegen, die für alle Instanzen der Extension zwingend zutreffen.
Deskriptive Merkmale sind dann optional, wenn sie eine dynamische bzw. temporäre Veränderlichkeit besitzen. Sie sind obligatorisch, sofern das Merkmal eine statische bzw. konstante Eigenschaft darstellt.

Es existieren folgende vordefinierte Beziehungs- bzw. Merkmalstypen:

- Ist-Teil-Von: Ein Konzept ist in einem anderen Konzept enthalten (Aggregation, Verfeinerung). Beispiel: "Abteilungen sind Bestandteil von Krankenhäusern"
- Ist-Ein: Ein Begriff ist ein Unterbegriff bzw. eine Spezialisierung eines anderen abstrakteren Oberbegriffs, wenn seine Extension Teil der Extension des Oberbegiffs ist. Instanzmerkmale des Oberbegriffs werden vom Unterbegriff geerbt.
  Beispiel: Ein Tier ist ein Lebewesen

  Es können folgende Sonderfälle unterschieden werden

  – Dynamische Rollenbeziehung können über Spezialisierungen mit der Qualität "optional" ausgedrückt werden. Bsp: "Ein Professor kann ein Dekan sein". Optionale Spezialisierungen drücken im Allgemeinen vor allem dynamische, d.h. zeitlich variabler Spezialisierung aus. Sie können jedoch auch für statische Spezialisierungsbeziehungen verwendet werden ("Eine *Person* kann ein *Arzt* sein") und daher zusätzlich oder alternativ als inverse Beziehung zu jeder obligatorischen Spezialisierung eingesetzt werden ("Ein *Arzt* ist eine *Person*").

    Dieser Beziehungstyp wird bei den semantischen Netzen auch als "may be a" im Gegensatz zu dem "is a" der normalen Spezialisierungsbeziehung bezeichnet [Rei91] (siehe hierzu auch [Per89]).

  – Beispielinstanzen können über Spezialisierungen mit der Qualität "Typmerkmal" ausgedrückt werden: Ein Begriff ist eine Instanz eines anderen Begriffs, wenn es direkt ein Element seiner Extension darstellt. Damit sind auch Konzepte höherer Ordnung möglich.
    Beispiel: *Hund*, *Katze*, *Maus* sind Beispiele bzw. direkte Instanzen einer *Tierart*

Die Entscheidung darüber, was als Merkmal und was als Begriff modelliert wird, ist im Allgemeinen diskursabhängig. So kann die Tatsache, dass Kunden in der Regel Produkte kaufen, sowohl als Merkmal eines Kunden ("*Kunde* kauft *Produkte*") oder aber als eigenständiger Begriff *Einkauf* modelliert werden.

Eine echte Teilmenge des Wissensmodells eines bestimmten Anwendungsbereichs sind die organisatorischen Einheiten, Rollen und Akteure des Organisationsmodells (siehe Abschnitt

4.3.1), die auch selbst wieder als Begriffe Gegenstand einer Interaktion sein können (Bsp: "*Sekretärin* reserviert *Raum* für einen bestimmten *Professor*").

Das Konzeptdiagramm

Das Wissensmodell kann in KoA$_{LA}$ sowohl textuell (siehe Kap. B.1) als auch zur Übersicht in Anlehnung an semantische Netze in Form eines sogenannten *Konzeptdiagramms* dargestellt werden. Während in der Projektdatenbank neben einer vollständigen textuellen Dokumentation aller Merkmale auch eine ausführlichere fachliche Definition der Begriffe zu finden ist, dient das Konzeptdiagramm vor allem der Übersicht und beschränkt sich daher im Allgemeinen auf eine Darstellung der wichtigsten Merkmale.

Abb. 4.1 zeigt als Beispiel ein Konzeptdiagramm aus dem Bereich der Kostenkalkulation von Hilfskraftverträgen im universitären Umfeld.

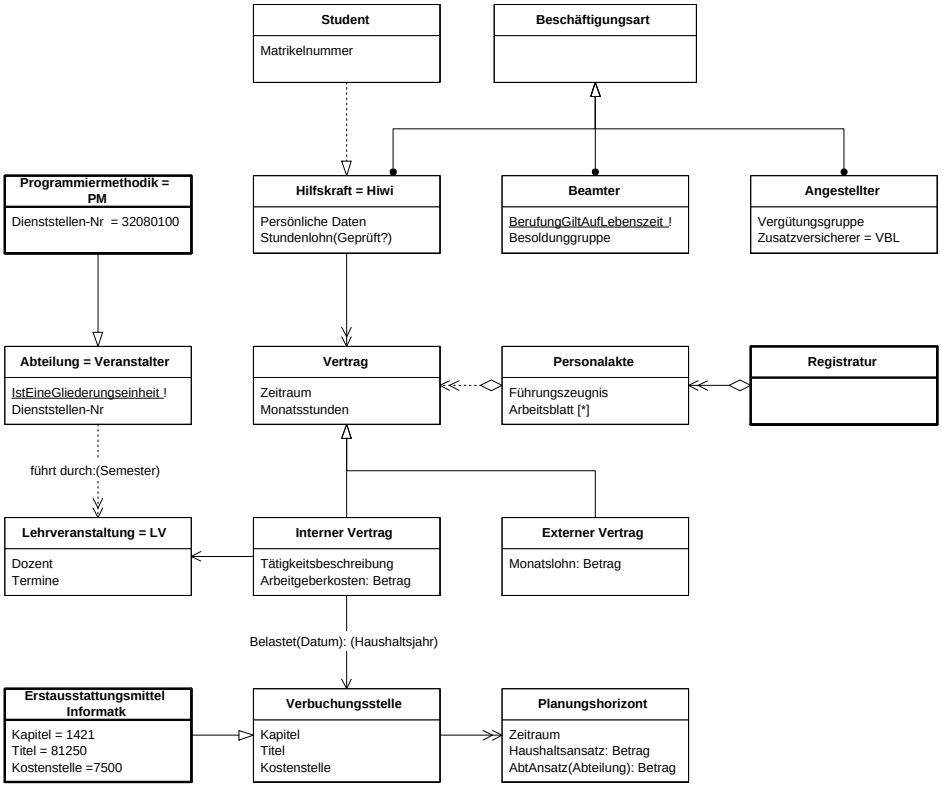


Abbildung 4.1.: Konzeptdiagramm für ein Wissensmodell aus dem Bereich der Kalkulation von Hilfskraftverträgen im Universitätsumfeld

Ein Konzept wird durch einen Begriff repräsentiert (z.B. "Abteilung"), der eine Menge von Konzeptinstanzen umfassen bzw. umschreiben kann (z.B. die Abteilung "Programmiermethodik und Compilerbau" der Informatik-Fakultät in Ulm, abgekürzt "PM"). Ein Konzept wird als Rechteck dargestellt, in dem als erstes der entsprechende Begriff notiert wird, optional gefolgt von einer Auflistung seiner Merkmale. Fachliche Synonyme werden in der Schreibweise

"Begriff = Synonym = ..." mit aufgeführt (eine "Abteilung" wird in bestimmten Kontexten auch als "Veranstalter" bezeichnet). Weitere Merkmale können durch Beziehungspfeile zu anderen Begriffen ausgedrückt werden, die je nach Wertigkeit der Beziehung mit einer Pfeilspitze (bedeutet "eine Beziehung zu genau einer Instanz") oder zwei Pfeilspitzen (bedeutet "eine Beziehung zu einer oder beliebig vielen Instanzen") versehen sind. Optional können typisierende Beziehungen auch mit einer genauen Kardinalitätsangabe versehen werden.

Eine Spezialisierungsbeziehung wird generell durch einen Pfeil mit leerer Dreiecksspitze dargestellt. Handelt es sich wie im Beispiel beim "Beamten" um eine deskriptive Beziehung ("ist eine Beschäftigungsart"), also um ein Merkmal des Konzepts selbst, kann dies grafisch auf unterschiedliche Weisen ausgedrückt werden. Entweder wird der Instanz-Begriff unterstrichen oder dick umrandet (wie im Beispiel die "Abteilung") oder der Spezialisierungspfeil selbst trägt einen kleinen ausgefüllten Kreis am Ausgangspunkt des Pfeils (wie bei den Beispielausprägungen der "Beschäftigungsart").

Ist die Extension eines Begriffs hingegen eine echte Teilmenge der Extension eines abstrakteren Begriffs, so handelt es sich um eine definitorische Beziehung, die sowohl ohne ausgefüllten kleinen Kreis notiert wird, als auch von einem Konzept ausgeht, das weder dick umrandet noch unterstrichen ist. Der abstraktere Begriff am Ende des Beziehungspfeils wird auch *Oberbegriff* genannt.

Spezialisierungen können durchaus eine höhere Ordnung besitzen (Beschäftigungsart besitzt als Beispielausprägung das Konzept eines "Angestellten", der als Ausprägung wiederum konkrete Personen besitzt). Weiterhin sind auch beliebig überlappende Kategorisierungen erlaubt ("Beamter" ist sowohl ein konkretes Beispiel für eine "Beschäftigungsart", als auch eine Spezialisierung eines Oberbegriffs "Person"). Will man eine Rollenbeziehung ausdrücken, so wird eine solche dynamische und deskriptive Beziehung als optionale Spezialisierungsbeziehung ausgedrückt und damit als gestrichelter Spezialisierungspfeil dargestellt (ein bestimmter "Student" kann jederzeit eine Hilfskraft sein/werden).

Eine Aggregationsbeziehung wird durch einen Pfeil mit kleiner Raute am Anfang dargestellt. Im Beispiel findet sich hier z.B die "Personalakte", die potentiell mehrere Verträge beinhaltet. Es wird keine weitere Unterscheidung zwischen starker und schwacher, logischer oder physikalischer Aggregation etc. getroffen, wie sie z.B. in UML in Form von "Aggregation" versus "Composition" zu finden ist.

Ein Merkmal kann sowohl grafisch (durch Beziehungspfeile) als auch textuell notiert werden und besitzt neben einem Namen auch einen Zielbegriff, mit dem der zu beschreibende Begriff in Beziehung gesetzt wird. Je nachdem, ob es sich dabei um eine typisierende oder instanziierende Beziehung handelt, sprechen wir im Folgenden statt vom "Zielbegriff" auch vom "Typ" oder "Wert" eines Merkmals. Die Angabe des Typs bzw. des Namens kann entfallen, wenn der Begriff identisch zum Namen des Aspekts ist (im Beispiel 4.1: Merkmal "Dozent" einer Lehrveranstaltung; Merkmal "Lehrveranstaltung" eines Vertrags). Anonyme instanziierende Merkmale wären im Beispiel eines "Baumes" etwa "=Pflanze" oder "=Grün" (in der textuellen Schreibweise). Für Merkmale mit Wahrheitswert (bzw. einstellige Prädikate) kann auch folgende Kurzschreibweise verwendet werden: "Merkmal?" : äquivalent zu "Merkmal: Wahrheitswert" oder "Merkmal!": äquivalent zu "Merkmal: Wahrheitswert = Wahr" (z.B. beim Baum: "schwer!", "lebendig!").

Statische Merkmale können von funktionalen Merkmalen unterschieden werden. Statische

Merkmale (wie z.B. "ein Dozent führt eine Reihe von Lehrveranstaltungen durch") beschreiben einfache statische Eigenschaften eines Konzepts, während funktionale Merkmale komplexeres Wissen darstellen können wie z.B. die Information, dass ein Vertrag – je nach Laufzeit und Absprachen – unterschiedliche Haushaltsjahre belasten kann. Dieser Umstand wird im Beispiel durch eine funktionale Abhängigkeit des Merkmals "Verbuchungsstelle" von einem bestimmten Datum modelliert. Die Parameter von funktionalen Merkmalen werden gleich hinter dem Merkmalsnamen in Klammern gesetzt (im Beispiel: "AbtAnsatz(Abteilung):...") und zwar unabhängig davon, ob das Merkmal textuell im Konzept-Rechteck oder als eigener Beziehungspfeil dokumentiert wird (vgl. im Beispiel "Belastet(Datum):..." und "AbtAnsatz(Abteilung):...").

Merkmale können auch selbst eigene Attribute besitzen. Diese werden textuell direkt nach dem Doppelpunkt dokumentiert, der dem Merkmalsnamen folgt, und in Klammern gesetzt. Im Beispiel findet sich hier z.B. die Belastung einer Verbuchungsstelle mit einem Vertrag, die als Zusatzattribut noch das Haushaltsjahr trägt, indem die Kosten zu verbuchen sind.

Da bei der Modellierung die Bedeutung der Begriffe im Vordergrund steht und nicht ihre Repräsentation, sind im Besonderen natürlich auch redundante Merkmale erlaubt, d.h. solche, deren "Werte" allgemein von anderen Merkmalen abgeleitet werden könnten, wie z.B. der "Abteilungsansatz", der auch als Summe aller Vertragskosten einer Abteilung in einem bestimmten Zeitraum berechnet werden könnte.

Diskussion

Die Wissensmodellierung nach KoA$_{\text{L}}$A weist im Vergleich zur konzeptionellen Modellierung anderer Softwareentwicklungsmethoden (z.B. den Klassen und Objektdiagrammen von UML oder der semantischen Datenmodellierung in Form erweiterter ER-Diagramme) eine Reihe von Vereinfachungen auf.

Auf semantischer Ebene der Modellierungskonzepte:

- Es wird nicht zwischen abstrakten ("Klassen") und konkreten Konzepten ("Instanzen") unterschieden. Beide werden als sprachliche Begriffe modelliert, die in Beziehung zueinander stehen können – im Besonderen auch in einer deskriptiven Spezialisierungsbeziehung.
- Es wird nicht zwischen den Merkmalen ("Attributen") und den semantischen Beziehungen von Konzepten unterschieden. Ein Merkmal ist gleichwertig zu einer definitorischen Beziehung, ein "Attributwert" zu einer instanziierenden Beziehung.
- Es wird kein prozedurales Wissen bzw. Verhalten modelliert ("Methoden") sondern nur statisches Wissen.
- KoA$_{\text{L}}$As Wissensmodell gehört zu der Klasse von semantischen Datenmodellen, die sich in der Darstellung von Beziehungen aus Vereinfachungsgründen (Reduktion der Modellierungskonzepte) hauptsächlich auf (mehrwertige) Attribute abstützt und hierzu keine eigenen Typkonstruktoren wie z.B Relationen (in der semantischen Datenmodellierung auch "aggregation" genannt!) oder die Mengenbildung ("association") einführt[9].

[9] zur Diskussion dieser zwei grundlegenden Philosophien der semantischen Datenmodellierung siehe[HK90]

Dies führt sowohl zu einer Reduktion der benötigten Modellierungskonzepte als auch zu einer Verstärkung der viewpoint-Orientierung bzw. der kontextorientierten Ausgestaltung des Wissensmodells: durch Auswahl der entsprechenden (parametrisierten) Attribute lässt sich die Navigationsmöglichkeit entlang einer Beziehung so festlegen, das ihre semantische Bedeutung klarer zu Tage tritt.
Beispiel: Der Sachverhalt, dass ein Arzt ab und zu bestimmte Bereitschaftdienste abzuleisten hat, wird im Kontext der Erstellung von Bereitschaftsdienstplänen aus Anwendersicht am verständlichsten so modelliert, dass dem Arzt folgendes Merkmal zugeschrieben wird: `"Bereitschaftdienst(Datum): Abteilung"` Die Modellierung der entgegengesetzten Richtung von der Abteilung zu all seinen zugeteilten Bereitschaftsärzten kann unterbleiben, da sie im gegebenen Kontext zur Vervollständigung des Anwendungswissens keinen bedeutenden Beitrag leistet. Mit der attributbasierten Beziehungsmodellierung geht ein höherer Freiheitsgrad einher, auch in Bezug auf eine gewollt unvollständige Modellierung.

- Der Abstraktionsgrad der Modellierung ist offen. Es gibt keine atomaren Konzepte ("Datentypen"), bis zu denen die Konzepte zu konkretisieren sind. Merkmale repräsentieren Wissen über Konzepte und dienen nur der Definition der Begriffsbedeutung, nicht der Spezifikation seiner Repräsentation (z.B. als Informationsstruktur). Die Definition wird nur so weit getrieben, wie sie für das subjektive und kontextabhängige Verständnis notwendig ist.

KoA$_{\text{L}}$A erweitert die Ausdrucksmächtigkeit der Modellierung in folgenden Punkten:

- In der Spezialisierung werden optionale und obligatorische Beziehungen unterschieden, so dass nun prototypische Begriffsdefinitionen möglich sind und echte Rollenbeziehungen darstellbar werden;
- Es werden Aspekte eines funktionalen Datenmodells inkorporiert (multi-argument, multi-valued attributes) [HK90];
- Typ- und Instanzebene können zur Darstellung von explizitem Wissen beliebig vermischt werden. Dabei wird gleichzeitig auch die Metamodellierung unterstützt, beispielsweise in Form von Konzepten, die andere abstrahierende Konzepte als Ausprägungen besitzen.

KoA$_{\text{L}}$As Wissenmodell dient im Gegensatz zu den in der Softwareentwicklung verwendeten Modellierungstechniken der reinen konzeptuellen Modellierung. Gegenstand der Beschreibung ist nicht die formale Spezifikation von Informationsinhalten einer Applikation, sondern eine weitgehend semiformale Definition der Begrifflichkeit des Anwendungsbereichs.

Die Modellierung von KoA$_{\text{L}}$A beginnt eine Ebene höher in der Begriffswelt der Anwender. Begriffe erhalten ihre Bedeutung als sprachliches Konstrukt aus ihrer Benutzung in einem bestimmten Kontext, der zu dokumentieren ist. Zur Definition neuer Begriffe werden ihre charakteristischen Merkmale dokumentiert, die mit Hilfe bereits definierter Begriffe beschrieben werden (->Hermeneutischer Zirkel). Die Transformation in Daten bzw. Objekte (Unterscheidung von Objekttypen, Attributen, Datentypen, Assoziationstypen, Kardinalitäten, Invarianten etc.) ist bereits ein Stück Entwurf und wird der SW-Spezifikation überlassen (siehe Kap. 5.8). KoA$_{\text{L}}$A beschränkt sich auf die Erfassung und Standardisierung der sprachlichen Begriffswelt.

4.3. Das Organisationsmodell

Das Organisationsmodell dient der Beschreibung der statischen Organisationsstrukturen eines Handlungssystems. Beschrieben werden die abstrakten oder konkreten, menschlichen oder technischen Protagonisten des betrachteten soziotechnischen Systems, ihre gegenseitigen Beziehungen und ihr Aktivitätsrepertoire. Ein Organisationsmodell definiert die globale Aufbauorganisation eines Handlungssystems und unterteilt sich in zwei Submodelle:

In der *Organisationsstruktur* wird der statische Aufbau der Organisation beschrieben, während mit den *Tätigkeitsprofilen* die dynamischen Aspekte der einzelnen Organisationseinheiten dokumentiert werden.

4.3.1. Organisationsstruktur

In der Organisationsstruktur werden alle Akteure des betrachteten Handlungssystems und ihre strukturellen Beziehungen zueinander beschrieben.

Eine Organisation ist in sogenannte *Organisationseinheiten* (OE) gegliedert, die von folgender Art sein können:

- abstrakte Organisationseinheiten, die aus anderen Organisationseinheiten aufgebaut sind. Beispiele: Unternehmen, Betrieb, Abteilung, Unterabteilung, Projektgruppe/Team, Ausschuss, Gremium, Filiale, Niederlassung, Rechenzentrum etc.
- konkrete Personen, die bestimmte Stellen übergeordneter OE besetzen können (bzw. von ihnen aggregiert werden)
- konkrete Elemente der technischen Infrastruktur. Elemente können dabei aktive technische Systeme oder Geräte sein (wie z.B. Informationssysteme, Webserver, Faxgerät, Drucker) oder aber auch passive Einrichtungsgegenstände physischer (wie z.B. Postfach, Schwarzes Brett, Aktenarchive) oder konzeptioneller Art (wie z.B. bestimmte Datenbestände etc.).

Da alle Organisationseinheiten im betrachteten Handlungssystem auch als aktiv handelnde Einheiten oder Interaktionspartner in Erscheinung treten können, werden sie im Folgenden auch als Akteure bezeichnet. Organisationseinheiten können ganze Unternehmen, einzelne Abteilungen, bestimmte, wirklich Personen oder technische Systeme sein. Sie sind die handelnden Akteure in der betrieblichen Realität.

Akteure werden ähnlich zu den Begriffen im Wissensmodell als sprachliche Konzepte modelliert. Ein Akteur lässt sich damit auch als Konzept definieren, das in "Interaktions"-Beziehungen zu anderen Konzepten steht.[10]

Im Allgemeinen wird das betrachtete Handlungsystem als Instanz beschrieben, die andere abstrakte oder konkrete Organisationseinheiten bzw. Akteure aggregiert und verfeinert. Akteure können dabei typisiert werden, d.h. in einer Spezialisierungsbeziehung zu einem anderen Akteurstyp stehen, oder bestimmten Rollen zugeordnet werden, d.h. in einer entsprechenden

[10] Die Interaktion tritt in KOA_{L}As Modellen damit als besonders ausgezeichneter Beziehungstyp in Erscheinung (siehe Kap. A.8), wird aber erst im Kooperationsmodell entsprechend thematisiert (siehe Kap. 4.4).

optionalen Spezialisierungsbeziehung stehen. Rollen explizieren bestimmte Aufgabenzusammenhänge, in denen ein Individuum im Laufe seiner Arbeitsaktivitäten stehen kann. Diese können u.U. natürlich auch mehrfach angenommen werden. Ein bestimmter Angestellter kann z.B. im Laufe seiner Tätigkeit Abteilungsleiter, Projektleiter, Geschäftsführer werden bzw. auch gleichzeitig Betriebsrat sein; ein bestimmtes Kombigerät kann zugleich als Telefon wie auch als Fax benutzt werden etc.

Stellen werden durch typisierende Aggregationsbeziehungen zu Akteurs-Typen oder -Rollen modelliert. Die Kardinalität der Aggregationsbeziehung verdeutlicht dabei die Anzahl der vorhandenen Stellen.

Im Modell werden sowohl die Typ- wie auch die Instanzebene gemischt berücksichtigt. Die Modellierung der Organisationsstruktur ist normalerweise weitgehend offen, d.h. dass die Organisationseinheiten nur so weit verfeinert werden, wie es zum Verständnis oder Beschreibung des Handlungsystems nützlich erscheint.

Eine Organisationsstruktur zeichnet sich auch durch weitere allgemeinere Beziehungen aus, die zwischen Akteuren des Handlungssystems feststellbar sind. Als Beispiel seien hier die Weisungsbefugnis oder die Personalverantwortung genannt ("Vorgesetzter von", "hat Personalverantwortung für") .

Das Konzeptdiagramm

Organisationsstrukturen werden in Form eines Konzeptdiagramms dargestellt, das sich auf Organisationseinheiten bzw. Akteure und deren Typen, Rollen, Merkmale bzw. Beziehungen beschränkt.

Menschliche Akteure oder Organisationseinheiten werden dabei in der Regel als grafische Strichmännchen dargestellt. Handelt es sich hingegen um einen aktiven technischen Akteur, wird ein Dreieck zu Darstellung verwendet. Passive Einrichtungsgegenstände wiederum werden als auf der Spitze stehende Quadrate notiert.

Abb. 4.2 stellt als Beispiel die Aufbauorganisation der Fakultät Informatik der Universität Ulm dar. Die Abstraktionsebene wird so gewählt, wie sie zur Untersuchung des zu gestaltenden Tätigkeitsbereich sinnvoll erscheint. Im Beispiel soll das Handlungssystem im Kontext der Ankündigung von Kolloqiumsvorträgen untersucht werden:

> Hierbei sind im Besonderen Herr Prof. K., der die Vortragsreihe organisiert und die Referenten requiriert, und Frau G., die als Dekanatssekretärin für die offizielle Bekanntgabe der Vorträge verantwortlich zeichnet, als Individuen interessant.
>
> Über die Vorträge selber müssen alle Angehörigen der Fakultät informiert werden, worunter im Besonderen auch die Studenten zu zählen sind. Die Kommunikation innerhalb der Fakultät findet sowohl in elektronischer Form mittels des eigenen Web/Email-Servers, als auch in altbewährter schriftlicher Form mittels eines zentralen Verteilers (dem "Postschrank") statt.
>
> Als direkter Veranstalter der einzelnen Vorträge treten diverse Professoren oder Assistenten in Erscheinung, die sich um die einzelnen Vorträge bzw. Referenten zu kümmern haben.

Das Konzeptdiagramm dient wie beim Wissensmodell nur der Übersicht. Eine ausführlichere Beschreibung der Akteure und ihrer Aufgaben findet sich zum Einen in der Projektdatenbank, zum Anderen im entsprechenden Tätigkeitsprofil (siehe Kap. 4.3.2).

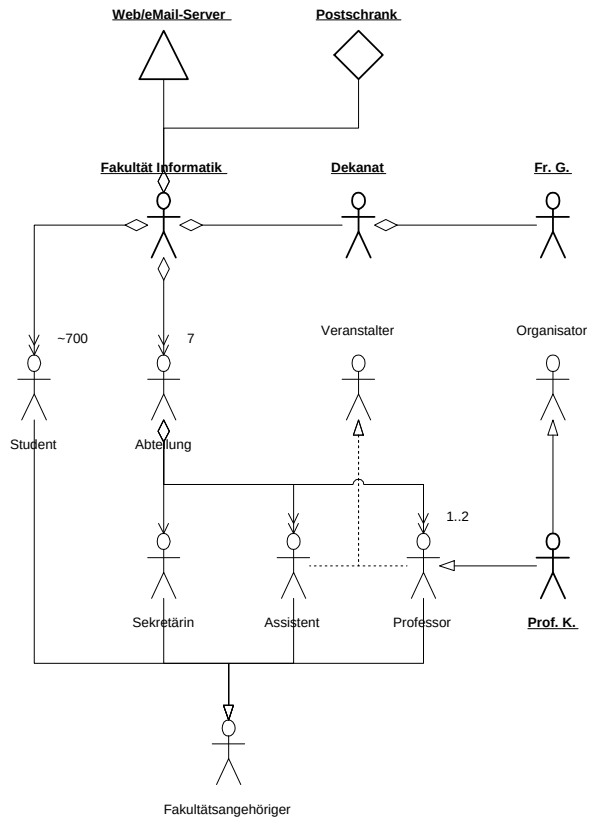


Abbildung 4.2.: Konzeptdiagramm für die Organisationsstruktur der Fakultät Informatik

Einzelne Organisationseinheiten lassen sich natürlich auch in Form weiterer Diagramme verfeinern. Zur Vereinfachung der Darstellung bietet es sich an, reale Personen oder technische Systeme durch Rollen oder Typen zu generalisieren. Im Gegenzug kann es zur Veranschaulichung sinnvoll sein, für den Kontext besonders relevante Stellen, Rollen oder Typen durch konkrete Akteure zu instanziieren.

4.3.2. Tätigkeitsprofile

In KoA$_{\text{L}}$As Tätigkeitsprofilen wird das Handlungsrepertoire der in der Organisationstruktur identifizierten organisatorischen Einheiten beschrieben. Sie definieren das Aktivitätspotential selbstgesteuerter (z.B. menschlicher) Akteure, das Leistungsprofil fremdbestimmter Akteure (z.B. technische Systeme) oder das Anforderungsprofil passiver Infrastrukturgegenstände in ihrem situativen Kontext (Anforderungen, Interaktionen mit anderen Akteuren, Randbedingungen etc.). Im Falle eines menschlichen Akteurs können sie im einfachsten Fall eine Stellenbeschreibung darstellen, im Falle einer abstrakten Einheit wie z.B. einer Abteilung oder eines ganzen Betriebs können sie aber auch einer abstrakten Beschreibung ihrer Geschäftsziele dienen.

Die Tätigkeitsprofile aller aktiven Akteure zusammen genommen stellen die zweite Komponente bzw. ein weiteres Submodell des Organisationsmodells dar.

Ein Tätigkeitsprofil beschreibt einen Akteur aus einer intentionalen Sicht heraus. Im Gegensatz bzw. in Ergänzung zu einer operationalen Beschreibung konkreter Tätigkeitssequenzen, wie es Gegenstand der Kooperations-Modellierung (siehe nächster Abschnitt) sein wird, werden in KoA$_L$A auch die Hintergründe bzw. der motivationale Kontext seiner Aktivität erfasst. Tätigkeitsprofile beschreiben das Handlungs-"Repertoire" der in der Organisationsstruktur identifizierten Akteure in deskriptiver implizierter Form, indem charakteristische Aktivitätsbereiche, Motive, Anforderungen, Randbedingungen und Ziele der handelnden Akteure dokumentiert werden. Die verschiedenen Aktivitäten der organisatorischen Einheiten stehen dabei in Aggregations- oder Spezialisierungsbeziehungen und werden durch *Bedingungen* beeinflusst.

Die der Modellierung zugrunde liegenden zentralen Konzepte sind die der *Aktivität* und die der *Bedingung*. Bedingungen und Aktivitäten werden ähnlich zu den Methoden der Zielstruktur-Analyse (siehe z.B. [AMP94, KL99, DvLF93]) in eine Baumstruktur gebracht, die den motivationalen Kontext der Akteurstätigkeiten deutlich macht.

Eine Aktivität steht für einen bedeutsamen Tätigkeitsbereich eines Akteurs. Eine Aktivität ist charakterisiert durch den oder die beteiligten Akteure, die Aktivitätsziele, die Randbedingungen der Aktivität und die während der Durchführung der Aktivität beobachtbaren Interaktionsbeziehungen mit anderen Akteuren. In der Beschreibung der Aktivitäten können auch weitere Merkmale definiert werden wie z.B. die Häufigkeit, die zeitliche Dauer oder die Frequenz der Aktivitätsausübung.

Eine Aktivität kann eine (menschliche) Handlung oder auch eine (maschinelle) Operation sein. Eine Aktivität umfasst im Normalfall eine große Menge potentieller Tätigkeiten, die zum Aktivitätsziel führen können. Aktivitäten lassen sich durch Unteraktivitäten verfeinern oder in Varianten spezialisieren. Die Modellierung beschränkt sich auf die charakteristischsten Aktivitätsbereiche und Randbedingungen. Aktivitäten werden dabei nur soweit verfeinert, wie es zum fachlichen Verständnis der Tätigkeit notwendig erscheint. Alle Sachverhalte, die die Durchführung einer Aktivität beeinflussen können, werden als *Bedingungen* modelliert. Hierunter fallen Motive, Anforderungen, Vereinbarungen, Voraussetzungen, Probleme oder Sonderfälle.

Die Definition einer Aktivität muss selbst keine Beschreibung ihrer konkreten Realisierung enthalten. Im Grunde wird mit einer KoA$_L$A-Aktivität darum eigentlich ein Aktivitätstyp bzw. ein operationalisierbares Ziel definiert, dem in der Realität eine Menge möglicher Tätigkeitssequenzen ("Szenarien") zugeordnet werden können. Wir reden darum im Folgenden statt von Aktivitäten auch oft von "Aktivitätsbereichen". Konkrete Ausprägungen einer Aktivität sind in KoA$_L$A erst Gegenstand von Aktivitätsmustern des Kooperationsmodells (s. Kap. 4.4.3).

Eine Konzentration auf die Aktivitätsziele fördert die Vollständigkeit in der Modellierung. Statt sich auf konkrete Ausprägungen einzelner Tätigkeiten der Arbeitspraxis zu konzentrieren, werden sie zu Aktivitätsbereichen zusammengefasst, die wiederum zu Unterbereichen verfeinert werden können. Letztendlich stützt sich jede Beschreibung menschlicher Aktivität auf vordefinierte Handlungskonzeptionen ab, die wiederum selbst situiert und damit nicht allgemeingültig festzulegen sind[11].

[11] Als Beispiel kann hier im Grunde jede menschliche Aktivität des Alltagslebens angeführt werden. Eine Beschreibung solcher - sich im Normalfall nie exakt wiederholbarer - Aktivitäten lässt sich im Grunde nie

Umgekehrt lassen sich aber viele Tätigkeiten durch Benennung ihres übergeordneten Zieles zusammenfassen und man kann ausgehend von globaleren Aktivitätsbereichen leicht andere Aktivitätsbereiche identifizieren, die wiederum Teilmengen enthaltener Tätigkeitssequenzen darstellen.

Nicht alle Ziele sind operationalisierbar (wie z.B. ein Wunsch "Kosten zu sparen" oder "die Bearbeitungszeit zu minimieren"). Diese gehen als *Bedingungen* in die Modellierung mit ein. Derartige Randbedingungen sind z.B. Regeln, Anforderungen, Vereinbarungen oder auch subjektive Wünsche oder Prinzipien, die in den dargestellten Aktivitäten berücksichtigt werden sollen oder müssen.

Besonders wichtige Randbedingungen sind auch unausgesprochene informelle Verträge, für deren Präzisierung im Lauf der Modellierung eine Konsensbildung notwendig ist und die dann in Form von offiziellen Anforderungen in die Modellierung aufgenommen werden können. Die Modellierung unterstützt damit eine Analyse der informellen sozialen Systeme, deren Berücksichtigung bei der Systemgestaltung von wesentlicher Bedeutung für eine effektive Arbeitsumgebung sind.

Eine andere Klasse von Bedingungen sind solche Sachverhalte, die Aktivitäten auslösen bzw. motivieren. [Dah97] unterscheidet in seiner Systemanalyse menschlichen Handelns hier Motiv, Ziel und Handlung. Das Motiv ist die Ursache für eine konkrete Zielsetzung, die zur Auslösung einer Handlung führt, welche zum Ziel führen und damit das Motiv befriedigen soll. Motive führen also neben Zielen noch ein weitere Abstraktionsebene in die Modellierung ein und tragen so zur Vervollständigung der Modellierung des menschlichen Handlungsrepertoires bei.

Neben Randbedingungen und Motiven einer Tätigkeit werden in KoA$_{\text{L}}$A auch noch *Problemsituationen*, die Ursache oder Folge einer Aktivität oder Bedingung sind, und *Sonderfälle*, die in den Aktivitäten besonders zu berücksichtigen sind, als weitere ausgezeichnete Bedingungen modelliert.

Allgemein beschreiben Bedingungen Sachverhalte, die zur Auslösung einer Aktivität beitragen, diese Aktivität in ihrem Ablauf beeinflussen oder durch die Aktivität selbst verursacht werden.

Beispiele für Bedingungen, die den Ablauf einer Aktivität beeinflussen, indem sie die Menge der möglichen Tätigkeitssequenzen festlegen bzw. einschränken sind beispielsweise:

- auf gesellschaftlicher Ebene: Konventionen, Gesetze, Umgangsformen, Rollen
- auf geschäftlicher Ebene: "Business Rules", Tarifverträge, firmenpolitische Richtlinien
- auf persönlicher Ebene: Prinzipien, Vorsätze, Werte, Vereinbarungen

Zusammenfassend können im Tätigkeitsprofil folgende Arten von Bedingungen gesondert gekennzeichnet werden:

Motiv Ein Motiv ist eine Bedingung, deren Ursprung der Akteur selbst ist. Beispiele:

- Geschäftsziele eines Unternehmens ("business goals")
- selbstgestecktes Ziel eines menschlichen Akteurs

in seiner konkreten Ausprägung festlegen, sondern nur in Form eines Handlungsmusters oder aber noch abstrakter in Form einer Zieldefinition ("Einkaufen", "Telefonieren", "Einen Brief schreiben" etc.).

- ein subjektiver Wunsch eines Akteurs

Anforderung Eine Anforderung ist eine Bedingung, deren Ursprung ein externer Akteur ist. Dies kann zum Beispiel eine offizielle Aufgabe oder interne Zielvorgabe als Bestandteil einer Stellenbeschreibung sein, oder eine spezifische Randbedingung, die von einer anderen organisatorischen Einheit eingefordert wird.

Problem Ein Problem ist ein Sonderfall eines Motivs. Problematische Tatbestände, die durch eine Aktivität entstehen können bzw. kausale Ursache für eine Aktivität sind. Kann durch ein Ausrufezeichen annotiert werden.

Sonderfall Sonderfälle sind Bedingungen, die eine Abweichung von der Regel darstellen und daher im Gestaltungsprozess gesondert berücksichtigt werden müssen. Sie werden auch als "Ausnahmen" bezeichnet. Sonderfälle führen in der beeinflussten Aktivität in der Regel zu einer expliziten "Ausnahmebehandlung". Kann durch ein Fragezeichen-Icon annotiert werden.

Das Profildiagramm

Tätigkeitsprofile werden sowohl textuell in der Projektdatenbank dokumentiert als auch grafisch in Form sogenannter Profildiagramme verdeutlicht.

Als Beispiel stellt Abb. 4.3 ein Tätigkeitsprofil der Personalabteilung III-2 der Universität Ulm dar. Ein wesentlicher Aufgabenbereich dieser Abteilung sind die von den einzelnen Fakultäten unterhaltenen Beschäftigungsverhältnisse bezüglich studentischer oder anderer Hilfskräfte. Für die Personalabteilung bedeutete dies im Besonderen, dass sie sowohl für die Kostenkalkulation und Genehmigung dieser Verträge zuständig ist, als auch für die Fakultäten Ansprechpartner für alle vertraglichen Belange ist.

Bedingungen werden im Profildiagramm als Textblöcke dargestellt, die mit geschweiften Klammern eingeschlossen sind und untereinander bzw. mit den beeinflussten oder beeinflussenden Aktivitäten durch gestrichelte *Wirkung*s-Pfeile verbunden sind. Optional können die Bedingungen dabei mit einer Kennzeichnung ihrer besonderen Art in Form einer Stereotyp-Kennzeichnung versehen werden (doppelte spitze Klammern wie in UML).

> Im Diagramm findet sich hier beispielsweise die *Anforderung* "Zuständig für die Hilfskraftangelegenheiten", die den grundlegenden und offiziell festgelegten Aufgabenbereich der Abteilung dokumentiert und sowohl andere Bedingungen impliziert (beispielsweise die Aufgabe der "Kostenkalkulation und Genehmigung") als auch Aktivitäten bewirkt bzw. motiviert (wie z.B. "Auskunft erteilen"). Eine andere Bedingung ist die interne Zielvorgabe der Abteilung, dass die Bearbeitung von Verträgen höchstens drei Wochen in Anspruch nehmen sollte. Diese Bedingung stellt ein *Motiv* dar, d.h. eine Anforderung, die die Abteilung sich selber stellt. Diese Bedingung beeinflusst den Aktivitätsbereich der Verwaltung der Beschäftigungsverhältnissen.

Durch Wirkungspfeile können logische Abhängigkeiten oder allgemeine Beeinflussungen zwischen Bedingungen, Akteuren und/oder Aktivitäten ausgedrückt werden. Die Art der Wirkung kann dabei optional durch eine textuelle Beschriftung deutlich gemacht werden.

> Im Beispiel "wünscht" sich ein Student etwa eine "pünktliche Auszahlung" seines Lohnes. Diese Bedingung wiederum beeinflusst die Auszahlungsaktivitäten der Personalabteilung, auch wenn sie wie im Beispiel (noch) keine offizielle Anforderung darstellt.

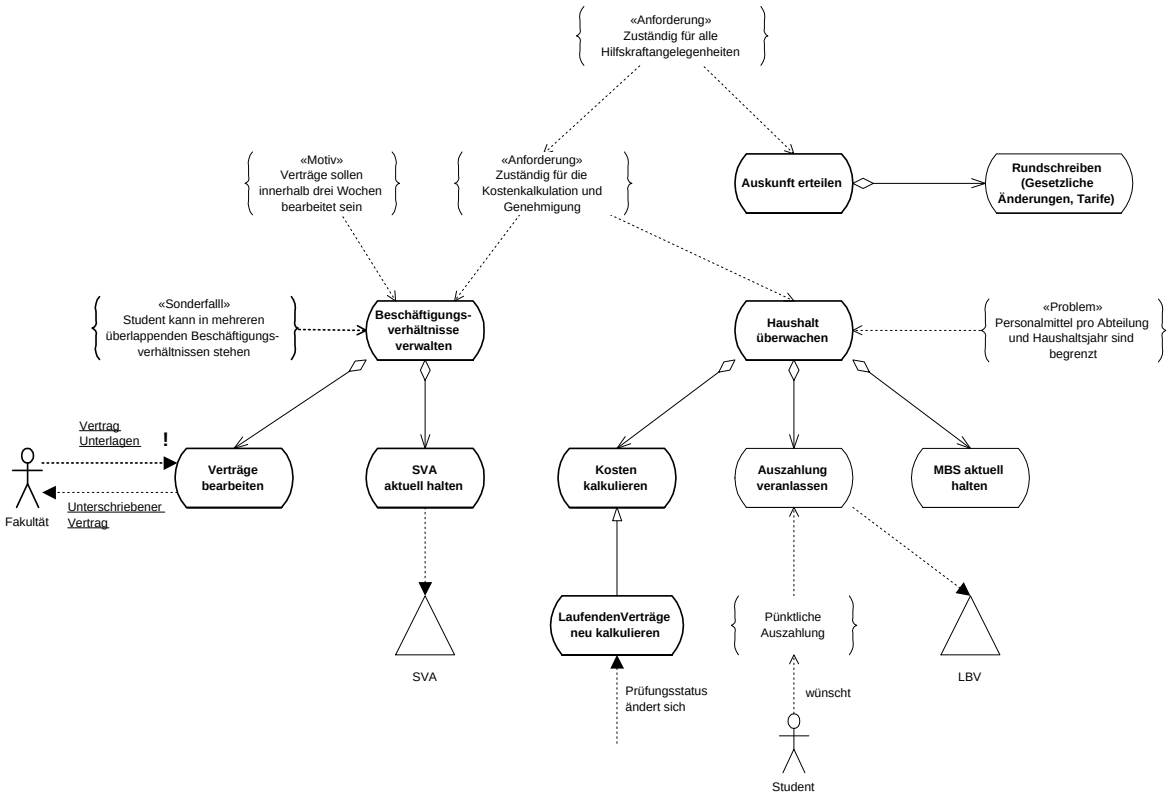


Abbildung 4.3.: Tätigkeitsprofil der Personalabteilung III-2

Aktivitäten werden im Profildiagramm ähnlich zu den “Activity States” von UML als Rechtecke mit abgerundeten Seiten dargestellt.

> Im Beispiel finden sich hierzu die Aktivität “Auskunft erteilen”, die von der Anforderung “Zuständig für alle Hilfskraftangelegenheiten” motiviert ist und noch weiter in die Aktivität “Rundschreiben” verfeinert wird.

Aktivitäten lassen sich generell über Aggregationsbeziehungen verfeinern oder auch durch Spezialisierungsbeziehungen zu Sonderfällen konkretisieren.

> Der Aktivitätsbereich “Kosten kalkulieren” kann z.B in der Form auftreten, dass die bereits einmal kalkulierten Kosten eines Vertrages erneut festgesetzt werden müssen, weil sich Vertragsbedingungen wie z.B. der Ausbildungsstand des Studenten ändern und dementsprechend seine Entlohnung anzupassen ist. Andere Aktivitätsbereiche der Personalabteilung sind die “Verwaltung von Beschäftigungverhältnissen” und die “Überwachung des Haushalts”.

Neben den Wirkungsbeziehungen zwischen Akteuren, Aktivitäten und Bedingungen werden in das Diagramm auch die wichtigsten Interaktionsbeziehungen zu externen Akteuren aufgenommen, die während der Durchführung der Aktivitäten zu beobachten sind. Interaktionen können dabei entweder in Folge der Aktivitätsdurchführung auftreten (ausgehende gestrichelte Interaktionspfeile) oder zur Durchführung einer Aktivität führen, dazu beitragen oder sie begleiten (eingehende gestrichelte Interaktionspfeile). Im Falle eines Auftrags umfasst die Aktivität nicht unbedingt auch die Aktivität der Durchführung des Auftrags. Falls der Pfeil allerdings ohne Beschriftung und nur mit einem Ausrufezeichen markiert ist, so modelliert

die Aktivität direkt die beauftragte Leistungserfüllung. In der Abbildung findet sich hierzu als Beispiel das "Bearbeiten von Verträgen" als Aktivität, die durch einen externen Auftrag veranlasst wird.

Zusätzlich zu Interaktionen mit anderen Organisationseinheiten können auch Interaktionen des betrachteten Akteurs mit sich selbst ausgedrückt werden (*interne* Interaktionen), indem der interaktionsauslösende oder -empfangende Akteur ungenannt bleibt. Dieser Formalismus kann z.B. zur Berücksichtigung von Erinnerungs- oder Erkenntnisprozessen (z.B. die Wahrnehmung von Ereignissen) ausgenutzt werden.

> Die Aktivität "Laufende Verträge neu kalkulieren" (s. Abb. 4.3) wird beispielsweise dann notwendig, wenn die Personalabteilung entdeckt, dass sich der Prüfungsstatus einer Hilfskraft ändert.

Bei der Modellierung dieses Ereignisses als interne Interaktion kann der Ursprung bzw. die eigentliche Quelle dieser Information bewusst unspezifiziert bleiben. Eine genauere Erläuterung der Interaktion als zentrales Modellierungskonzept von KoA$_{\text{L}}$A findet sich weiter unten in Kapitel 4.4.1.

Diskussion

Erkenntnisse der Arbeitspsychologie [Uli94] zeigen, dass die Aufgabe (im Englischen "task") das zentrale Konzept in der Beschreibung konkreter Arbeitssituationen ist. Dies lässt sich auch am Erfolg von Use-Cases oder Szenarien absehen, die ursprünglich als neues Konzept in aktuelle objektorientierte Analysemethoden eingeführt wurden und mittlerweile auch in viele andere Vorgehensweisen der Softwaretechnik Eingang gefunden haben. Eine Beschreibung der Systemnutzung in Form von Aktivitätsbeschreibungen ist der Vorstellungswelt von Anwendern und sonstigen Interessengruppen angemessener als eine technische Spezifikation von Systemen in Form von Objekten oder Datenflüssen.

Neben dieser eher operationalen Sichtweise haben sich in der Praxis allerdings auch weitergehende deskriptive Ansätze bewährt:

"*Experience in a number of case studies has convinced us that taking a functional or activity-based view of a system at this stage leads to significantly lower quality requirements than taking a view of a system as supporting contractual and world-related obligations*" [BCDS93]

Organisationseinheiten bzw. Akteure besitzen im Allgemeinen verschiedene Rollen bzw. Verantwortlichkeiten (Erwartungen, Anforderungen, Aufgaben, Rechte und Pflichten), aus denen ihr Verhalten folgt ("*the behavior of an agent is consequence of its mental states such as beliefs, intentions, capabilities, goals, commitments etc.*" [Sho93]). Sie unterwerfen sich bestimmten Regeln (z.B politische Vorgaben) und Prozeduren (z.B. standardisierte Vorgehensweisen) und besitzen subjektive Ziele und Wünsche.

Diese Erkenntnis hat auch Eingang in neuere Ansätze der Organisationstheorie gefunden. Die Offenlegung der persönlichen Handlungsziele und Beweggründe ist Bestandteil des "organizational learning" [Fre90], das eine gegebene Organisation nicht als starren "Mechanismus", sondern als ein lebendiges Wesen begreift, welches neben der offiziellen und oft fremdbestimmten Zielstruktur (bedingt durch die Aufgabenteilung) auch die selbstbestimmten persönlichen Ziele und Motive berücksichtigt. Organisation wird dabei als andauernder Prozess gesehen, der subjektive Gegebenheiten miteinbezieht, analysiert und bewusst macht.

In der Analyse des motivationalen Kontexts lassen sich laut Sutcliffe [SM93] im Wesentlichen drei Verfeinerungsebenen von Zielen unterscheidet. Während im "policy level" übergeordnete und abstrakte Ziele definiert werden, werden im "functional level" die dafür notwendigen konkreten Handlungsziele benannt und diese im "domain level" letztendlich zu Handlungsanweisungen konkretisiert bzw. rationalisiert.

Modelliert man diese Ziele in KoA$_{L}$A in Form von Aktivitäten eines Tätigkeitsprofils und setzt sie in entsprechende Verfeinerungs- oder Spezialisierungsbeziehungen, entsteht eine baumartige Struktur, die die abstrakten Ziele als Wurzel und die konkretisierten Handlungsziele als Blätter besitzt.

Laut Dardenne hilft eine solche baumartigen Repräsentation in der Modellierung Vollständigkeit zu erreichen, indem der Modellierungsprozess angeleitet und strukturiert wird, z.B. in Form eines hierarchischen Top/Down-Vorgehens wie beim "Goal-directed Requirements Acquisition" nach [DvLF93]. Dardenne verfeinert hier die Systemziele inkrementell in Form eines And/Or-Graph, was auch in KoA$_{L}$A analog mittels Aggregations- und Spezialisierungsbeziehungen ausgedrückt werden kann. Ein Vorteil baumartiger Repräsentationen liegt in ihrer einfachen Verständlichkeit. Das globale Bild ("the big picture") ist auch Laien bzw. Anwendern sofort zugänglich.

Ziel der Modellierung ist dabei ein vollständige Dokumentation des Aktivitätspotentials (Menge möglicher Aktivitäten, die an den Tag gelegt werden), nicht unbedingt auch eine vollständige Dokumentation der Motivation oder einer umfassenden Cause-Effect-Struktur.

In der Typisierung der Ziele unterscheidet Sutcliffe desweiteren "prescriptive goals" (von außen vorgeschrieben), "descriptive goals" (operationalisierbare Handlungsziele) und "adverbial goals" (Randbedingungen). Auch in KoA$_{L}$A findet sich diese klare Unterscheidung: prescriptive und adverbial goals werden als Bedingungen (Anforderungen, Motive, Randbedingungen etc.) modelliert und descriptive goals als Aktivitäten beschrieben bzw. zu Aktivitätsmustern verfeinert.

Ziele und Szenarien sind laut Sutcliffe für eine vollständige Modellierung der Anwendungsumgebung beide unabdingbar. Goals leiten die Erhebung der Szenarien, während Szenarien wiederum essentiell für das Verständnis und die Erhebung der Ziele sind. Für eine Modellierung der Ziele anstatt von konkreten Arbeitsabläufen (wie z.B. in Form konkreter Geschäftsprozesse) spricht auch, dass Ziele stabiler als die sie realisierenden Prozesse sind [AMP94].

In KoA$_{L}$A finden sich diese zwei Sichten – Ziele und Szenarien – in den Tätigkeitsprofilen des Organisationsmodells und in den Aktivitäts- und Interaktionsmustern des Kooperationsmodells wieder.

4.4. Das Kooperationsmodell

Das Kooperationsmodell beschreibt die globale Ablauforganisation eines betrachteten soziotechnischen Handlungssystems. Es definiert ein "Drehbuch" der täglichen Arbeitspraxis, in dem es deren charakteristischsten Handlungen und Dialoge beschreibt. Das Drehbuch wird

durch entsprechende Diagramme visualisiert und in der Projektdatenbank um eine Beschreibung des spezifischen Kontexts ergänzt.

Das Kooperationsmodell besteht aus folgenden Teilmodellen:

- In der *Organisationsstruktur* werden die statischen, kommunikativen und motivationalen Beziehungen aller im Organisationsmodell identifizierten Akteure beschrieben.
- Mittels der *Kommunikationsmuster* wird das Interaktionsverhalten der Akteure im betrachteten Handlungssystem dargestellt
- Durch die *Aktivitätsmuster* werden die charakteristischsten Ausschnitte akteurübergreifender Arbeitsabläufe dokumentiert

Technische Informationssysteme dienen der Speicherung, dem Transport und der Verarbeitung von Informationen, die in Form sprachlicher Symbole kodiert werden. Die Effektivität und Akzeptanz eines solchen Systems ist in hohem Maße von einer adäquaten Ausgestaltung dieser vom System verstandenen "Sprache" bzw. konzeptionellen Begriffswelt abhängig. Dies betrifft dabei natürlich nicht nur den reinen Wortschatz bzw. die Informationsstruktur sondern auch ihre angemessene Interpretation innerhalb der vom System zur Verfügung gestellten Verarbeitungsfunktionen.

Verschiedenste Ansätze der Sprachphilosophie und Erkenntnistheorie (Heidegger, Wittgenstein, Searle etc.) vertreten die Ansicht, dass sich die Bedeutung des gesprochenen Wortes nur im Kontext menschlicher Tätigkeit interpretieren lässt. Die sprachliche Begriffswelt ist kein Abbild vorgegebener Wirklichkeit, sondern sie entwickelt sich erst iterativ aus den Bedürfnissen menschlicher Kooperation heraus.[12] Um die Sprache eines Handlungssystems wirklich zu verstehen, sind also auch die Aktivitäten dieses Systems zu untersuchen.

Umgekehrt kann eine Analyse der Kommunikationsstrukturen und -muster dabei helfen, die zentralen Aktivitätsbereiche eines Systems herauszuarbeiten.[13]

Um eine optimale Gestaltung eines IS zu erzielen, ist es darum unumgänglich, die praktische Rolle des Systems in der alltäglichen Arbeitswirklichkeit zu untersuchen. Um hier die vielseitigen Beziehungen der Akteure im Handlungssystem aufzudecken, bietet es sich an, ihre Kommunikationsbeziehungen und Interaktionsmuster zu untersuchen.

[12] Nach Heidegger entsteht der Zwang zur sprachlichen bzw. rationalen Erfassung von Sachverhalten erst da, wo der Handlungsfluss "zusammenbricht", d.h. Probleme bewältigt werden müssen. Ein Hammer wird dem Zimmermann beim Einschlagen eines Nagels nur dann als eigenständiges Objekt bewusst, wenn er z.B. auseinanderbricht. In der eigentlichen Tätigkeit ist ein solches *"zuhandenes"* Werkzeug dem Benutzer nicht bewusst, d.h. es stört diesen Fluss auch nicht.

[13] In der Sprachphilosophie und der Erkenntnistheorie wird die menschliche Sprache aus den verschiedensten Blickwinkeln untersucht. Sprache ist laut Heidegger immer nur in Beziehung zu einem gegebenen Hintergrund interpretierbar. Dieses Hintergrundwissen wird jedoch durch Handlungen bzw. Kommunikation fortlaufend verändert (-> "Hermeneutischer Zirkel"). Die Bedeutung sprachlicher Begriffe lässt sich daher nur im aktuellen Tätigkeitskontext bestimmen (Wittgenstein: "Meaning is use"). An die Stelle des Rationalismus&Dualismus Descartscher Prägung tritt die Idee des praktischen Handelns als Grundlage, Folge und Ziel von Erkenntnisprozessen.

Sprache dient selbst als Werkzeug, das Handlungsmöglichkeiten schafft und darüber die Wirklichkeit ändert (Searle spricht daher auch von "Speech Actions"). Eine wesentliche Funktion der Sprache ist z.B. das Erzeugen bzw. Unterhalten eines Netzwerks aus wechselseitigen Verpflichtungen zum Zwecke der Handlungskoordination. Die wechselseitige Kopplung der Sprechenden untereinander führt dabei zur Ausbildung von Regelmäßigkeiten im Sprachgebrauch – zu Konversationsmustern.

4.4.1. Kooperationsstruktur

In diesem Teilmodell wird der statische Kontext der Kooperationen dokumentiert. Wesentliches Augenmerk liegt hier im Besonderen auf den Kommunikations- bzw. Kooperationsbeziehungen, die in der realen Arbeitspraxis eingegangen werden und den gegenseitigen Erwartungshaltungen.

In der Sprachphilosophie wurde die natürliche Sprache als Kommunikationsmittel ausführlich studiert [WF87]. Ein maßgebliches Ergebnis dieser Bestrebungen ist die Erkenntnis, dass eine verbale Nachricht zusätzlich zum propositionellen Inhalt auch eine pragmatische Funktion besitzt. Searl spricht hier darum auch von "Speech Acts" [Sea69, Ort02, KKB95]. Sprachliche Interaktionen werden dabei in unterschiedliche Äußerungstypen unterschieden wie z.B. mitteilen, auffordern, fragen, beschreiben, warnen, ersuchen, entschuldigen, anordnen, versprechen, kommentieren, kritisieren etc.

In KoA$_L$A wird eine Interaktion darum auch als Kooperation aufgefasst, die über die reine Aktivität selbst und den Transport von Informationen bzw. etwaigen Materialien hinaus auch eine übergeordnete Funktion erfüllen kann.

Beispiel: "Medizinstudent stellt Antrag zur Teilnahme am Praktischen Jahr" umfasst nicht nur das von Studentenseite durchgeführte Ausfüllen und Einschicken des Antrags an das PJ-Büro nebst dessen Entgegennahme durch das PJ-Büro, sondern auch eine Willensäußerung des Studenten zur Teilnahme am Zuteilungsverfahren und der Bereitschaft, ein Praktisches Jahr abzuleisten.

Neben den Interaktionsbeziehungen werden in der Kooperationsstruktur auch Kooperationsbeziehungen und Erwartungshaltungen berücksichtigt. Ausgedrückt werden hierbei nur statische Beziehungen. Es wird keine Aussage über die zeitliche Abfolge oder kausale Abhängigkeiten von Interaktionen oder Kooperationen getroffen.

Zusammenfassend stehen damit folgende Modellierungskonzepte zur Verfügung:

Interaktionen Kommunikationsbeziehungen zweier oder mehrerer Akteure. Als Interaktion werden in KoA$_L$As Kooperationsstruktur ganz unterschiedlichste Interaktionstypen modelliert,wie z.B.

- die reine Übermittlung von Informationen (z.B. Wahlergebnisse, Kandidatennamen etc.) oder materiellen Objekten (z.B. Akten, Stimmzettel, Dateien etc.)
- der normale Sprechakt, der zwar immer auch mit einer Informationsübermittlung einhergeht, allerdings darüber hinaus auch immer einem bestimmten Zweck dient (z.B. der "Bestellung von Ware", der "Anmeldung für eine bestimmte Lehrveranstaltung", der "Wahl eines Personalrats" etc.)
- die Übermittlung bzw. Initiierung von Aufträgen bzw. "Befehlen" zur Delegation von Aktivitäten
- die Ausübung rein physikalischer "Einwirkungen" auf ein Gegenüber ("Leeren einer Wahlurne", "Ausfüllen eines Stimmzettels"), meistens in Zusammenhang mit passiven Gegenständen der Organisationsstruktur (z.B. "Postfach leeren" als Interaktion zwischen einem menschlichen Akteuren und dem Gegenstand "Postfach", oder "Fax schicken")

Kooperationen Gemeinsame Aktivitäten zwei oder mehrerer Akteure, die über reine Kommunikationshandlungen hinausgehen. In der Regel auf einer sehr abstrakten Ebene beschrieben.

Erwartungshaltungen Randbedingungen für Akteure, deren Ursprung jeweils andere Akteure sind. Damit werden auch solche indirekten Einflüsse modelliert, die in ihrer Vereinigung einen Ausschnitt des sozialen Netzes der gegenseitigen Verpflichtungen und Vereinbarungen explizit machen. Zusätzlich zu dem im Kap. 4.3.2 definierten Bedingungstyp "Anforderung" kann hier auch noch der Bedingungstyp "Vereinbarung" unterschieden werden, der eine von mehreren Akteuren eingegangene gegenseitige Verpflichtung, einen sogenannten "social contract" darstellt.

Das Kontextdiagramm

Die Organisationsstruktur wird textuell in der Projektdatenbank dokumentiert und grafisch in Form eines oder mehrerer Kontextdiagramme verdeutlicht. Abb. 4.4 zeigt als Beispiel die Kooperationsstruktur eines Handlungsystems, wie es sich an der Universität Ulm bei der Durchführung einer Gremienwahl ergibt.

Im Kontextdiagramm werden die wichtigsten Akteure und deren kooperationsbezogenen Abhängigkeiten dokumentiert. Akteure werden dabei wie in den Diagrammen der Organisationsmodellierung dargestellt (siehe Kap. 4.3.1).

> Wichtige Akteure einer Gremienwahl sind z.B. die Wahlleitung, die Wahlhelfer und natürlich die Wähler.

Unter den Kooperationsbeziehungen zwischen diesen Akteuren werden Kooperationen, Erwartungen und Interaktionen unterschieden.

Kooperationen werden wie die Aktivitäten des Tätigkeitsprofils als Rechtecke mit abgerundeten Seiten notiert (siehe Kap. 4.3.2) und mit allen beteiligten Akteuren verbunden.

> Im Falle der Gremienwahl kooperieren beispielsweise Wahlhelfer und Wahlleitung bei der Auszählung des Wahlergebnisses. Die Briefwahl hingegen ist eine direkte Kooperation von Wahlleitung und Wähler, d.h. sie findet insbesondere unabhängig von Wahlhelfern oder Wahllokalen statt.

Erwartungen werden als Bedingungen modelliert und sind daher entsprechend als Textblöcke in geschweifte Klammern gesetzt und mit den verursachenden bzw. beeinflussten Akteuren mit einer gestrichelten Linie verbunden.

> Die Wahlleitung steht im Fallbeispiel hier beispielsweise unter einem erhöhten Erwartungsdruck des Rektors der Universität, eine korrekte Wahl durchzuführen, nachdem eine frühere Wahl wegen Formfehlern wiederholt werden musste.

Zusätzlich zu den im Kap. 4.3.2 definierten Bedingungstypen können im Kontextmenü auch "Vereinbarungen" unterschieden werden.

> In der Abbildung wird hierzu beispielsweise die notwendige enge Zusammenarbeit angeführt, die Wahlleitung und Wahlprüfungsausschuss zum Zweck der Überprüfung des Wahlergebnisses vereinbart haben.

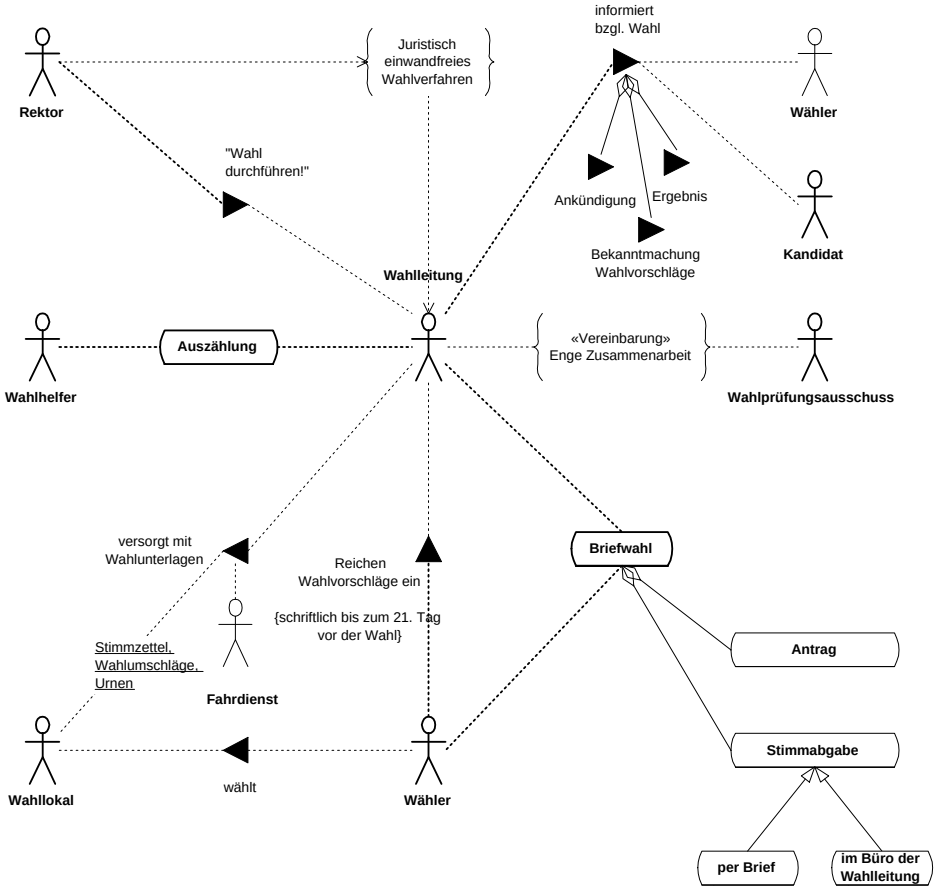


Abbildung 4.4.: Kontextdiagramm für die Durchführung von Gremienwahlen

Eine genaue textuelle Erläuterung bzw. explizite Ausarbeitung der angegebenen Randbedingungen oder Vereinbarung findet sich in der Projektdatenbank.

Interaktionsbeziehungen werden, wie in den Profildiagrammen bereits erwähnt, in Form von Pfeilen mit ausgefüllter Dreieckspitze dargestellt (siehe Kap. 4.3.2), die die beteiligten Akteure verbinden. Die Pfeilspitze kann dabei alternativ entweder am Ende der Verbindungslinie notiert werden oder aber auch in deren Mitte gesetzt werden. Da die Interaktion ein zentrales Modellierungskonzept ist, das in vielen Diagrammen von KoA$_{\text{L}}$A auftaucht, soll dieses Konzept nun im Folgenden etwas umfassender dargestellt werden, als es in Bezug auf das Kontextdiagramm eigentlich notwendig wäre.

Die Interaktion Eine Interaktion stellt eine besondere Art der Kooperation zweier oder mehrerer Akteure dar: die der Kommunikation. [Die91] definiert Kommunikation als eine wechselseitige Beeinflussung des Verhaltens der Kommunikationspartner durch den Gebrauch von Sprache. KoA$_{\text{L}}$A liegt ein erweiterter Kommunikationsbegriff zugrunde, indem sich hier Kommunikation nicht alleine auf die sprachliche Verständigung beschränkt. Jegliche Einflussnahme auf einen menschlichen oder technischen Interaktionspartner wird in KoA$_{\text{L}}$A als Interaktion aufgefasst, sei es die Übermittlung physischer Objekte wie z.B eines Briefes oder auch die di-

rekte physische Einflussnahme z.B. in Form der Bedienung von Eingabegeräten wie Tastatur und Maus oder das Aushängen von Aushängen an einem schwarzen Brett.

Eine Interaktion umfasst alle Einzelaktivitäten, die der direkten Beeinflussung des Interaktionspartners dienen, jedoch keine Aktivitäten, die der Vorbereitung der Interaktion dienen oder die lokal von den beteiligten Akteuren während oder in Folge der Interaktion durchgeführt werden. Eine Interaktion hat immer genau einen Initiator bzw. Sender und einen oder mehrere mögliche Empfänger. Eine Interaktion kann einen komplexen Ablauf von Einzelinteraktionen (= Dialog) subsumieren.

Bei der Darstellung einer Interaktion werden in den Modellen bzw. Diagrammen von KoA$_{\text{L}}$A allgemein folgende Aspekte unterschieden:

Ziel: Die pragmatische Funktion der Interaktion.

Interaktionspfeile gehen vom Initiator aus und werden mit dem Interaktionsziel beschriftet. Die Beschriftung wird in der Regel subjektbezogen und aktiv formuliert (“bestellt”, “bewirbt sich”).

Interaktionen, die der Vermittlung eines Auftrags dienen, werden hervorgehoben, indem die Beschriftung in Anführungszeichen gesetzt und mit einem Ausrufezeichen notiert wird. Der Auftrag wird normalerweise in der neutralen Infinitivform formuliert: ”Akte holen!”. Damit wird sowohl die delegierte Aufgabe selbst dokumentiert als auch das weitergehende Interaktionsziel. Bei Werkzeugen lässt sich auch die aktive Formulierung einsetzen, wenn man ihr Zuhandensein besonders betonen will (”kalkuliert Festlegung!”, ”erstellt Tagesabrechnung!”). Diese Formulierung unterstreicht die aktive Rolle des Initiators, indem ihm nicht nur die Übermittlung des Auftrags, sondern auch die aktive Durchführung der (eigentlich delegierten) Handlung direkt zugeordnet wird. Die Rolle des “Werkzeug” bzw. des Auftragnehmers als eigentlicher (stellvertretend handelnder) Akteur tritt in den Hintergrund.

Eine Interaktion ist eine Aktivität, die von den beteiligten Akteuren gemeinsam durchgeführt wird. Sie umfasst alle interaktionsbezogenen Aktivitäten der Partner, die zur Durchführung einer erfolgreichen Kommunikation notwendig sind. Im Diagramm selbst wird jedoch nur das Ziel dieser Interaktionsaktivitäten aufgeführt. Eine ausführliche Beschreibung dieser Aktivitäten kann in der Projektdatenbank notiert werden.

Inhalt: Die in der Kommunikation vermittelten Objekte werden in der Nähe der Rezipienten am Beginn oder Ende des Pfeils vermerkt. Diese Objekte können dabei sowohl materieller (z.B. bestimmte Dokumente) als auch immaterieller Natur sein (z.B. Informationsobjekte). Definiert werden diese Objekte in Form sprachlicher Konzepte als Bestandteil des Wissensmodells (s. Abschnitt 4.2).

Während einer Interaktion können Inhalte immer auch in beide Richtungen “fließen”; Konzepte können also an beiden Enden des Pfeils notiert sein. Um Interaktionen und Konzepte physikalischer Beschaffenheit zu verdeutlichen, können auch Pictogramme verwendet werden.

Partner: Im Falle einer 1:1-Kommunikation führt ein Interaktionspfeil vom Initiator zum

Empfänger. Im Falle einer 1:n-Kommunikation führt ein Strich vom Initiator zur Mitte einer Seite der gefüllten Dreiecksspitze, während die gegenüberliegende Spitze des Dreiecks mit allen Empfängern verbunden wird. Ist der Initiator nicht eindeutig festzulegen (echte Kooperationen, Dialoge, n:1-Kommunikationen) oder will man das nicht, wird diese Beziehung nicht als Interaktion, sondern als Kooperation modelliert.

Resource: Akteure, die in der Interaktion nur als Kommunikationsmittel, Medium oder Werkzeug benutzt werden, werden mit einer der beiden restlichen Dreiecksseiten verknüpft, die nicht mit dem Initiator in Verbindung stehen.

Randbedingungen: Als normale Aktivität kann eine Interaktion auch bestimmten Randbedingungen unterliegen. Interessant sind hier z.B. die Häufigkeit, die zeitliche Dauer, die zeitlichen Abstände einer Interaktion etc. Diese Randbedingungen lassen sich als Bedingung modellieren. Auf diese Weise lassen sich im Diagramm optional auch Gestalt und Form der Interaktion festlegen. Bedingungen von Interaktionen werden notiert, indem sie (in geschweiften Klammern gesetzt) direkt neben dem Interaktionspfeil vermerkt werden.

Alle diese Aspekte einer Interaktion sind in ihrer grafischen Darstellung in den diversen Diagrammen von KoA$_{\text{L}}$A optional. Eine Interaktionsbeziehung kann beispielsweise einen reinen "Kommunikationskanal" (ohne alle Annotationen), einen Informations- oder Gegenstands-Fluss (nur der Inhalt wird annotiert) oder einen vollständigen Sprech-Akt (mit Funktionsangabe) ausdrücken. In Tätigkeitsprofilen (s.o.) oder Konversationsdiagrammen (s.u.) können mitunter auch Interaktionen auftauchen, die nur einen Partner besitzen und damit reflexive Interaktionen (z.B. Erinnerungsvorgänge, Bewusstwerdung etc.) darstellen bzw. die Modellierung bewusst unvollständig bzw. "offen" lassen, falls z.B. der Ursprung eines Ereignisses oder der Empfänger von erstellten Dokumenten unbekannt oder irrelevant ist.

Komplexe Kooperationen oder Interaktionen können im Kontextdiagramm ähnlich zum Profildiagramm über Aggregationsbeziehungen verfeinert oder über Spezialisierungsbeziehungen konkretisiert werden. Abb. 4.4 zeigt hier als Beispiel die Interaktion "informiert bzgl. Wahl", die in die Einzelinteraktionen "Ankündigung" "Bekanntmachung Wahlvorschläge" und "Ergebnis" verfeinert wird oder die Kooperation "Briefwahl", die in die Einzelaktionen "Antrag" und "Stimmabgabe" verfeinert wird. Desweiteren wird die Stimmenabgabe bei der Briefwahl in die zwei Fälle unterschieden, dass ein Briefwähler entweder per Brief abstimmt oder aber direkt bei der Wahlleitung vorstellig wird und dort den Stimmzettel ausfüllt.

4.4.2. Konversationsmuster

Während im Teilmodell der Kooperationsstrukturen vor allem die statischen Aspekte von Kooperationsbeziehungen dargestellt werden, beschreiben die nun im Folgenden vorgestellten Submodelle der Konversations- und Aktivitätsmuster diese Kooperationen aus dynamischer Sicht.

Das dynamische Verhalten eines komplexen Systems wie das einer bestimmten soziotechnischen Organisation kann auf Grundlage einer Modellierung ihrer Aktivitätsebene nie vollständig erfasst werden. Das mögliche Verhaltensrepertoire autonomer und mit "freiem Willen"

ausgestatteter menschlicher Akteure ist hierzu in der Realität zu vielfältig. In der klassischen Geschäftsprozessmodellierung beschränkt man sich demzufolge auch auf eine Darstellung der charakteristischsten und wichtigsten Muster solcher Aktivitäten, die "Geschäftsprozesse". Neuere Sichtweisen begreifen einen solchen Geschäftsprozess jedoch nicht mehr nur als eine vordefinierte und abgeschlossene Kette von zu koordinierenden Einzelaktivitäten, sondern sehen ihn als einen offenen Aktivitätsbereich, indem die beteiligten Akteure autonom und über Interaktionen kooperierend Probleme lösen: "*We view a business process as a collection of autonomous, problem solving agents which interact with others when they have interdependencies*".[YS99]

Statt einer prozessorientierten Sicht auf die Aktivitätsbereiche eines soziotechnischen Systems bietet sich darum auch eine abstraktere, interaktionsorientierte Sicht an, die auf der Ebene der zu beobachtenden Kommunikation ansetzt. Das Kommunikationsverhalten des soziotechnischen Systems lässt sich hier z.B. auf Basis von Konversationsmustern beschreiben, die sich regelmäßig wiederholende Sequenzen von Interaktionen zu Dialogen zusammenfassen (siehe hierzu z.B. auch die "discourse structure"-Diagramme von [WF87]). Die zu beobachtenden Interaktionen stellen zum Einen oft die eigentliche Ursache für das Tätigwerden der beteiligten Akteure dar, zum Anderen geben sie den Aktivitäten der Akteure aber auch einen umfassenden Kontext, der das Ziel, das Motiv oder die Randbedingungen der Aktivität weitgehend festlegt.[14]

Konversationsstrukturen stellen systemische Bereiche dar, die Ergebnis sozialer oder betrieblicher Vereinbarungen und Konventionen sind. Diese stellen den eigentlichen Rahmen für das praktische Handeln jedes Akteurs dar und sind der Ausgangspunkt jeder Bemühung, Kontextinformationen zu erfassen und zu dokumentieren.[15]

Die spezifische Form bzw. das Muster der etablierten Konversationen ist in Sozialsystemen normalerweise weitgehend festgelegt, etabliert oder standardisiert und daher seltener Verände-

[14] Konversationen werden z.B. auch in [HMW98] als höhere Abstraktionsebene eingeführt, um den Übergang von der Prozessmodellierung zum softwaretechnischen Systementwurf bruchloser zu gestalten als es mit der Geschäftsprozessmodellierung auf Grundlage von Use-Cases alleine möglich wäre.

[15] Die besondere Rolle der Kommunikation bei der Erfassung bzw. Beschreibung praktischer Arbeitswirklichkeit lässt sich auch aus Ergebnissen der Philosophie bzw. Erkenntnistheorie ableiten. Nach Martin Heidegger [WF87] ist menschliches Arbeitshandeln überwiegend "unbewusst". Erst bei "Brüchen" im Arbeitsfluss wird Bewusstsein notwendig, beispielsweise in der Benutzung von Werkzeugen, wenn diese Werzeuge ihre Funktion nicht erfüllen (wenn z.B. beim Hämmern der Hammer bricht), oder in arbeitsteiligen Arbeitsprozessen, wenn Arbeitspartner Erwartungen bzw. Vereinbarungen nicht erfüllen. Nach Heidegger und anderen "Konstruktivisten" entstanden und erstehen Worte nur, um solche Aktivitätsbrüche zu beseitigen. Begriffe beziehen ihre Bedeutung bzw. Daseinsberechtigung also allein aus den Aktivitätsprozessen, die sie ermöglichen.

Im Umkehrschluss kann daher aus den zu beobachtenden Brüchen und den verwendeten Begriffen auch wieder auf die eigentliche Aktivität zurückgeschlossen werden. Beschreiben wir systemische Bereiche der Kommunikation incl. den thematisierten Begriffen, so identifizieren wir damit auch die wesentlichen Aktivitätsmuster und -inhalte. Konversationen sind notwendige Mittel zum Zweck und können daher ein Weg sein, um diesen Zweck in Form übergreifender Aktivitätsfolgen aufzuspüren und daraus wiederum Tätigkeitsprofile zu definieren.

Diese Ideen sind auch aus Sicht der Gestaltung relevant: Eine "unterbrechungsfreie" Tätigkeit ist effizienter als eine Tätigkeit, die aufgrund von "Brüchen" Interaktionen erforderlich macht. Kommunikationen sind daher Anzeichen für Brüche im Handlungsablauf, die durch die Systemgestaltung möglichst minimiert werden sollten. Erst in zweiter Linie sind dann natürlich auch die Arbeitstätigkeiten selbst zu optimieren, z.B. durch Rationalisierung einzelner Tätigkeiten, was wiederum zu neuen Interaktionsbeziehungen führen kann.

rungen unterworfen als die spezifische Ausprägung der ausgelösten Aktivitäten der einzelnen Akteure selbst, die oft weniger reglementiert sind. Der Kontext einer Aktivität ist stabiler als deren konkrete Ausprägung.

Das Konversationsdiagramm

Konversationsmuster werden grafisch in Form von Konversationsdiagrammen dargestellt und textuell in der Projektdatenbank dokumentiert.

Konversationsdiagramme beschreiben mögliche sich wiederholende Sequenzen von Akteurs-Interaktionen in Form einer Art nichtdeterministischen Zustandsübergangsdiagramms, dessen Zustandsknoten grafisch durch die gerade aktiven Akteure besetzt sind und deren mögliche Übergänge durch die von diesem Akteur initiierten Interaktionen ausgelöst werden.

Abb. 4.5 zeigt als Beispiel das globale Konversationsmuster für die Durchführung einer Gremienwahl an der Universität Ulm, wie sie auch in der offiziellen Wahlordnung festgelegt ist.

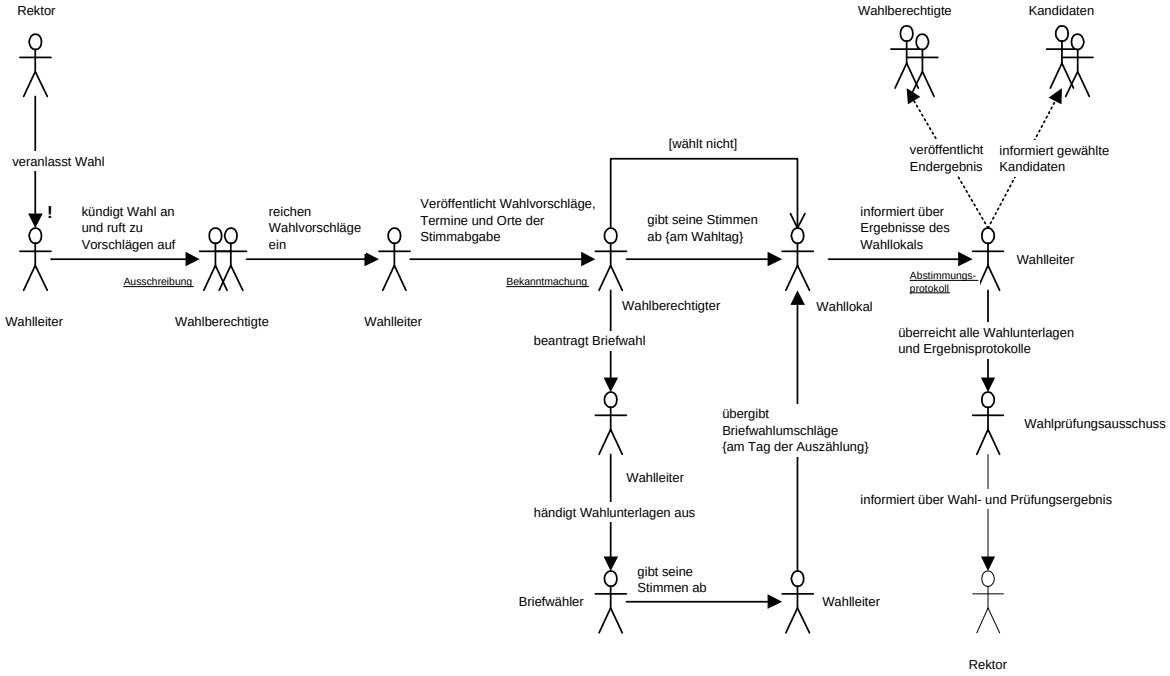


Abbildung 4.5.: Konversationsdiagramm für die Durchführung einer Gremienwahl

Ein Konversationsdiagramm verwendet zur Darstellung von Akteuren und Interaktionen die gleichen grafischen Elemente, wie sie schon in den Diagrammen des Organisationsmodells (siehe Kap. 4.3) und der Kooperationsstruktur (siehe Kap. 4.3.1) verwendet wurden.

Die Abstraktionsebene in der Modellierung der beteiligten Akteure wechselt dabei innerhalb eines Diagramms beliebig zwischen konkreter Instanz, Rolle, Typ und Einheiten höherer Organisationsebenen (wie im Beispiel das Wahllokal oder auch die Gruppe aller Wahlberechtigten

etc.) und beschränkt sich in der Darstellung normalerweise gezielt auf einzelne wichtige Interaktionsbeziehungen. Ein Konversationsmuster dient somit in der Regel nicht der umfassenden Spezifikation des Kommunikationsverhaltens eines gesamten Systems sondern eher der gezielt offenen Beschreibung einzelner Aktivitätskontexte dieses Systems. Konversationsdiagramme stellen hierzu gewollt "unvollständige" Beschreibungen einzelner, wichtiger und sich wiederholender Interaktionssequenzen bzw. Kommunikationsszenarien dar.

> In der Abbildung beginnt die Konversation beispielsweise mit dem Auftrag des Rektors zur Durchführung der Wahl, setzt sich fort in der Ankündigung der Wahl gegenüber allen Wahlberechtigten und konkretisiert sich zur Beschreibung des Abstimmungsverfahrens auf der Ebene des einzelnen Wahlberechtigten, ehe zum Schluss der Wahlleiter das Gesamtergebnis an den Wahlprüfungsausschuss übergibt und dieser das Ergebnis an den Rektor zurückmeldet.
>
> Wie die einzelnen Wahllokale mit dem Wahlleiter zusammenarbeiten und in welcher Weise die Stimmabgabe der zahlreichen Wahlberechtigten koordiniert und ausgewertet werden, bleibt im Beispiel undefiniert.

Neben dem Zustandsübergang per Interaktion sieht das Konversationsdiagramm auch die Möglichkeit eines einfachen Zustandswechsels vor, der durch einen einfachen Pfeil dargestellt wird. Diese Art eines "Kontrollflusses" kann ebenso wie auch eine Interaktion desweiteren durch bestimmte Eingangsbedingungen (*Voraussetzung*s- oder Guard-Bedingungen) eingeschränkt werden. Statt diese Bedingung ähnlich zu den Bedingungen im Profildiagramm zu notieren und durch Wirkungspfeile mit den Interaktionen zu verbinden, können diese Bedingungen auch in eckige Klammern gesetzt neben dem Interaktions- oder Zustandswechsel-Pfeil notiert werden. Im dargestellten Diagramm findet sich hierzu z.B. ein Übergang vom Wahlberechtigten zum Wahllokal in dem Falle, dass ein Wahlberechtigter nicht zur Wahl geht bzw. ein Briefwähler zwar die Briefwahl beantragt, jedoch seine Stimmzettel nie einschickt.

Konversationsdiagramme können auch nichtdeterministisch sein, da Zustandsübergänge nicht unbedingt mit einer Bedingung verknüpft, ausgehende Interaktionen nicht unbedingt verschieden sein müssen und Bedingungen auf ausgehenden Pfeilen nicht unbedingt disjunkt sein müssen. Konversationsdiagramme beschreiben einen Möglichkeitsraum, sie spezifizieren kein Verhalten von Automaten.

Neben den eigentlichen dynamischen Zustandsübergängen können in Konversationsdiagramme auch weitere statische Interaktionsbeziehungen aufgenommen werden. Im Diagramm findet sich hier als Beispiel die Bekanntmachung des Wahlergebnisses gegenüber den Wahlberechtigten und den Kandidaten. Diese Beziehungen werden gestrichelt notiert und stellen reine Zusatzinformationen dar, die insbesondere keine Auswirkung auf den Zustand der dargestellten Konversation nehmen.

Will man ein Kontextdiagramm mit mehr Kontextinformationen anreichern, so kann man die Akteurs-Knoten des dargestellten Graphen noch um die von den Akteuren im jeweiligen Dialogzustand initiierten Aktivitäten bzw. Aktivitätsmuster ergänzen. Abb. 4.6 zeigt ein derart "angereichertes" Konversationsdiagramm für das Prozedere beim Abschluss von Hilfskraftverträgen.

> Der Student bewirbt sich bei einer Abteilung offiziell, indem er ein sogenanntes "Hilfskraft-Antragsformular" ausfüllt, das die Abteilung dann an Herrn M. weiterreicht. Parallel hierzu informiert sich der Student jedoch noch bei Frau G. über das weitere Prozedere und erhält ein Merkblatt bzgl. aller beizubringender Unterlagen.
>
> Herr M. kalkuliert mit Hilfe der Applikation "Hilve" die Kosten des Vertrages und überprüft das Budget. Sind die Kosten gedeckt, so wird ein offizielles Vertragsdokument erstellt, das noch vom

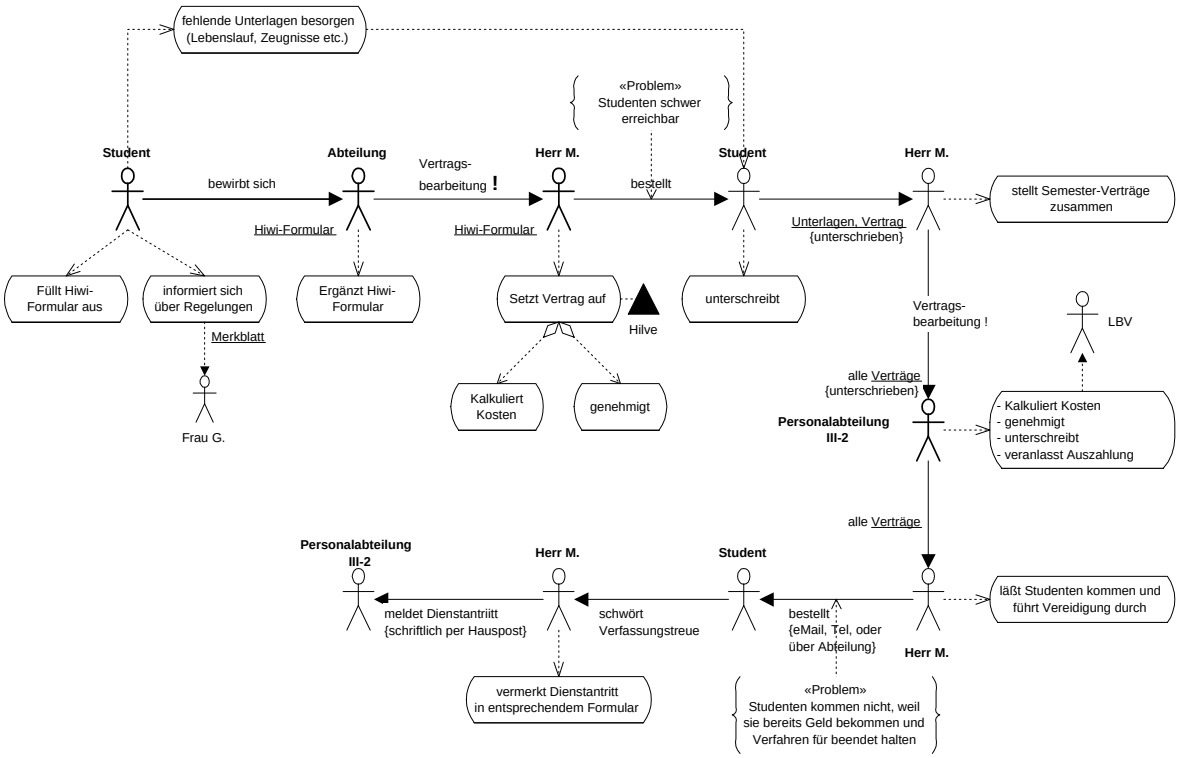


Abbildung 4.6.: Konversationsdiagramm für den Abschluss von Hilfskraftverträgen in der Fakultät Informatik

> Studenten zu unterschreiben ist, bevor es an die Personalabteilung III-2 weitergereicht werden kann.
>
> Diese führt eine erneute Kosten- und Budgetüberprüfung durch, unterschreibt den Vertrag von Seiten des Arbeitgebers, veranlasst die Zahlung über das LBV und schickt den fertigen Vertrag an Herrn M. zurück.
>
> Sobald er den entsprechenden Studenten habhaft werden kann, vereidigt er diesen und meldet den offiziellen Dienstantritt schriftlich per Formular an die Personalabteilung.

Die initiierten Aktivitäten werden mit gestrichelten Wechselpfeilen mit dem Akteur verbunden. Einzelne Aktivitäten lassen sich dabei auch durch Aggregationsbeziehungen in Unteraktionen verfeinern, wie im Beispiel das Aufsetzen des Vertrages in die beiden Unteraktivitäten Kalkulation und Genehmigung. Die Wechselpfeile werden gestrichelt gezeichnet, da sie nur der Zusatzinformation dienen und insbesondere keinen Einfluss auf die dargestellte Konversation nehmen.

Diskussion

Konversationsdiagramme zeichnen mögliche Ketten von Ereignissen vor. Bei der Erstellung der Diagramme liegt der Hauptaugenmerk des Modellierers jeweils auf den möglichen (Re-)-Aktionen der einzelnen Interaktionspartner. Konversationsdiagramme sind *deskriptiv*, sie beschreiben mögliches Verhalten autonomer Interaktionspartner.

Die in der Systementwicklung zur Beschreibung von Aktivitäten weit verbreiteten Diagram-

me wie z.B. Geschäftsprozessdiagramme oder die "Activity Diagrams" von UML sind von ihrem Charakter her in der Regel eher *präskriptiv*. Sie beschreiben bestimmte Tätigkeitsfolgen, die von den beteiligten Akteuren zur Erreichung ihrer Ziele durchgeführt werden bzw. durchgeführt werden müssen.

KoA_LAs Aktivitätsdiagramme, wie sie im nächsten Abschnitt beschrieben werden, sind nützlich zur Dokumentation von Handlungsplänen, die feste Abfolgen von Einzelaktivitäten festschreiben. Konversationsdiagramme hingegen dienen eher der explorativen Untersuchung möglicher Kommunikationsmuster. Während das Verhalten technischer Systeme noch mit Aktivitätsdiagrammen potentiell vollständig erfasst und definiert werden kann bzw muss, sind soziotechnische Systeme selbst auf der Ebene ihrer Kommunikationsmuster alleine nur unvollständig formalisierbar.

Die Knoten der Konversationsdiagramme identifizieren bestimmte Zustände der an der Konversation beteiligten Akteure und beschreiben ihre potentiellen kommunikationsbezogenen Reaktionsmöglichkeiten. In ihrer Vereinigung dokumentieren die zentralen Konversationsmuster das mögliche interaktive Verhaltensrepertoire autonomer – im Falle menschlicher Akteure auch mit freiem Willen ausgestatteter – Akteure, ohne es jedoch deterministisch festzulegen. Die tatsächlich stattfindenden Aktivitäten, d.h. das konkrete operative Verhalten selbst, ist erst Gegenstand des nächsten Teilmodells, das im folgenden Abschnitt beschrieben wird.

4.4.3. Aktivitätsmuster

Mit dem Teilmodell der Aktivitätsmuster kennt KoA_LA noch eine zum Teilmodell der Konversationsmuster orthogonale dynamische Sicht auf die Kooperationen innerhalb eines soziotechnischen Systems.

Während sich die Darstellung in den Konversationsdiagrammen auf diskrete Kommunikationsereignisse und die aktiven Wechselbeziehungen zwischen den beteiligten Akteuren konzentriert, legen KoA_LAs Aktivitätsdiagramme ihren Schwerpunkt eher auf die Erfassung der kontinuierlichen und kooperativen Tätigkeiten dieser Akteure.

Aktivitätsmuster beschreiben charakteristische Tätigkeitsfolgen eines oder mehrerer Akteure des betrachteten Handlungssystems. Die dargestellten Aktivitäten können dabei unterschiedlichen Abstraktionsebenen entstammen und sowohl akteurübergreifende globale Geschäftsprozesse darstellen als auch konkrete Szenarien elementarer Tätigkeiten einzelner Akteure[16]. Sie beschreiben dabei in der Regel Ausschnitte *möglicher* Aktivitätsbereiche der dargestellten Akteure, d.h. sie legen das Verhalten eines Akteurs dabei nicht deterministisch fest. Im Fall menschlicher Akteure werden diese als eigenständig handelnde Individuen angesehen, deren Arbeitstätigkeit nur ausschnittsweise erfasst und kontextreich beschrieben wird, anstatt den Versuch zu unternehmen, diese vollständig und formal zu spezifizieren. Eine vollständige Modellierung menschlichen Tätigseins ist von vornherein ausgeschlossen, da Aktivitäten immer situiert sind, d.h. an die spezifische Situation abgepasst werden [Suc87]. Die notwendigen

[16] Ein "globales", d.h. akteurübergreifendes Aktivitätsmuster kann immer auch als "lokale" Aktivität einer abstrakten Organisationseinheit angesehen werden. Ein Geschäftsprozess ist nichts anderes als ein bestimmtes Aktivitätsmuster eines als eine Ganzheit betrachteten Unternehmens bzw. als ein Aktivitätsmuster des "Unternehmens-Nutzungssystems" (Kunden plus Unternehmen).

Arbeitsschritte, um ein Ziel zu erreichen, sind wie die Zielsetzung selbst immer von der konkreten Situation abhängig, in der sich der handelnde Akteur befindet. Sie sind nie wirklich im vorhinein festschreibbar. Nach Suchman können Pläne keine vollständige Repräsentation realer Arbeitstätigkeit sein. Sie sind vielmehr kognitive Konstruktionen, die entweder vor oder nach der Handlungsrealisierung imaginiert werden. Pläne sind nach Suchman damit eher eine "resource for action" als ein "rigidly executed program".

Nicht destotrotz bewähren sich Pläne in Standardsituationen als Leitfaden für bewährte Vorgehensweisen und als Diskussionsmedium zur Darstellung menschlichen Arbeitshandelns (so z.B. im "Business Process Reengineering", der Geschäftsprozessanalyse, der Definition der "Core Workflows" im Rational Unified Process oder bei der Dokumentation der Vorgehensweise von KoA_LA selbst).[17] KoA_LAs Aktivitätsmuster dienen daher als Grundlage zur gemeinsamen Diskussion von real stattfindenden oder zukünftig vorgesehenen Tätigkeitsmustern des betrachteten Handlungsystems, ohne diesen Tätigkeitsraum im vornherein wirklich festzulegen oder einzuschränken.

Unverzichtbar sind solche Diskussions-Modelle immer dann, wenn es sich um arbeitsteiliges standardisiertes Handeln oder um die Steuerung, Darstellung, Vereinbarung oder Diskussion von Handeln dreht und das Handeln damit zum Gegenstand einer rationalen Auseinandersetzung wird. Hier müssen die Aufgaben bzw. Pflichten jedes Beteiligten klar sein und der globale Prozess in seiner Ablaufstruktur – auch nach außen hin (Kunde, Management) – transparent sein (Bsp: Das Wahlprozedere muss nicht nur der Wahlleitung und den Wahlhelfern klar sein, es muss auch den Wählern gegenüber deutlich kommuniziert werden).

Durch die Einführung technischer Systeme in ein Handlungssystem werden Teile seiner Aktivitätsmuster zwangsläufig formalisiert bzw. fest definiert. Durch die begrenzte Funktionalität und das festgeschriebene Verhalten des implementierten Systems und seiner Benutzerschnittstelle werden die möglichen Nutzungsszenarien normalerweise weitgehend vorgeschrieben bzw. eingeschränkt. Als Gestaltungsziel ist dieser Determinismus bzw. Inflexibilität jedoch normalerweise weitgehend unwillkommen. Selbst in Anwendungen wie den Workflow-Systemen, die Aktivitätsflüsse weitgehend formalisieren und damit auch kontrollierbar machen sollen, kommt man um die Berücksichtigung von "Ad Hoc"-Abweichungen nicht herum, wenn man "erfolgreiche", d.h. in der Arbeitspraxis nützliche und akzeptierte Systeme erzielen möchte. Es ist darum unabdingbar, im Gestaltungsprozess auch diesen nichtdeterministischen Nutzungskontexts eines Softwaresystems zu berücksichtigen und möglichst weitgehend zu erfassen.

Das Aktivitätsdiagramm

Ergänzend zu der im Weiteren vorgestellten grafischen Form der Darstellung von Aktivitätsmustern in Form von Aktivitätsdiagrammen tritt auch eine textuelle Beschreibung der Aktivitätsmuster in der Projektdatenbank (z.B. in der Form der *narrative task scenarios* nach [RC95]). Als eine nützliche Technik zur Dokumentation von Aktivitätsbeschreibungen hat sich der strukturierte Freitext ("Templates") schon in der Beschreibung von Use-Cases bewährt [Coc00], die mittlerweile nun des öfteren auch zur Modellierung von Geschäftspro-

[17] "*... the core workflows cannot present the real nuances of scheduling "real work", for they cannot depict the optionality of activities or iterative nature of real projects. Yet they still have value as a way for us to understand the process by breaking it into smaller 'areas of concern*" [RUP98]

zessen eingesetzt werden (siehe z.B. [Hru98]). Diese Technik lässt sich natürlich auch für KoA$_L$As Aktivitäten nutzbringend einsetzen. Siehe hierzu auch den Abschnitt 5.5.5.

Die textuelle Form der Beschreibung ist als erster Schritt für den Nutzer oft einfacher zu verstehen und zu validieren. Zur Diskussion ist hingegen eine abstraktere bzw. kompaktere Darstellung als Diagramm sinnvoller und nützlicher.

Die Notation zur grafischen Darstellung von Aktivitätsmustern orientiert sich im Wesentlichen an den "Activity Diagrams" von UML[Rat97], vereinfacht sie dabei jedoch weitgehend. Wie auch von Paech [Pae00] vorgeschlagen, beschränkt sich KoA$_L$A dabei auf die Aktivitäten in Kombination mit "Daten-" und "Kontrollflüssen" zwischen diesen[18]. Im Gegensatz zu UML oder Aris werden Datenflüsse wie auch Bedingungen nur als zusätzliche Annotation auf dem Wechselpfeil angebracht und nicht als eigene grafische Elemente eingeführt (siehe hierzu auch eine ausführlichere Diskussion von KoA$_L$A in Kap. 6.3).

Abb. 4.7 zeigt als Beispiel ein Aktivitätsmuster für die globale Ablauforganisation einer Gremienwahl, wie sie an der Universität Ulm durchgeführt wird.

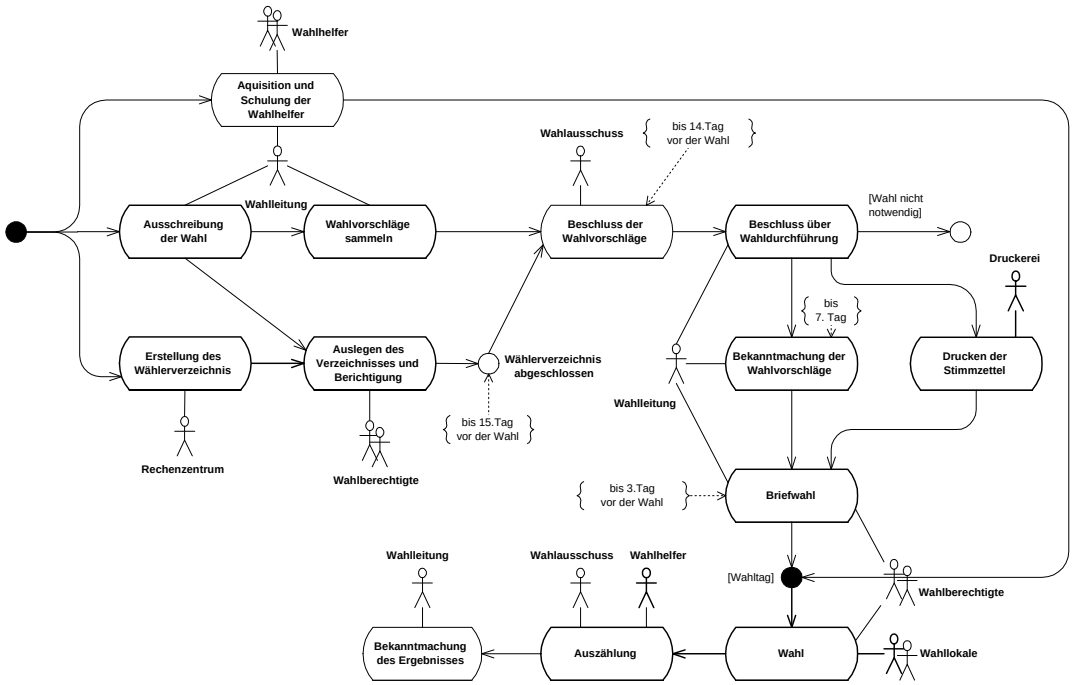


Abbildung 4.7.: Aktivitätsdiagramm für den globalen Ablauf einer Gremienwahl an der Universität Ulm

Ausgezeichnete Zustände im Aktivitätsfluss werden in einem Aktivitätsdiagramm als kleine Kreise notiert und mit einer Beschreibung des Zustands annotiert. Ein Zustand wird dann als

[18] Die Hinzunahme von Ereignissen/Bedingungen ist laut Paech nur gerechtfertigt, wenn man komplexe Kontrollflüsse zu modellieren hat. Die Modellierung würde sonst zu feingranular bzw. unübersichtlich (Beispiel: Ereignisprozessketten von Aris [Sch99a]). Die Hinzunahme von Datenspeichern ist desweiteren hingegen nur bei der Modellierung komplexer Synchronisationsmechanismen angezeigt.

ausgefüllter Kreis dargestellt, wenn *alle* eingehenden Wechsel stattfinden, bevor der Zustand erreicht ist und *alle* ausgehenden Wechsel unabhängig voneinander "nebenläufig" vonstatten gehen können. Solche Zustände werden im Folgenden auch als "Synchronisationspunkte" bezeichnet. So muss im Beispiel vor der eigentlichen Wahl sowohl die Briefwahl abgeschlossen sein als auch die beteiligten Wahlhelfer fertig ausgebildet worden sein.

Alle Zustände oder Aktivitäten, die keine eingehenden Zustandswechsel besitzen, sind potentiell nebenläufige Anfangszustände bzw -aktivitäten des dargestellten Musters. Zustände oder Aktivitäten, von denen keine weiteren Wechsel ausgehen, sind hingegen potentiell nebenläufige Endzustände bzw -aktivitäten.

> Im Beispiel findet sich hier ein Synchronisationspunkt als Anfangszustand, ein normaler Zustand als Endzustand (tritt ein, falls die Wahl aufgrund der wenigen eingegangenen Wahlvorschläge nicht notwendig ist) und die Wahlbekanntmachung als eine Aktivität, die das normale Ende des Wahlablaufs markiert.
>
> Ausgehend vom Synchronisationspunkt am Beginn des Aktivitätsmusters finden sich Wechsel zu unabhängigen Handlungsfäden wie der Wahlhelferschulung, der Wahlausschreibung und der Wählerverzeichniserstellung, die zu Beginn der Organisation "parallel" stattfindenden können.

Da Anfangsaktivitäten eines Musters implizit immer auch nebenläufig sein können, ist der Anfangs-Synchronisationspunkt in der Darstellung eigentlich überflüssig. Er dient hier ausschließlich der Lesbarkeit. Ein unausgefüllter Kreis kennzeichnet einen "normalen" Zustand, der von beliebigen eingehenden Wechseln aktiviert und alternativ durch genau einen ausgehenden Wechsel verlassen wird. Ein Beispiel hierfür findet sich im nächsten Diagramm (s.u.).

Aktivitäten werden im Aktivitätsdiagramm wie in den Diagrammen des Organisationsmodells als Rechtecke mit abgerundeten Seiten dargestellt und mit allen beteiligten Akteuren durch Linien verbunden. Aktivitäten sind implizit immer auch Synchronisationspunkte, d.h. alle vorausgehenden Aktivitäten der betrachteten Aktivität sind abgeschlossen, bevor die Aktivität selbst durchgeführt wird, und alle nachfolgenden Aktivitäten sind potentiell nebenläufig.

> In der Abbildung findet sich hierzu zum Beispiel das "Auslegen des Wählerverzeichnisses und dessen Berichtigung", das erst dann ausgeführt wird bzw. werden kann, wenn die Wahl durch die Wahlleitung ausgeschrieben wurde und das Wählerverzeichnis durch das Rechenzentrum erstellt und geliefert wurde. Hingegen können nach dem offiziellen Beschluss über die Wahldurchführung sowohl die Wahlvorschläge bekanntgemacht als auch unabhängig davon die Stimmzettel hergestellt werden.

Wechselpfeile, Aktivitäten oder Zustände können wie im Tätigkeitsprofil mit beliebigen Bedingungen annotiert werden. Zusätzlich zu den dort bereits eingeführten Stereotypen einer Bedingung tritt im Aktivitätsdiagramm noch der Typ "Voraussetzung" hinzu (auch "Guard"-Bedingung genannt), der das Ein- bzw. Auftreten einer Aktivität, eines Wechsels oder eines Zustands von einer zusätzlichen in Freitextform formulierten Bedingung abhängig macht. Diese Bedingungen können an Stelle einer separaten Darstellung in Form eines in geschweifte Klammern gesetzten Textblocks, der mit dem beeinflussten Modellelement durch Wirkungspfeile verbunden wird, auch in eckigen Klammern direkt neben der jeweiligen grafischen Ausdrucksform des Modellelements notiert werden.

Während das obige Diagramm ein akteurübergreifendes Aktivitätsmuster zeigt, stellt das Diagramm in Abb. 4.8 ein lokales Muster für die Bearbeitung von Hilfskraftverträgen aus der Sicht eines Sachbearbeiters der Personalabteilung dar.

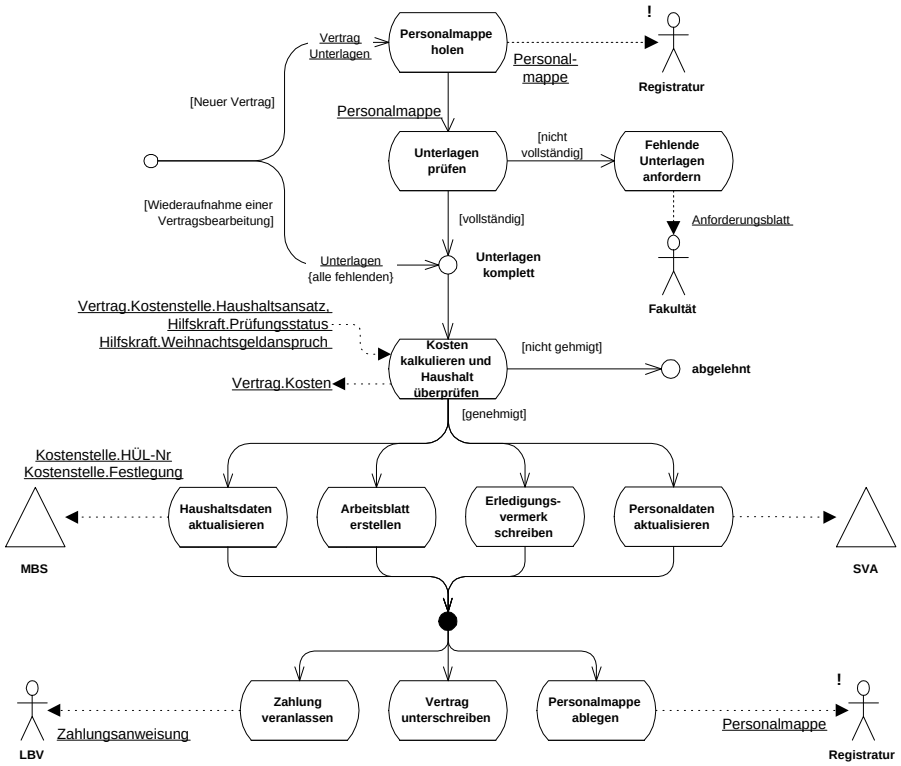


Abbildung 4.8.: Ein Aktivitätsdiagramm für die lokale Bearbeitung von Hilfskraftverträgen durch die Abteilung III-2 der Personalabteilung der Universität Ulm

Nach einer erfolgreichen Überprüfung der Unterlagen (auch an Hand der Personalmappe) werden die Vertragskosten von diesem berechnet und die Vertragsdaten in die Nachbarsysteme MBS (Mittelbewirtschaftung) und SVA (Personalverwaltung) eingepflegt.

Im gleichen Zuge wird ein Arbeitsblatt erstellt und Erledigungsvermerke geschrieben. Diese gehen dann zusammen mit den sonstigen Unterlagen in der Personalmappe zurück in die Registratur.

Begleitend hierzu wird der Vertrag durch Unterschrift gültig gemacht und die Auszahlung durch die LBV (Landesamt für Besoldung und Versorgung) veranlasst.

Handelt es sich um eine Wiederaufnahme eines Falles, bei dem Unterlagen fehlten, so beginnt der Aktivitätsfluss im Zustand “Unterlagen vollständig”. Der Eingangszustand des Diagramms kennt hierzu zwei alternative Übergänge, die jeweils mit der zutreffenden Voraussetzungs-Bedingung beschriftet sind (“[neuer Vertrag]” oder “[Wiederaufnahme einer Vertragsbearbeitung]”).

Um auch den Kontext einzelner Aktivitäten zu konkretisieren, können in das Aktivitätsdiagramm auch statische Interaktionsbeziehungen mit aufgenommen werden. Hierzu werden die Interaktionspfeile wie im Tätigkeitsprofil auch mit einer gestrichelten Linie gezeichnet, die Aktivität und externen Interaktionspartner verbindet.

Ein solcher Partner kann z.B. ein technisches System wie im Diagramm das “Mittel Bewirtschaftungs System” (MBS) sein oder menschliche Organisationseinheiten wie z.B. das “Landesamt für Besoldung und Versorgung” (LBV).

Die Aktivität “Personalmappe holen” ist beispielsweise ein Auftrag, welcher der Registratur gegeben wird. Fehlende Unterlagen werden angefordert, indem der zuständigen Fakultät ein Anforderungsblatt zugestellt wird etc.

Interaktions- wie auch Wechselpfeile können mit Konzepten beschriftet werden, die beim Wechsel bzw. der Interaktion “transportiert” werden. Im Diagramm findet sich hier beispielsweise die Personalmappe, die von der Registratur geliefert und zur Prüfung der Unterlagen herangezogen wird.

Interaktionsbeziehungen können auch reflexiv bzw. “intern” sein, in welchem Fall dann der eigentliche Partner am Ende des Interaktionspfeils ausgelassen und grafisch durch die kommunizierten Konzepte ersetzt wird.

> Im Rahmen der Aktivität “Kosten kalkulieren” informiert sich der Sachbearbeiter z.B. über den Haushaltsansatz der dem Vertrag zugeordneten Kostenstelle, über den Prüfungsstatus und den Weihnachtsgeldanspruch der Hilfskraft, um die Kosten des Vertrages zu ermitteln und diesen genehmigen zu können.

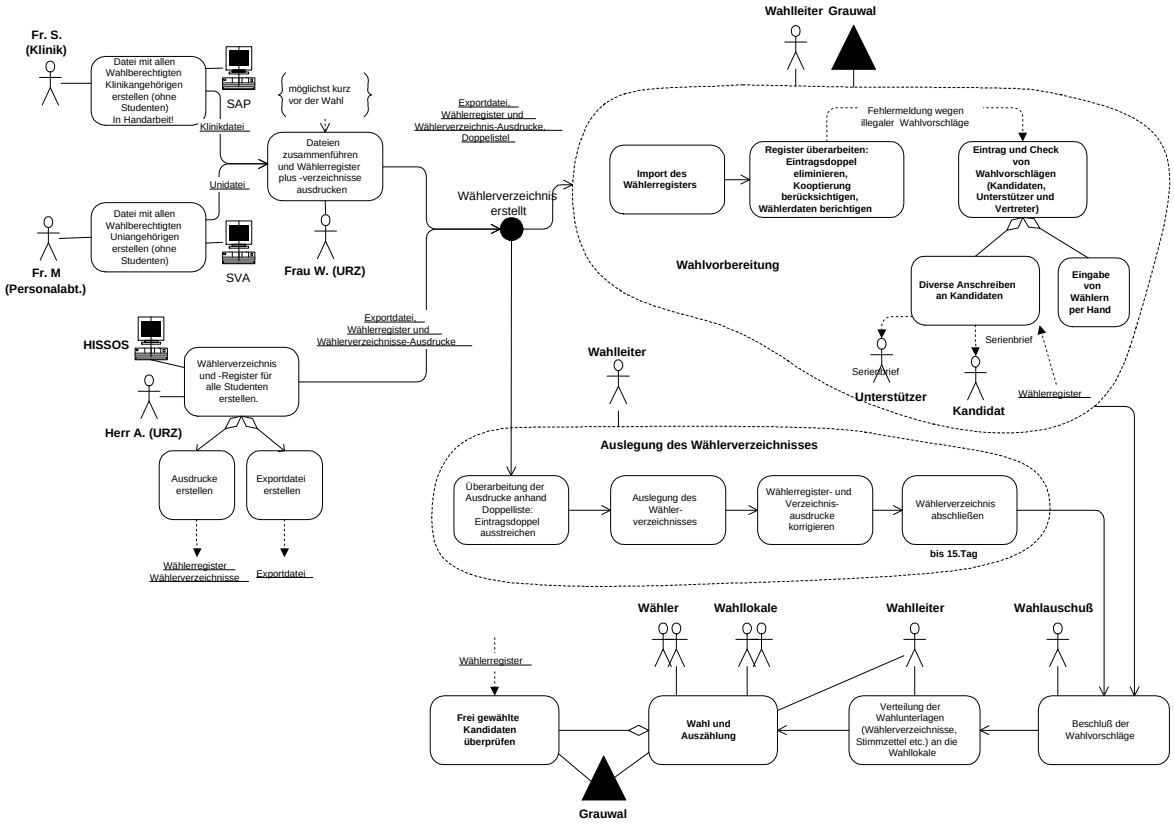


Abbildung 4.9.: Ein Aktivitätsdiagramm zur Darstellung der Gremienwahldurchführung nach Einführung des Systems und im besonderen Kontext des Wählerverzeichnisses

Als weiteres Beispiel zeigt Abb. 4.9 ein spezialisiertes Aktivitätsdiagramm, das die Aktivitäten einer Gremienwahl nach Einführung des neuen Softwaresystems “Grauwal” aus einem bestimmten Gesichtspunkt beleuchtet, und zwar im Hinblick auf das Wählerverzeichnis. Das Diagramm verfeinert bzw. konkretisiert hierzu bestimmte Aktivitäten, um Ursprung, Bedeutung und Verwendung des Wählerverzeichnisses deutlich zu machen. Konkretisierungen werden dabei entweder durch Aggregationen oder durch Angabe von Aktivitätsmustern ausgedrückt.

Im Diagramm findet sich hier als Beispiel die Erstellung des Studenten-Wählerverzeichnisses, das in die zwei aggregierten Aktivitäten Ausdruck und Dateierstellung verfeinert wird. Die verschiedenen Aktivitäten bei der Auslegung des Wählerverzeichnisses durch den Wahlleiter werden durch eine gestrichelte Linie eingeschlossen, die dadurch eine übergeordnete Aktivität kennzeichnet. Der von dieser Linie eingeschlossene Teil des Aktivitätsmusters bildet damit ein eigenes Aktivitätsmuster, das die Durchführung der Aktivität "Auslegung" konkretisiert.

Durch diese Schreibweise wird auch implizit ausgedrückt, dass alle eingeschlossenen Aktivitäten zusätzlich zu etwaigen lokalen Akteuren auch die Akteure der übergeordneten Aktivität besitzen. Zu einer Aktivität können dabei durchaus auch mehrere Teilmuster angegeben werden, die die Aktivität nur unvollständig konkretisieren. In der Abbildung sind beispielsweise nur die Aktivitäten der Wahlvorbereitung verfeinert worden, die im Hinblick auf das Wählerverzeichnis relevant sind.

Diskussion

KoA$_L$As Aktivitätsdiagramm soll im Gegensatz zum UML-Aktivitätsdiagramm keinen prozeduralen kausal-logischen *Ablauf* ([HK99]S.162) innerhalb eines technischen Systems präskriptiv *vorschreiben*, sondern es soll den möglichen *Handlungsspielraum* der autonomen Akteure eines soziotechnischen Systems deskriptiv *beschreiben*. KoA$_L$As Aktivitätsdiagramme sind darum nichtdeterministisch, kennen Kooperationen und Aktivitätsaggregationen: Letztendlich entscheiden die Akteure selbst über ihr Vorgehen und diese Entscheidung ist immer wesentlich von der real vorgefundenen Situation abhängig, nicht nur von einem etwaigen vereinbarten oder vorgesehenen Handlungsplan. Kooperationen sind Aktivitäten mit mehreren Akteuren, deren spezifischer Anteil an der gemeinsamen Aktivität nicht weiter konkretisiert wird. Aktivitäten sind darüber hinaus in der Regel eher inhaltlich beschrieben, so dass ihre genaue Operationalisierung bzw. Ablauf normalerweise weitgehend offen bleibt. Aggregationen schließlich können verwendet werden, um die Abstraktionsebene der Modellierung beliebig zu mischen oder auch nur teilweise zu konkretisieren, je nach darzustellender Sicht.

KoA$_L$As Aktivitätsdiagramme stellen keine technischen Spezifikationen dar, sondern dienen als Kommunikationsmedium für Gestaltungsprozesse mit Anwendern des zukünftigen Softwaresystems. Die verwendeten grafischen Modellierungselemente sind daher bei KoA$_L$A weitgehend vereinfacht und reduziert. Das Konzept "Zustand" bzw. die "Aktivität" wird zusätzlich auch für das Konzept "Ereignis" und "Synchronisationsbalken" verwendet. Es gibt keine zusätzlichen Modellierungskonzepte wie den Verzweigungs-Diamanten, Start oder Endzustände von UML. Die Diagrammelemente sind desweiteren mit allen bekannten Elementen der anderen Modelle kombinierbar.

4.4.4. Zusammenfassung

Zur Modellierung von Kooperationen zwischen den Akteuren eines soziotechnischen Systems stellt KoA$_L$A insgesamt drei verschiedene Modellarten zur Verfügung.

In der *Kooperationsstruktur* werden die Kooperationen aus einer statischen Sicht heraus dargestellt, während Konversations- und Aktivitätsmuster die dynamischen Aspekte dieser Kooperationen beleuchten.

Konversations- und *Aktivitätsdiagramme* stellen das praktische Arbeitshandeln aus orthogonalen bzw. komplementären Sichten dar.

Konversationen stellen die zu beobachtende Arbeitspraxis als Sequenzen diskreter Ereignisse dar, die aus dem "spontanen" (ergo nichtdeterministischen) Verhalten autonomer Akteure resultieren. Sie versuchen, das Gebiet *möglichen* Verhaltens umfassend zu dokumentieren, indem sie sich auf die charakteristischen Kommunikationsabläufe konzentrieren. Durch die Modellierung von Konversationen erzielt man eine abstrakte Darstellung des motivationalen Rahmens der zu beobachtenden Tätigkeit. Das soziotechnische System wird dadurch (im Sinne der Systemtheorie) als Menge von Akteuren modelliert, die sich durch Interaktionen und gegenseitigen Erwartungshaltungen gegenseitig beeinflussen. Die zu beobachtende reale Tätigkeit ist aus den Interaktionen, den Erwartungshaltungen und dem subjektiven Tätigkeitsprofil ableitbar.

Aktivitätsmuster schließlich beschreiben diese realen Tätigkeiten als geordnete bzw. musterhafte Abfolgen kontinuierlicher Aktivitätsbereiche, die alleine oder kooperativ durchgeführt werden. Im Gegensatz zu den Konversationsmustern kennt ein Aktivitätsmuster dabei auch nebenläufige Handlungsfäden. Obwohl Aktivitätsdiagramme gut zur präskriptiven Festlegung von Handlungsplänen oder "Arbeitsabläufen" (bzw. "Geschäftsprozessen") herangezogen werden können ("Was soll getan werden? Was wird normalerweise getan?"), können sie ähnlich zu den Konversationsdiagrammen auch zur Beschreibung nichtdeterministischer emergenter Aktivität genutzt werden ("Was kann alles geschehen?"). Diese unterscheidet KoA_LAs Diagramme von den Activity-, Sequence- oder Collaboration-Diagrammen von UML, die ausschließlich zur deterministischen Spezifikation des operationalen Verhaltens eines technischen Artefakts dienen, und darum keinen Raum für freie Entscheidungen, Auslassungen und andere Nichtdeterminismen bieten.

Als Hintergrund zum Verständnis der zu beobachtenden Aktivitäten kennt KoA_LA damit zusammenfassend folgende Perspektiven bzw. Kontexte von Aktivitäten:

Akteur: Eine Organisationseinheit, welche eine Aktivität durchführt oder an einer Kooperation teilnimmt und seine Verfeinerungs- (Organisationsstrukturen) bzw. Generalisierungsstruktur (Typisierung, Rollen)

Begriff: Ein mentales Konzept, das die Aktivität begleitet oder das Gegenstand der Aktivität ist

Aktivität: Ein Tätigkeitsbereich eines oder mehrerer Akteure und seine Verfeinerungs- (Teilaktivität bzw. übergeordnete Aktivität) bzw. Generalisierungsstruktur (Ausnahmebehandlungen)

Interaktion: Eine kommunikative Kooperation zweier oder mehrerer Akteure und ihre Verfeinerungs- bzw. Generalisierungsstruktur

Bedingung: Eine Bedingung, die einen Akteur allgemein beeinflusst oder speziell bei der Durchführung einer Aktivität oder Interaktion zu beachten ist: Randbedingung, Motiv, Ziel, Anforderung, Aufgabe etc.

Aktivitätsmuster: Eine sich regelmäßig wiederholende Abfolge von Einzeltätigkeiten

Konversation: Eine sich regelmäßig wiederholende Abfolge von Interaktionen

5. Die Vorgehensweise von Koala

"[This] divorce of art from technology is completely unnatural.
It's just that it's gone on so long you have to be an archaeologist
to find out where the two separated."

Pirsig[1]

In den folgenden Abschnitten wird nun eine Möglichkeit beschrieben, wie man in der ersten Phase der Softwareentwicklung – dem "Kontextdesign" – zur Ausgestaltung von Informationssystemen schrittweise vorgehen kann.

Ich nenne KoA_LA bewusst eine *Vorgehensweise*, um mich damit von der Standardbedeutung einer "Methode" abzugrenzen. Der Begriff Methode wird normalerweise mit einer weitgehend abgeschlossenen und wiederholbaren formalisierten Verfahrensweise zur Erreichung definierter Entwicklungsziele assoziiert. Die Rolle einer Methode im Umfeld eines Softwareentwicklungsprozesse ist jedoch vielmehr die eines epistemologischen Rahmens der Definitions- und Entwicklungsprozesse [AWHVW98]. Ein Methode ist laut Avison eher als eine Metapher anzusehen, die in einer bestimmten Entwicklungssituation immer wieder neu zu interpretieren und weiterzuentwickeln ist, anstatt sie als eine fertige präskriptive Beschreibung realer Aktivitäten aufzufassen.

Ich beschreibe im Folgenden einen weitgehend offenen Handlungsrahmen, der zwar die wichtigsten Aktivitäten des Kontextdesigns aufzeigt und sie durch ihre Ergebnisse spezifiziert, sie in ihrer Durchführungsreihenfolge und einer genaueren Tätigkeitsbeschreibung aber weitgehend offen lässt[2].

Obwohl ich eine Reihenfolge vorschlage, können viele der geschilderten Aktivitäten in Wirklichkeit auch überlappend oder alternierend durchgeführt werden, genauso wie sich in der praktischen Projektarbeit auch Tätigkeiten des Softwaredesigns mit denen des Kontextdesigns im Allgemeinen mischen werden. Dies ist letztlich darauf zurückzuführen, dass sich Diskussionen mit Anwendern nie wirklich streng auf ein Themengebiet fokussieren lassen und in Diskussionen über die alte und/oder neue Anwendungsumgebung automatisch immer auch Ideen oder spezifische Details des zukünftigen IS thematisiert werden.

Die folgende "Anleitung" beschreibt auf Grundlage von Einzelaktivitäten *ein* mögliches Aktivitätsmuster für das konkrete Kontextdesign. Ein reales Prozedere kann sich zwar an diesem "roten Faden" ausrichten, ist letztendlich aber natürlich von der aktuellen Aufgabenstellung abhängig. Demzufolge ist die anschließende Beschreibung eines "musterhaften" Vorgehens auch tendenziell auf das gewählte Fallbeispiel ausgerichtet.

[1] Zen and the Art of Motorcycle Maintenance, 1974.

[2] Ganz im Einklang mit Herbert Weber: "*Ich denke inzwischen, dass 'Softwareprozesse' ein in die Sackgasse führendes Konzept sind. [...] Software Engineering zu organisieren, entspricht für mich heute der Aufgabe 'Communicating Communities' zu organisieren*" [Web01]

Als Beispiel zur Darstellung der einzelnen Ergebnisse verwende ich das bereits bekannte Projekt "Apvel", in dem ein Softwaresystem zur Verwaltung und Verteilung der Ärzte im Praktischen Jahr zu realisieren war (siehe Kap. 1).

Die Beschreibung der Einzelaktivitäten ist auf eine Erläuterung der Aktivitätsziele fokussiert und nicht auf die Spezifikation ihres konkreten Ablaufs, d.h. ihrer verfeinerten Einzelschritte. Insbesondere finden sich hier im Allgemeinen keine spezifischen Angaben über die Organisation der Einzelaktivitäten bzw. den Einsatz bestimmter Techniken. Viele der Aktivitäten bedingen eine enge Zusammenarbeit zwischen Anwendern und Entwicklern. Wie man diese Zusammenarbeit bzw. Ergebnisse erzielen kann – über Interviews, Workshops, regelmäßige Treffen, Fragebögen, ethnografische Studien etc. – soll kein Schwerpunkt der Darstellung sein.

Eine grobe Verfeinerung der Aktivitäten findet sich im Folgenden aber in Form von "*Leitfragen*" wieder, die Teilziele der beschriebenen Einzelschritte des Kontextdesigns definieren.

5.1. Überblick

KoA$_{\text{L}}$A begreift die Gestaltung eines Informationssystems als ein iteratives und evolutionäres Vorgehen, das stark modellgestützt durchgeführt wird. Beginnend mit der Modellierung der aktuellen Arbeitssituation wird sukzessive der Anwendungskontext des neuen Softwaresystems umgestaltet bzw. entworfen, indem das neue IS selbst dort als neuer Akteur eingeführt und an Hand des Modells des neuen Anwendungskontexts validiert wird.

Abbildung 5.1 stellt die Vorgehensweise von KoA$_{\text{L}}$A in Form eines Aktivitätsdiagramms dar:

Nach der Untersuchung der Motivation für das Projekt wird zunächst das Projektziel in Form einer gemeinsamen Vision dokumentiert. Nach der Analyse des Ist-Zustands ("Kontextanalyse"; siehe verfeinertes Aktivitätsdiagramm in Abb. 5.5, Kap. 5.4) wird das System als neuer Akteur definiert ("Systemgestaltung"; siehe verfeinertes Aktivitätsdiagramm in Abb. 5.12, Kap. 5.5) und in seinem Anwendungskontext dargestellt ("Fiktion"; siehe verfeinertes Aktivitätsdiagramm in Abb. 5.17, Kap. 5.6).

Die Gestaltung findet in iterativen Schritten statt, solange bis das dokumentierte Systemprofil seine endgültige Form erhalten hat und von allen Interessengruppen akzeptiert wird.

KoA$_{\text{L}}$As Modelle dienen sowohl der Entwicklung als auch der Validierung der Gestalt eines IS. Entwicklungsziel ist dabei die Definition des Aktivitätsprofils des Softwaresystems inkl. der Beschreibung seines Interaktions- und Kooperationsverhaltens in Bezug auf seinen konkreten Anwendungskontext.

Validiert wird die Systemgestalt durch die Interessengruppen, indem sie an Hand der erstellten Modelle die "Passgenauigkeit" des Systems in Bezug auf seinen Anwendungskontext bewerten. Hierbei sind in erster Linie die Aktivitäten aller betroffener Akteure und darüber hinaus die Erwartungshaltungen aller Interessengruppen relevant.

Bei den Akteuren der im Folgenden beschriebenen Einzelaktivitäten unterscheide ich in der weiteren Dokumentation der Vorgehensweise nach KoA$_{\text{L}}$A folgende Personengruppen bzw. Projektrollen:

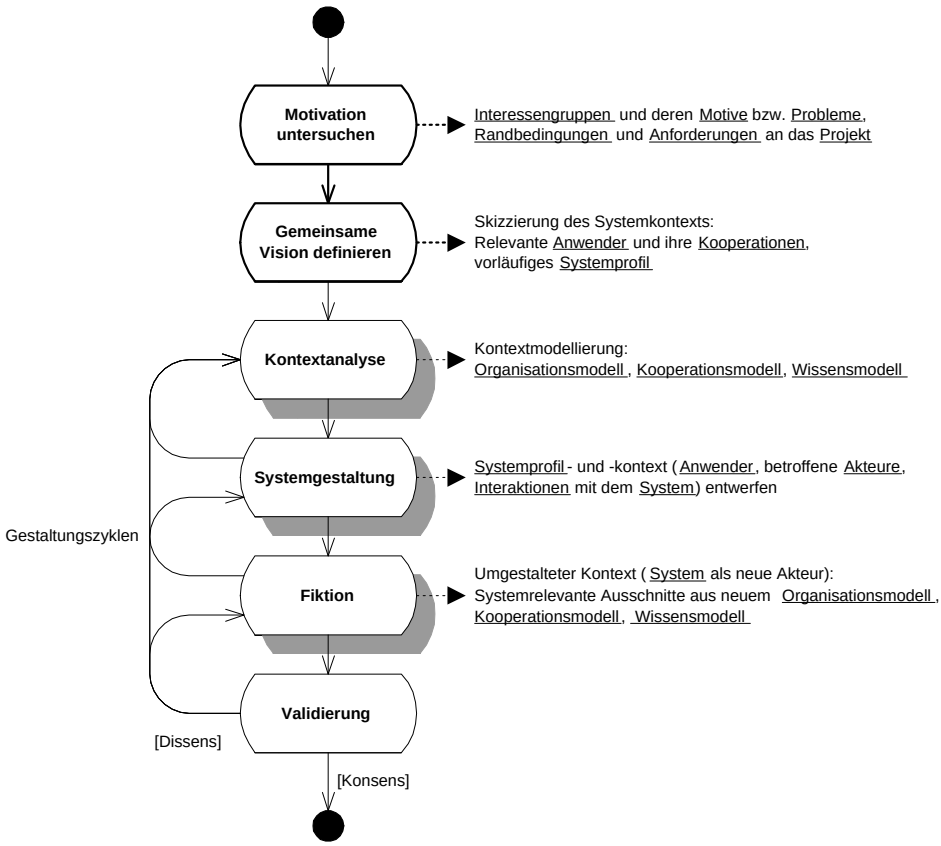


Abbildung 5.1.: Die Vorgehensweise nach KoA$_L$A

Interessengruppen Die Gruppe der Vertreter aller derjenigen, die ein eigenes Interesse am zukünftigen Softwaresystem oder seiner Entwicklung anmelden. Die Gruppe umfasst potentiell alle vom System direkt oder indirekt Betroffenen wie z.B. die Auftraggeber, das Management, etwaige Kunden, die Marketingabteilung, aber auch die Entwickler, die Wartungsprogrammierer etc. In der Literatur werden die Mitglieder der Interessengruppen auch als "Stakeholder" bezeichnet. Eine besonders wichtige Interessengruppe stellt im Allgemeinen die Gruppe der direkten Anwender dar (siehe nächster Abschnitt).

Anwender Diese Gruppe umfasst alle menschlichen Akteure, die in ihrer praktischen Arbeit direkt oder indirekt vom Informationssystem betroffen sind. Das sind sowohl die direkten Nutzer des neuen Softwaresystems als auch andere Akteure, mit denen diese Nutzer kooperieren bzw. die in ihrer Tätigkeit von Ergebnissen, Aktivitäten oder Erfordernissen des neuen Softwaresystems beeinflusst werden.

Entwickler Damit werden alle Personen bezeichnet, die in der Rolle des Auftragnehmers die Entwicklung des neuen Softwaresystems betreiben. Diese Rolle umfasst damit alle Analysten, Designer, Programmierer etc., die das neue IS konkret entwerfen und realisieren wollen.

Die Entwickler wenden KoA$_L$A an, um den neuen Aktivitätskontext des IS in enger Zusammenarbeit mit Anwendern und anderen Interessengruppen auszugestalten. Das reale Auftreten der Entwickler bzw. ihre Rolle hängt dabei vom vorherrschenden Gestaltungs-

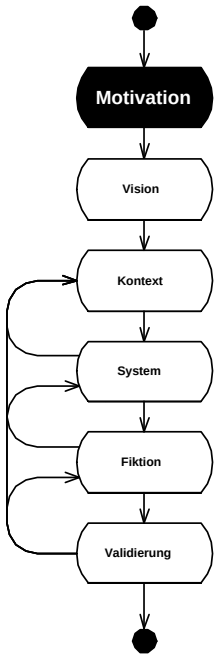
Paradigma ab und kann das ganze Spektrum vom “technical expert” , über den “facilitator” , den “agent for social progress” bis hin zum “change catalyst” abdecken [HK89].

Die Modelle von KoA$_L$A werden im Allgemeinen immer auch grafisch in Form von Diagrammen veranschaulicht. Die grundlegende und vollständige Dokumentationsform ist jedoch die ausführliche textuelle Beschreibung aller Modellelemente und ihrer Beziehungen in der Projektdatenbank.

Aus Gründen der Übersichtlichkeit liegt der Schwerpunkt in der folgenden Darstellung aber nichtsdestotrotz auf der grafischen Beschreibung in Form textuell erläuterter Bespieldiagramme.

Beispiele für die mögliche Form einer vollständigen semiformalen textuellen Dokumentation von KoA$_L$A-Modellen finden sich im Anhang (siehe Kap. B).

5.2. Motivation



Ziel: Identifikation aller direkt oder indirekt vom Projekt betroffenen Interessengruppen und deren projektrelevante Motive bzw. Probleme, Wünsche und Ziele

Die erste Aufgabe im Kontextdesign ist die Dokumentation des Projektkontexts.

Das Projekt selbst ist eine Aktivität, die zum Einen durch ein definiertes Projektteam durchgeführt wird und zum Anderen bestimmten Randbedingungen unterliegt, die im Wesentlichen von den betroffenen Interessengruppen diktiert werden. Deren Wünsche, Ansprüche, Probleme und Ziele bilden den Aktivitätskontext, in dem das Projekt durchgeführt wird. Dieser Kontext beinhaltet alle Motive der Projektarbeit.[3] Ein Projekt ist nur dann erfolgreich, wenn seine Ergebnisse den wahren Bedürfnissen gerecht werden. Eine umfassende Untersuchung der projektbezogenen Erwartungshaltung ist daher unerlässlich, auch wenn diese Erwartungshaltung

[3] Karl Wiegers spricht in diesem Zusammenhang auch von “Business Requirements”[Wie99], Boehm von “Win-Conditions”[BBHL94].

später nur zum Teil in Form der Spezifikation des anvisierten Softwaresystems objektiviert werden wird.

Die zu berücksichtigenden Interessengruppen sind nicht unbedingt in ihrer konkreten Arbeitspraxis vom Informationssystem betroffen, genau wie sie auch nicht zwangsläufig an der konkreten Ausgestaltung des Softwaresystems beteiligt werden. Alle Interessengruppenvertreter ("stakeholder") zusammen stellen jedoch die eigentlichen (direkten oder indirekten) Auftraggeber des Projekts dar und definieren den grundlegenden Anforderungsrahmen, der bei der weiteren Ausgestaltung implizit zu berücksichtigen sein wird bzw. dem die noch zu treffenden Entwurfsentscheidungen unterzuordnen und in den die in der Gestaltung noch zu definierenden Anforderungen einzupassen sind.

Es wird zunächst dieser Projektkontext analysiert, bevor dann im nächsten Schritt der Projektarbeit der Anwendungskontext näher untersucht wird, in dem das neue Softwaresystem eingesetzt werden soll.

Zur Modellierung

KoA$_{\text{L}}$A modelliert den Projektkontext in Form eines statischen Kooperationsmodells (siehe Kap. 4.4.1) mit allen Interessengruppen als Akteuren und ihren Anforderungen an das Projekt, ihren Motiven und Zielen als Bedingungen, die sie an den virtuellen Akteur "Projekt" stellen.

Das Projekt kann dabei weiteren "anonymen" sonstigen Randbedingungen und einschränkenden Sachzwängen unterliegen. Die Motive für die Entwicklung des Informationssystems selbst können damit in Form von Problemen, Zielsetzungen oder auch Wünschen der Akteure thematisiert werden. Damit werden sowohl die übergeordneten Anforderungen als auch globale Randbedingungen der Projektarbeit explizit gemacht.

Abb. 5.2 zeigt ein Kontextdiagramm für das Fallbeispiel Apvel:

> Im Projekt sind mehrere Interessengruppen zu berücksichtigen. Neben dem eigentlichen Kunden (die Leiterin des Studiensekretariats) und der zukünftigen Nutzerin des Systems (Frau L.) ist auch die Systembetreuung und die Entwicklungsmannschaft selbst in Betracht zu ziehen.
>
> Ursprung und Grund für die Neurealisierung ist eine Änderung der der Zuteilung zugrundeliegenden Verteilordnung. Die der Studiumsorganisation zugrundeliegende Studienordnung bzw. Verteilordnung ist vollständig zu erfüllen.
>
> Bei der Realisierung ist zu berücksichtigen, dass sich die Nutzerin des neuen Systems eine Arbeitserleichterung im Vergleich zur Nutzung der alten Anwendung erhofft und den Entwicklern im aktuellen Projektkontext außerdem nur ein Zeitraum von vier Personenmonaten zur Verfügung steht[4].

Zum Vorgehen

Um den Kontext des Projekts zu definieren, sind alle Interessengruppen zu bestimmen, zu befragen und zur Mitarbeit einzuladen. In den ersten gemeinsamen Diskussionen werden hier bereits unterschiedliche Standpunkte und Sichtweisen aufgedeckt und Konflikte thematisiert.

[4] Das Projekt wird im Rahmen einer Lehrveranstaltung (das "Softwaregrundpraktikum" im dritten Semester des Informatik-Studiengangs der Universität Ulm) durchgeführt, für die nur begrenzte Mittel bzw. Zeit zur Verfügung stehen.

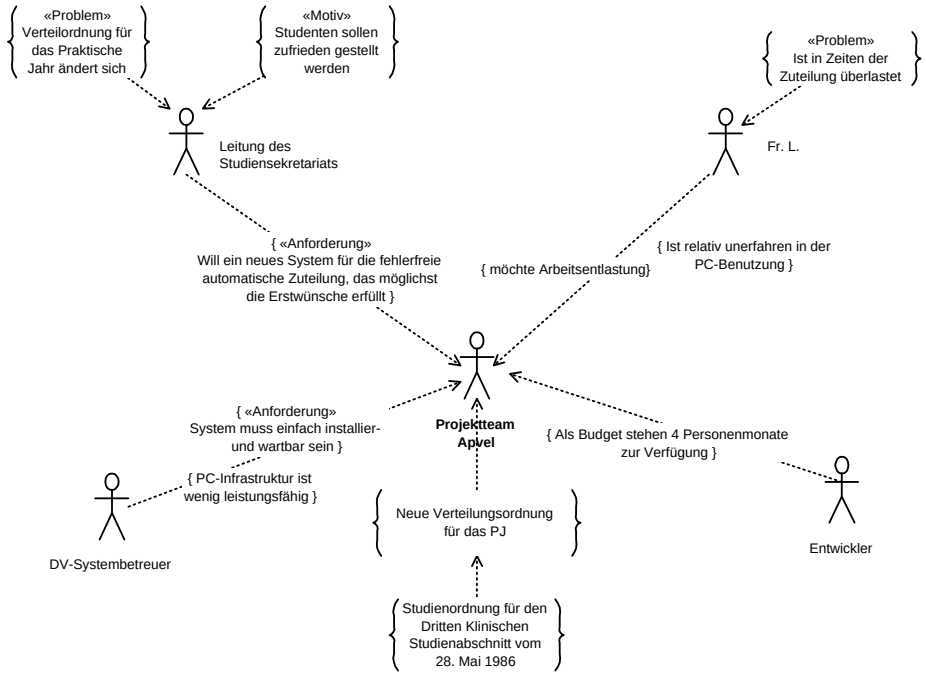


Abbildung 5.2.: Kontextdiagramm zur Darstellung des Projektkontexts

Es gilt die Problemsituation des Projekts selbst zu analysieren. Ein möglicher Rahmen für diese ersten Besprechungen ist z.B. ein Workshop, zu dem alle interessierten Parteien gemeinsam zu einem ersten “Brainstorming” eingeladen werden.

Im Einzelnen sind folgende Leitfragen zu klären[5]:

Warum soll überhaupt ein neues Informationssystem entwickelt werden?

Was sind die initialen Gründe zur Begründung des Projekts? Wie, Warum und Wann ist das Projekt initiiert worden?

Im Beispiel: Die eigentliche Motivation für das Projekt besteht darin, dass sich die Verteilordnung für das Praktische Jahr geändert hat und das aktuell eingesetzte System nicht mehr entsprechend angepasst werden kann.

Welche Interessengruppen gibt es?

Wer ist vom System betroffen? Wer meldet Ansprüche an? Wer kann die Interessen der verschiedenen Gruppierungen vertreten?

Es gilt die Akteure zu identifizieren, die Anforderungen an die Projektarbeit stellen (dürfen). Damit beginnt implizit bereits ein Teil der Organisationsmodellierung.

[5]Die Leitfragen zu einem Abschnitt des Kontextdesigns sind zwar jeweils in eine Reihenfolge gebracht, in der sie sinnvollerweise auch zu bearbeiten sind, sie können allerdings natürlich je nach den spezifischen Erfordernissen des Projekts auch in beliebig anderer Reihenfolge bzw. in einer spezifischen Auswahl betrachtet werden. Darüber hinaus listen diese Aufstellungen zwar die wichtigsten Teilaufgaben eines Getaltungsschritts auf, sie sind jedoch darin auch nicht zwingend vollständig.

Im Beispiel: Auftraggeber ist das Studiensekretariat. Neben der zukünftigen Anwenderin, Frau L. sind auch noch die DV-Systembetreuer und die von den Entwicklern vorgegebenen Randbedingungen zu berücksichtigen.

Was sind ihre Motive zur Mitarbeit am Projekt?

Welche projektrelevanten Interessen besitzen diese Gruppen? Welche Motive, Wünsche, Probleme oder Ziele veranlassen sie zur Mitarbeit am Projekt? Wie sehen die Hintergründe für die von den Gruppen angemeldeten Forderungen an das Projekt aus?

Ein Wissen um die motivationale Ausgangslage und den subjektiven Interessen dient dazu, das gegenseitige Verständnis zu fördern und etwaige Konfliktherde zu entschärfen.

Im Beispiel: Die Leitung des Studiensekretariats will ein System für die möglichst automatische fehlerfreie Zuteilung, das möglichst viele Erstwünsche erfüllt. Frau L. möchte ein System, dass sie vor allem in Stoßzeiten mehr als das alte System entlastet etc.

Welche konkreten Anforderungen stellen diese Gruppen an das Projekt?

Welche offiziellen allgemeinen (Wünsche, Erwartungen) oder konkreten Anforderungen (z.B. Resourcen wie Kosten, Zeit, Personen; gesetzliche Regelungen etc.) stellen diese Gruppen an das *Projekt?*

Hierbei steht nicht das konkrete Softwaresystem im Vordergrund sondern das Projekt und seine Ziele. Konkrete Vorstellungen über "features" des System sind vor allem Gegenstand des nächsten Schrittes, der "Vision" (siehe nächster Abschnitt). Bei der Untersuchung der Anforderungen ist hier nicht nur die "Kundenseite" sondern auch die Entwicklerseite zu berücksichtigen.

Ergebnis der Klärung dieser Erwartungshaltung sollten die initialen (Rand-)Bedingungen sein, die von allen externen Akteuren gestellt werden. Sie stellen eine Art "Stoffsammlung" dar, die an dieser Stelle noch nicht weiter "analysiert" bzw. konkretisiert oder widerspruchsfrei gemacht werden. Der Gestaltungsraum ist damit zwar noch offen, allerdings schon einmal vorläufig eingegrenzt.

Im Beispiel: Frau L. möchte ein System, das einen möglichst geringen Einarbeitungsaufwand verursacht, die Entwickler haben für die Realisierung nur vier Personenmonate Zeit, etc.

Welchen allgemeinen Randbedingungen untersteht das Projekt?

Welchen sonstigen "anonymen" Anforderungen – das sind Anforderungen ohne einen sie direkt einfordernden Interessensvertreter – unterliegt das Projekt?

Beispiel hierfür sind Gesetze, Verordnungen, Regeln, "business rules" etc. Diese Anforderungen sind oft implizit, d.h. sie werden oft nicht direkt durch Interessengruppenvertreter geäußert, sondern müssen von den Entwicklern aktiv aufgedeckt bzw. explizit gemacht werden.

Im Beispiel: Die neue Verteilordnung ist vollständig zu erfüllen, das Projekt ist spätestens zum Start des nächsten Praktischen Jahres fertigzustellen, etc.

KoA$_{\text{L}}$A beginnt mit den Motiven, Problemen und Zielen des Projekts, also auf einer abstrakten Ebene, ohne bereits konkrete Anforderungen an das IS festzulegen. Dies ist umso wichtiger, als

Benutzer oft keine spezifischen Anforderungen an ein IS spezifizieren können, weil sie dessen Möglichkeiten, Einschränkungen, zukünftige Nutzungssituation nicht kennen oder antizipieren können oder Folgen auf ihre Arbeitswelt nicht überblicken. Nutzer haben eher ein "Problem", das der "Kollege Computer" lösen soll, können aber nicht angeben, wie. Ausgehend von einer Analyse der Probleme, Ziele, Randbedingungen kann der Analyst aber in der Folge (s.u.) iterativ mit dem Nutzer zusammen eine Idee für das System entwickeln. Hauptaufgabe des Analysten ist es dabei, die Vorstellungskraft des Nutzers zu fördern.

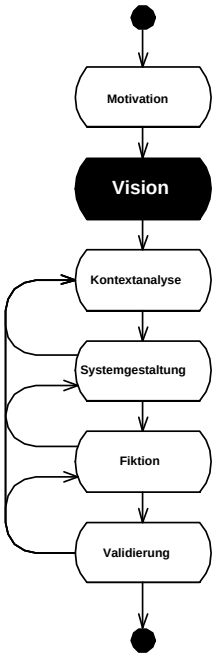
In der Modellierung kann potentiell auch "anonym" mit den Motiven oder Problemen begonnen werden, die im Vordergrund stehen, ohne gleich eine Zuordnung zu Interessengruppen zu versuchen. Dies ist vor allem nützlich, um die ersten zentralen Wünsche bzw. Ideen für die Ausgestaltung des Informationssystems aufzunehmen und die Hauptprobleme zu erfassen, die das neue IS beseitigen oder lindern soll.

Im Anschluss an die Dokumentation des Projektkontexts wird das Team definiert, das den folgenden Gestaltungsprozess durchführen soll. Es werden dabei alle Personen festgelegt, die konkret an der Analyse und der Definition des Problems aktiv (Workshop-Teilnehmer, Validierungs-Gremien) oder passiv (Interview-Partner, Fragebogen-Zielgruppe, ethnografische "Studienobjekte" etc.) beteiligt werden sollen bzw. können. Das sind aus zeitlichen oder politischen Gründen natürlich oft nur eine Untermenge der oben definierten Interessengruppen. Essentielle Kandidaten auf Seiten des "Kunden" sind hierfür vor allem die realen Anwender und sonstige in ihrer praktischen Arbeitstätigkeit betroffenen Akteure. Auf Seiten des "Auftragnehmers" werden Entwickler ausgewählt, die im betroffenen Anwendungsgebiet möglichst über Vorwissen verfügen und den Gestaltungsprozess aufgrund ihrer "social skills" erfolgreich anleiten können.

Bei der Modellierung der Anforderungen an das Projekt werden unweigerlich bereits vereinzelt konkrete Anforderungen an das SW-System artikuliert werden. Dies führt die Modellierung nahtlos in die nun folgende Visionsphase über.

5.3. Vision

lat. visio, visionis: das Sehen, der Anblick, die Erscheinung



Ziel: Erste vorläufige Skizze des Softwaresystems und seiner Anwender- bzw. Nachbarsysteme

Der Auftraggeber besitzt vor dem Projektstart in der Regel bereits ein erstes Bild von der Ausgestaltung des IS, das er im Prozess der Entscheidungsfindung bei der Initiierung des Projekts entwickelt hat. Diese ersten Ideen, Vorstellungen und Wünsche sollten möglichst frühzeitig in einer Art "Ideensammlung" dokumentiert werden, indem der direkte Nutzungskontext des Systems und sein Profil vorneweg einmal skizziert wird.

Zur Modellierung

Um die initiale Idee für eine Eingliederung des IS in der Anwendungsumgebung zu beschreiben, wird die direkte Umgebung des Systems in einem Kontextdiagramm (siehe Kap. 4.4.1) dargestellt und anhand der Beziehungen der Akteure zueinander bzw. mit dem System ausschnittsweise skizziert. Erste Vorstellungen zum Aufgabenbereich des zukünftigen IS werden hingegen in Form eines Tätigkeitsprofils eines neuen technischen "System"-Akteurs beschrieben.

Die Modellierung umfasst damit sowohl Fragmente eines Organisationsmodells – das Tätigkeitsprofil des Systems – als auch Teile eines ersten Kooperationsmodells für das direkte Systemumfeld – die Interaktionsbeziehungen zwischen dem System und seinen externen Akteuren. Das Tätigkeitsprofil kann sowohl grafisch in Form eines Profildiagramms als auch textuell dokumentiert werden. Die Interaktionsbeziehungen hingegen werden in der Regel nur grafisch als Kontextdiagramm dargestellt.

Die einzelnen Modellierungskonzepte von KoA$_{\text{L}}$A werden in der Modellierung folgendermaßen eingesetzt:

Akteure Das System selbst bzw. unterscheidbare Komponenten des Systems werden als eigene technische Akteure dargestellt. Zusätzlich werden auch alle menschlichen oder technischen Akteure in das Modell mit aufgenommen, mit denen das System interagiert.

Aktivitäten An den "System"-Akteur können beliebige Aufgaben delegiert werden (der "Computer" als intelligente "Hilfskraft"). Diese Aufgaben werden im Profildiagramm des Systems als Aktivitäten modelliert. Komplexe Aktivitäten können hier in weitere Unteraktivitäten verfeinert oder in ihrem Ablauf in Form von Aktivitätsdiagrammen dokumentiert werden. Die meisten Aktivitäten stellen jedoch einfache Aufträge dar, die nicht mehr weiter verfeinert werden müssen.

Abstrakte Aktivitäten, die vom System und anderen Akteuren gemeinsam durchgeführt werden, können auch als Kooperationen im Kontextmodell aufgeführt werden. Das betrifft z.B. solche Aktivitäten, für die man noch keine akteurspezifische Aufteilung der einzelnen Teilaktivitäten vornehmen will.

Interaktionen Aufträge werden in der Regel von bestimmten Nutzergruppen initiiert und können daher auch gut in Form eines Kontextdiagramms dokumentiert werden. Darin werden die Auftraggeber nebst ihren Aufträgen bzw. Interaktionen und mit den in diesem Zusammenhang thematisierten fachlichen Begriffen bzw. betroffenen Informationsobjekten dokumentiert.

Bedingungen Motive, Anforderungen, Randbedingungen und Probleme (Bsp: "es darf nicht passieren, dass ...", "das darf nicht länger dauern als ...", "am liebsten wäre mir, dass ...", "wie haben oft Probleme mit ..."), die das System betreffen, werden im Kontextdiagramm oder im Tätigkeitsprofil des Systems als Bedingungen des entsprechenden Stereotyps modelliert.

Aktivitätsmuster Erste Vorstellungen zu Dialog- bzw. Anwendungsabläufen können in Form von Konversations- oder Aktivitätsmustern dokumentiert werden, die erste Beispiel-Nutzungssequenzen dokumentieren und damit ein vorläufiges Nutzungskonzept vorstellen.

Abb. 5.3 zeigt als Beispiel eine erste Skizze für den angedachten Kontext des neuen IS. Gezeigt wird ein Kontextdiagramm, das die wichtigsten Anwender des Systems und ihre zentralen Kooperationen bzw. Interaktionen aufführt:

> Die Leiterin des Studiensekretariats will am System nicht operativ arbeiten, sondern sich nur über den Stand der Bewerbung bzw. Zuteilung informieren.
>
> Die eigentliche Bearbeitung der Bewerbungen und die Verteilung wird von der Sekretärin durchgeführt.
>
> Alternativ zur Bewerbung per schriftlichem Hauptantrag können sich Studenten auch online per Webinterface bewerben, über das sie sich auch eine Auflistung der aktuellen Bewerbungen ansehen können.
>
> Die Notwendigkeit zur Synchronisierung des Datenbestands von Webserver und IS wird in Form einer gemeinsamen Aktivität vermerkt, aber noch nicht weiter ausgearbeitet.
>
> Das IS soll desweiteren eine Schnittstelle zum eingesetzten Textverarbeitungssystem realisieren, um Serienbriefe generieren zu können.

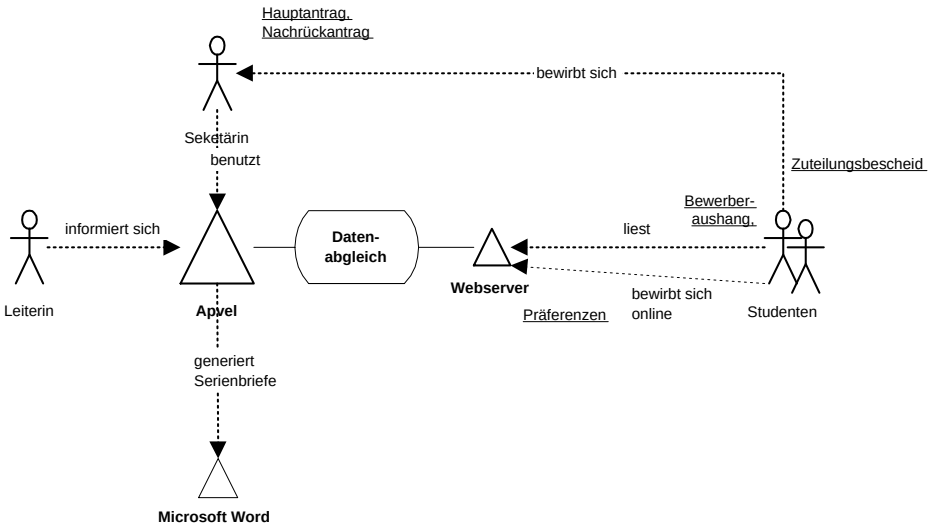


Abbildung 5.3.: Erste Skizze des Systemkontexts

Abb. 5.4 zeigt einen ersten Entwurf des Tätigkeitsprofils des angedachten Informationssystems. Wesentliche Bestandteile sind dabei die Aktivitäten, die das System leisten soll (seine "Aufgaben"): Neben der notwendigen Verwaltung der Stammdaten und der Generierung entsprechender Reports und Listen soll das System vor allem die automatische Zuteilung übernehmen. Ein ausgezeichneter Teilbereich dieser Aktivität ist die Durchführung der Nachverteilung.

Die Darstellung des Profils spiegelt die jeweiligen Sichtweisen aller Interessengruppen wieder. Sowohl die Auswahl der dargestellten Aktivitäten und ihre spezifische Verfeinerung, als auch die Annotationen in Form von Bedingungen etc. richten sich nach den subjektiven Bedeutungen, die von den Interessengruppen den jeweiligen Komponenten zugemessen werden.

Die Unteraktivität "Nachrückverfahren" wird beispielsweise deshalb eigens als Aktivität aufgeführt, da die Nachverteilung im alten System nur unzureichend berücksichtigt wurde. Dadurch, dass sie nun als eigenständige Aktivität des Systems definiert wird, tritt ihre Bedeutung für das neue System augenfälliger zutage.

Alle Aktivitäten und Bedingungen können optional gewichtet werden, um ihre Bedeutung zusätzlich zu kennzeichnen. Daraus wird ersichtlich, dass z.B. die statistischen Auswertungen oder die Generierung von Serienbriefen keine hohe Priorität besitzen und die jeweiligen Realisierungsarbeiten evtl. zurückgestellt werden könnten. Die automatische Zuteilung, das Nachrückverfahren und die anderen Reports hingegen werden als besonders wichtig eingestuft.

Zum Vorgehen

Das Softwaresystem wird als virtueller Akteur modelliert, der eine eigene "Stellenbeschreibung" erhält und ein eigenständiges Kommunikationsverhalten aufweist.

Dieser Ansatz wird der Beobachtung gerecht, dass Auftraggeber häufig die Tendenz besitzen, das System in ihrem Sprachgebrauch zu personifizieren: Das System "soll etwas bestimmtes

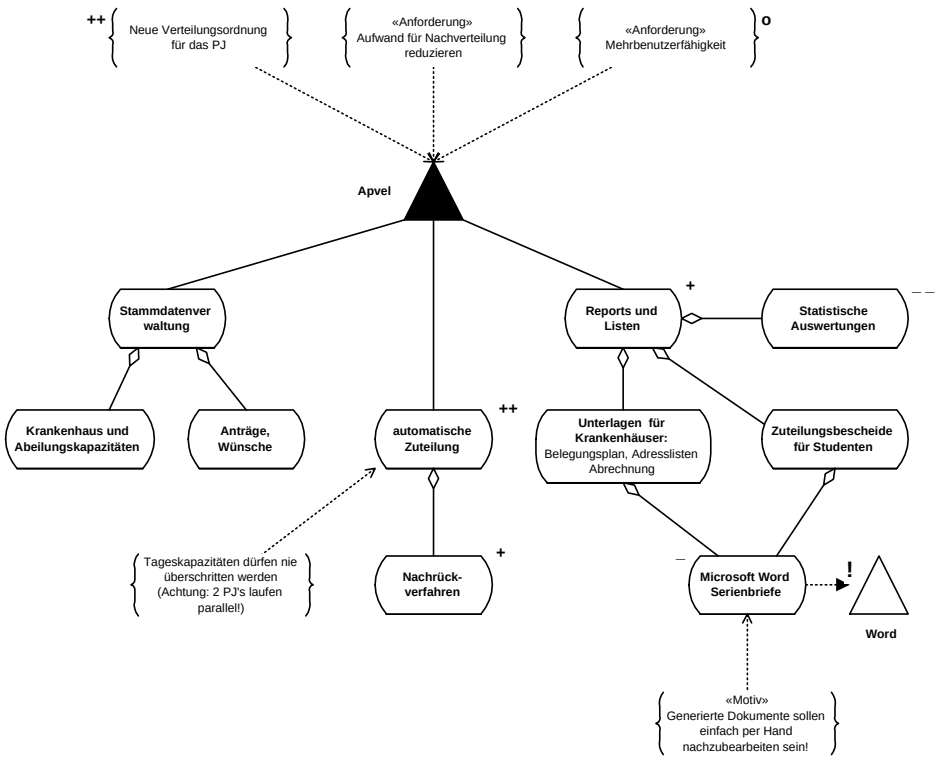


Abbildung 5.4.: Vorläufiges Profil des Informationssystems

tun", "hat die Aufgabe, ...", es muss eine bestimmte Leistung vollbringen etc. Optional kann das System dabei auch in verschiedene Komponenten "zerlegt" werden, die wiederum jeder für sich als Akteur mit spezifischem Tätigkeits- und Kommunikationsprofil dargestellt werden.

In enger Zusammenarbeit mit den Vertretern aller Interessengruppen wird hierzu zunächst die Umgebung des Systems in Form eines Kontextdiagramms skizziert, das die externen Akteure (Nutzer oder Fremdsysteme) in Beziehung zum System-Akteur setzt. Dabei werden sowohl Kooperationen (Aktivitäten mehrere Akteure zusammen) als auch Interaktionen zwischen System und den externen Akteuren dokumentiert. Die Kooperationen werden dabei in der Regel auf einer recht abstrakten und unspezifischen Ebene formuliert (z.B: Anwender "nutzt" das System). Sie können allerdings auch konkret werden, wenn es sich z.B. um ein bereits existierendes technisches Nachbarsystem handelt und die geforderte Interaktion bereits hinreichend feststeht (z.B. System transferiert XML-Datei per ftp auf den Webserver "www.verwaltung.uni-ulm.de").

Darauf folgend wird eine erste "Stellenbeschreibung" für den Softwaresystem-Akteur angelegt (rein textuell oder grafisch als Profildiagramm). Es wird dokumentiert, "was" das System tun soll, welchen Randbedingungen es dabei unterliegt und welche Prioritäten diese einzelnen Funktionen besitzen.

Um die Kreativität zu fördern, sollten sich die Entwickler in dieser Phase möglichst zurücknehmen, was Bemerkungen zur Machbarkeit, Sachzwängen, Kosten etc. anbetrifft. Die Interessenvertreter sollen ihre Vorstellungen frei äußern, ohne von den Entwicklern bereits in

bestimmte Bahnen (mögliche Lösungsvarianten) gelenkt zu werden und damit in ihrer Denkwelt tendenziell beeinflusst zu werden. Es handelt sich hier vorerst nur um die Vision einer Problemlösung, die erst in den folgenden Phasen schrittweise auf ihre Tragfähigkeit hin untersucht und entsprechend angepasst wird. Hauptaufgabe des Analytikers ist hier vor allem die Moderation, um zu einem gemeinsamen Konsens der Interessengruppen zu gelangen.

Nach einer ersten Modellierung des Systemprofils werden dessen Vorgaben schrittweise hinterfragt und zur Verfeinerung bzw. Abstraktion der Aktivitäten zu Einzeltätigkeiten oder übergreifenden Zielen des Systems benutzt. Auf diese Weise werden evtl. automatisch wieder einige Projektziele der Motivierungsphase angesprochen und es wird somit wieder der Projektkontext thematisiert. Die wesentliche Randbedingung "das System muss die neue Verteilordnung berücksichtigen" ist beispielsweise gleichzeitig auch eine wesentliche Motivation für das Projekt selbst, da das Altsystem die neuen Änderungen nicht berücksichtigt bzw. ein Umbau zu teuer sein würde.

Ein wichtiges Ziel dieser Phase ist die Etablierung einer *gemeinsamen* Anschauungsweise – einer *Vision* – über die grundlegenden Ziele, Leistungen und Randbedingungen des Systems, die zur Lösung und/oder Schlichtung von späteren Interessenskonflikten bei der konkreten Ausgestaltung benutzt werden kann. Die dokumentierten Ziele, Bedingungen und Anforderungen an das System sollten daher widerspruchsfrei sein. Auf dieser frühen und abstrakten Ebene ist also bereits jetzt ein Konsens aller Beteiligten zu etablieren.

Die spezifizierten Anforderungen sind noch keineswegs vollständig, sie beschreiben nur den *gemeinsamen* Ausgangspunkt für den nun folgenden Gestaltungsprozess. Ein Konsens gelingt auf Grundlage der allgemein gehaltenen Ziele noch sehr viel einfacher als auf Grundlage konkreter Einzelfunktionen, die das System bieten soll. Eine weitere Möglichkeit, Interessenkonflikte zu entschärfen, besteht in der gemeinsamen Priorisierung der verschiedenen Systemleistungen bzw. Randbedingungen. Dabei sind sowohl der Nutzen (Wichtigkeit aus Anwendungssicht) als auch die Kosten zu berücksichtigen. Weit auseinanderliegende Priorisierungen durch die Interessenvertreter sind dabei oft Anzeichen von Missverständnissen, die noch ausgeräumt werden müssen, um eine gemeinsame Vision zu "formen" [BBHL94].

Die Modellierung wird von folgenden Leitfragen geleitet:

Welche Systemkomponenten können unterschieden werden?

> Kann das System evtl. jetzt schon in mehrere weitgehend unabhängige Komponenten zerlegt werden? Spielen diese Systemteile evtl. unterschiedliche Rollen (Maschine, Werkzeug, autonomer intelligenter Akteur etc.)?
>
> Die Modellierung ist für jede dieser Komponenten getrennt durchzuführen.
>
> Im Beispiel: Das Informationssystem Apvel (als Mehrplatzapplikation) und die Webapplikation Apvel (zur Unterstützung der Bewerbung und als Publikationsmedium). Im Weiteren wird vor allem das Kernsystem für die Sachbearbeiter betrachtet.

Zu welchen Akteuren steht das System in Beziehung?

> In welchen Interaktionsbeziehungen steht das System? An welchen Kooperationen soll das System teilnehmen? Welche Aufträge werden von externen Akteuren gestellt, welche Aufgaben werden an externe Akteure weiterdelegiert? Welche Kommunikationsflüsse sind zu identifizieren?

Im Beispiel: Zur Erstellung von Serienbriefen soll Microsoft Word verwendet werden.

Welche Begrifflichkeiten werden dabei thematisiert?

Welche sprachlichen Begriffe werden in den Kommunikationsbeziehung benutzt? Welche konzeptuellen oder auch physisch realen "Informationseinheiten" werden übermittelt? Welche Dokumente, Reports, Dateien, Listen, Formulare, Übersichten, Informationen werden verwendet?

Hier wird bereits ein Teil der Begriffswelt des Systems definiert.

Im Beispiel: Der Webserver stellt eine Übersicht über die aktuellen Bewerbungen zur Verfügung ("Bewerberaushang"), der Student teilt mit seinem Antrag "Präferenzen" für seine Zuteilung mit etc.

Welche Aufgaben soll das System übernehmen?

Hier befinden wir uns schon im Tätigkeitsprofil einer Systemkomponente. Es sind hier die Aufgaben und Verpflichtungen des Softwaresystems zu dokumentieren und zu priorisieren.

Im Beispiel: die automatische Zuteilung, die Generierung von Reports und Listen etc.

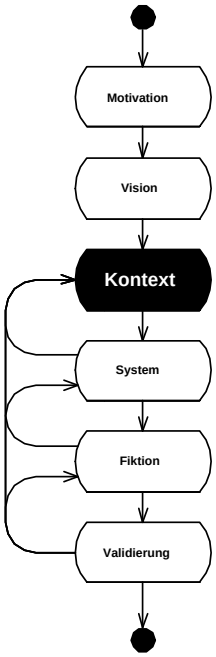
Welchen Randbedingungen unterliegt das System?

Welche konkreten gesetzlichen Regelungen, Systemeinschränkungen, Vorbedingungen sind vom System zu beachten?

Im Beispiel: Das System soll mehrbenutzerfähig sein, soll die neue Verteilungsordnung berücksichtigen etc.

Das wirkliche intuitive Verständnis der Anforderungen an ein IS ist für den nachhaltigen Projekterfolg letztendlich wichtiger als jede Formalisierung. Dieses Verständnis nährt sich – wie für jeden anderen Typ von Softwaresystemen auch – in erster Linie aus der Kenntnis der Systemumgebung. Zum Verständnis der Rolle von Informationssystemen ist die reale Arbeitspraxis zu begreifen, die maßgeblich von diesen Softwaresystemen mitgeprägt wird. Die Modellierung dieser Umgebung ist Gegenstand der im nächsten Abschnitt beschriebenen Phase. Die Vision fasst dieses Verständnis in den wesentlichen Punkten zusammen. Sie ist normalerweise bereits bei Projektstart existent. Ein Ergebnis der folgenden Phasen der Kontextanalyse und Systemgestaltung kann allerdings auch eine fundamentale Neuausrichtung sein.

5.4. Kontextanalyse



Ziel: Analyse des Arbeitszusammenhangs, in dem das neue Softwaresystem eingesetzt werden soll

Die Kontextanalyse dient der Untersuchung und Modellierung des praktischen Arbeitsumfelds, in dem das neue Softwaresystem zukünftig eingesetzt werden soll.

Die Modellierung selbst wird von den Entwicklern durchgeführt. Durch entsprechende Interviews, Fragebögen, Workshops, Recherchen, Beobachtungen, Studien etc. lernt der Systemanalyst die Arbeitswelt der Anwender kennen. Er modelliert die konkrete Arbeitspraxis, in dessen Kontext das neue IS eingebettet werden soll.

Bei der Erfassung des Aktivitätskontexts kommen oft wichtige versteckte Wünsche und Ziele der Interessengruppen (Bsp: “Besonders die Erstwünsche sollten möglichst erfüllt werden, damit Studenten nicht vorbeikommen oder sich in der Fakultät beklagen”), organisatorische Einschränkungen (“Die Ergebnisse des 2. Staatsexamen sind erst kurz vor dem Praktischen Jahr bekannt”), Randbedingungen (“Jeder Ulmer Student hat ein Anrecht auf einen Praktikumsplatz, auch wenn er sich erst eine Woche nach Praktikumsbeginn bewirbt”) und Vorgaben (“Das Prüfungsamt muss Adressetiketten für Anschreiben an die Praktikumsteilnehmer bekommen”) zum Vorschein.

Zur Modellierung

Der Arbeitsbereich wird in Form der verschiedenen KoA$_{\text{L}}$A-Modelle dokumentiert. Der Systemanalyst erstellt ein Organisations-, Kooperations- und Wissensmodell für das relevante Anwendungssystem.

Das Softwaresystem selbst kommt in diesen Modellen in der Regel vorerst noch nicht vor. Sobald allerdings Ideen für die Rolle des Softwaresystems vorliegen und arbeitsorganisatorische Auswirkungen auf den Arbeitskontext deutlich werden, ist es notwendig, im Rahmen einer

zusätzlichen Iteration das Kontextmodell sukzessive zu einem Modell der zukünftigen Arbeitsumgebung weiterzuentwickeln. Nimmt das System selbst starken Einfluss auf die praktischen Arbeitsabläufe, z.B. indem es selbst Arbeitsanweisungen ausgibt (Einsatzplanungssysteme, Workflowsysteme etc.) oder die Kommunikationsbeziehungen der anderen Akteure verändert, wird der System-Akteur und seine Kooperationsbeziehungen auch selbst in die Kontextmodellierung mit eingeschlossen. Besitzt das neue System hingegen eher den Charakter eines Werkzeugs, das lokale einzelne Tätigkeiten oder Interaktionen von Akteuren unterstützt, aber keinen Einfluss auf akteurübergreifende Aktivitätsmuster besitzt, so wird es selbst in der Kontextmodellierung nicht auftauchen.

Aufgabe der Kontextanalyse ist letztlich die Modellierung einer bestimmten Organisation. Diese kann aus verschiedenen Perspektiven betrachtet werden. [Rem89] unterscheidet hier folgende Gesichtspunkte bzw. Bedeutungen des Begriffs "Organisation":

Struktur: faktorielle Sichtweise: die Organisation als materielles System. Führt zu folgenden Fragestellungen: Wie ist die Organisation aufgebaut? Welche Aufgaben werden bearbeitet? Welche Stellen, Personen, Sachmittel, Informationsobjekte sind zu identifizieren?
Ein eingeführter Begriff in diesem Kontext ist die "Aufbauorganisation". Eine Organisation kann in diesem Sinne als ein System instrumenteller Rollenerwartungen aufgefasst werden, die jeweils an bestimmte Positionen (Stellen) gebunden sind. Rollen werden von Positionsinhabern als auch von Außenstehenden an eine Postion geknüpft (= soziale und instrumentelle Erwartungen).
KoA$_{\text{L}}$A modelliert diesen Aspekt einer Organisation in seinem Organisationsmodell.

Verhalten: aktionelle Sichtweise: die Organisation als "Lebensform". Hier steht das äußere "Erscheinungsbild" eines organisierten zweckgerichteten Sozialsystems im Mittelpunkt des Interesses, das ein festes System von Regeln kennt, welche das Verhalten der in ihm tätigen und der mit ihm in Berührung kommenden Menschen steuert.
KoA$_{\text{L}}$A modelliert diesen anthropomorphen Aspekt einer Organisation in Form seines Kooperationsmodells.

Funktion: instrumentelle Sichtweise: die Organisation als Mittel zur Erreichung von Zielen. Dieser Perspektive führt zu Fragestellungen nach dem "Wie": wie geht man zur Erreichung der Ziele vor? Welche Aufgaben haben die Beteiligten, welche Verhaltensregeln gibt es, welche Handlungsanweisungen können identifziert werden, etc. Die sogenannte "Ablauforganisation" steht im Vordergrund.
KoA$_{\text{L}}$A beschreibt diesen Aspekt einer Organisation zu einem Teil in Form der Tätigkeitsprofile als Untermodell des Organisationsmodells, zum anderen Teil in Form seiner Aktivitätsmuster als Untermodell des Kooperationsmodells.

Eine Reihe von betroffenen Akteuren der betrieblichen Arbeitspraxis wurden bereits in der Motivationsphase als Interessengruppen bzw. in der Visionsphase als Systeminteraktionspartner identifiziert. Bei der Untersuchung des Arbeitsumfelds können hier natürlich weitere Akteure aufgedeckt werden.

Zum Vorgehen

Abb. 5.5 zeigt das Vorgehen bei der Kontextmodellierung:

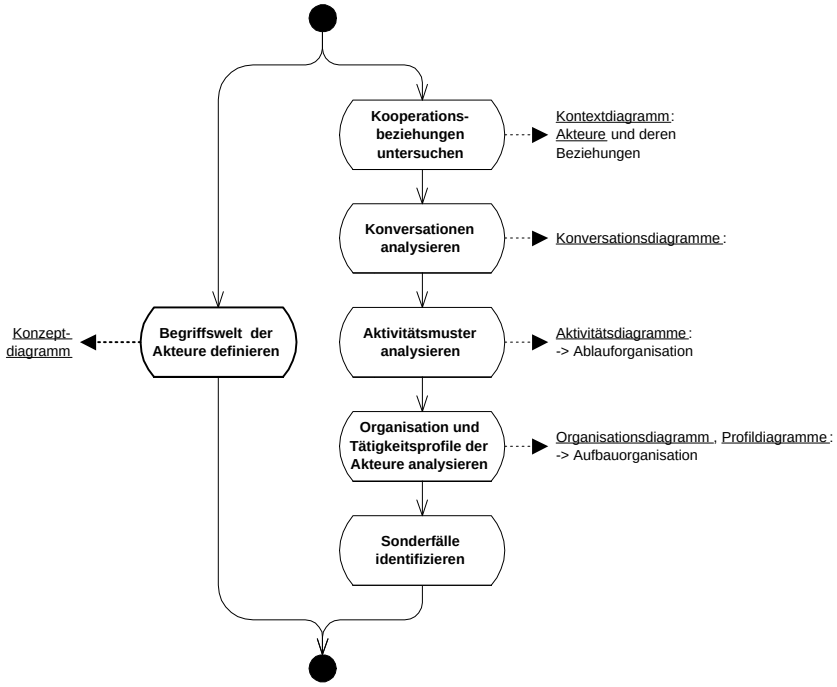


Abbildung 5.5.: Das Vorgehen bei der Kontextmodellierung

Die Modellierung beginnt mit der Darstellung statischer Interaktions- und Kooperationsbeziehungen zwischen den beteiligten Akteuren in Form eines Kontextdiagramms, das die wichtigsten Abhängigkeitsbeziehungen aller Akteure des Anwendungsbereichs aufführt.

Nachdem im Folgenden dann auch die Dynamik des Anwendungsbereichs in Form von globalen Konversations- und Aktivitätsmustern untersucht und modelliert wurde, wird auch die globale Organisationsstruktur und die einzelnen subjektiven Tätigkeitsprofile der Akteure dokumentiert.

Parallel zu diesen Modellierungsaktivitäten wird das spezifische Fachwissen des Anwendungsbereichs sukkzessive verfeinert bzw. vervollständigt und durch ein Wissensmodells dargestellt.

Die angegebenen Schritte der Modellierung mit KoA$_L$A geben eine logische Reihenfolge der Modellierungsaktivitäten an. Sie sagen nichts darüber aus, in welcher Form und wann diese Modellierungstätigkeiten in realen Projekten durchgeführt werden. Sind viele Betroffene zu berücksichtigen, kann es z.B. Sinn machen, diese durch einzelne Entwickler isoliert und evtl. gleichzeitig zu befragen und die Modellierung erst anschließend gemeinsam im Designteam durchzuführen. Wenn Kunden Zeit und Interesse mitbringen, könnte man die Modellierung alternativ auch vor Ort und in enger Zusammenarbeit mit dem Kunden durchführen. Als erster Schritt vor der Modellierung kann es dabei sehr nützlich sein, sich die einzelnen Ideen, Fragen, Erkenntnisse, Probleme, Konfliktpunkte, Randbedingungen, Ziele etc., die während der Interviews oder der gemeinsamen Treffen angesprochen werden, als einfache Textlisten zu dokumentieren, um sie dann später gesammelt in Designsitzungen zu diskutieren, zu strukturieren und zu einem gemeinsamen Bild des globalen Anwendungskontexts zusammenzusetzen. Eine gute Technik zur gemeinsamen Strukturierung solcher “Rohdaten” stellt z.B. die Erstel-

lung von "affinity diagrams" dar, wie sie in [BH98] beschrieben wird.

Eine wesentliche Aufgabe der Kontextmodellierung ist das Explizitmachen impliziten Wissens, um dieses Wissen der rationalen Auseinandersetzung zugänglich und entsprechende Erkenntnisse über Abläufe, Ziele, Randbedingungen etc. im Gestaltungsprozess berücksichtigen zu können. Ein menschlicher Akteur rationalisiert seine Aktivitäten, Ziele und Aufgaben in der Regel nicht vollständig. Viele Aspekte seines Tuns sind im Allgemeinen situiert und daher nicht a priori analysier- und darstellbar [Suc87]. Die Grundlagen seiner Aktivitäten und spontanen Entscheidungen sind vielmehr oft nur aus Beobachtungen seiner Alltagsaktivitäten ableitbar (Suchman spricht hier von "tacit knowledge"). Im Gespräch ist es daher oft nützlich, das implizite Wissen über den Umweg der Beobachtung, Diskussion und Beschreibung seiner Alltagspraxis Schritt für Schritt zu erforschen und explizit zu machen (Was passiert, wenn?). Siehe hierzu den Abschnitt 5.4.4.

Ein anderer erfolgversprechender Weg führt über die Erforschung von Interaktionsbeziehungen und den sich wiederholenden Konversationsmustern (siehe z.B. [WF87, AHL92, BRLD94]). Die Wirklichkeitsbeschreibung fällt aus der ereignisorientierten Sicht von Kommunikationsabläufen oft einfacher als in der aktionsorientierten logischen Sicht von Aktivitätsabläufen. Die Aktivitätssicht ist eher für lokale Handlungspläne oder organisatorische Tätigkeitsfestlegungen sinnvoll. Die reale Arbeitspraxis ist hingegen überwiegend nichtdeterministisch und daher oft besser in Form einer deskriptiven Beschreibung greifbar. Siehe hierzu den Abschnitt 5.4.2.

5.4.1. Kooperationsbeziehungen

> **Ziel**: Überblick über die Kooperationsbeziehungen aller Akteure des Handlungssystems

Der erste Schritt der Kontextmodellierung besteht in der Analyse und Bestimmung der statischen Kooperationsbeziehungen aller Akteure. Eine betriebliche Organisation stellt ein stark arbeitsteiliges Handlungssystem dar. Ein erster Überblick hierüber lässt sich erzielen, indem man die wichtigsten menschlichen und technischen Akteure identifiziert und deren Kommunikationsbeziehungen untersucht. Damit lässt sich ein erster Einblick in die primären Kooperationsbereiche der Anwendungsumgebung erreichen.

Zur Modellierung

Dargestellt werden diese Beziehungen primär an Hand eines oder mehrerer Kontextdiagramme (siehe Kap. 4.4.1), die die Akteure des Anwendungssystems und ihre Kooperationsbeziehungen zeigen. Optional werden diese Beziehungen zusätzlich auch textuell erläutert, indem sowohl die Art, der Umfang und der mögliche Ablauf dieser Kooperationen ausführlich beschrieben werden. Die Akteure der Kontextdiagramme können sowohl konkrete Personen der Anwendungsumgebung darstellen, bestimmte Arten oder Rollen von Mitarbeitern modellieren, oder auch für ganz abstrakte Organisationseinheiten des Unternehmens stehen.

Unter den Beziehungen der Akteure können folgende Arten unterschieden werden:

Kooperationen Gemeinsame Aktivitäten zweier oder mehrerer Akteure mit einem gemeinsamen Ziel.

Kooperationen werden als Aktivitäten modelliert, die mit den beteiligten Akteuren verbunden werden.

Die hier modellierten Kooperationen werden in der Regel auf einer hohen Abstraktionsstufe angesiedelt sein, die erst in den folgenden Schritten näher konkretisiert werden. Optional können die Kooperationen allerdings auch im Kontextdiagramm in Form aggregierter Aktivitätsbereiche verfeinert werden.

Interaktionen Stellen eine besondere Untergruppe der Kooperationen dar. Interaktionen sind gemeinsame Aktivitäten mehrerer Akteure, die der kommunikativen oder sonstigen gegenseitigen Einwirkungen der Interaktionspartner untereinander dienen.

Interaktionen werden als Interaktionspfeile dargestellt, die in Form von Begriffen optional auch mit dem kommunizierten Informationsgehalt annotiert werden.

Interaktionen werden im Normalfall von einem ausgezeichneten Akteur initiiert und dienen einem bestimmten gemeinsamen Ziel [6] bzw. erfüllen eine bestimmte Funktion (wie z.B. in Apvel die "Einreichung eines Wahlvorschlags" oder der "Bestellung eines Produkts"). Die einzelnen lokalen Aktivitäten der Akteure, die durch diese Interaktionen möglicherweise hervorgerufen werden, fallen nicht darunter. Interaktionen sind einfach oder komplex, d.h. sie können auch mehrere Einzelinteraktionen umfassen und damit den Charakter eines Dialogs besitzen, der aus einer Sequenz von gegenseitigen Kommunikationsakten besteht (siehe Abschnitt 5.4.2). Im Kontextdiagramm können solche Interaktionen auch durch Aggregationsbeziehungen mit mehreren ausgezeichneten Einzelinteraktionen verbunden dargestellt werden.

Erwartungen Akteure nehmen auch indirekt Einfluss auf andere Akteure, indem sie die Erfüllung von Vereinbarungen einfordern oder konkrete Anforderungen an andere Akteure stellen.

Diese Erwartungen werden als entsprechend stereotypisierte Bedingungen modelliert, die mit den betroffenen bzw. verursachenden Akteuren über gestrichelte Pfeile verbunden sind und dadurch das soziale Netz der gegenseitigen Verpflichtungen und Vereinbarungen explizit machen.

Soziale "Verträge" wie Vereinbarungen, stillschweigende Übereinkünfte und gegenseitige Selbstverpflichtungen sind die eigentliche Motivation bzw. Voraussetzung für jegliches organisiertes Handeln in menschlichen arbeitsteiligen Handlungssystemen.

In Abb. 5.6 wird als Beispiel der statische Kontext aller Akteure im Umfeld von Apvel dargestellt:

> Das PJ-Büro steht in einer Vielzahl von Kooperations- und Interaktionsbeziehungen zu Akteuren seiner Umgebung:

[6] Würden die Ziele der Interaktionspartner nicht existent sein oder differieren, käme keine erfolgreiche Interaktion zustande. Im Fall von Einwirkungen auf Gegenständen sind natürlich nur die Ziele der aktiven Akteure relevant.

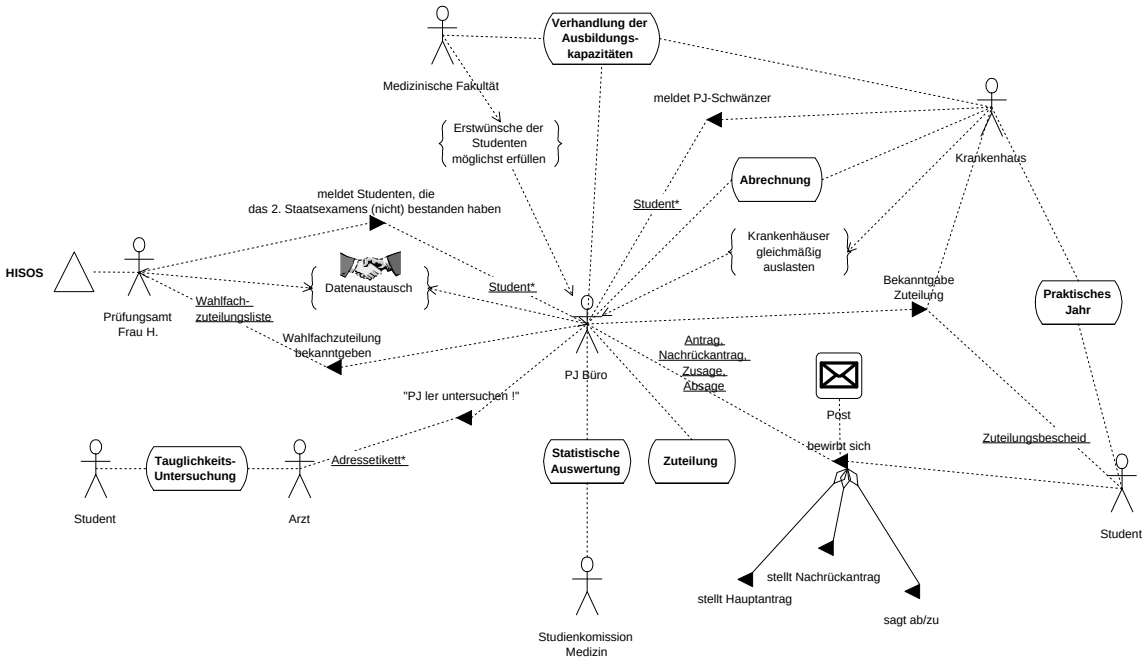


Abbildung 5.6.: Kontextdiagramm für das Anwendungssystem

- In Zusammenarbeit mit den Krankenhäusern und der medizinischen Fakultät wird vor Beginn der Bewerbungsphase eines PJ die Anzahl der Ausbildungsplätze vereinbart (=*Kooperation*).
- Mit Eröffnung des Anmeldeverfahrens bewerben sich die Studenten auf dem Postweg, bekommen einen Zuteilungsbescheid, bestätigen die Zuteilung oder stellen einen Nachrückantrag (=*Interaktionen*).
- Nach der erfolgreichen Zuteilung müssen alle PJ-Studenten vor Antritt ihres PJ von einem Facharzt untersucht werden (=*Kooperationen*). Daher erhält dieser Arzt eine Reihe von Adressaufklebern für die Einladungsschreiben (=*Interaktion*).
- Die Krankenhäuser informieren das PJ-Büro über alle Studenten, die ihr Praktisches Jahr antreten. Studenten, die nicht erscheinen, werden gemeldet (=*Interaktionen*).
- Im Anschluss an ein Praktisches Jahr rechnet das PJ-Büro jeden Praktikumsplatz mit den Krankenhäusern ab (=*Kooperation*).
- Das Prüfungsamt wird über die Wahlfachzuteilung informiert, damit es die Studenten zu entsprechenden Prüfungen anmelden kann. Im Gegenzug wird das PJ-Büro über die Ergebnisse des zweiten Staatsexamens auf dem Laufenden gehalten, damit Praktikumsplätze von Studenten, die die Zulassungsvoraussetzungen nicht erfüllen, wieder für die Nachverteilung zur Verfügung gestellt werden können (=*Interaktionen*, *Erwartungen*). Studenten, die diese Prüfung nicht bestehen, werden vom Praktischen Jahr ausgeschlossen. Deren Plätze stehen darum anderen Studenten zur Verfügung, deren Erstwunsch z.B. bis dahin noch nicht entsprochen werden konnte.
- Die Studienkommission interessiert sich regelmäßig für statistische Daten wie

z.B. den Grad der Erfüllung von Erstwünschen, die Bewerbungszahlen für spezifische Wahlfächer, den Anteil von Auslandsaufenthalten etc. (=*Kooperation*).

Zum Vorgehen

Für die Modellierung der Kooperationsbeziehungen müssen zunächst die wichtigsten Akteure des direkten Anwendungsumfelds identifiziert werden. Kandidaten von relevanten Akteuren finden sich dabei zum Einen in Vertretern der in der Phase der Motivation identifizierten Interessengruppen und den in der Phase der Vision bereits anvisierten Anwendern des neuen Softwaresystems. Die restlichen relevanten Akteure ergeben sich aus der sukzessiven Untersuchung und Ergänzung aller sonstigen Abhängigkeiten bzw. Kooperationen der aktuellen Arbeitspraxis.

Durch Interviews, Workshops, Fragebögen etc. zum Aufdecken der etablierte Kooperations- und Kommunikationsbeziehungen entwickelt sich sukzessive das erweiterte Bild eines Beziehungsgeflechts zwischen den Beteiligten. Dieses Geflecht wird zunächst grafisch auf einer möglichst abstrakten Ebene in einem globalen Kontextdiagramm skizziert und nur in den Bereichen detailliert/verfeinert, in denen das zukünftige System voraussichtlich eine Rolle spielen wird.

Auf einer sehr hohen Abstraktionsstufe kann man hier die Zusammenarbeit bestimmter Akteure z.B. auch in Form einer einzigen Aktivität, d.h. Kooperation modellieren, an der all diese Akteure beteiligt sind, und die erst in der Folge fallweise sukzessive weiter spezifiziert, d.h. in Einzelaktivitäten oder Interaktionen aufgesplittet wird. In unserem Fallbeispiel könnte man z.B. die Kooperation "das PJ-Büro nimmt in Zusammenarbeit mit der Studienkommission eine statistische Auswertung des PJ vor" nach und nach in die relevanten Subaktivitäten (wie z.B. "PJ-Büro berechnet Wunscherfüllungsgrad") und enthaltene Interaktionen (wie z.B. "PJ-Büro übermittelt Statistiken am Ende des Nachrückverfahrens") verfeinern.

Relevant für die Modellierung sind immer solche Modellelemente, die durch das neue IS betroffen sein könnten. Eine Verfeinerung einer Kooperation und die damit verbundene Zuordnung von Verantwortung für Einzelbereiche der Aktivität an die beteiligten Akteure ist immer dann interessant, wenn zukünftige Nutzer des Systems beteiligt sind.

Um die Funktionalität des zukünftigen Systems einzugrenzen (wird in Kap. 5.5.5 behandelt), ist es notwendig, die lokalen Tätigkeitsprofile von Anwendern des Systems genau zu kennen bzw. festzulegen. Die in Kap. 5.4.5 beschriebene Modellierung der Tätigkeitsprofile basiert daher zum Einen auf einer sukzessiven Konkretisierung der im Kontextdiagramm noch recht abstrakt identifizierten Kooperationsbeziehungen, um so die individuellen Aufgabenbereiche der beteiligten Akteure herauszuarbeiten, zum Anderen auf einer Untersuchung der Folgen bzw. Ursachen der festgestellten Interaktionsbeziehungen (siehe hierzu die folgenden Abschnitte der Kontextanalyse).

Bei der Modellierung der Kooperationsbeziehungen sind damit zusammenfassend folgende Leitfragen zu beantworten:

Welche Akteure sind für das Anwendungssystem relevant?

Aus welchen menschlichen und technischen Akteuren besteht das relevante Anwendungssystem, d.h wer sind die Betroffenen der praktischen Arbeitswirklichkeit in der Umgebung des neuen Informationssystems?

Mit der Auswahl dieser Akteure ist der Gegenstandsbereich bereits eingegrenzt, dessen Analyse in den Gestaltungsprozess des neuen IS einfließen kann. Der betrachtete Ausschnitt der realen Arbeitspraxis sollte daher von Anfang an nicht zu knapp bemessen sein, da das Anforderungsprofil der Interessengruppen zum Projektbeginn oft noch sehr inhomogen ist.

Mögliche Kandidaten für die Akteure finden sich in den Akteuren der Motivations- und Visionsmodellen.

Im Beispiel: Das PJ-Büro, der Student, die Krankenhäuser, das Prüfungsamt, die Medizinische Fakultät und die Studienkommision.

Welche Kooperationen sind zu identifizieren?

Welche gemeinsamen Aktivitäten der Akteure können unterschieden werden? Wie lassen sich Aktivitätsbereiche verfeinern oder auch verallgemeinern?

Hier treten die Abhängigkeiten der Akteure in ihrer praktischen Arbeitstätigkeit zutage.

Im Beispiel: Die Verhandlung der Ausbildungskapazitäten, die Abrechnung, die statistische Auswertung etc.

Welche Erwartungshaltungen existieren?

Welche Anforderungen und Erwartungshaltungen bestehen zwischen den Akteuren? Welche Vereinbarungen beeinflussen die praktische Arbeitswirklichkeit?

Im Beispiel: Das Krankenhaus will seine Kapazitäten gleichmäßig auslasten, die Erstwünsche der Studenten sollen möglichst erfüllt werden etc.

Welche Kommunikationsbeziehungen lassen sich identifizieren?

Welche Akteure treten als Sender bzw. Auslöser, welche als Empfänger auf? Welche Resourcen (Werkzeuge, Kommunikationssysteme, etc.) werden dabei verwendet? Welche Begrifflichkeiten werden angesprochen? Welche Dokumente werden transportiert? Welche "Funktion" erfüllt die Interaktion?

Im Beispiel: Studenten bewerben sich, das PJ-Büro gibt die Zuteilung bekannt, das PJ-Büro gibt dem Prüfungsamt die zugeteilten Wahlfächer bekannt etc.

Auf Ebene der Darstellung im Kontextdiagramm wird man sich normalerweise auf einer relativ abstrakten Ebene bewegen. In der textuellen Beschreibung der einzelnen Kooperationen bzw. Interaktionen hingegen steht einer umfang- und kontextreichen Erläuterung derselben nichts im Wege.

Hauptaufgabe bei der Modellierung der Kooperationsbeziehungen ist die Identifikation aller Beteiligten und eine erste Übersicht bzw. Eingrenzung ihrer primären Abhängigkeitsbeziehungen. Die sukzessive Detaillierung der relevanten Abhängigkeiten findet automatisch auch in den nächsten Stufen der Kontextmodellierung statt. Die Modellierung konzentriert sich

auf die wesentlichen konzeptionellen Kommunikations- und Kooperationsbeziehungen der Beteiligten unabhängig von ihrer etwaigen späteren technischen Unterstützung durch das neue Softwaresystem und dient damit als erster Ausgangspunkt für die folgende Detaillierung der Tätigkeitsprofile und Aktivitätsmuster aller relevanter Anwender des neuen Systems.

Nachdem die Statik der gegenseitigen Abhängigkeitsbeziehungen nun ausreichend dokumentiert wurde, geht es in den folgenden zwei Schritten darum, sich die Dynamik dieser Abhängigkeiten näher anzusehen.

5.4.2. Konversationen

Ziel: Skizzierung des Handlungsrahmens

Die Interaktionsbeziehungen der Akteure werden nun weiter untersucht, indem die zentralen Interaktionsmuster der Anwendungsumgebung beschrieben werden. Aus diesen Kommunikationsmustern lässt sich gut auf die wichtigsten Arbeitsabläufe in der Umgebung des neuen Softwaresystems schließen, die bei der Ausgestaltung des Systems zu berücksichtigen sein werden. Konversationen definieren zentrale Handlungsrahmen, von denen aus die konkrete Arbeitspraxis verstanden werden muss bzw. von denen aus viele Aktivitäten ihre Daseinsberechtigung ableiten (siehe hierzu z.B. auch [KKB95]).

Konversationen setzen die bereits im vorigen Abschnitt beschriebenen Interaktionen und Kooperationen in einen größeren Kontext bzw. verfeinern solche Interaktionen, indem sie charakteristische Muster für deren Ablaufstruktur beschreiben.

Viele der Methoden, die zur Analyse der Anwendungswelt Verwendung finden (z.B. das "Business Modelling" von RUP, oder allgemein die Methoden der Geschäftsprozessmodellierung) konzentrieren sich ausschließlich auf die Modellierung von Aktivitätsmustern in Kombination mit einer Analyse der begleitenden Informationsflüsse. Die eigentlichen Ursachen, Veranlassungen oder Motive der zu beobachtenden Aktivitäten bleiben damit weitgehend unberücksichtigt. Damit einhergehend werden somit potentiell auch Optionen der Um- und Ausgestaltung von Tätigkeitsbereichen ignoriert, da nur ihre aktuelle Ausprägung, nicht jedoch ihr Hintergrund bzw. ihre eigentliche Funktion und alternative Umsetzungsmöglichkeiten thematisiert werden. Als Auslöser für Aktivitäten der einzelnen Akteure treten hier höchstens einzelne Informationsflüsse wie z.B. die Überstellung bestimmter Dokumente auf. Beschrieben werden damit im Allgemeinen nur bestimmte Inkarnationen von akteurübergreifendem Verhalten, nicht deren eigentliche Veranlassung. Konzentriert man sich in der Analyse der zu beobachtenden Aktivitätsmuster hingegen auf die Kommunikationsebene, tritt die Veranlassung der einzelnen Akteure in den Vordergrund.

Über die reine Überstellung der einzelnen Dokumente oder Informationen hinaus lassen sich weitere kommunikationsorientierte Einflüsse der beteiligten Akteure identifizieren, die deren Aktivitäten letztlich motivieren: Akteure veranlassen andere Akteure zu bestimmten Leistungen, sie stellen Anfragen, sagen Leistungen zu oder informieren über bestimmte Sachverhalte. Alle diese Interaktionen können bei Empfängern unterschiedlichste Reaktionen – wie z.B. weitere Interaktionen – hervorrufen. Im Allgemeinen lassen sich in den Interaktionssequenzen systemische – d.h. sich wiederholende, standardisierte – Bereiche identifizieren, die in $\text{KoA}_{\text{L}}\text{A}$ als Konversationen modelliert werden.

Konversationen stellen damit eine höhere Modellierungsebene des Anwendungsbereichs dar. Sie modellieren systemische Kommunikationsbereiche, die den motivationalen Kontext für die zu beobachtenden Aktivitäten der beteiligten Akteure bilden. Die einzelnen Aktivitätsmuster werden in KoA$_{L}$A erst im nächsten Schritt modelliert (siehe Abschnitt 5.4.4).

Zur Modellierung

Die charakteristischsten Konversationen werden sowohl in Form von Konversationsdiagrammen dokumentiert (siehe Kap. 4.4.2) als auch textuell in der Projektdatenbank in ihrem Ablauf beschrieben.

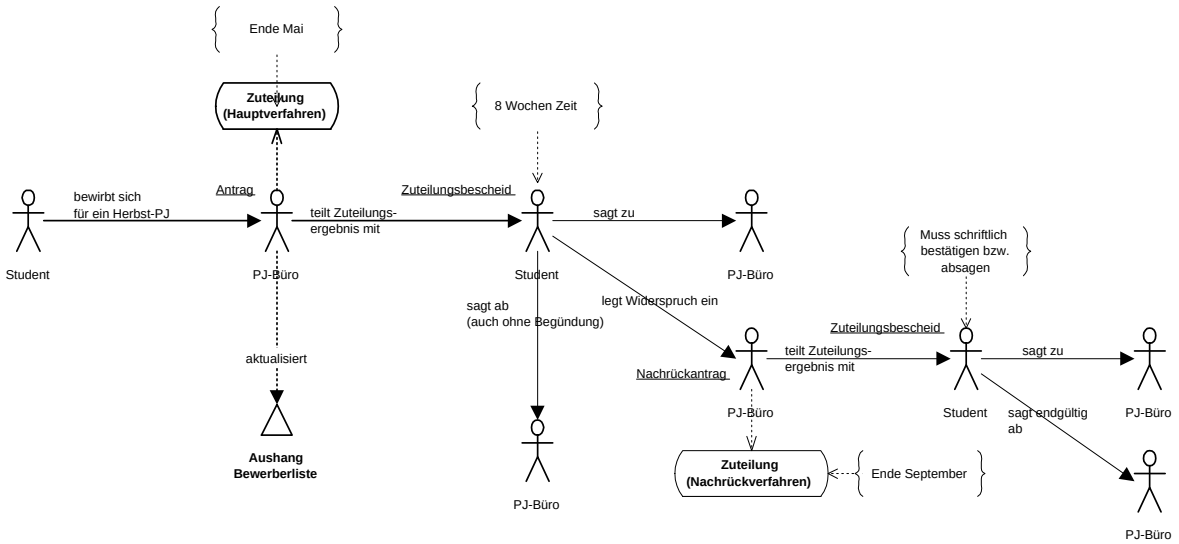


Abbildung 5.7.: Die Bewerbung als Konversationsdiagramm

In Abb. 5.7 findet sich als Beispiel ein Konversationsdiagramm für den Ablauf des Bewerbungsverfahrens für das Praktische Jahr:

> Die Studenten bewerben sich beim PJ-Büro für die Teilnahme an einem Praktischen Jahr. Die aktuelle Bewerbungssituation wird in Form eines Aushangs bekanntgegeben.
>
> Nachdem das PJ-Büro die Hauptverteilung durchgeführt hat, wird allen Bewerbern ein Zuteilungsbescheid zugestellt. Wenn die Bewerber mit ihrer Zuteilung zufrieden sind, sagen sie dem PJ-Büro schriftlich zu und das Bewerbungsverfahren ist beendet.
>
> Wenn sie in Form eines Nachrückantrags einen Widerspruch einlegen, so werden sie im Nachrückverfahren erneut zugeteilt. Die endgültige Zuteilung muss von den Studenten wiederum "quittiert" werden, indem sie der Zuteilung schriftlich akzeptieren oder ablehnen.
>
> Bei korrekter Absage können sich die entsprechenden Studenten im nächsten Jahr ohne Nachteile in Kauf nehmen zu müssen erneut bewerben. Fehlen sie unentschuldigt, so werden sie bei erneuter Bewerbung nur nachrangig berücksichtigt.

Das dargestellte Konversationsdiagramm zeigt den Verlauf des zentralen Dialogs zwischen Antragsteller und PJ-Büro, der den eigentlichen Handlungsrahmen für die Tätigkeiten des PJ-Büros bilden. Alle Aktivitäten des PJ-Büros zielen darauf ab, genau diese Kommunikation erfolgreich zum Abschluss zu bringen, bzw. werden durch dieses Ziel motiviert oder veranlasst. Ein Verständnis dieses Kommunikationsmusters ist daher für alle Projektbeteiligten ein guter

Ausgangspunkt für die Analyse und Modellierung der Arbeitswirklichkeit aller betroffenen Akteure.

Eine andere zentrale Konversation wäre beispielsweise auch die regelmäßige Kommunikation zwischen den Lehrkrankenhäusern und dem PJ-Büro, die die Kapazitätsverhandlung, die vorläufige und endgültige Übermittlung des Belegungsplans, die Nachverhandlung von Kapazitäten, die Meldungen des Krankenhauses über Abbrüche und die abschließende Abrechnung umfasst.

Zum Vorgehen

Die Modellierung muss folgende Leitfragen klären:

Welche typischen Konversationsmuster sind zu beobachten?

Wie sehen typische Interaktionssequenzen aus und welche Akteure sind daran beteiligt? Welche Dialoge oder akteurübergreifende Interaktionsmuster wiederholen sich und bilden einen Handlungsrahmen für die zu beobachtenden Aktivitäten? Welche Dialoge sind festgeschrieben oder geregelt, welche haben sich etabliert?

Sinnvollerweise geht man dabei sukzessive alle Akteure durch, die man bei der Modellierung der Kooperationsbeziehungen identifiziert hat und untersucht, in welchen musterhaften Interaktionsabläufen sie beteiligt sind. Die wichtigsten Konversationen, die die Aktivitäten im Anwendungssystem am besten erklären, werden modelliert.

Im Beispiel: Potentielle Kandidaten der Modellierung sind die Konversationen bezüglich einer normalen Bewerbung, der Verhandlung der Ausbildungskapazitäten, der Zusammenarbeit von Prüfungsamt und PJ-Büro etc.

Welche Begrifflichkeiten werden dabei thematisiert bzw. transportiert?

Welche sprachlichen Begriffe werden in den Einzelinteraktionen benutzt? Welche konzeptuellen oder auch physisch realen "Informationseinheiten" werden übermittelt? Welche Informationen oder Dokumente, Reports, Dateien, Listen, Formulare etc. werden transportiert? Mit der Definition und Beschreibung dieser Begriffe beginnt bereits die Modellierung einer gemeinsamen Begriffswelt (s. nächster Abschnitt) bzw. die Vervollständigung der Begriffswelt, die bereits in der Vision Gestalt angenommen hat.

Anhand des Lebenslaufs zentraler Informationsobjekte lassen sich wiederum oft auch wichtige zusammengehörige Interaktionssequenzen aufdecken, die in Form eigener Konversationsdiagramme dokumentiert werden sollten.

Im Beispiel: Der Antrag, der Zuteilungsbescheid, die Zu/Absage etc.

Welche Aktivitäten zeichnen die identifizierten Dialogzustände aus?

Zur Verdeutlichung einer Konversation können in das Diagramm auch zentrale Aktivitätsbereiche mit aufgenommen werden, die in den jeweiligen Zuständen der skizzierten Konversation ausgelöst werden. An Hand dieser Aktivitäten lässt sich u.U. leichter auf die praktische Bedeutung einzelner Interaktionssequenzen schließen.

Im Beispiel: Nachdem sich alle Bewerber angemeldet haben, wird die Hauptverteilung durchgeführt. Danach werden die Bewerber über ihre Zuteilung informiert.

In der Modellierung der Konversationen geht es vor allem um die Beschreibung der standardisierten und etablierten Hauptdialoge. Diese sollen einen Überblick über das allgemeine Geschehen bieten, indem sie den zentralen Handlungsrahmen definieren. Es werden weniger Beispiele konkreter Interaktionssequenzen dokumentiert als vielmehr musterhaftes Kommunikationsverhalten.

In der Modellierung der Aktivitäten (s.u.) hingegen geht es um die konkrete Beschreibung von potentiell nebenläufigen und ereignisabhängigen Aktivitätsmustern, also um eine umfassendere Dokumentation möglicher Aktivitätsbereiche (nicht nur bezogen auf Interaktionen). Hierbei sind dann auch konkrete Beispiele wertvoll, die das Verhalten bzw. den Aktivitätsfluss in ganz bestimmten Fällen (z.B. in Ausnahmesituationen) verdeutlichen.

5.4.3. Begriffswelt

Ziel: Dokumentation der Fachsprache

Das Verständnis der Begriffswelt des betrachteten Anwendungsbereichs ist essentiell für jeden Projekterfolg und entwickelt sich kontinuierlich während der gesamten Projektlaufzeit weiter. Dieser Lernprozess ist nicht auf einen spezifischen Schritt in der Vorgehensweise begrenzt sondern er begleitet alle anderen Schritte. Die Ergebnisse des Lernprozesses, das Fachwissen, wird jedoch an einem Ort bzw. in *einem* Modell zusammengetragen und hier in Form der Begriffswelt dokumentiert.

Ein Teil dieses Wissens findet sich in der bisherigen Modellierung in Form der von den Interaktionen thematisierten Begriffe. Sie bilden zusammengenommen die Fachsprache (oder "Thesaurus") des Anwendungsbereichs. Dabei geht es hier in erster Linie nicht um Datenelemente und deren Struktur, sondern um eine konzeptionelle Definition der in der sprachlichen Kommunikation verwendeten Begrifflichkeiten. Begriffe werden dabei an Hand ihrer Merkmale und Beispielausprägungen aus Anwendersicht beschrieben.

Der gesamte Gestaltungsprozess findet in dieser Begriffswelt statt. Ohne ein Verständnis der Anwendungswelt lässt sich keine erfolgreiche Kommunikation zwischen Kunde und Analyst bewerkstelligen. Hier spielen Fachbegriffe, Konzepte, Metaphern und mentale Modelle eine große Rolle. Diese werden im Rahmen des Begriffsweltmodells möglichst präzise und kontextreich definiert.

Zur Modellierung

Die Begriffswelt wird vor allem in textueller Form dokumentiert (s. Abschnitt B). Zur ersten Übersicht kann aber auch ein Konzeptdiagramm dienen, das nur die wichtigsten Begriffe und deren Abhängigkeiten beschreibt (siehe Kap. 4.2).

In Abb. 5.8 ist als Beispiel ein Ausschnitt der Begriffswelt von Apvel als Konzeptdiagramm dargestellt:

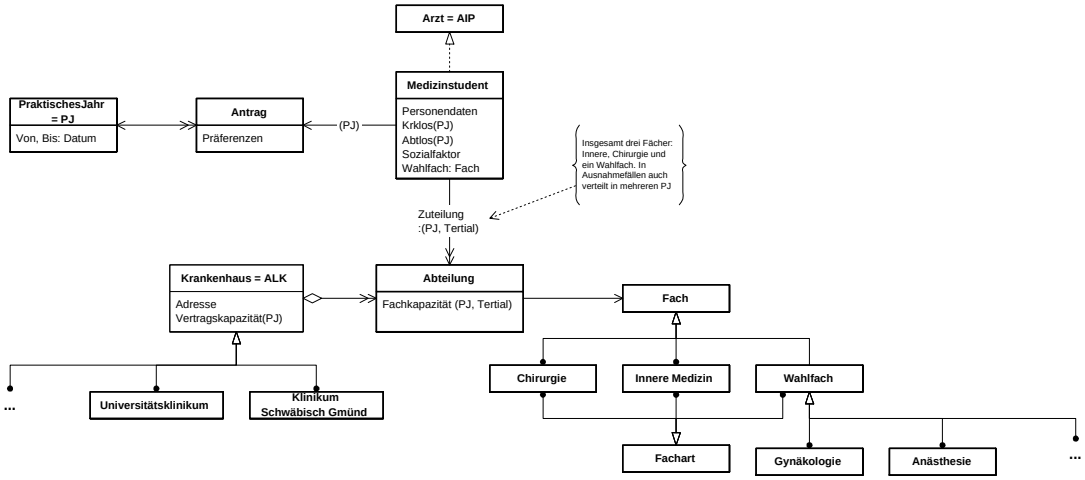


Abbildung 5.8.: Die Begriffswelt von Apvel als Konzeptdiagramm

Jeder Antrag eines Studenten zur Teilnahme an einem Praktischen Jahr bezieht sich auf genau ein Praktisches Jahr. Sobald ein Student im Praktischen Jahr ist, wird er auch als AIPler bezeichnet ("Arzt im Praktikum").

Neben den Pflichtfächern "Chirurgie" und "Innere Medizin" muss er noch ein drittes Fach belegen, das als "Wahlfach" bezeichnet wird, weil der Student in seinem Antrag hierzu bis zu drei Wünsche äußern kann. Das Diagramm führt als Beispiel eines Wahlfachs nur die Gynäkologie und die Anästhesie auf (ein Platzhalter "..." steht für den Rest).

Das Praktische Jahr teilt sich in drei Abschnitte (die "Tertiale") auf, in denen der Student jeweils in genau einem Fach ausgebildet wird.

Jedes akademische Lehrkrankenhaus ("ALK") besitzt hierzu neben Abteilungen der Facharten "Chirurgie" und "Innere Medizin" noch mindestens eine zusätzliche Abteilung, die eine weitere Fachart aufweist.

Zum Vorgehen

Bei dem Versuch, die im Kontext des Projekts verwendete Begriffswelt zu klären und eindeutig zu definieren, treten oft Mehrdeutigkeiten zutage: Es werden Homonyme und Synonyme identifiziert, Anwender besitzen unterschiedliche Konzeptionen für gleiche Begriffe. Es werden neue projektspezifische Begrifflichkeiten entwickelt und eine neue gemeinsame und eindeutige Fachsprache definiert. Dies ist inhärente Aufgabe jeder Anforderungserhebungs- und Analysetätigkeit und dient der Sicherung der Kommunikationsbeziehungen innerhalb des Projekts.

Die Bildung einer gemeinsamen Begriffswelt gelingt nur, wenn die unterschiedlichen Sichtweisen und Sprachen aller Betroffenen integriert werden. Auch alle bereits eingeführten Begriffe, die in alten Verfahren und Systemen begründet sind, müssen neu auf ihre Tragfähigkeit überprüft werden. Bei einer genauen Analyse des Arbeitskontextes im Hinblick auf implizit verwendete Konzepte können nicht nur althergebrachte sondern auch neue Begriffe zu Tage treten, die erst noch zu definieren sind. Die Begriffe stellen damit Design-Artefakte für die Gestaltung der zukünftigen Arbeitsumgebung dar. Sie müssen einen ausreichenden Grad von Kohärenz aufweisen, d.h. im Arbeitskontext intuitiv verständlich bzw. benutzbar sein.

Projekte initiieren oft die Bildung einer eigenen Begriffswelt, die sich im Projektfortgang

immer weiter "standardisiert". Das ist letztendlich auch die Voraussetzung bzw. Folge jedes Einsatzes von Softwaresystemen, deren Benutzerschnittstelle immer fest definierte Konzeptionen einführen und damit einen eng begrenzten und sehr spezifischen "Sprachschatz" besitzen (s. [Dut94]).

Bei der Definition dieser gemeinsamen Sprache geht man von den verwendeten Fachbegriffen aus, die in der Arbeitspraxis verwendet werden und die daher auch schon Eingang in die bisherige Modellierung gefunden haben.

Zur Definition der Begriffswelt kann folgenden Leitfragen nachgegangen werden:

Welche Begriffe spielen in der Anwendungsumgebung eine zentrale Rolle?

Welche abstrakten oder konkreten Konzepte werden in den identifizierten Interaktionsbeziehungen thematisiert? Welches spezifische Fachwissen wird für die praktische Arbeitstätigkeit benötigt?

Name und Bedeutung der zentralen Fachbegriffe müssen geklärt und dokumentiert werden.

Im Beispiel: der Antrag, das Praktische Jahr, das Fach etc.

Wie lassen sich Zweideutigkeiten eliminieren?

Welche Synonyme können identifiziert werden? Wie können Homonyme "eindeutig" gemacht werden?

Um den Sprachgebrauch im Projekt eindeutig zu machen, kann es erforderlich sein, sich auf bestimmte Begriffsnamen zu verständigen und evtl. auch eine neue gemeinsame Sprache festzulegen. Nicht zuletzt führt auch das IS neue Begrifflichkeiten ein, die möglichst mit allen Projektbeteiligten abzustimmen sind. Hierzu sind auch alle Synonyme (verschiedene Begriffe für das gleiche Konzept) zu dokumentieren und Homonyme (gleiche Begriffe für unterschiedliche Konzepte) zu beseitigen.

Im Beispiel: der "AIP", das "ALK", "Ausbildungsfach" versus "Wahlfach" etc.

Welche Merkmale weisen die Begriffe auf?

Wie kann man Begriffe durch Rückgriff auf bereits definierte Begriffe charakterisieren oder beschreiben? Welche Aspekte lassen sich unterscheiden, welche abhängigen Informationen tragen sie?

Im Beispiel: Ein Antrag enthält Präferenzen, ein Medizinstudent hat einen Sozialfaktor etc.

Wie können die Begriffe generalisiert oder spezialisiert werden?

Welche Verallgemeinerungen oder Spezialisierungen der Begriffe gibt es?

Im Beispiel: Die "Gynäkologie" und die "Anästhesie" sind Beispiele für Fächer, die von Studenten gewünscht werden können (daher der Name "Wahlfach"). Jedes Wahlfach stellt wie auch die "Chirurgie" oder die "Innere Medizin" ein normales Ausbildungsfach dar etc.

Welche typischen Konzeptausprägungen gibt es?

Welche typischen Beispielinstanzen können gefunden werden, die in der Anwendungsumgebung eine besondere Rolle spielen?

Im Beispiel: Die "Chirurgie" und die "Innere Medizin" sind Beispiele für vordefinierte (Pflicht-)Fächer, die jedes Krankenhaus anbieten muss.

Da der Prozess der gemeinsamen Erstellung eines Wissensmodells in der Regel mehrere Iterationen und Überarbeitungen benötigt und die fachliche Beschreibung der Fachbegriffe und Beispiele im Vordergrund steht, bietet sich für diese Abstimmungszyklen primär die textuelle Darstellungsweise an (siehe Abschnitt B).

Die grafische Darstellung des Wissensmodells in Form eines Diagramms eignet sich hingegen besser zur zusammenfassenden Präsentation der Zwischen- oder Endergebnisse dieses Gestaltungsprozesses.

5.4.4. Aktivitätsmuster

Ziel: Beschreibung der Ablauforganisation

Als Nächstes werden die zentralen Aktivitätsmuster modelliert. Ausgehend von den in 5.4.1 identifizierten Kooperationsbeziehungen und den in 5.4.2 beschriebenen Interaktionsbeziehungen werden nun akteurübergreifende Aktivitäten untersucht, welche die dokumentierten Kooperationen bzw. Interaktionen beinhalten und damit in einen größeren Zusammenhang setzen. Es werden die zentralen Aktivitätsbereiche gesucht und beschrieben, die sich in der Alltagspraxis wiederholen und die wesentlichen Aktivitätsbereiche des Anwendungssystems aufdecken. Desgleichen werden nun auch komplexe Kooperationsbeziehungen dahingehend untersucht, ob sie durch Angabe eines Aktivitätsmusters verfeinert werden können oder sollen. Diese systemischen Aktivitätsbereiche [WF87] modellieren zentrale Anteile der aktuellen Ablauforganisation im Anwendungssystem.

Zur Modellierung

Alle zentralen Aktivitätsmuster werden grafisch in Form von Aktivitätsdiagrammen (siehe Kap. 4.4.3) beschrieben und textuell erläutert.

Abb. 5.9 zeigt in Form eines Aktivitätsdiagramms den globalen Ablauf des Zuteilungsverfahrens im Überblick:

> Vom 1. bis 15. Mai können sich die Medizinstudenten beim PJ-Büro für das Praktische Jahr bewerben, das sich vom 15. Okt bis zum 20. Sept des folgenden Jahres erstreckt.
>
> Nach der Hauptverteilung Ende Mai werden die Studenten über das Ergebnis der Zuteilung schriftlich informiert. Auch alle Lehrkrankenhäuser bekommen in der Folge einen vorläufigen Belegungsplan, der bereits auch externe und verspätete Bewerbungen mit berücksichtigt. Zur gleichen Zeit bietet sich den Studenten die Gelegenheit, ihrer Zuteilung entweder zuzustimmen oder sie abzulehnen, indem sie einen Nachrückantrag stellen.
>
> Mitte September schreiben die Medizinstudenten bzw. die "Antragsteller" dann das zweite Staatsexamen. Bestehen sie diese Prüfung nicht, so ist ihnen die Teilnahme am PJ verwehrt und ihre Plätze können nun im Nachrückverfahren neu besetzt werden. Dieses Verfahren findet kurz vor dem Beginn des Praktischen Jahres statt.

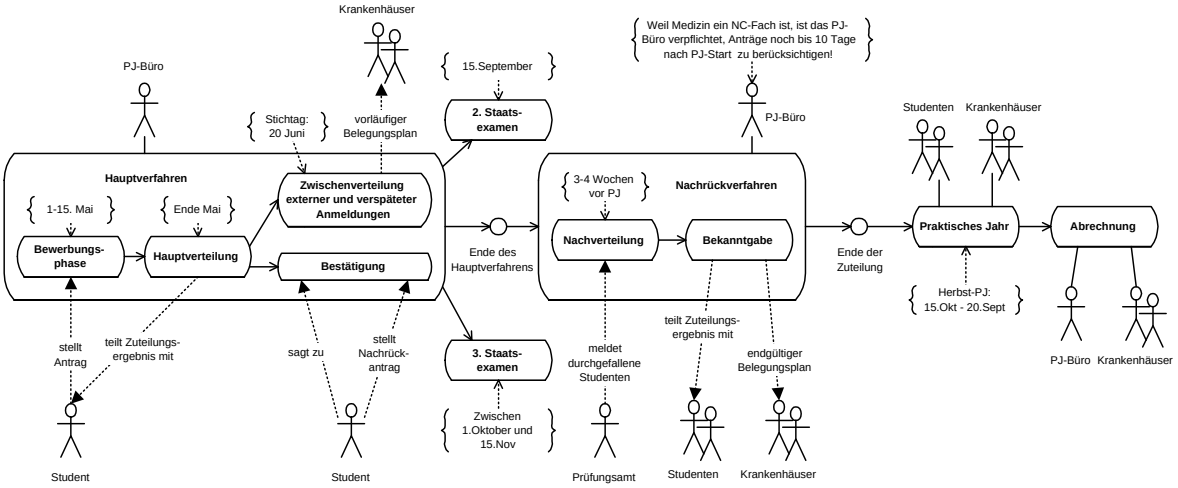


Abbildung 5.9.: Ein Aktivitätsdiagramm für das Zuteilungsverfahren im Überblick

Studenten, die das Praktische Jahr beendet haben schreiben im Anschluss daran das dritte Staatsexamen. Bestehen sie nicht, so müssen sie entsprechende Abschnitte des PJ wiederholen. Hierzu muss ihnen auch ein entsprechender Ausbildungsplatz zur Verfügung gestellt werden. Nach der Bekanntgabe der Zuteilungsergebnisse findet das Praktische Jahr statt. Nach Abschluss eines Praktischen Jahres rechnet das PJ-Büro mit allen Lehrkrankenhäusern ab, indem sie ihnen eine pauschale Aufwandsentschädigung ausbezahlt.

Das Diagramm konzentriert sich auf die Darstellung der logischen Abfolge von Einzeltätigkeiten und damit auf die Beschreibung von musterhaften Aktivitätsbereichen. Der Kontext dieser Einzeltätigkeiten wird jedoch ebenso mit in die Darstellung aufgenommen, indem auch eine Auswahl der charakteristischsten Interaktionsbeziehungen und sonstigen Randbedingungen der Einzelaktivitäten aufgeführt werden. Während der Bestätigungsphase z.B. kann eine Reihe von Interaktionen mit Studenten stattfinden. Die Einen sagen ihrer Zuteilung schriftlich zu, die Anderen lehnen ihre Zuteilung ab und stellen einen Nachrückantrag. Stichtag für Bewerbungen externer Studenten um einen Platz im Herbst-PJ ist der 20. Juni etc.

Zum Vorgehen

Als Aktivitätsmuster werden vor allem die zentralen Verfahren der Ablauforganisation dokumentiert. Auf der abstraktesten Ebene werden hier z.B. “Geschäftsprozesse” modelliert, die sich etabliert haben oder fest vorgegeben sind (in unserem Beispiel z.B. der Ablauf des ganzen Zuteilungsverfahrens mit Bewerbungsphase, Zuteilung, Nachverteilung etc.). Für ein Verständnis der Arbeitspraxis ist ein Überblick über die wichtigsten Handlungsmuster der betrachteten Organisation notwendig.

Im Laufe der Modellierung sind folgende Leitfragen zu behandeln:

Welche globalen Handlungsmuster lassen sich identifizieren?

Welche zentralen wiederkehrende und akteurübergreifende Verfahren, Prozeduren oder Handlungsmuster lassen sich identifizieren?

Kandidaten für zu modellierende Aktivitätsmuster finden sich z.B. in den Kooperationen, die in 5.4.1 identifiziert wurden.

Im Beispiel: Die Verhandlung der Ausbildungskapazitäten, das Bewerbungs- und Verteilverfahren, die Abrechnung etc.

Wie lassen sich diese Verfahren als Aktivitätsmuster beschreiben?

Aus welchen Einzelaktivitäten bestehen diese Verfahren? In welcher Reihenfolge werden diese ausgeführt? Sequentiell oder nebenläufig? Sind die Aktivitätswechsel von bestimmten Bedingungen abhängig? Von welchen?

Im Beispiel: Das Bewerbungsverfahren kann in die Bewerbungsphase, die Hauptverteilung, die Zwischenverteilung etc. unterschieden werden.

Welche Akteure sind an den Einzelaktivitäten beteilgt?

Aktivitäten können von einem oder mehreren Akteuren kooperativ durchgeführt werden. Dabei sind sowohl menschliche als auch technische Akteure relevant. Die Akteure sollten sich auch in den Diagrammen aus Abschnitt 5.4.1 wiederfinden (und umgekehrt).

Im Beispiel: Im Bewerbungs- und Verteilverfahren sind die Studenten, das PJ-Büro und die Krankenhäuser direkt beteiligt. Indirekt spielt auch das Prüfungsamt eine Rolle.

Mit welchen Akteuren wird interagiert?

Beim Ablauf einer Aktivität kann mit anderen Akteuren interagiert werden. Diese Interaktionen stellen eine wesentliche Kontextinformation für das Verständnis der Aktivität dar. Interaktionen können Ursache oder Folge von Aktivitäten sein.

Im Beispiel: Das PJ-Büro meldet Prüfungsergebnisse. Studenten und Krankenhäuser werden über die Zuteilungsergebnisse informiert etc.

Welche Ereignisse im Ablauf sind von besonderer Bedeutung?

Bestimmte Punkte im Ablauf besitzen oft eine besondere Bedeutung. Sie stellen erreichte Meilensteine dar oder kennzeichnen mögliche Alternativen.

Im Beispiel: Ende des Hauptverfahrens, Ende der Zuteilung.

Welchen Randbedingungen unterliegt das dargestellte Verfahren?

Einzelaktivitäten, Aktivitätswechsel oder Zustände unterliegen oft bestimmten Nebenbedingungen – oft z.B. zeitlicher Art. Aktivitäten sind bis zu einem bestimmten Zeitpunkt abzuwickeln; Aktivitätswechsel können nur bei bestimmten Ergebnissen oder Bedingungen stattfinden; Zustände treten nur bei bestimmten Ereignissen auf etc.

Im Beispiel: Anträge können noch bis 10 Tage vor PJ-Beginn eingereicht werden, etc.

Welche übergeordneten Aktivitätsbereiche lassen sich identifizieren?

Eine intuitive Tätigkeitsbeschreibung stützt sich als Erstes auf die konkrete Tätigkeit zur Erfassung von Aktivitätsmustern oder Szenarien. Die Konkretisierung der Einzeltätigkeiten endet dabei jeweils bei den uninteressanten (sprich “beherrschten”) Handlungen, deren Ablauf bereits klar ist.

Der übergeordnete Sinn der erfassten Aktivität bzw. die übergeordneten Aktivitätsbereiche werden in der Regel erst nach einem längeren Denkprozess definiert. Jedoch ist diese Abstraktion und Voraussetzung für Gestaltungs-Alternativen sehr wichtig!

Im Beispiel: Das Hauptverfahren und das Nachrückverfahren.

Nachdem nun die Kommunikationsbeziehungen, Kooperationen und die wichtigsten gemeinsamen Tätigkeitsmuster beschrieben sind, werden nun auch die einzelnen Aufgabenbereichen der einzelnen Akteure identifiziert und beschrieben.

Aktivitätsmuster bieten sich als Einstieg in die Modellierung der Tätigkeitsprofile der beteiligten Akteure geradezu an, da sie bereits deren Aufgabenbereiche andeuten und einen Verständnisrahmen für deren Einzeltätigkeiten bieten. Die lokale Aktivität der einzelnen Akteure wird in einen größeren Zusammenhang gesetzt und in ihrem zeitlichen Ablauf verdeutlicht.

5.4.5. Organisation

Ziel: Beschreibung der Aufbauorganisation

Von den bisher identifizierten Kooperations- und Interaktionsbeziehungen, Konversations- und Aktivitätsmustern lassen sich erste Bestandteile des zu berücksichtigenden Anwendungsbereichs ableiten. Wir haben sowohl einen Teil der Akteure als auch einen großen Ausschnitt ihres Aufgabenbereichs kennengelernt. Die Organisation wird nun sowohl in ihrer globalen Struktur als auch in ihren lokalen Aufgabenbereichen näher untersucht.

Struktur

Im ersten Schritt wird nun zunächst die "Aufbauorganisation" beschrieben, d.h. die statischen Beziehungen aller identifizierten Akteure.

Eine Organisation ist in der Regel aus mehreren Organisationseinheiten aufgebaut, die bis hinunter zu "Stellen" verfeinert werden können. Eine Stelle wird von konkreten Akteuren besetzt, die bestimmte Aufgaben erfüllen. Mehrere dieser Aufgaben können oft an bestimmte Rollen geknüpft werden, die von Akteuren zu erfüllen sind. Diese Rollen sind nützlich bzw. notwendig, um von einzelnen Individuen abstrahieren zu können und deren Tätigkeitsprofile in einem einzigen Rollen-Profil konsolidieren zu können.

Zur Modellierung

Die Organisationstruktur wird in Form eines Konzeptmodells (siehe Kap. 4.3.1) beschrieben, das die verschiedenen Organisationseinheiten als abstrakte oder konkrete Akteure beschreibt und ihre statischen (Aggregations-, Spezialisierungs-) bzw. dynamischen (Rollen-) Beziehungen dokumentiert.

Abb. 5.10 zeigt als Beispiel den projektrelevanten Teil der Aufbauorganisation des Studiensekretariats:

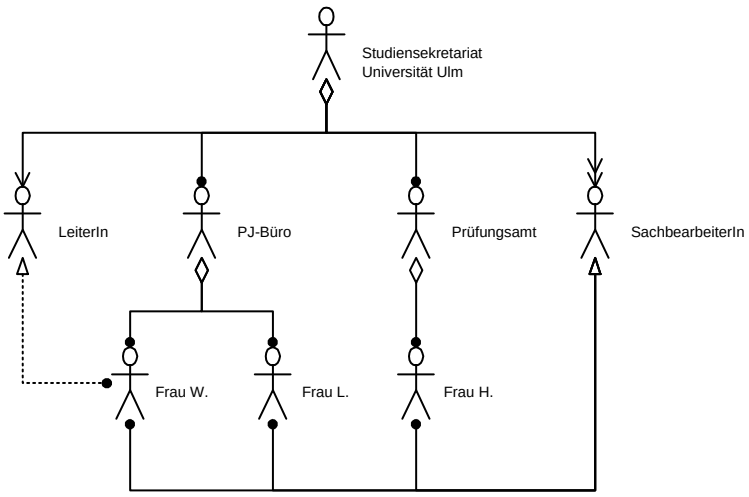


Abbildung 5.10.: Ein Ausschnitt der Aufbauorganisation des Studiensekretariats als Konzeptdiagramm

Das IS soll im Wesentlichen das PJ-Büro in seiner Arbeit unterstützen. Dieses Büro besteht aus zwei Akteuren, der Leiterin des Studiensekretariats und einer zusätzlichen Sachbearbeiterin.

Das PJ-Büro arbeitet eng mit Fr. H. zusammen, die die Aufgaben des Prüfungsamtes wahrnimmt.

Zum Vorgehen

Ausgehend von allen bisher identifizierten Akteuren wird nun die Aufbauorganisation des betrachteten Anwendungsbereichs untersucht und beschrieben.

Bei der Modellierung sind folgende Leitfragen zu klären:

Aus welchen Akteuren besteht das Nutzungssystem?

Hier finden sich potentiell alle Akteure der bisherigen Modellierung wieder. Es werden alle menschlichen und technischen Akteure identifiziert, die vom neuen Softwaresystem betroffen sind.

Im Beispiel: Frau W., Frau L und Frau H.

Wie sind die identifizierten Akteure korreliert?

Welche Beziehungen existieren zwischen den Akteuren? Wie können Akteure zu organisatorischen Einheiten zusammengefasst bzw. verfeinert werden? Dabei sind sowohl kurz- (z.B. Projektteams, Interessengruppen) wie langlebige Organisationseinheiten (z.B. Abteilung, Stelle) zu berücksichtigen.
Welche sonstigen Beziehungen existieren zwischen den Organisationseinheiten (z.B. Weisungsbefugnis, Installierend, Auflösend)? Welche Typen und Instanzen gibt es?

Im Beispiel: Frau W. und Frau L. bilden das “PJ-Büro”, eine Unterabteilung des Studiensekretariats. Daneben gibt es noch das Prüfungsamt, dem Frau H. zugeteilt ist etc.

Welche Generalisierungs- und Spezialisierungsbeziehungen exisitieren?

Welche Typen können unterschieden werden? Welche Beispielinstanzen sind existent?

Im Beispiel: Frau W. , Frau L und Frau H. sind SachbearbeiterInnen.

Welche Zuständigkeitsbereiche können unterschieden werden?

Lassen sich einzelne Aufgabengebiete identifizieren, die von der aktuellen Organisationsstruktur unabhängig sind, d.h. delegiert werden können? Können Aktivitätsbereiche in Form von Rollen expliziert werden, um so von der aktuellen Aufgabenverteilung zu abstrahieren? Lassen sich auf diesem Weg bereits potentielle Nutzergruppen des zukünftigen IS ableiten?

Im Beispiel: Frau W. ist Leiterin des Studiensekretariats.

Tätigkeitsprofile

Nach der Darstellung der aufbaulogischen Struktur der Organisation folgt nun die aufgabenlogische Beschreibung seiner Bestandteile. Es werden die Ziele, Aufgaben, Anforderungen, Randbedingungen und Aktivitäten der einzelnen Bestandteile der Organisation näher untersucht. Dabei wird nicht nur deren Aktivität selbst, sondern auch ihr umfassender Kontext analysiert.

Der Fokus der Modellierung wechselt von der globalen Organisations- und Kommunikationssicht zur subjektiven Ebene der beteiligten Personen bzw. vorhandener technischer Systeme und ihrer Aktivitäten. Die Analyse der Arbeitsbereiche und ihre Gestaltung erfordert eine besonders starke Beteiligung bzw. Berücksichtigung der Betroffenen, so dass auch hier die Verständlichkeit und Einfachheit des Modells im Vordergrund steht.

Zur Modellierung

Die Tätigkeitsprofile werden für jeden Akteur sowohl grafisch in Form eines Profildiagramms (siehe Kap. 4.3.2) als auch in Form einer textuellen Beschreibung dokumentiert. Die grafische Darstellung ist dabei vor allem in der ersten Skizzierung des Tätigkeitsbereichs eines Akteurs nützlich, da die gemeinsame Diskussion durch die flexibel erweiterbare baumartige Repräsentation der Aktivitäten wie auch der übergeordneten Ziele optimal gefördert wird. Diese Darstellung wurde daher auch in der Phase der Vision bevorzugt. Über die gemeinsame explorative Abstraktion – z.B. von konkreter Aktivität zu umfassender Aktivität hin zu Zielen, Oberzielen und Motiven – und/oder Konkretisierung von Modellelementen – z.B. von Zielen zu Aktivitäten und zu Einzeltätigkeiten – wird sukzessive ein umfassenderes Verständnis der Aktivitätsumgebung entwickelt.

Zur weiteren Detaillierung der Aktivitätsbeschreibung können auch Aktivitätsdiagramme benutzt werden, die szenarisch oder mustergültig einzelne Tätigkeitsfolgen zur konkreten Ausführung der spezifizierten Aufgaben dokumentieren. Sie beschreiben aktuelle Handlungsmuster oder mögliche Handlungspläne für die Umsetzung der im Tätigkeitsprofil spezifizierten Aufgabenbereiche. Im Apvel-Beispiel könnte man hier z.B. das Verfahren angeben, wie man

in der Verteilung vorgegangen ist, als man sie noch von Hand ohne Systemunterstützung durchgeführt hat.

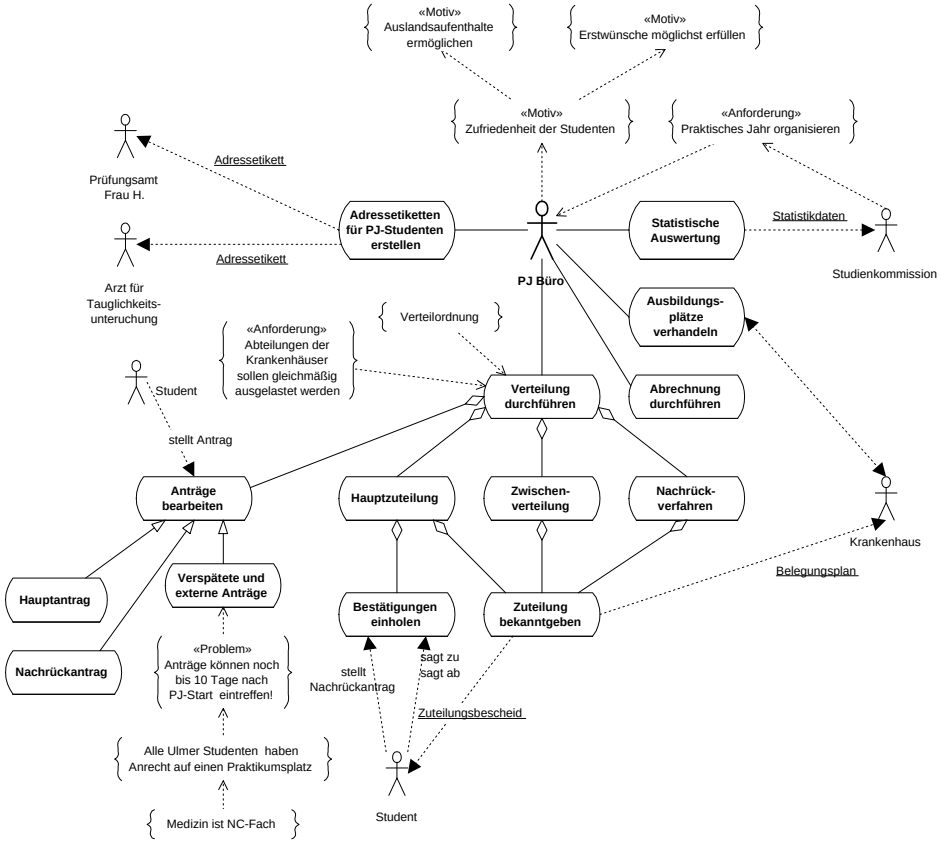


Abbildung 5.11.: Ein Profildiagramm für das Tätigkeitsprofil des PJ-Büros

Abb. 5.11 zeigt als Beispiel ein Profildiagramm für das allgemeine Tätigkeitsprofil des PJ-Büros:

Zentrale Tätigkeitsbereiche sind die Bearbeitung der verschiedenen Arten von Anträgen, die Zuteilung der Ausbildungsplätze in den diversen Verfahrensabschnitten und die Vor- und Nachbereitung der Zuteilung in Form der Verhandlung der Ausbildungsplätze mit den einzelnen Krankenhäusern, der Veröffentlichung und statistischen Auswertung der Zuteilungsergebnisse und der Durchführung der Abrechnung mit den einzelnen Krankenhäusern.

Der Aktivitätsbereich Verteilung wird in die Unterbereiche “Haupt”-, “Zwischen”- und “Nachverteilung” unterschieden während die Aktivität “Anträge bearbeiten” in die Varianten “Hauptantrag”, “Nachrückantrag” und “Verspätete und externe Anträge” unterteilt wird.

Das Diagramm enthält desweiteren Zusatzangaben über einschränkende oder motivierende Randbedingungen der einzelnen Aktivitätsbereiche.

Die Bearbeitung verspäteter Anträge hat die Tatsache zu berücksichtigen, dass Anträge noch bis kurz vor dem Start des PJ eintreffen können. Dies begründet sich aus dem gesetzlichen Anspruch von Ulmer Medizinstudenten auf einen Praktikumsplatz.

Desweiteren werden im Diagramm auch Anforderungen und Motive aufgeführt, die die Arbeit des PJ-Büros global beeinflussen bzw. alle Tätigkeiten des Büros originär verursachen bzw. beeinflussen.

Leitmotiv der Arbeit des PJ-Büros ist beispielsweise die Erzielung von Zufriedenheit auf Seiten der Medizinstudenten. Dies umfasst zum Einen die Erfüllung ihrer Erstwünsche bezogen auf

die Zuteilung zu Krankenhäusern und Wahlfächern, zum Anderen die Berücksichtigung ihrer Auslandsaufenthalte.

Andere Einflussgrößen sind die von externen Akteuren gestellten Anforderungen wie z.B. die zentrale Aufgabe des PJ-Büros "Organisation des Praktischen Jahres", die dem PJ-Büro von der Studienkommission gestellt wird.

Zum Vorgehen

Ausgehend von den bisherigen Ergebnissen der Kooperationsmodellierung und dem Organisationsstrukturdiagramm werden die Aktivitäten aller organisatorischen Einheiten identifiziert und modelliert, die entweder von eingehenden Interaktionen ausgelöst werden (können) oder ausgehende Interaktionen zur Folge haben.

Die Kooperationen, Konversationen und Aktivitätsmuster der Arbeitsumgebung (Globalsicht) stellen die Grundlage zur Analyse der Tätigkeitsprofile der einzelnen Akteure dar (Lokalsicht). Durch die Analyse von Zuständigkeitsbereichen werden nun die lokalen Aktivitäts- und Interaktionsanteile der an den Kooperationen beteiligten Akteure identifiziert. Dabei stehen die lokalen Tätigkeitsprofile solcher Akteure im Vordergrund, die vom zukünftigen System potentiell unterstützt werden sollen. Ausgehend von den Kooperationen und Kommunikationen mit der Umgebung werden nun alle lokalen Aktivitäten und ihre Motive, Randbedingungen etc. abgeleitet und verfeinert. Darauf aufbauend kann dann in der Systemgestaltung (siehe Abschnitt 5.5) die Umgestaltung der lokalen Profile aufgrund des neuen Systems erfolgen, indem die lokalen Aktivitäten bzgl. der Interaktionen mit dem System verfeinert bzw. umgestaltet werden.

Um den Aufwand zur Dokumentation der Tätigkeitsprofile zu minimieren, werden die realen Akteure möglichst nicht jeder für sich beschrieben, sondern stellvertretend durch Akteurs-Rollen oder -Typen modelliert. Durch geeignete Generalisierungen kann der Spezifikationsaufwand weiter verringert werden. Desweiteren kann es auch nützlich und ausreichend sein, nur das Profil der Organisationseinheiten anzugeben, statt Einzelprofile von aggregierten Akteuren zu beschreiben.

Um die Tätigkeitsprofile erstellen zu können, halten die Entwickler engen Kontakt zu den betroffenen Anwendern. Beginnend mit einer Sammlung der einzelnen Aufgabenbereiche werden diese sukzessive in Form eines Baums strukturiert, der auf der einen Seite die übergeordneten Aktivitäten bzw. verursachenden Motive identifiziert und auf der anderen Seite die Aktivitäten in Teilaktivitäten zerlegt und verfeinert, zumindest soweit, wie sie für die spätere Systemunterstützung Relevanz besitzen.

In der Analyse des abstrakteren Kontexts jeder einzelnen Aktivität geht der Entwickler über ihre aktuelle Ausprägung hinaus, indem er die Hintergründe, übergeordnete Aktivitäten, Motive, Interaktionsbeziehungen und Randbedingungen der Aktivität dokumentiert und somit hieraus in der späteren Phase der Systemgestaltung auch alternative Vorgehensweisen entwickeln bzw. durchspielen kann. Durch Aufdecken der motivationalen Ursprünge der beobachtbaren Tätigkeiten eines Akteurs entwickelt sich sukzessive ein Verständnis für seine "Problemsituation". Je mehr über seine Motive, Ziele, Wünsche und Aufgaben bekannt ist, desto einfacher lässt sich zum Einen das Modell seines Tätigkeitsprofils vervollständigen und zum Anderen die Passgenauigkeit eines Softwaresystems sicherstellen bzw. bewerten, das den modellierten Akteur in seiner Arbeit unterstützen soll.

Die Abstraktion der aktuellen Arbeitstätigkeit in Form eines Tätigkeitsprofils ist überwiegende Aufgabe der Entwickler, die die betroffenen Akteure befragen, sie in ihrer praktischen Arbeit beobachten und Hintergründe ihrer Tätigkeiten aufdecken. Da Außenstehende in größerem Abstand zum Geschehen stehen, fällt es ihnen potentiell leichter, das Geschehen im Überblick zu betrachten und damit in einem größeren Kontext darstellen zu können.

Zur Modellierung einer Einzeltätigkeit sieht $\text{KoA}_\text{L}\text{A}$ neben der deskriptiven Beschreibung auch die Möglichkeit der Dokumentation von Aktivitätsmustern vor. Die Darstellung solcher akteurlokaler Handlungspläne in Form von Aktivitätsdiagrammen stellt oft einen guten ersten Einstieg in die Profilmodellierung dar. Ziel der Modellierung ist aber eine möglichst umfassende Darstellung aller Aufgabengebiete des Akteurs, nicht so sehr der genaue Ablauf im Einzelnen.

In den meisten modernen Arbeitsorganisationen sind die zu beobachtenden Tätigkeiten nicht starr vordefiniert. Eine "tayloristische" Arbeitsteilung, die Arbeitsabläufe fest vorschreibt, ist hier eher selten zu finden. Vorherrschend sind hingegen Arbeitsplätze mit ausgeweiteter Selbstverantwortung, in denen flexibel auf gegebene Bedingungen reagiert werden kann. In der Modellierung konzentriert man sich daher auf eine möglichst vollständige Erfassung aller Aktivitätsbereiche bzw. Randbedingungen dieser Aktivitäten. Eine Darstellung des logischen Ablaufs in Form von Handlungsplänen ist meist unnötig. Ein Beispiel im Apvel-Projekt hierfür ist das Verfahren der Zuteilung, dessen Ergebnis zwar durch eine Verteilordnung weitestgehend reglementiert wird, für das jedoch im Modell kein explizites Vorgehen beschrieben wird. Die konkrete Umsetzung bleibt auch in ihrer Modellierung dem Ausführenden weitgehend freigestellt.

Bei der Modellierung ist folgenden Leitfragen nachzugehen:

Wie lassen sich Kooperationen in Einzelaktivitäten zerlegen?

> Welchen Anteil nehmen die beteiligten Akteure an den Kooperationen, die im Kooperationsmodell dokumentiert wurden. Welche Teilaufgaben haben sie zu verantworten?
>
> Im Beispiel: Das PJ-Büro muss die Abrechnung vornehmen, Adressetiketten erstellen, etc.

Welchen Aktivitäten lassen sich die identifzierten Interaktionen zuordnen?

> Welche lokalen Tätigkeitsbereiche führen zur Auslösung der dokumentierten Interaktionen? Was sind die übergeordneten Ziele der Kommunikationstätigkeit? Welche Aktivitäten werden durch die Interaktionen ausgelöst oder beeinflusst?
>
> Die in den dokumentierten Kooperationsbeziehungen und Konversationen thematisierten Interaktionen können in der Regel lokalen Aktivitätsbereichen zugeordnet werden. Sowohl beim "Sender" als auch beim "Empfänger" lassen sich die Interaktionsaktivitäten einer umfassenderen Aktivität zuordnen.
>
> Im Beispiel: das PJ-Büro muss Verträge erfassen bzw. bearbeiten. Das Ergebnis der Zuteilung muss bekanntgegeben werden etc.

Welche Aktivitätsbereiche der Akteure lassen sich identifizieren?

> Was sind die wichtigsten Alltagsaktivitäten der Akteure? Welche Aktivitätsmuster wiederholen sich in der Arbeitspraxis?

Im Beispiel: Das PJ-Büro bearbeitet Anträge und führt Verteilungen durch.

Wie lassen sich Aktivitäten weiter verfeinern oder zusammenfassen?

Welche Unteraktivitäten können in den dokumentierten Aktivitätsbereichen unterschieden werden? Welche übergeordneten Aktivitätsbereiche bzw. Ziele oder Motive werden mit den Aktivitäten verfolgt?

Dabei sind vor allem solche Aktivitäten interessant, die potentielle Kandidaten für eine Unterstützung durch das zukünftige Softwaresystem sind.

Im Beispiel: Die Verteilung zergliedert sich in drei Unterphasen: Haupt-, Zwischen- und Nachrückverfahren.

Welche alternativen Handlungsstränge können unterschieden werden?

Lassen sich Aktivitäten in alternative Klassen unterteilen, die unterschiedliche Problemsituationen abdecken? Können grundlegende Varianten der Aktivitätsbereiche ausgemacht werden?

Im Beispiel: Die Pflege von Anträgen kann in drei Fälle unterschieden werden: Haupt-, Nachrück- und externe Verträge.

Welche Motive oder Randbedingungen lassen sich identifizieren?

Aus welchen Motiven, Aufgaben oder Anforderungen heraus werden die identifizierten Aktivitäten durchgeführt? Welche Randbedingungen sind dabei zu beachten? Welche Bedingungen werden durch andere Akteure gestellt? Welche Bedingungen oder Probleme können durch die Aktivitäten begründet oder verursacht werden?

Aktivitäten können evtl. auch neue Problemsituationen hervorrufen bzw. aufdecken, die wiederum zu neuen Aktivitäten führen können.

Im Beispiel: Das oberste Motiv für die Arbeit des PJ-Büros ist die Zufriedenheit der Studenten. Dieses Ziel geht sogar der Erfüllung von Krankenhaus-Wünschen vor. Die Auslastung der Krankenhausabteilungen sollte möglichst gleich sein etc.

5.4.6. Sonderfälle

Ziel: Eruierung notwendiger Ausnahmebehandlungen bzw. Regelabweichungen

Als letzter Schritt der Kontextmodellierung werden Sonderfälle thematisiert, die in der Alltagspraxis auftauchen und dort eine außergewöhnliche Abweichung vom Standardprozedere verursachen (können). Sonderfälle kommen bereits ganz natürlich auch in den früheren Phasen der Kontextmodellierung zur Sprache, werden nun aber explizit angesprochen, gesammelt und textuell dokumentiert. Sonderfälle der Arbeitstätigkeit sind in der Gestaltung besonders zu berücksichtigen, da die Effektivität eines Arbeitsplatzes stark mit der Fähigkeit korreliert ist, mit Ausnahmefällen effizient umgehen zu können [Sac95] und die Behandlung dieser Ausnahmefälle

- in der Arbeitspraxis oft Probleme bereitet, die durch das neue IS entschärft oder gelöst werden sollen

- oft sehr "teuer" sind, wenn sie vom IS unterstützt werden sollen, und daher
- vom IS oft nicht direkt "unterstützt" werden, aber trotzdem geeignet zu behandeln sind.[7]

Zur Modellierung

Sonderfälle ziehen eine Erweiterung aller Kontextmodelle nach sich. Sie werden in Form globaler Bedingungen modelliert und können beliebige Modelle beeinflussen. Ausnahmen können ihren Niederschlag in der Modellierung auch direkt in Form zusätzlicher Spezialisierungen bestimmter Aktivitäten finden, die Beispiele der Behandlung von Ausnahmen dokumentieren.

Im Folgenden einige Beispiele für Sonderfälle aus dem Apvel-Projekt:

- Wenn sich ein Student einmal beworben hatte und zugeteilt wurde, dann aber weder zu- noch abgesagt hat oder seiner Stelle unentschuldigt ferngeblieben ist, wird er bei einer erneuten Bewerbung nachrangig behandelt.
- Fällt ein Student durch das dritte Staatsexamen, muss er entsprechende Fächer des PJ wiederholen. Da die Prüfungsergebnisse erst bekanntgegeben werden, wenn das nächste PJ bereits gestartet ist, müssen solche Studenten im Normalfall kurzfristig per Hand in die zu wiederholenden Tertiale eingeteilt werden.
- Bestimmte Studenten werden bevorzugt behandelt. Alleinstehende mit Kind dürfen sich z.B. die zeitaufwändige Chirurgie als 3. Tertial wünschen und können ihre übrigen Fehltage damit komplett zum Schluss nehmen.
- Bis zu zwei Tertiale können im Ausland verbracht werden; davon eines sogar auch nur zur Hälfte. Entsprechende Vereinbarungen sind in der Zuteilung zu berücksichtigen. Da Auslandsaufenthalte jedoch oft "platzen" (Vereinbarungen halten nicht oder Aufenthalte werden abgebrochen), sind entsprechende Vorbereitungen zu treffen.
- In Ausnahmefällen kann in einem Tertial auch auf Teilzeitbasis gearbeitet werden. Der Zeitraum der Beschäftigung verlängert sich dann entsprechend.

Sonderfälle stellen besondere Konstellationen, Anforderungen oder Bedingungen heraus, die bei der Gestaltung des Systems zu beachten sind. Gerade solche Sonderfälle fallen in einer methodischen und "auf Effizienz getrimmten" Vorgehensweise gerne unter den Tisch, da sie "lästige" Ausnahmen von der Regel darstellen. Nichtsdestotrotz entscheidet gerade die Art und Weise der notwendigen Ausnahmebehandlung über die Akzeptanz des Softwaresystems beim Endanwender.

Zum Vorgehen

In der Modellierung sind folgende Leitfragen zu bearbeiten:

Welche Sonderfälle gibt es?

[7] Bei der späteren Ausgestaltung des Informationssystems (siehe Abschnitt 5.5) spielen diese Fälle eine besondere Rolle. Sofern sie Einfluss auf die spätere Systemnutzung besitzen, sind sie in der Systemgestaltung zu berücksichtigen und es ist für sie in der Fiktionsphase (Abschnitt 5.6.1) ein Nutzungskonzept zu entwickeln.

Welche Ausnahmesituationen gibt es? Welche Ereignisse verursachen ein Ausnahmeverfahren? Welche besonderen Bedingungen machen in der Arbeitspraxis Probleme? Für welche Spezialfälle wird viel Arbeitszeit benötigt? Mit welchen Komplikationen muss gerechnet werden?

Sonderfälle werden als Bedingungen modelliert.

Im Beispiel: Falls ein zugeteilter Student das zweite Staatsexamen nicht besteht, wird er bei der nächsten Bewerbung zu einem Praktischen Jahr wie ein Erstbewerber behandelt etc.

Welche Abweichungen von der Regel ergeben sich regelmäßig?

Welche besonderen Ausnahmeverfahren gibt es und warum? Welche Abweichungen vom Normalverfahren lassen sich beobachten?

Dokumentiert werden solche Sonderfallbehandlungen als Spezialisierungen von Aktivitäten.

Im Beispiel: Alleinstehende mit Kind werden "bevorzugt" behandelt. Teilzeitbeschäftige leisten ihr Praktisches Jahr entsprechend länger ab etc.

Was passiert, wenn ...?

Wie wird in den "anderen" bzw. selteneren Fällen verfahren?

Diskussion aller Varianten im Ablauf. Diese Diskussion sollte vor allem von den Entwicklern stimuliert werden, indem sie die dokumentierten Standard-Abläufe schrittweise hinterfragen. Konversations- und Aktivitätsmuster werden entsprechend ergänzt.

Im Beispiel: Was passiert, wenn ein Student im Praktischen Jahr krank wird? Was passiert, wenn ein Auslandsaufenthalt im letzten Moment "platzt"?

Sonderfälle werden als eigenes ausgezeichnetes Modellkonzept eingeführt, um sowohl die Modellierung als auch den Gestaltungsprozess geeignet fokussieren zu können. Einige Fälle finden ihren Niederschlag zwar oft bereits auch in Form ausgezeichneter Kommunikations- oder Aktivitätsmuster und werden damit in den Kooperationsmodellen und Tätigkeitsprofilen bereits – evtl. auch partiell – berücksichtigt. Ihrer Natur nach sind diese Sonderfälle jedoch auch allgemeine übergeordnete "Aussichtspunkte" bzw. bestimmte Blickwinkel, die den Prozess der Modellierung und Gestaltung beeinflussen und lenken. Desweiteren stellen sie zusätzlich auch nützliche Ausgangspunkte für die Identifikation von geeigneten Ausnahmeszenarien bei der Beschreibung des Nutzungskonzepts dar (siehe Fiktionsphase, Abschnitt 5.6.1).

5.4.7. Zusammenfassung

Die Kontextanalyse dient einer umfassenden Modellierung der aktuellen Arbeitspraxis im Umfeld des neu einzuführenden IS. Erst mit der genauen Kenntnis der Tätigkeitsbereiche und Abhängigkeiten aller betroffener Akteure kann ein Gestaltungsprozess initiiert werden, der zu einem Softwaresystems führt, dass sich erfolgreich in diesen Tätigkeitsbereich integriert.

Ausgehend von einer Darstellung aller Akteure des Tätigkeitsbereichs und ihrer abstrakten Kooperations- bzw. Kommunikationsbeziehungen wird die Alltagspraxis zunächst anhand der

vorzufindenden Standard- Konversations- und Aktivitätsmuster skizziert, bevor man dann den lokalen Tätigkeitsbereich jedes einzelnen Akteurs näher konkretisiert und um ihren Kontext in Form von Motiven, Randbedingungen, Anforderungen und Interaktionsbeziehungen erweitert.

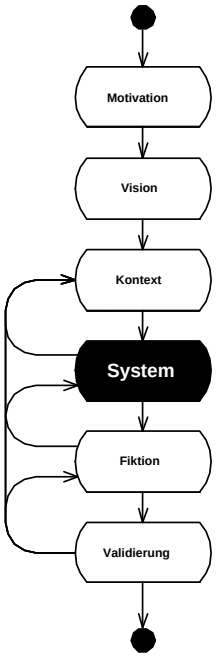
Die detaillierte Beschreibung aktueller Aktivitäten in Kombination mit der Identifikation und Definition der Anlässe, Begleiterscheinungen und Randbedingungen dieser Aktivität schafft Raum für die Diskussion von alternativen Umgestaltungoptionen wie z.B. durch die Einführung neuer Softwaresysteme.

Im Falle der Ablösung von Altsystemen kann die Kontextanalyse auch Reengineering-Tätigkeiten implizieren. Ausgehend von dem abzulösenden System, seinem gegebenem User-Interface und dem aktuellen Nutzungskonzept sind die zentralen Konzepte, Aufgaben, Kooperationsziele des Softwareeinsatzes zu identifizieren und damit die eigentlichen Anforderungen der Nutzer bzw. des Unternehmens im Kontext des Altsystems aufzudecken.

In standardisierten Kontexten kann man evtl. auch auf generische "Domänenmodelle" zurückgreifen, welche den Anwendungskontext ganzer Klassen von Problemsituationen modellieren, und diese als Grundlage für eine detailliertere Analyse der konkreten Situation verwenden. Eine solche Wiederverwendung von Produkten des Requirements Engineering fordert z.B. Jackson [Jac94a].

Die Skizzierung des neuen IS ist nun Inhalt des nächsten Schritts von KoA$_{\text{L}}$A, der *Systemgestaltung*.

5.5. Systemgestaltung



Ziel: Definition des IS als neuer technischer Akteur aus der Sicht der Anwender (das "Lastenheft")

Eine ursprüngliche Vorstellung über das Leistungsprofil des neuen Softwaresystems findet sich bereits in den Ergebnisdokumenten der Visionierungsphase wieder.

In der Gestaltungsphase nimmt der System-Akteur nun konkretere Gestalt an. Basierend auf der vorläufigen Ideensammlung (Vision) und der gemeinsam gefundenen Priorisierung der geforderten Systemmerkmale wird der Akteur weiter spezifiziert – und evtl. in einzelne "intelligente Assistenten" aufgesplittet –, in den Arbeitskontext eingesetzt und in seinen Interaktionsbeziehungen und Dialogabläufen untersucht.

Schon mit der im letzten Kapitel beschriebenen Analyse, also der Modellierung des globalen Arbeitskontexts beginnt der eigentliche Gestaltungsprozess, da durch Auswahl und Modellierung der betroffenen Realweltausschnitte bereits eine Selektion und Projektion des betrachteten Gegenstandsbereichs stattfindet und sich in dieser Phase unweigerlich Ideen zur weiteren Ausgestaltung des Informationssystems entwickeln.

Die Phase der Systemgestaltung dient nun zur expliziten Ausgestaltung des neuen Softwaresystems, das nun als neuer Akteur in Erscheinung tritt und bestehende Kooperationen bzw. Aktivitäten unterstützt und/oder neue Kooperationen bzw. Konversationen zur Folge hat. Die ersten Ideen für diesen Akteur in Form der Vision (Abschnitt 5.3) werden noch einmal hinterfragt bzw. weiterentwickelt.

Die eigentliche Entwicklung der Systemgestalt beginnt zwangsläufig schon mit der Kontextmodellierung, genau wie sich der modellierte Kontext automatisch mit Einführung des System-Akteurs ändert und zu einer erneuten Iteration der Kontextmodellierung führen kann. Als Aktivität ist die Systemgestaltung allerdings der Kontextmodellierung logisch nachgeordnet, da nur mit dem Wissen über den aktuellen Zustand auch eine optimale Systemunterstützung entwickelt werden kann.

Zum Vorgehen

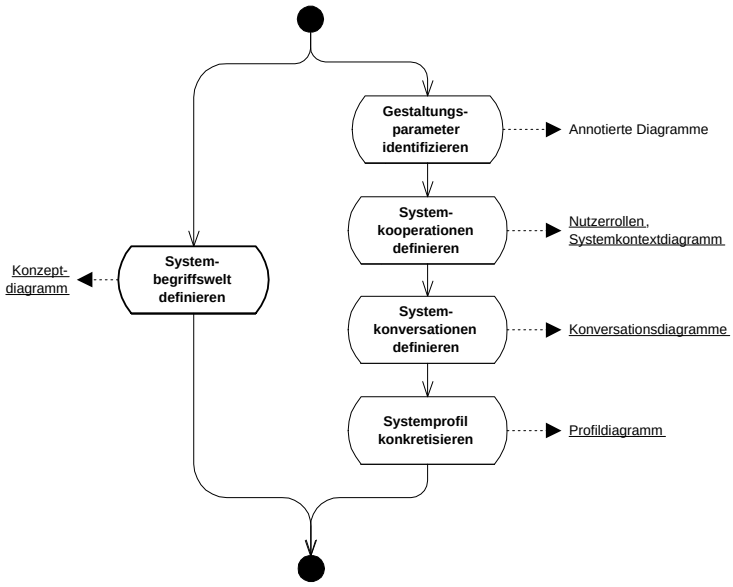


Abbildung 5.12.: Das Vorgehen bei der Gestaltung

Abb. 5.12 zeigt das Vorgehen in der Gestaltungsphase. Ziel dieser Phase ist eine Modellierung des zukünftigen Nutzungssystems, also eine Beschreibung der Tätigkeitsbereiche des neuen Informationssystems und seiner engeren Umgebung, d.h seiner Nutzer und technischer Nachbarsysteme.

Das Vorgehen gliedert sich in die folgenden Abschnitte:

> Nach der Identifikation der durch das neue Softwaresystem betroffenen Bestandteile ausgehend von den bereits dokumentierten Modellen der Kontextmodellierung – das sind z.B. alle Akteure und ihre Aktivitäten bzw. Interaktionen, die vom neuen Softwaresystem betroffen sein werden –
>
> wird zunächst der Systemkontext skizziert. Hier wird eine erste Idee für die direkte Umgebung des Softwaresystems, also seine menschlichen Anwender und technischen Nachbarsysteme, zusammen mit ihren Interaktions- und Kooperationsbeziehungen entworfen und dokumentiert.
>
> Im nächsten Schritt werden diese Beziehungen an Hand einer Beschreibung von charakteristischen Interaktionsmustern zwischen dem System und den Akteuren der Anwendungsumgebung konkretisiert und in Form von Konversationsdiagrammen dokumentiert.
>
> Die Beschreibung des definitiven lokalen Tätigkeitsprofils des IS folgt erst im letzten Schritt. Hier werden die einzelnen Aufgabenbereiche des Systems, ihre Randbedingungen und begleitenden Interaktionsbeziehungen, ihre Struktur und evtl. auch einzelne lokale Aktivitätsmuster festgelegt und ausführlich dokumentiert.
>
> Parallel zu den genannten Aktivitäten findet eine sukzessive Konkretisierung des “Sprachschatzes” des Softwaresystems statt. Die Systembegriffswelt wird suk-

zessive aus der allgemeineren Begriffswelt der Anwendungsumgebung und seiner Akteure entwickelt bzw. abgeleitet.

KoA$_L$A modelliert das Softwaresystem als eigenständigen Akteur mit spezifischem Interaktionsverhalten. Dieses Leitbild von KoA$_L$A ist maßgebend für die weiterführende Gestaltungstätigkeit. Es unterscheidet sich wesentlich von dem werkzeug- oder maschinenorientierten Leitbildern, wie sie in den klassischen Softwareentwicklungsmethoden vorherrschen und für die weitere Konkretisierung und Spezifikation technischer Softwaresysteme verwendet werden:

Das Softwaresystem

- tritt als selbständiger, u.U. auch autonomer Akteur in Erscheinung, der ein spezifisches Aufgabengebiet besitzt. Seine Tätigkeiten werden nicht auf operationaler Ebene formal spezifiziert – z.B. als einzelne maschinelle Operationen mit spezifischer Schnittstelle und deterministischen Transformationen des System-Zustandsraums –, sondern in der Sprache der Anwender auf abstrakterer Ebene deklarativ über Aufgabenbereiche und nur inhaltlich festgelegte Interaktionsbeziehungen beschrieben. Nicht die Ausprägung seiner lokalen Tätigkeiten stehen im Vordergrund der Dokumentation, sondern die Rolle des Systems in Bezug zu seiner Anwendungsumgebung, seine organisatorische Einbettung in ein Netz von kooperierenden und daher auch kommunizierenden Akteuren.

- wird als "intelligent" angesehen. Es trifft u.U. auch selbständige Entscheidungen[8], so dass sein Verhalten auch nicht vollständig spezifiziert werden muss/kann. Nicht die genaue Ausprägung seines Verhaltens genießt in der Darstellung oberste Priorität, sondern die Dokumentation der Ursprünge (Aufgaben, Ziele, Aktivitätsbereiche), Hintergründe (Interaktionen, Motive, Probleme), Randbedingungen und Auswirkungen (Bedingungen, Interaktionen, Probleme) seines Verhaltens.

- interagiert mit seiner Umgebung. In der Modellierung steht dabei nicht die Form oder das Medium der Kommunikation im Vordergrund sondern ihr Ziel. Infolgedessen ist die Darstellung auch nicht auf die Ebene konkreter technischer Konstrukte wie Funktionsaufrufe, Nachrichten, Signale oder gar konkrete Interaktionen mit Komponenten grafischer Benutzeroberflächen beschränkt, sondern kann beliebig abstrahierte Wirkungsflüsse in Form allgemeiner Kommunikationsereignisse zwischen Akteuren umschreiben.

- besitzt ein eigenes Weltbild bzw. fachliches Wissen. Nicht die Datenstruktur – die Syntax – steht im Vordergrund der Modellierung, sondern die Definition einer konsistenten Begriffswelt aus Anwendersicht – die Semantik, die der später zu entwickelnden Benutzerschnittstelle des Softwaresystems und den damit realisierten mentalen Modellen zugrunde liegen wird. Nicht die Darstellung bzw. Struktur, sondern die Bedeutung der kommunizierten Konzeptionen (inkl. deren Verdeutlichung über Beispiele) steht bei der Modellierung im Vordergrund.

Das IS wird also als "menschlicher" Akteur modelliert, mit dem kommuniziert werden kann, der bestimmte Aufgaben besitzt und Aufträge ausführt, der selbsttätig bestimmte Interaktionen durchführt (Einsatzplanungssysteme, Workflowsysteme) etc. Die Leitfragen der Ge-

[8] Auf der Ebene der technischen Spezifikation muss dieses Verhalten dann natürlich formalisiert werden, da ein Softwaresystem nie wirklich "selbständig" ist. Zum Zweck einer kompakten und anwenderfreundlichen intuitiven Beschreibung des Systems-Akteurs während der Gestaltungsphase wird ihm jedoch vorläufig eine eigene "Intelligenz" zugebilligt.

staltung lauten: “Welche Hilfskräfte wünschen sich die beteiligten Interessengruppen? Welche Aufgaben sollen sie übernehmen? Welche Ergebnisse werden von ihnen erwartet? Wie sollen sie sich verhalten?”

Im Rahmen der Ausgestaltung des Informationssystems ist in Abhängigkeit von der verfolgten IuK-Strategie des Unternehmens auch eine Entscheidung über den Softwaresystemtyp zu treffen (s. [Kos62]). Wir betrachten hier schwerpunktartig den Typ betriebliches Informationssystem, d.h. ein nutzergesteuertes interaktives System, das den autonomen Mitarbeiter einer Organisation als “Werkzeug” dient.

Systeme zur Unterstützung der Kommunikation bzw. Koordination, wie sie z.B. oft bei stark arbeitsteiligen Organisationen eingesetzt werden (Workflowsysteme, Groupware, Planungssysteme etc.), werden hier nur insoweit berücksichtigt, wie solche Systeme auch als eigenständige “Werkzeuge” aufgefasst werden können. Auf die Gestaltung der von solchen Systemen zu implementierenden Koordinationsstrukturen und -prozesse wird nicht näher eingegangen.

Ich stelle im Folgenden nun die einzelnen logischen Schritte vor, die zur Beschreibung des Systems und seines direkten Umfelds durchgeführt werden. In der Realität werden diese Schritte oft eng verzahnt und iterativ vonstatten gehen, genauso wie sich die Phase der Systemgestaltung im Allgemeinen auch mit der Phase der Kontextmodellierung überlappen wird.

An die Systemgestaltung schließt sich die Phase der Fiktion an, in der die Folgen der Ausgestaltung für die Anwendungsumgebung kontextreich dargestellt und diskutiert werden. Zur Diskussion und Abwägung grundlegender Gestaltungsvarianten kann es sinnvoll bzw. notwendig sein, alle Varianten vollständig auszuarbeiten und ihre “Folgen” in eigenständigen Fiktions-Modellen darzulegen.

5.5.1. Gestaltungsparameter

Ziel: Identifikation der Umgestaltungsoptionen

Um die zukünftige Systemumgebung zu skizzieren und vorläufig einzugrenzen, werden zusammen mit den Anwendern und Interessengruppen im ersten Schritt erst einmal die wichtigsten potentiellen Gestaltungsobjekte identifiziert bzw. festgelegt. Das sind Elemente der Anwendungsumgebung, die vom neuen IS betroffen sind bzw. von ihm unterstützt werden sollen.

Zur Modellierung

Hierzu werden in den Diagrammen der Kontextmodellierung die Modellelemente entsprechend hervorgehoben (z.B. farblich markiert), die vom neu einzuführenden System betroffen sind bzw. sein sollen. Hierbei werden vor allem die wichtigsten zukünftigen Anwender des Systems identifiziert und die Aktivitäten bzw. Interaktionen bestimmt, die vom zukünftigen System unterstützt werden sollen.

In Abb. 5.13 ist als Beispiel ein entsprechend annotiertes Kontextdiagramm von Apvel dargestellt. Alle Elemente, für die eine Systemunterstützung angedacht ist, wurden fett markiert:

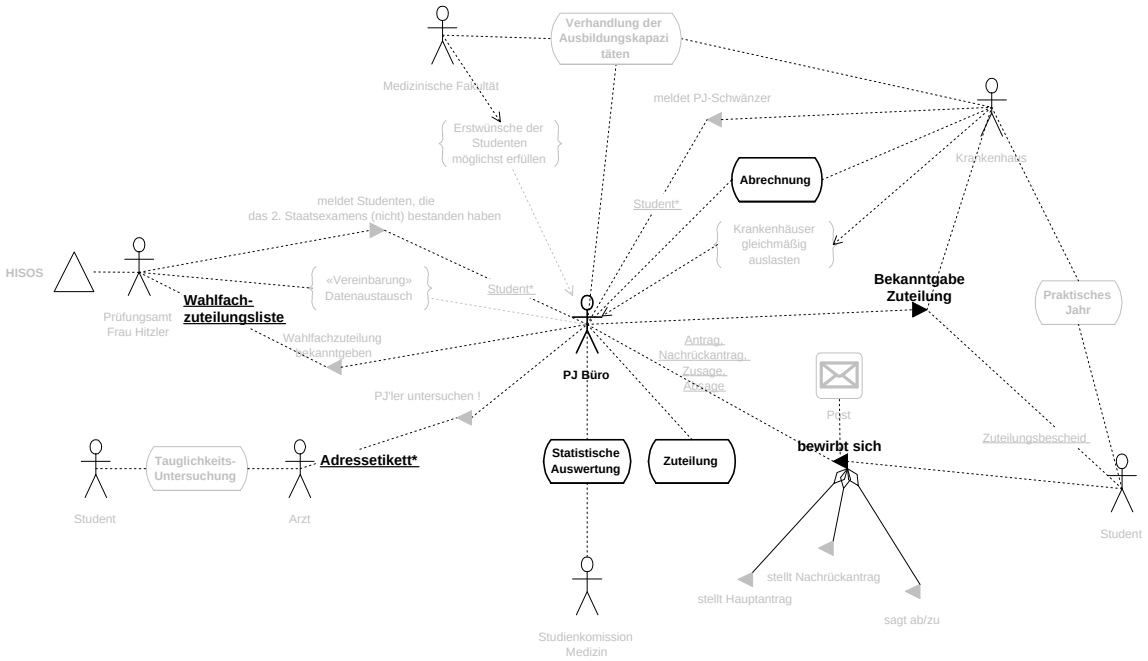


Abbildung 5.13.: Gestaltungsparameter von Apvel

Im Beispiel soll alleine das PJ-Büro in seiner täglichen Arbeit direkt unterstützt werden. Der Student wird nicht markiert, da er von der Einführung nur indirekt betroffen ist.

Im besonderen Maße soll die Bewerbung und die Bekanntgabe der Zuteilungsergebnisse an die Krankenhäuser und Studenten vom System unterstützt werden, da diese Aufgabe momentan noch sehr viel Arbeitszeit bindet. Die anvisierte Bewerbungsmöglichkeit und die Einsicht in die Zuteilungsergebnisse per Web-Server entsprechen einer Unterstützung der entsprechenden Interaktionen zwischen PJ-Büro und Student.

Die Kommunikation mit dem Arzt, der die Tauglichkeitsuntersuchungen durchführt, soll insofern unterstützt werden, als das System die notwendigen Adressetiketten erzeugen soll.

Desweiteren ist auch die Wahlfachzuteilungsliste zu generieren, die das Prüfungsamt benötigt.

An lokalen Aktivitäten soll vor allem die Zuteilung unterstützt werden, die momentan noch alleine vom PJ-Büro durchgeführt wird und nun vom System übernommen werden soll.

Desweiteren soll das System auch die statistische Auswertung vereinfachen und die Abrechnung unterstützen.

Die genaue Form der Unterstützung wird erst in den nächsten Schritten sukzessive entwickelt und beschrieben.

Zum Vorgehen

Um eine gegebene Arbeitssituation durch Einführung neuer Werkzeuge wie z.B. eines neuen IS zu verbessern, sind die Aktivitätsbereiche aller Akteure des Nutzungssystems im Zusammenhang zu untersuchen. Dabei sind sowohl die Kooperationen der Akteure untereinander als auch lokale Aktivitäten der Akteure selbst zu betrachten. Einzeltätigkeiten der Akteure sind z.B. daraufhin zu untersuchen, ob automatisierbare Anteile identifiziert werden können. Derartige operationalisierbare Tätigkeitsbereiche bzw. “Operationen” sind von den nicht automatisierbaren “menschlichen Fertigkeiten” zu unterscheiden (s.[DR97]).

Ideen zu Möglichkeiten und Alternativen werden eng mit den betroffenen Interessengruppenvertretern zusammen erarbeitet (z.B. in einem Workshop). Hierbei spielen zum Einen die in ihrer Arbeit direkt betroffenen Anwender des neuen Systems und zum Anderen aber auch die beteiligten Entwickler eine zentrale Rolle. Vor allem die Entwickler wissen um technische Möglichkeiten, deren Kosten und potentiellen Nutzen, verfügbare Komponenten oder Anwendungssysteme, notwendige Sicherungssysteme etc.

Hauptmedium bzw. -diskussionsmittel der Gestaltung sind dabei die bisher in der Kontextmodellierung entwickelten Modelle. Durch die Einführung des neuen System-Akteurs und seine weitere Ausgestaltung werden u.U. die globalen Konversations- und Aktivitätsmuster verändert, die als Ausgangspunkt für die Ausarbeitung der einzelnen Gestaltungsalternativen durch die Betroffenen der Umgestaltung dienten. Je nach Umfang dieser Auswirkungen oder der Anzahl möglicher alternativer Gestaltungsformen kann es sinnvoll sein, die erstellten Modelle unter Einbeziehung der alternativen System-Akteure neu zu erstellen.

Zur Herausarbeitung von Gestaltungsoptionen kann nach folgenden Leitfragen vorgegangen werden:

Welche Akteure sollen in ihrer Arbeit unterstützt werden?

Welche Akteure sind die zukünftigen direkten Nutzer des Softwaresystems?

Im Beispiel: das PJ-Büro.

Welche Interaktionen sollen unterstützt werden?

Welche Interaktionen sind durch die Einführung des Systems betroffen? Für welche Kommunikationsbeziehungen stellt das System ein entsprechendes Medium zur Verfügung? An welchen Interaktionen soll das System beteiligt werden?

Im Beispiel: Student “bewirbt sich” beim PJ-Büro, “Bekanntgabe Zuteilung” durch das PJ-Büro, das PJ-Büro gibt “Wahlfachzuteilung bekannt” etc.

Welche Aktivitäten sollen unterstützt werden?

Welche Aktivitäten sind kritisch und könnten automatisiert, unterstützt bzw. verbessert werden, weil sie z.B. zu fehleranfällig sind, zu viele Resourcen verbrauchen etc. ?

Im Beispiel: die “Zuteilung”, die “Statistische Auswertung” und die “Abrechnung”.

Parallel zu dieser Festlegung über die Eingriffspunkte der Umgestaltung werden natürlicherweise auch bereits Optionen der verwendeten Technologie diskutiert und erste Ideen für die Art und Zielrichtung der Umgestaltung angedacht. Um diese Ideen für den weiteren Gestaltungsprozess nutzbar zu machen, sollten sie dokumentiert werden, beispielsweise in einfachen Textlisten.

Die Aufstellung der Umgestaltungskandidaten findet in engem Dialog der Entwickler mit den Anwendern statt. Wie schon Bowers und Pycock bei der iterativen Entwicklung von Softwaresystemen festgestellt haben, sind es oft die Entwickler, die Probleme der Benutzer bemerken, Vorschläge zu Verbesserungen machen und bewusst Alternativen zur Diskussion stellen ([UU93], S.213). Von sich aus artikulieren Benutzer eher selten konkrete Anforderungen. Oft werden erst mit der Vorstellung technischer Möglichkeiten und Realisierungsoptionen durch den Entwickler konkrete Anforderungen thematisiert bzw. “generiert”.

Der Gestaltungsprozess von KoA$_{\text{L}}$A wird daher normalerweise oft vom Entwickler geleitet bzw. vorangetrieben, indem er Möglichkeiten der technischen Unterstützung aufzeigt, Alternativen der Umgestaltung vorstellt und in Absprache mit den Anwendern die wichtigsten Gestaltungsparameter festlegt, die im weiteren Vorgehen berücksichtigt werden sollen. Fokussiert wird der hierzu notwendige Abstimmungsprozess vor allem anhand der bisher erstellten Diagramme, die in übersichtlicher und komprimierter Weise ein Modell der aktuellen Arbeitspraxis darstellen.

Anhand dieser Modelle ergeben sich natürlicherweise auch Diskussionen über umfassendere organisatorische Um- bzw. Neugestaltungsoptionen. Es stellt sich die Frage, welche Interaktionen oder gar Akteure evtl. überflüssig sind oder abgelöst werden könnten bzw. wie die Organisation global geändert werden könnte, um mit weniger Arbeits- und Kommunikationsaufwand dieselben Effekte zu erzielen.

Alternativ zu diesem vom Entwickler gesteuerten Entwurfsverfahren können natürlich auch Brainstorming-Techniken eingesetzt werden, die für eine erfolgreichere Partizipation der Anwender Sorge tragen. Siehe hierzu z.B. die in [KM91] beschriebenen "Zukunftswerkstätten".

Sobald die grobe Richtung der Umgestaltung des Anwendungsumfelds festliegt, wird nun im nächsten Schritt die Rolle des neuen IS anhand seiner Einbettung in die nähere Umgebung grob beschrieben.

5.5.2. Systemkooperationen

Ziel: Skizzierung der direkten Systemumgebung

In diesem Schritt wird der Kooperationskontext des neuen IS grob festgelegt. Wesentliche Aufgabe dabei ist die Dokumentation der zu identifizierenden Anwenderrollen des zukünftigen Softwaresystems und all ihrer Kooperationen bzw. Interaktionen mit dem System.

Zur Modellierung

Dargestellt wird dieser Kontext im Wesentlichen durch ein Kooperationsdiagramm, welches die statischen Kooperationsbeziehungen des Systems und seiner direkten Umgebung – das sind seine Anwender und technischen Nachbarsysteme – beschreibt und das zusätzlich auch textuell erläutert bzw. verfeinert wird. Ziel der Modellierung ist eine erste Festlegung des Zusammenspiels von Softwaresystem und seiner näheren Umgebung. Dabei werden die Anwender möglichst kategorisiert und stellvertretend in Form von Nutzerrollen-Akteuren beschrieben. Andere technische Nachbarsysteme wie auch das IS selbst werden als technische Akteure dokumentiert.

Will man das Modell in gemeinsamen Gestaltungssitzungen zusammen mit den Anwendern entwerfen bzw. sukzessive weiterentwickeln, ist es notwendig, das Modell einfach und in seiner Darstellung verständlich zu halten. Hierzu können auch aussagekräftige grafische Symbole beitragen, die zur Darstellung bzw. Annotierung von menschlichen oder technischen Akteuren, Interaktionen oder Kooperationen benutzt werden.

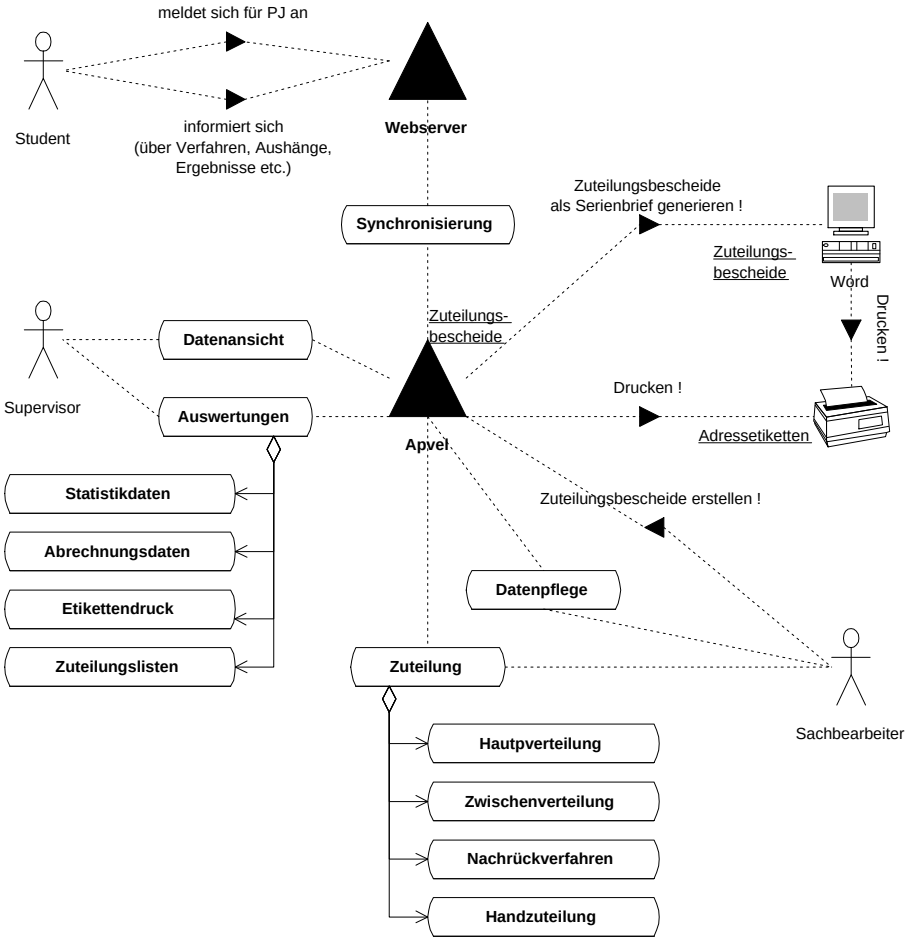


Abbildung 5.14.: Ein Kontextdiagramm für den direkten Systemnutzungszusammenhang

Abb. 5.14 zeigt ein Kontextdiagramm, das die Kooperationen des neuen IS bzw. seiner Komponenten mit seiner direkten Umgebung skizziert. Neben den wichtigsten Interaktions- und Kooperationsbeziehungen dokumentiert es in Form der Kooperationspartner auch die verschiedenen Anwenderkategorien und technischen Nachbarsysteme, mit denen es das IS zu tun hat.

> Im Beispiel sind zwei verschiedene Nutzerkategorien zu unterscheiden: Der "Sachbearbeiter" ist für die Erfassung der Anträge zuständig, kümmert sich um die Aktualisierung der Stammdaten (Krankenhaus- und Abteilungsdaten) und ist letztendlich für alle Tätigkeiten verantwortlich, die die Zuteilung selbst betreffen.
>
> Der "Supervisor" hingegen sieht sich die Stamm- und Bewegungsdaten nur an und führt diverse Auswertungen durch. Er kann sich statistische Übersichten und Zuteilungslisten erstellen lassen, Adressetiketten drucken oder sich abrechnungsrelevante Daten ausgeben lassen.

Der Abstraktionsgrad der dargestellten Kooperationen und Iterationen variiert mit ihrer Bedeutung für die zukünftige Arbeitssituation der Anwender des Systems. Eine weiterführende Konkretisierung der Kooperation Datenpflege findet an dieser Stelle z.B. noch nicht statt,

da die konkrete "Realisierung" dieser Kooperation nur von vernachlässigbarer Bedeutung für eine Beurteilung der neuen Arbeitssituation zu sein scheint. Genauso verhält es sich mit den Aktivitäten, die zur Synchronisierung des Webservers mit dem Apvel-System notwendig sind. Deren genaue Ausgestaltung ist für die zukünftige Arbeitspraxis der menschlichen Akteure weitgehend irrelevant. Der Kooperationsverlauf für das Zuteilungsverfahren jedoch wird im folgenden Schritt noch einmal aufgegriffen, da die Zuteilung die wesentliche Aufgabe des Softwaresystems darstellt und die Arbeitssituation auch insofern stark beeinflusst, als die Reihenfolge der verschiedenen zuteilungsbezogenen Kooperationen "Hauptverteilung", "Zwischenverteilung", "Nachrückverfahren", "Handzuteilung" durch die sogenannte "Verteilordnung im Praktischen Jahr" gesetzlich stark reglementiert ist.

Konkrete Interaktionen statt abstrakter Kooperationen werden im Kontextdiagramm immer dann verwendet, wenn es sich bei der gemeinsamen Aktivität um definitive Kommunikationsakte handelt, in denen sowohl Sender als auch Empfänger feststehen, und die in keine weiteren akteurspezifischen Tätigkeiten verfeinert werden sollen. Durch die Verwendung von Auftrag-Stereotypen können sie darüber hinaus bereits an dieser Stelle dazu verwendet werden, um akteurspezifische Zuständigkeitsbereiche anzudeuten. Das Apvel-System erhält im Diagramm z.B. den Auftrag "Zuteilungsbescheide erstellen". Damit ist bereits auch ein Teil seines Aktivitätsprofils festgelegt. Die Form der Interaktion selbst bleibt hingegen unspezifiziert und wird erst später in der Softwarespezifikation (s. Kap. 5.8) festgelegt, in der die konkrete Benutzerschnittstelle entworfen wird.

Die Kooperation "Statistikdaten" hingegen legt noch in keiner Weise fest, wie ein Supervisor mit dem System interagieren kann, um statistische Auswertungen durchzuführen. Eine Konkretisierung hin zu Interaktionsmöglichkeiten, Einzelfunktionen des Systems und notwendigen Tätigkeiten des Supervisors bleibt damit späteren Gestaltungsschritten überlassen. Zur Konkretisierung solcher Kooperationen ist es notwendig, die spezifischen Anteile der beteiligten Akteure an der gemeinsamen Aktivität herauszuarbeiten und damit zum Einen die genauen Zuständigkeiten des Systems und seiner Anwender festzulegen und zum Anderen die Interaktionsschnittstelle zwischen System und Anwender zu konkretisieren.

Diese weitergehenden Gestaltungsschritte finden sich in den Abschnitten 5.5.5 bei der Festlegung des Systemprofils und im Abschnitt 5.6.1 bei der Definition der Nutzerprofile. Die Interaktionsschnittstelle hingegen wird im Abschnitt 5.4.2 bei der Modellierung typischer Systemkonversationen thematisiert und bei der folgenden Festlegung des Systemprofils und insbesondere bei der Beschreibung der Nutzungsaktivitäten in den Nutzerprofilen fortlaufend konkretisiert, indem das Kooperationsmodell um neue Interaktionen zwischen System und Nutzern ergänzt oder bereits definierte abstrakte Interaktionen verfeinert werden.

Im konkreten Anwendungsfall tritt Frau L. (eine Mitarbeiterin des PJ-Büros) dem System gegenüber in der Rolle des "Sachbearbeiters" auf. Sie ist für die Organisation und Durchführung der Zuteilung verantwortlich. Die Leiterin des Studiensekretariats hingegen verwendet das System überwiegend in der Rolle des "Supervisor", um sich auf dem aktuellen Stand zu halten und am Ende des Praktischen Jahres die notwendige Abrechnung durchzuführen. Im Vertretungsfall – z.B. bei Krankheit oder Urlaub – kann es jedoch durchaus sein, dass die Leiterin oder aber eine Hilfskraft in die Rolle des "Sachbearbeiters" wechselt, um das Tagesgeschäft des PJ-Büros aufrecht zu erhalten. Während der Systemgestaltung beschränken wir uns in der Modellierung jedoch auf Nutzerrollen. Die Arbeitssituation der realen Akteure, die in der Arbeitspraxis in diesen Nutzerrollen auftreten, wird erst in der nächsten Phase von

KoA_LA thematisiert – der Fiktion.

Zum Vorgehen

Das Zusammenspiel der Akteure wird über Interaktionen und Kooperationen modelliert, welche die verschiedenen Partner in Beziehung setzen. Um die Übersichtlichkeit zur gewährleisten werden im Diagramm nur die wichtigsten Kooperationen und Interaktionsmöglichkeiten aufgeführt bzw. in abstrahierter Form dargestellt. Um mehrere konkrete Einzelinteraktionen zusammenzufassen, können abstrakte Kooperationen oder Interaktionen definiert werden, welche diese Interaktionen subsumieren. Alternativ kann das Kooperationsmodell des IS natürlich auch in mehreren Kontextdiagrammen dargestellt werden, die z.B. spezifisch für jede Anwenderkategorie oder jedes technische Nachbarsystem angelegt werden. Eine verfeinerte Darstellung der Kooperationsbeziehungen kann in textueller Form erfolgen.

Kooperationen bzw. Interaktionen werden sukzessive immer dann auf eine elementarere Interaktionsebene verfeinert, wenn die entsprechenden Konversationsmuster (siehe nächste Abschnitte) bzw. die einzelnen Interaktionsbeziehungen für die Gestaltung relevant sind. Das ist z.B. der Fall, wenn die einzelnen Interaktionen unterschiedliche Anwender-Rollen betreffen oder wenn die Arbeitssituation wesentlich durch den Ablauf der Kooperation bestimmt wird. Interaktionen werden im Allgemeinen erst im Schritt des Systementwurfs (siehe Abschnitt 5.8, Softwarespezifikation) bei der Gestaltung der Benutzeroberfläche vollständig in konkrete Kommunikationsabläufe transformiert bzw. verfeinert.

Das System selbst kann potentiell auch durch mehrere Akteure beschrieben werden, die jeweils ganz bestimmte Einzelkomponenten des Systems modellieren. Die Modellierung des Systemkontexts dient in erster Linie der vorläufigen Abstimmung der Rolle des neuen IS. Die Kooperationsbeziehung zwischen Akteuren des Systems (zwischen Systemkomponenten) und zwischen System und Anwendern bzw. Nachbarsystemen müssen hierzu nicht vollständig angegeben oder verfeinert werden. Eine Skizze der wichtigsten Abhängigkeitsbeziehungen reicht aus, um Struktur und Charakter der anvisierten Systemunterstützung hinreichend festzulegen. Die weitere Konkretisierung dieser Kooperationen findet sukzessive in den nächsten Abschnitten der Systemgestaltung statt. Erst dort wird über eine genauere Aufgabenzuordnung entschieden, d.h. welcher Akteur welchen Anteil an Einzelaktivitäten der Kooperation übernimmt.

In der Modellierung sind folgende Leitfragen zu klären

Aus welchen Komponenten besteht das Softwaresystem?

> Ist das Softwaresystem sehr groß oder komplex, so kann es Sinn machen, einzelne abgrenzbare Teile des Systems durch eigene technische Akteure zu modellieren.
>
> Im Beispiel wird das Kernsystem des Akteurs “Apvel” dargestellt, während seine Webkomponente als Akteur “Webserver” modelliert wird.

Welchen Rollen können die Anwender des neuen Systems zugeordnet werden?

> Können die Anwender des Systems in verschiedene Kategorien unterschieden werden?

Im Beispiel werden die beiden Kategorien "Sachbearbeiter" und "Supervisor" unterschieden, die ein unterschiedliches Interessen- oder Zuständigkeitsprofil von potentiellen Nutzern des Softwaresystems modellieren. Der "Sachbearbeiter" ist für die Pflege der Daten zuständig während der "Supervisor" hauptsächlich daran interessiert ist, sich über den Status der Verteilung informieren zu können und Auswertungen vorzunehmen.

Mit welchen technischen Nachbarsystemen interagiert das Softwaresystem?

Im Beispiel sind zwei technische Akteure zu identifizieren. Zum Einen ein Drucker, der die notwendigen Adressetiketten erzeugen soll, zum Anderen das Programm Microsoft Word, das in der Büroumgebung zum Generieren und Nachbearbeiten der Serienbriefe Verwendung finden soll.

Wie interagiert das System mit seiner Umgebung?

Welche Interaktionen seiner Nachbarn betreffen das Softwaresystem? Welche Interaktionen führt das Softwaresystem mit seinen Nachbarn durch?

Im Beispiel interagiert Apvel mit Word zur Generierung von Serienbriefen für die Zuteilungsbescheide und mit dem Drucker zur Erzeugung der Adressetiketten. Der Student interagiert mit dem Webserver zur Anmeldung und zur Einsicht in Zuteilungsinformationen. Der Sachbearbeiter gibt dem System einen Auftrag zur Generierung der Zuteilungsbescheide.

An welchen Kooperationen nehmen die Akteure teil?

Viele weitere Interaktionen werden im Beispiel aus Gründen der Übersichtlichkeit in Form allgemeiner Kooperationen modelliert, die einzelne Interaktionen und lokale Aktivitäten zusammenfassen. Die Kooperation Hauptverteilung z.B. subsumiert neben dem eigentlichen Auftrag an das System zur Durchführung der Hauptverteilung auch Interaktionen des Systems zur Rückmeldung des Ergebnisses der Zuteilung, lokale Aktivitäten des Sachbearbeiters zur Anpassung von Kapazitäten etc.

Alle Kooperationen werden nur allgemein beschrieben, nicht aber in ihrem Ablauf oder ihrer Struktur festgelegt. Bei der Handzuteilung wird z.B. völlig offen gelassen, wie diese "realisiert" wird, d.h welche Aufgaben bzw. Möglichkeiten die beteiligten Akteure besitzen und welche Form ihre Interaktionsbeziehungen annehmen werden. Optional können auch Kooperationen und Interaktionen der Nachbar-Akteure untereinander mit in die Modellierung aufgenommen werden.

Im Beispiel kooperieren Webserver und das Kernsystem zum Zwecke der Datensynchronisierung, es besteht eine Interaktionsbeziehung zwischen dem Akteur "Word" und dem "Drucker" etc.

Welchen Randbedingungen unterliegen die Kooperationen?

Welche Anforderungen stellen Anwender und System an den jeweils anderen? Welche Rechte und Pflichten besitzen die Kooperationspartner? Welche Erwartungen existieren?

5.5.3. Systemkonversationen

Ziel: Skizzierung typischer Systemnutzungssequenzen

Im nächsten Schritt werden die im vorigen Schritt definierten abstrakten System-Kooperationen nun in Form von Konversationen weiter konkretisiert bzw. dort identifizierte Einzelinteraktionen in einen allgemeineren Kontext gesetzt. Durch die Modellierung von Systemnutzungskonversationen wird die Vision des Softwaresystems bereits in Richtung auf eine reale Fiktion konkretisiert und vervollständigt. Es werden neue Interaktionen zwischen Anwendern und System definiert, die den Aufgabenbereich des System-Akteurs weiter eingrenzen.

Interaktionen mit Systemkomponenten können Zustandsabhängigkeiten unterworfen sein und daher wiederkehrende Muster bilden. Solche Dialogstrukturen können in gleicher Weise wie auch die Interaktionsbeziehungen des Kooperationsmodells der Kontextmodellierung als Konversationen dargestellt werden. Ein ähnliches Konzept findet sich z.B. in den *"conversations"* von [WB95] wieder.

Konversationen beschreiben wiederkehrende Muster im Kommunikationsverhalten zwischen dem System und seinen verschiedenen Anwendern. Sie beschreiben Interaktionssequenzen, die beispielsweise durch den Systementwurf so vorgeschrieben werden (sollen), oder auch Kommunikationen, welche die Systembenutzung besonders gut charakterisieren.

In dieser Form können auch zusammenhängende, zielorientierte und wiederkehrende Benutzungsszenarien ("operating procedures") beschrieben werden, die zur Erzielung bestimmter Nutzwerte durchgeführt werden und die sich in klassischen objektorientierten Analysemethoden als "essential use cases" wiederfinden[9] (siehe z.B. [CL99]). Systemkonversationen können dazu benutzt werden, auf kommunikationsorientierter Ebene eine abstrakte Benutzerschnittstelle des Systems zu skizzieren (für eine weiterführende Diskussion siehe auch Kap. 6.3).

Zur Modellierung

Interaktionssequenzen werden entweder auf Anwenderseite als abstrakte Interaktionen modelliert und ausführlich textuell beschrieben (siehe dazu auch Abschnitt 5.6.1) oder sie werden als besondere Form eines Aktivitätsmusters – als Konversation – modelliert und grafisch in Form von Konversationsdiagrammen dargestellt bzw. textuell beschrieben.

Zur Modellierung der Systemnutzer werden Akteure verwendet, die sowohl reale Anwender als auch Nutzerkategorien bzw. Anwenderrollen darstellen können. Die Interaktionen mit dem System stellen größtenteils Aufträge an das System dar, d.h Tätigkeiten, die dem System befohlen werden können, bzw. Funktionen, die das System autonom oder nichtautonom ausführt

[9] Das Konzept des Use-Case wird in den etablierten Ansätzen allerdings oft ausschließlich zur Spezifikation der Systemschnittstelle benutzt – oft sogar sehr feingranular bis hinunter zur Verwendung spezifischer Oberflächen-Widgets (diese Art von Use-Cases nennt Constantine "concrete use cases"). KOA$_{\text{L}}$A hingegen verwendet das Konzept der Konversation hier auf einer deskriptiven kommunikationsorientierten Abstraktionsebene. Nicht die Form ihrer Durchführung bzw. Umsetzung, sondern ihr Inhalt bzw. Ziel steht im Vordergrund. Desweiteren ist eine Konversation auch nicht auf den direkten Dialog eines Systembenutzers mit dem System beschränkt, sondern kann auch akteurübergreifende Kommunikationssequenzen umfassen (für eine weiterführende Diskussion siehe Kap. 6.3).

bzw. ausführen kann. In der Modellierung solcher Einzelaufträge kann also bereits eine Vervollständigung des in der Vision skizzierten Systemprofils stattfinden. Die Interaktionen des Systems mit seinen Anwendern modellieren hingegen größtenteils Reaktionen des Systems auf vom Anwender initiierte Interaktionen hin.

Bei der Modellierung der einzelnen Interaktionen steht immer ihre pragmatische Funktion im Vordergrund. Die genaue Art und Weise der Interaktion selbst, d.h. ihre genaue Ausprägung, ist dabei erst einmal sekundär. Im Fall der Interaktion mit einem Werkzeug wie z.B. einem IS wird im Normalfall nicht weiter spezifiziert, *wie* diese Interaktion genau realisiert wird. Es wird erst einmal alleine das Ziel der Werkzeugnutzung dargestellt. Eine Detailierung der Tätigkeiten, die vom Anwender bei der Nutzung des Werzeugs zu leisten sind, ist erst Aufgabe der späteren Softwarespezifikation. Erst hier wird die Benutzeroberfläche detailliert definiert.

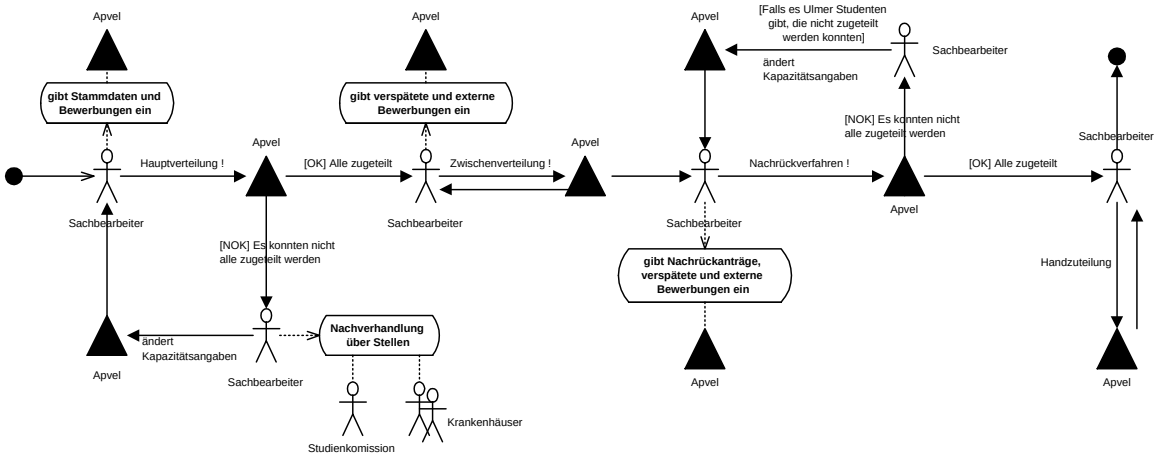


Abbildung 5.15.: Eine charakteristische Zuteilungskonversation für Apvel als Konversationsdiagramm

In Abb. 5.15 wird als Beispiel ein charakteristisches Interaktionsmuster für die Systemnutzung während der Durchführung einer Zuteilung als Konversationsdiagramm vorgestellt. Das Diagramm beschreibt im groben den kompletten Arbeitsprozess der Zuteilung, wie er sich im Zusammenspiel zwischen Sachbearbeiter und Softwaresystem darstellt.

Die dargestellten Aktivitäten, die mit einer gestrichelten Linie mit den Akteuren verbunden sind, dienen jeweils nur zur ergänzenden Kontextbeschreibung, mit welchen Aktivitäten die Sachbearbeiter im jeweiligen Zustand der Konversation beschäftigt sind.

Betrachtet man die Einzelinteraktionen und deren Abfolge, wird klar, wie das System zur Durchführung einer Zuteilung benutzt wird bzw. welchen Einschränkungen es dabei unterliegt:

> Ein Sachbearbeiter führt nach Eingabe der Bewerbungen die Hauptverteilung in der Regel nur einmal durch, d.h. er wiederholt sie nur so oft, bis sie einmal erfolgreich durchgeführt werden konnte.
>
> Nach der Hauptverteilung führt er beliebig oft eine Zwischenverteilung durch, und zwar jeweils nachdem wieder eine bestimmte Anzahl externer oder verspäteter Bewerbungen eingetroffen ist.
>
> Sobald alle Nachrückanträge eingegeben wurden und die Einspruchfrist abgelaufen ist, führt er das Nachrückverfahren durch und bearbeitet das Ergebnis evtl. noch per Hand.

Nach der Zwischenverteilung darf keine Hauptverteilung stattfinden, genauso wie auch nach dem Nachrückverfahren keine Zwischenverteilung mehr initiiert werden kann.

Zum Vorgehen

Systemkonversationen dienen der groben Skizzierung der Systemnutzung aus kommunikationsorientierter Sicht. Sie stellen den Verlauf standardisierter Kooperationen bzw. Kommunikationen als musterhafte Interaktionssequenz dar. In der Auswahl und bei der Darstellung von Systemkonversationen konzentriert man sich auf solche Ausschnitte der Systemnutzung, die den Umgang mit dem System prägnant darstellen und die Rolle des Systems besonders gut charakterisieren.

Der Abstraktionsgrad in der Modellierung des Nutzungssystems (bezogen auf die Konkretisierung der Benutzerschnittstelle) ist natürlich im Prinzip beliebig verfeinerbar. In der Gestaltungsphase beschränkt sich KoA$_{\text{L}}$A aber auf die konzeptionelle Kommunikationsebene und führt keine weitere Gestaltung in Richtung der konkreten Benutzeroberfläche durch; d.h. grafisches Layout, Festlegung von Bedienelementen etc. bleibt außen vor. Optionale Verfeinerungen ergeben sich allerdings durch Aufteilung des IS in verschiedene System-Komponenten oder auch Werkzeuge und deren Konkretisierung über eine Definition ihres Interaktionsverhaltens.

Will man den Darstellungsbereich im Beispiel weiter verfeinern, um z.B. das System in Hinblick auf die Dateneingabe genauer festzulegen bzw. auszugestalten, könnte man auch die Kooperation der Antragseingabe weiter konkretisieren, indem man sie nun in Form einer Konversation modelliert. Konversationen können hier nützlich sein, um verschiedene Ideen für den logischen Ablauf der Systembedienung zu dokumentieren. Beispielsweise könnte man hier festlegen, das der Sachbearbeiter nicht stur alle Personendaten auf einmal eingeben muss, sondern hierbei vom System unterstützt wird, indem es auf Eingabe der Matrikelnummer hin mit dem kompletten Datensatz antwortet, sofern der Student bereits einen früheren Antrag gestellt hat oder das System diese Daten per Zugriff auf das Studentenverwaltungssystem ermitteln konnte. Desgleichen könnte hiermit offengelegt werden, wie der Sachbearbeiter reagieren kann bzw. welche Unterstützung das System in den Fällen bietet, in denen ein Student bereits Teile eines PJ abgeleistet hat oder der Antrag inkonsistente Präferenzangaben enthält.

Alle Konversationen zusammengenommen definieren zum Einen die abstrakte Benutzerschnittstelle des anvisierten neuen IS, die zur Fremdsteuerung des Softwaresystems eingesetzt werden kann und zum Anderen das selbstgesteuerte kommunikative Verhalten des Softwaresystems in Bezug auf seine Außenwelt (anderen technische oder menschliche Akteure).

5.5.4. Systembegriffswelt

Ziel: Definition der Systemsprache

Im nächsten Abschnitt wird nun das spezialisierte Wissen und damit die Sprache des IS näher festgelegt.

Das System selbst wird nur einen Bruchteil der sprachlichen Begriffe des Anwendungsbereichs kennen bzw. realisieren. Desweiteren wird es auch neue Begrifflichkeiten bzw. Konzeptionen verwenden, die ein Anwender zur erfolgreichen Benutzung bzw. Interaktion mit dem System

verstehen muss[10]. Die Systembegriffswelt definiert das interne Wissen des Softwaresystems in Form von Begriffen, die sich eng an die in der Anwendungsumgebung gesprochene Sprache anlehnen. Dabei werden die einzelnen Konzepte in ihrer Bedeutung definiert, nicht aber in ihrer (Daten-)Struktur! Das ist erst Aufgabe der späteren Softwarespezifikationsphase (siehe Abschnitt 5.8).

Zur Modellierung

Die Sprache bzw. das Wissen des Systems wird in KoA$_L$A in Form eines Wissensmodells definiert (siehe Kap. 4.2), das grafisch als Konzeptdiagramm verdeutlicht und alternativ oder ergänzend textuell spezifiziert werden kann. Es definiert den Wortschatz des Systems, der sowohl in den Interaktionen des Systems mit seinen Nachbar-Akteuren verwendet wird als auch Gegenstandsbereich für die internen Aktivitäten des Systems ist.

Das System wird immer nur einen Ausschnitt der Alltagssprache der Anwendungsumgebung realisieren, so dass sowohl Form als auch Inhalt möglicher Systeminteraktionen im Allgemeinen eng reglementiert sind. Das Wissensmodell des Systems ist in der Regel ein Ausschnitt aus dem Wissensmodell der Anwendungsumgebung, allerdings optional vereinfacht und erweitert um systemspezifische Konzepte.

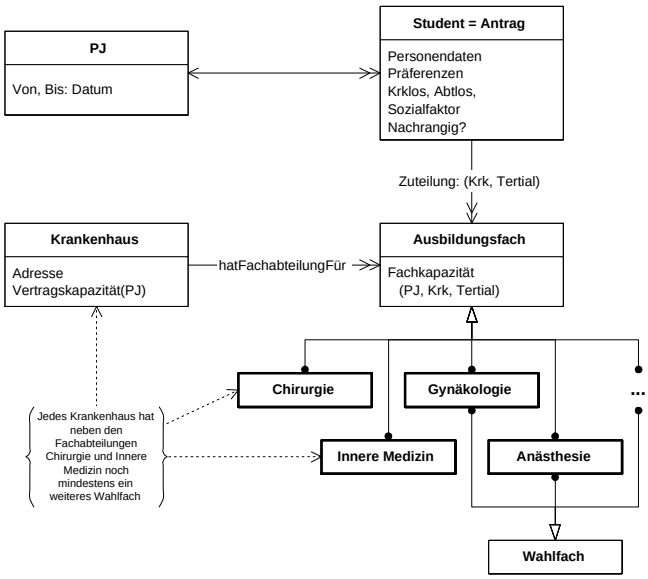


Abbildung 5.16.: Ein Konzeptdiagramm für die Begriffswelt des Softwaresystems

In Abb. 5.16 wird als Beispiel ein vereinfachtes Konzeptdiagramm für das Wissensmodell des Apvel-Systems dargestellt, indem nun nicht mehr zwischen dem Studenten und seinen Anträgen unterschieden wird – beispielsweise um die Benutzeroberfläche zu vereinfachen bzw. die Projektkosten zu minimieren:[11]

[10] [BR97] redet hier von Kontroller-Klassen, [Pae00] von "Hilfsdaten" der Benutzeroberfläche

[11] Diese Vereinfachung wird hier aus "didaktischen" Gründen vorgenommen. Einmal um im Fallbeispiel das Gestaltungsproblem für den Kunden zu vereinfachen, und zum Anderen, um dem Leser die unterschiedlichen

Das System kennt keine Studenten an für sich, sondern pro PJ nur mehr eine Reihe von Anträgen, die konzeptionell mit dem Studenten gleich gesetzt werden. Dies hat natürlich eine Reihe von Implikationen für die Systemfunktionalität, die mit den Anwendern abgestimmt werden müssen. Beispielsweise kann nun nicht mehr überprüft werden, ob ein Student evtl. schon an einem anderen Praktischen Jahr teilgenommen hat und ob seine Fächerfolge konsistent ist. Dieser Nachteil kann z.B. teilweise dadurch abgefangen werden, dass beim Neuanlegen eines Antrags anhand der Matrikelnummer überprüft wird, ob diese bereits in einem früheren Jahr verwendet wurde und der Anwender davon in Kenntnis gesetzt wird, sodass dieser nun von Hand den Sachverhalt überprüfen kann.

Entscheidungen in der Wissensmodellierung können also durchaus auch rückwirkenden Einfluss auf die Gestaltung der Systemkooperation besitzen. Sinnvollerweise ist im angesprochenen Fall eine Konkretisierung der Kooperation "Anlegen eines Antrags" durchzuführen, entweder rein textuell oder auch umfassender durch ein eigenes Konversationsdiagramm.

Desweiteren wird im Beispiel nun auch nicht mehr zwischen Abteilungen und Ausbildungsfächern unterschieden. Das System kennt pro Krankenhaus nun eine Reihe virtueller Fachabteilungen, die nichts mehr mit konkreten Abteilungen der Krankenhäuser zu tun haben. Beispielsweise hat das Klinikum Ulm in Wirklichkeit fünf verschiedene reale Chirurgie-Abteilungen, in denen Studenten jeweils verschiedene Abschnitte ihrer Fachausbildung ableisten. Dem System ist jedoch nur bekannt, dass das Klinikum in diesem Fach ausbildet und wieviel Ausbildungsplätze in dieser Fachrichtung zur Verfügung stehen. Ein solches Konzept einer "Fachabteilung" ist ein Beispiel für eine neue Begrifflichkeit, die erst mit dem neuen Softwaresystem Eingang in den Sprachgebrauch der Anwendungsumgebung findet.

Als ein Beispiel für die Konkretisierung von Konzepten wird dem Studenten als neuer Aspekt seine "Nachrangigkeit" zugeordnet. Dieses Merkmal bestimmt, ob der Student in der automatischen Verteilung erst nach der Zuteilung aller anderen "normalen" Studenten berücksichtigt werden soll. Dies wird z.B. benötigt, um Studenten zu "bestrafen" die sich bereits in einem vorhergehenden Praktischen Jahr beworben, aber ihre Stelle unentschuldigt nie angetreten haben.

Bei den Präferenzen wird in der Projektdatenbank außerdem textuell konkretisiert, ob zu einem Wunsch-Krankenhaus beliebige Fachabteilungen gewünscht werden können oder ob hier nur Fachabteilungen des Wunsch-Krankenhauses Eingang finden können.

Zum Vorgehen

Für die Systemgestaltung relevant sind nicht nur die eigentlichen Begrifflichkeiten des Systems, sondern auch alle konzeptionellen Elemente, die die Systemschnittstelle thematisiert und verwendet. Das sind beispielsweise auch sonstige vom System verwendete oder erzeugte Artefakte wie z.B erstellte Protokolle, Dateien, Belegungspläne, Aushänge etc.

Zusammengenommen werden in Form der Systembegriffswelt alle begrifflichen Konzepte definiert, die für die vollständige Definition des Leistungsprofils des neuen Softwaresystems essentiell sind. Es werden die fachsprachlichen Einheiten definiert, mittels derer sowohl Leistungen als auch Interaktionsverhalten des neuen IS-Akteurs aus Sicht der Anwender verständlich und eindeutig definiert werden können.

In der fertigen Benutzerschnittstelle finden sich später allerdings in der Regel auch viele weitere Begriffe der Anwendungsumgebung wieder, die nicht direkt zur Begriffswelt des Systems gehören, jedoch in der Kontextmodellierung bereits Eingang in das globale Begriffswelt-Modell gefunden haben. Beispielsweise werden die Begriffe "Antrag" oder "Pflichtfach" in der Benutzerschnittstelle wohl durchaus verwendet werden, um dem Benutzer die Bedeutung seiner

Modellierungsebenen (Handlungs-, Nutzungs, Softwaresystem) deutlich vor Augen zu führen. Die Unterscheidung von Student und Antrag wird später in der Softwarespezifikation wieder eingeführt werden, um die Spezifikation aus formalen Gesichtspunkten konsistenter zu machen bzw. konzeptionell zu vereinfachen (siehe Kap. 5.8).

Eingaben (z.B. bei der Präferenzeingabe) klar zu machen, auch wenn das System selbst diese Begrifflichkeiten nicht kennt oder es ihre Bedeutung nicht klar unterscheidet.

5.5.5. Systemprofil

Ziel: Definition der Softwareanforderungen

In diesem Schritt wird nun das Leistungsprofil des neuen Systems konkretisiert. Das neue Softwaresystem wird aus "menschlicher" Interaktionssicht als intelligenter Akteur modelliert, der mit den Begrifflichkeiten der globalen Begriffswelt umzugehen weiß. Als neuer Kommunikationspartner ist er Bestandteil der realen Welt und verändert die etablierten Kommunikationsnetze.

Der System-Akteur definiert alle Anforderungen an das neue Softwaresystem aus Anwendersicht, indem seine sichtbaren Effekte auf die reale (Um-)Welt deutlich gemacht werden (siehe [GGJZ00] zur Definition von Anforderungen). Diese Effekte sind bei betrieblichen Informationssystemen vor allem aus den Kommunikations- und Kooperationsmustern ersichtlich, an denen das Softwaresystem beteiligt ist.

Das Systemprofil definiert ein Lastenheft für die weitergehende Softwareentwicklung. Sobald die Gestaltungsphase abgeschlossen ist und als Ergebnis von KoA$_{L}$A ein validiertes Systemprofil vorliegt, kann ein konkretes Softwaresystem entworfen und beschrieben werden, das diesem Anforderungsprofil entspricht. Erst dort wird also eine konkrete Instanz thematisiert, die das beschriebene Stellenprofil ausfüllen kann. In der SW-Spezifikation werden die Dialoge zu grafischen Benutzerschnittstellen konkretisiert und die Aktivitäten zu Operationen verfeinert und formalisiert, die auf Grundlage der im Datenmodell definierten Datenstrukturen arbeiten. Erst hier werden also die in den definierten Interaktions- und Kommunikationsbeziehungen verwendeten "internen" und "externen" Repräsentationsformen bzw. Kommunikationsmedien relevant! Diese Spezifikation der Systeminstanz dient dann als Pflichtenheft für die weitere Realisierung des IS.

Das Systemprofil von KoA$_{L}$A definiert die Rolle des Informationssystems, nicht dessen konkrete Ausprägung. Dieses Profil wird dabei in einer auf die Sprache und Vorstellungswelt der Anwender angepassten Darstellungsweise dokumentiert. Das IS wird hierzu in anthropomorpher Weise ebenso wie alle anderen menschlichen Anwender und Betroffene als Akteur mit eigener "Stellenbeschreibung" modelliert, mit dem normal kommuniziert werden kann und der auch selbständig kommunikativ tätig werden kann.

Ergänzend zum Systemprofil werden im ersten Schritt der darauf folgenden Fiktionsphase dann auch die einzelnen Nutzerprofile konkretisiert. Die Aktivitäten der Anwender werden dann um Systembenutzungsaktivitäten verfeinert bzw. um Konversationen mit dem Softwaresystem ergänzt.

Zur Modellierung

Das Systemprofil wird sowohl grafisch als auch textuell dokumentiert.

In der Visionsphase von KoA$_L$A wurde bereits ein vorläufiges Tätigkeitsprofil des neuen Softwaresystems skizziert. Dieses Diagramm kann als Ausgangspunkt der Modellierung verwendet werden, indem es sukzessive an die Ergebnisse des Gestaltungsprozesses angepasst wird, in dem nun alle konkreten Aufgaben- bzw. Aktivitätsbereiche des Systems dokumentiert und verfeinert werden. Hierfür können auch bewährte Techniken wie z.B. das Cooperative Goal Modelling von [Ale98] eingesetzt werden.

Alternativ bzw. ergänzend hierzu wird das Systemprofil umfassend textuell spezifiziert. Im Folgenden findet sich als Beispiel ein Abschnitt aus der textuellen Definition des Tätigkeitsprofils von Apvel:

```
AKTEUR          Apvel

ANFORDERUNG     Mehrplatzfähigkeit +
Es ist davon auszugehen, dass in der Regel nur ein Sachbearbeiter mit dem
System operativ arbeiten wird. Jedoch sollte ein lesender Zugriff durch einen
anderen Sachbearbeiter oder Supervisor auch parallel stattfinden können.
Im Vertretungsfall sollte ein Sachbearbeiter auch von einem anderen
Arbeitsplatz aus mit dem System arbeiten können.

ANFORDERUNG     Losnummern ++
Die Losnummern für die Krankenhaus- und Abteilungszuteilung sollen vom System
beim Anlegen des Satzes automatisch einmalig bestimmt werden und danach nicht
mehr geändert werden können. Durch diese feste "Verdrahtung" der
Losnummernvergabe soll auch nach Außen dokumentiert werden, dass Anträge vom
PJ-Büro gerecht behandelt werden und von den Sachbearbeitern nicht willkürlich
behandelt werden können.

ANFORDERUNG     Fächerreihenfolge
Die automatische Zuteilung sollte eingeschränkt "steuerbar" sein. Es muss die
Möglichkeit geben, eine Fächerreihenfolge festzulegen, z.B. Wahlfach im ersten
Tertial, Innere im dritten. Dies geschieht vor der Zuteilung und damit zu
einem Zeitpunkt, da weder das zugeteilte Krankenhaus noch das zugeordnete
Wahlfach feststehen.
Eine persönliche Ausnahmesituation, die diese Sonderbehandlung rechtfertigen
kann, ist z.B. bei Alleinerziehende gegeben, die die Chirurgie wegen den
24h-Schichten am Schluss machen wollen,um dort möglichst alle restlichen
Fehltage nehmen zu können

...

AKTIVITÄT       VerwaltetAnträge ++
                Verwaltung aller Antrags- und Wunschdaten der Studenten
=               NeuenAntragAufnehmen
BEDINGTDURCH    Fächerreihenfolge, Losnummern
INTERAKTION     PflegtAntragsdaten <Antrag< Sachbearbeiter
                Diese abstrakte Interaktion des Sachbearbeiters schließt alle
                Aktionen zur Pflege der Antragsdaten ein: Anlegen, Lesen,
                Ändern, Löschen
AUFGABE         NeuenAntragAufnehmen <Antrag< Sachbearbeiter
                Anlegen eines neuen Antrags für ein bestimmtes Praktisches
                Jahr
INTERAKTION     WeistAufWiederholerHin >> Sachbearbeiter
ABLAUF          Legt der Sachbearbeiter einen Antrag für einen Studenten an,
                dessen Matrikelnummer bereits in einem alten Antrag und in
```

```
                einem früheren PJ verwendet wurde, so wird der Benutzer darauf
                hingewiesen. Er erhält somit die Möglichkeit, die Notwendigkeit
                der Nachrangingkeit festzustellen und beim Antrag nachträglich
                zu vermerken.

AKTIVITÄT       AutomatischeZuteilung ++
                Unterstützung der automatischen Zuteilung der Studenten auf ihre
                Praktikumsplätze
=               Hauptverteilung, Zwischenverteilung, Nachverteilung, Handzuteilung
AUFGABE         Hauptverteilung <PJ< Sachbearbeiter ++
                Zuteilung aller in einem PJ rechtzeitig angemeldeten Ulmer
                Studenten auf die akademischen Lehrkrankenhäuser unter
                Berücksichtigung der angegebenen Präferenzen, Sozialfaktoren
                und Losnummern
BEDINGTDURCH    Fächerreihenfolge
ABLAUF          Die Verteilung wird nach folgendem Muster abgewickelt:
                Die Studenten werden erst komplett auf die Lehrkrankenhäuser
                verteilt, bevor eine weitergehende Zuordnung zu ihren
                Wahlfächern stattfindet. Sowohl bei der Krankenhaus-, als auch
                bei der Wahlfachzuordnung werden bis zu drei priorisierte
                Wünsche berücksichtigt. Im Konfliktfall werden sowohl
                der Sozialfaktor als auch eine Losnummer ausgewertet. Ist
                keiner der Wünsche  erfüllbar, kann eine beliebige Zuordnung
                vorgenommen werden.
INTERAKTION     MeldetVerteilungsErgebnis >> Sachbearbeiter
                Das System teilt dem Sachbearbeiter das Ergebnis der Zuteilung
                mit. Es kann vorkommen, das aufgrund eines Kapazitätsmangels
                nicht alle Studenten verteilt werden können. Dann liegt es am
                Anwender, die Kapazitäten neu zu setzen und die
                Hauptverteilung zu wiederholen.

AUFGABE         Zwischenverteilung <PJ< Sachbearbeiter
                Automatische Verteilung aller externen und verspäteten Anträge
                eines PJ, die seit der letzten Hautpverteilung bzw.
                Zwischenverteilung neu eingepflegt wurden und noch nicht
                verteilt sind. Der Sachbearbeiter stößt von Zeit zu Zeit die
                Zwischenverteilung an, sobald wieder genug Anträge
                eingetroffen sind.
INTERAKTION     MeldetVerteilungsErgebnis >> Sachbearbeiter
                Das System teilt das Ergebnis der Verteilung mit. Insbesondere
                wird auf die nicht zuteilbaren und sonstigen Studenten
                hingewiesen, deren Erstwunsch nicht erfüllt werden konnte.

AUFGABE         ZuteilungsbescheideGenerieren
                <Student[*]:Zuteilungsbescheid[*]< Sachbearbeiter +
                Für eine bestimmte Menge von zugeteilten Studenten
                Zuteilungsbescheide in Form von Word-Serienbriefen generieren
INTERAKTION     WähltStudentenAus <Student[*]< Sachbearbeiter
                Der Sachbearbeiter wählt die Menge der Studenten aus, für die
                Zuteilungsbescheide generiert werden sollen.
AUFTRAG         GeneriereSerienbrief >> Word
                Apvel verwendet die Serienbrieffunktion von Word zur
                Generierung der Zuteilungsbescheide
VARIANTE        Zuteilung
                Nach jeder Zuteilung (Haupt, Zwischen oder Nachverteilung)
                werden die neu zugeteilten Studenten automatisch "markiert",
                so dass eine explizite Auswahl der Studenten vor der
```

```
                Generierung unnötig ist.
AUFGABE         StatistikenBerechnen << Supervisor -
                Das System soll die Anträge und das Zuteilungsergebnis
                statistisch auswerten

AUFGABE         WunscherfüllungsStatistik << Supervisor
SPEZIALFALLVON  StatistikenBerechnen
                Apvel berechnet den Grad der Erfüllung der verschiedenen Orts-
                und Wahlfachwünsche über alle Anträge eines bestimmten
                Praktischen Jahres hinweg

AKTIVITÄT       Synchronisierung +
                Apvel synchronisiert sich regelmäßig mit dem Webserver der
                Verwaltung, um zum Einen die eingegangenen Anmeldungen zu
                bearbeiten und zum Anderen die publizierten Aushänge bzgl.der
                Wunsch- oder Zuteilungsstatistiken zu aktualisieren
INTERAKTION     Aktualisierung >> Webserver
                die publizierten Aushänge werden stündlich aktualisiert
INTERAKTION     AnträgeHolen >> Webserver
                alle neu eingegangenen Anträge vom Webserver holen, überprüfen
                und ins System übernehmen
```

Im ersten Abschnitt dieser textuellen Dokumentation des System-Profils finden sich drei Beispiele für Anforderungen, die an den System-Akteur gestellt werden. Diese Anforderungen beschreiben allgemeine Bedingungen, die das System erfüllen muss bzw. die bei der Ausführung von Systemaktivitäten zu beachten sind. Hierbei handelt es sich im Allgemeinen um sogenannte "nichtfunktionale" Anforderungen, die sich nicht auf eine bestimmte zu erbringende Systemleistung beziehen, sondern auf Randbedingungen oder Motive, die bei der Erbringung der eigentlichen Systemfunktionen zu berücksichtigen sind. Die Anforderung "Mehrplatzfähigkeit" spezifiziert hier z.B. eine Bedingung der zukünftigen Einsatzumgebung des Systems, die sehr wohl die weitere softwaretechnische Entwicklung des Systems und seiner Nutzerschnittstelle beeinflussen wird, aber keine eigentliche Funktionalität bzw. erbrachte Leistung des Systems dokumentiert. Im gleichen Sinne beeinflusst die Anforderung "Losnummer" die Aktivität Antragsverwaltung zwar, sie dokumentiert aber keinen eigenständigen Funktionsbereich des Systems. Ebenso dokumentiert auch die "Fächerreihenfolge"-Bedingung das zugrundeliegende Motiv für diese Anforderung bzw. für die betroffenen Aktivitätsbereiche. Die Notwendigkeit zur Verwaltung einer Fächerreihenfolge als Vorgabe für die automatische Zuteilung etwa ergibt sich aus dem Wunsch nach einer Möglichkeit zur Berücksichtigung von Härtefällen wie der von Alleinerziehenden.

Operationalisierbare Anforderungen hingegen werden als Aktivitätsbereiche spezifiziert, die sich in der Arbeitspraxis des Systemakteurs manifestieren. Solche Aktivitäten werden in AKTIVITÄT- oder AUFGABE-Abschnitten dokumentiert. Letztere definieren Aufgabenbereiche, die direkt von anderen Akteuren eingefordert werden können, sie führen daher auch den Auftraggeber mit auf.

Aktivitäten können, wie Bedingungen (Anforderungen) auch, priorisiert werden. Die wichtigsten Aktivitätsbereiche des Apvel-Systems sind z.B. die Antragsverwaltung und die automatische Verteilungsfunktionalität. Weniger wichtig sind die Statistikfunktionen.

Die Beschreibung einer Aktivität gliedert sich in verschiedene Abschnitte, die jeweils durch ein Schlüsselwort eingeleitet werden. Nach dem eindeutigen Namen der Aktivität folgt optional

die Priorität (–,-,0-+ oder ++) und eine deskriptive Definition der Aktivität – z.B. durch Angabe der Aktivitätsziele. Im Abschnitt "BEDINGTDURCH" werden alle Bedingungen bzw. Anforderungen angegeben, die die aktuelle Aktivität betreffen. Die Aktivität Antragsverwaltung beispielsweise wird durch die beiden Bedingungen "Fächerreihenfolge" und "Losnummer" beeinflusst.

Alle ein- und ausgehenden Interaktionen, die während des Ablaufs der Aktivität auftreten können, werden in "INTERAKTION" oder "AUFTRAG"-Abschnitten zusammen mit dem Interaktionspartner, der Richtung der Interaktion und den vermittelten Informationsobjekten angegeben. Zur Aufgabe "ZuteilungsbescheideGenerieren" etwa gehört eine Interaktion mit dem Sachbearbeiter, der die Menge der Studenten festlegt, für die die Bescheide zu generieren sind. Ebenso ist dieser Aktivität auch der Auftrag des Apvel-Systems an Word zur Erstellung eines Serienbriefs zuzuordnen. Die vermittelten Informationsobjekte der Aufgabe bestehen zum Einen in der Menge der Studenten, die vom Initiator der Interaktion zum Empfänger übermittelt werden, und zum Anderen aus der Menge an Zuteilungsbescheiden, die als Ergebnis der Interaktion wieder zurück zum Initiator übertragen werden.

Zu jeder Aktivität können optional im ABLAUF-Abschnitt ein oder mehrere mögliche Varianten ihrer konkreten Durchführung angegeben werden. Diese Varianten beschreiben typische Aktivitätsmuster, die bei der Durchführung der Aktivität zu beobachten sind bzw. definieren Handlungspläne, die vom System zur Erreichung des spezifizierten Aktivitätsziels eingehalten werden sollen. Solche Muster können ergänzend oder alternativ auch grafisch in Form von Aktivitätsdiagrammen dargestellt werden. Im Apvel-Projekt wird beispielsweise die Hauptverteilung durch einen ABLAUF-Abschnitt konkretisiert, der das vorgeschriebene Vorgehen bei der Zuteilung der Studenten erläutert. Desweiteren wird in der Aktivität "NeuenAntragAnlegen" bereits der nähere Interaktionsverlauf erläutert, und zwar *ohne* auf eine weitergehende Ausgestaltung der Benutzerschnittstelle Bezug zu nehmen. Zusätzlich zu einem ABLAUF-Abschnitt, in dem der reguläre Ablauf beschrieben wird, kann eine Aktivität auch VARIANTE-Abschnitte aufweisen, in denen alternative Abläufe der Aktivität beschrieben werden. "ZuteilungsbescheideGenerieren" thematisiert im Beispiel die Variante, dass die Bescheide direkt im Anschluss an eine Verteilung durchgeführt wird. Dann kann die Auswahl der Studenten entfallen, für die ein Bescheid generiert werden soll, weil diese Studenten bereits durch die Verteilung selbst für die Generierung selektiert wurden. Ablaufvarianten können alternativ auch durch spezialisierte Aktivitäten modelliert werden, so dass VARIANTE-Abschnitte auch als Kurzschreibweise solcher Spezialisierungen aufgefasst werden können.

Aktivitätsbereiche können partiell durch andere Aktivitäten verfeinert werden. Dies wird über einen =-Abschnitt spezifiziert, der in einer Liste alle aggregierten Aktivitäten aufführt. Im Apvel-Beispiel ist hier z.B. der Aktivitätsbereich "AutomatischeZuteilung" definiert, der die Aktivitäten "Hauptverteilung", "Zwischenverteilung", "Nachrückverfahren" und "Handzuteilung" beinhaltet. Diese "Unter"-Aktivitäten beschreiben Teilbereiche der umfassenden Aktivität ohne ihre genaue Einordnung in den umfassenderen Ablauf angeben zu müssen. Diese kann allerdings optional in textuell oder grafisch dokumentierten Aktivitätsmustern geschehen, die die Definition der umfassenden Aktivität ergänzen.

Aktivitäten wie Interaktionen können eine beliebige Abstraktionsebene aufweisen. Sie werden nur so weit verfeinert, wie es für eine Beurteilung und Validierung der gewählten Systemgestaltung notwendig zu sein scheint. Die Aktivität Antragsverwaltung etwa wird nur partiell in die Aktivität "AntragAufnehmen" verfeinert, da diese Funktion die Behandlung der Anträge

von Studenten verdeutlicht, die bereits schon einmal einen Antrag gestellt haben.

Zur Verfeinerung kann nicht nur die Aggregation eingesetzt werden, sondern auch die Spezialisierung. Im Beispiel konkretisiert die Aufgabe "WunscherfüllungsStatistik" z.B. einen Spezialfall der Aufgabe "StatistikenBerechnen".

Zum Vorgehen

Das Systemprofil ist als Verfeinerung bzw. Konkretisierung der bisherigen Modelle aufzufassen. Ausgehend von dem vorläufig in der Vision skizzierten Systemprofil und den in den bisherigen Gestaltungsschritten definierten Interaktions- und Kooperationsbeziehungen des Systems mit seinen Anwendern findet nun eine definitive Zuordnung von Verantwortlichkeiten zwischen Nutzer und System statt. Das anvisierte IS wird in Form eines Akteurs spezifiziert, der fest definierte Aktivitäten an den Tag legt, dabei fest definierte Interaktionen ermöglicht bzw. durchführt und fest definierte Randbedingungen berücksichtigt. Alle bisher definierten Kooperationen mit Anwendern des System-Akteurs werden nun so weit verfeinert, bis die systemspezifischen Anteile und das Interaktionsverhalten des System-Akteurs deutlich werden und vollständig definiert werden können. Die Anteile der Anwender werden hingegen erst in der Fiktionsphase konkretisiert (siehe Abschnitt 5.6.1).

Alle autonomen oder durch Interaktionen veranlassten Tätigkeiten des Systems werden als Aktivitäten des System-Akteurs modelliert. Bei Informationssystemen haben viele dieser Aktivitäten die "Funktion" einer Informationsverarbeitung. Beschreibungen der Verarbeitung werden sich daher auf die Konzepte der bereits definierten System-Begriffswelt beziehen. Bei der Beschreibung dieser Funktionalitäten besteht der Vorstellungsbereich bzw. die Begriffswelt damit weiterhin in den Konzepten der Anwendungsumgebung. Die Funktion wird in Begriffen einer praktischen Arbeitstätigkeit beschrieben, nicht auf der Basis von technischen softwaresystemspezifischen Interna.

Eine Konkretisierung der Systemaktivitäten erlaubt oft auch einen Rückschluss auf das notwendige Systemwissen. Um z.B. die Synchronisierung von Apvel mit dem Webserver näher zu konkretisieren, müsste man die Art der ausgetauschten Informationsobjekte detaillieren und vor allem auch die im Web publizierten Dokumente konkretisieren. Auch dies ist ein Beispiel dafür, dass die Gestaltungsschritte in der Regel niemals streng sequentiell durchgeführt werden sondern oft iterativ und überlappend.

Zur gemeinsamen Definition der Systemaufgaben ist für die ersten Schritte die grafische Dokumentationsform vorzuziehen. Hier wird z.B. das entsprechende Profildiagramm der Visions-Phase diskutiert und erweitert. Zur Dokumentation des endgültigen Tätigkeitsprofils des Systems ist hingegen eher die Textform vorzuziehen, die alle geforderten Punkte in der Sprache der Anwender der Reihe nach auflistet und mehr Platz zur konkreten freitextlichen Definition der einzelnen Anforderungen bietet.

Aus Gründen der Übersichtlichkeit kann es sinnvoll sein, den Systemakteur in unterschiedliche Komponenten zu zerlegen, die einen kohärenten Aufgabenbereich des Systems modellieren. Der Systemakteur kann hierzu in mehrere aggregierte Sub-Akteure zerlegt werden, deren Beschreibungen dann die Aufgaben- und Anforderungsdefinitionen des Systems geeignet partitionieren und so deren Zusammenhang und Abhängigkeiten deutlicher machen.

Im Folgenden nun eine Auswahl an Leitfragen, an denen sich die Erstellung des Systemprofils orientieren kann:

Welche Leistungen steuert das System zu den Kooperationen bei?

Welchen eigenen Anteil hat das System an den identifizierten Kooperationen mit den Anwendern?

Solche Leistungen werden entweder als allgemeine und abstrakte Aufgabenbereiche beschrieben wie z.B. "VerwaltetAnträge" oder zu einzelnen Aufgaben konkretisiert, die von außen, d.h. von Anwendern eingefordert werden können. "ZuteilungsbescheideGenerieren" ist z.B. eine solche Aufgabe, die vom Sachbearbeiter in Form eines Auftrags angestoßen werden kann.

Im Beispiel: Das System verwaltet Anträge, berechnet Statistiken und synchronisiert sich selbsttätig mit dem Webserver.

Welche Interaktionen begleiten die Systemaktivitäten?

Es werden alle Interaktionen beschrieben, die während der Aktivität zwischen der Umgebung und dem System selbst stattfinden. Das umschließt alle Interaktionen, die von Anwendern des Systems oder vom System selber initiiert werden. Bei den Interaktionen unterscheidet man zwischen allgemeinen kommunikativen Handlungen und den speziellen Kommunikationen zur Übermittlung eines Auftrags.

Im Beispiel: Die SachbearbeiterIn pflegt einen Antrag ein, das System beauftragt Word mit der Generierung eines Serienbriefs etc.

Wie lassen sich diese Einzelleistungen konkretisieren?

Der Abstraktionsgrad der dargestellten Aktivitätsbereiche lässt sich durch Verfeinerung (Aggregation) oder Spezialisierung schrittweise konkretisieren. Konkretisierungsschritte werden immer dann unternommen, wenn sie für das Verständnis und die Ausgestaltung des Interaktionsverhaltens des Softwaresystems notwendig erscheinen.

Im Beispiel: Die Wunscherfüllungs-Statistik ist eine Beispielform der statistischen Aufbereitung, die Nachverteilung ist eine Teilaufgabe der automatischen Zuteilung etc.

Wie wichtig sind die Einzelleistungen bzw. Bedingungen für die Projektziele?

Jeder Aktivität wird eine Priorität (++, +, 0, -) zugewiesen, mit der diese Aktivität im weiteren Gestaltungs- und Validierungprozess berücksichtigt werden soll. Ebenso werden die verschiedenen Bedingungen gewichtet. Konflikte können oft gelöst bzw. entschärft werden, indem unwichtigere Ziele/Bedingungen entsprechend abgeschwächt oder auch korrigiert werden, oder indem Aktivitäten spezialisiert dargestellt oder verallgemeinert werden.

Im Beispiel: Die automatische Zuteilung ist die wichtigste Aufgabe des Systems, die Statistik-Funktion eher unwichtig etc.

Welche Tätigkeitsmuster können identifiziert werden?

Können einzelne Aktivitäten des Systems zu Ablaufbeschreibungen bzw. Aktivitätsmustern konkretisiert werden?

Im Apvel-Beispiel wird hier z.B. das Vorgehen bei der Zuteilung textuell konkretisiert. Möglich ist natürlich auch eine grafische Beschreibung von Aktivitätsmustern durch Aktivitätsdiagramme. Diese Ablaufbeschreibungen stellen Beispiele einer möglichen Operationalisierung der deskriptiv definierten Aktivitäten dar.

Im Beispiel: Das Anlegen eines Antrags und die Hauptverteilung wird textuell genau beschrieben.

Welche Randbedingungen muss das System bei seinen Aktivitäten beachten?

Diese nichtfunktionalen Anforderungen werden als Bedingungen dokumentiert. Alle Aktivitäten, die von diesen Bedingungen direkt betroffen werden, referenzieren diese (im Beispiel in Form von BEDINGTDURCH-Abschnitten).

Im Beispiel: Die Mehrplatzfähigkeit, die Behandlung der Losnummern etc.

Welche Motive beeinflussen die Aktivitäten?

Was ist der motivationale Hintergrund für das Leistungsprofil des Systems? Welche übergeordneten Ziele sollen mit den Aktivitäten erzielt werden?

Bedingungen können dazu benutzt werden, die Motivation hinter den Systemaktivitäten deutlich zu machen oder auch um z.B. das "Design Rational" der Gestaltung explizit zu machen.

Im Beispiel: Die Fächerreihenfolge.

5.5.6. Zusammenfassung

Ziel der Systemgestaltung ist eine Beschreibung des zu realisierenden Softwaresystems aus Anwendersicht. In enger Zusammenarbeit mit den Interessengruppen wird das Leistungsprofil des Systems schrittweise entworfen, kontextreich beschrieben und konkretisiert.

Zusammengenommen mit dem in der nächsten Phase zu erstellenden Nutzungskonzept, das die Systemnutzung beispielhaft aus Anwendersicht beschreibt, konkretisiert das Systemprofil im Wesentlichen die Rolle des Systems, wie sie bereits im Abschnitt 5.5.2 in Form der Systemkooperationen auf abstrakter Ebene thematisiert ist. Die dort skizzierten Kooperationsbereiche werden im Systemprofil (siehe Kap. 5.5.5) zu lokalen Aktivitäten des Softwaresystems verfeinert, während die Interaktionen der Softwaresystem- und Anwender-Akteure in Form der Systemkonversationen (siehe Kap. 5.5.3) konkretisiert werden.

Ziel der Gestaltungsphase ist eine gemeinsame Beschlussfassung über die Ausgestaltung des System-Akteurs (Tätigkeitsprofil, Interaktionsverhalten, Begriffswelt etc.). Eine Validierung erfolgt erst im Anschluss an die nächsten Phase – der Fiktion –, in der die Auswirkungen der gewählten Systemgestalt auf die Anwendungsumgebung – die konkrete Arbeitspraxis aller Betroffenen – thematisiert, dargestellt und diskutiert wird.

Zum Vorgehen

In der Gestaltungsphase selbst werden unweigerlich diverse unterschiedliche Ideen zur Ausgestaltung des System-Akteurs aufgeworfen und diskutiert. Ist eine Konsensfindung unmöglich, so kann es notwendig sein, die verschiedenen Gestaltungsvarianten explizit durchzumodellieren und auch durch eine eigene Fiktionsphase (s.u.) getrennt in ihren Auswirkungen zu untersuchen. Die verschiedenen Varianten sind zu bewerten, indem ihre Vor- und Nachteile gegenübergestellt werden, ihr Kosten/Nutzen-Verhältnis untersucht, die Durchführbarkeit ("feasability study") des entsprechenden Projekts bewertet oder auch eine "impact analysis" (siehe [Eas88]) für alle beteiligten Interessengruppen durchgeführt wird.

Eine Konsensfindung kann dadurch erleichtert werden, dass man versucht, die einzelnen Funktionen und Aspekte des IS gemeinsam in ihrer Wichtigkeit zu priorisieren. In dieser Diskussion kristallisiert sich oft ein großer Teil des "Design Rational" heraus, d.h. der Randbedingungen, denen der Gestaltungsprozess selbst unterliegt und die die eigentlichen Beweggründe für die gewählte Form der Ausgestaltung des Softwaresystems darstellen. In der Diskussion und Bewertung der Lösungsalternativen stellen sich unweigerlich unterschiedliche Schwerpunkte heraus, die die einzelnen Betroffenen setzen. Diese Ergebnisse können genutzt werden, um das Projektkontextmodell aus der Motivationsphase weiter zu verfeinern.

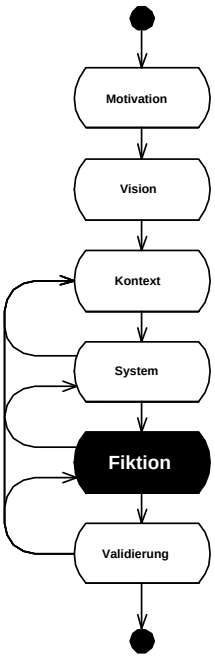
Die beschriebene Vorgehensweise ist von ihrem Charakter her ein Verfeinerungs- und Konkretisierungsvorgang, der sinnvollerweise iterativ durchgeführt wird. Damit lassen sich einzelne Umgestaltungsideen schrittweise entwickeln und in ihren Auswirkungen untersuchen. Um das Gestaltungsergebnis rückverfolgbar zu machen und die ausgewählten Alternativen mit nachvollziehbaren Argumenten untermauern ("Design Rational") zu können, sollte der Gestaltungsvorgang in Form sogenannter "Design-Entscheidungen" protokolliert/dokumentiert werden. Inhalt der Beschreibung sind dabei die identifizierten Gestaltungsobjekte ("issues" nach [BBHL94]), die diskutierten Gestaltungsoptionen ("options") und die gemeinsam beschlossenen Ausgestaltungen ("agreement").

In der konkreten Ausgestaltung des technischen Akteurs kann es helfen, generische Problemsituationen zu unterscheiden ("Problem Frames" nach [Jac94a]), die unterschiedliche standardisierte Rollen von Informationssystemen unterscheiden. Nützliche mentale Konzepte für die Systemgestalt können dabei folgende sein: Das System als "Wissensbasis", "Rechenknecht", "Werkzeug der Informationsverarbeitung", "Einsatzleiter" (wie z.B. bei Workflow- und Einsatzplanungssysteme), "autonomer Agent" oder "Kommunikationsmedium". "Problem Frames" können benutzt werden, um spezifische Fragenkataloge bzw. Drehbücher zu entwickeln, die in den Gestaltungsworkshops – z.B. zur spezifischen Entwicklung von Gestaltungsvarianten – eingesetzt werden können.

In den nun folgenden Vorgehensschritten der Fiktion wird der Einfluss des Softwaresystems auf die Arbeits- und Kommunikationsabläufe der Akteure in der Anwendungsumgebung szenarisch dargestellt, um sie mit den Betroffenen abstimmen und diskutieren zu können.

5.6. Fiktion

lat. fingere, fictum: formen, bilden, ersinnen



Ziel: Illustration der zukünftigen Anwendungswelt

In der Phase der Fiktion antizipieren und dokumentieren die Entwickler nun szenarisch den Einsatz des IS im anvisierten neuen Arbeitskontext. Ziel ist dabei eine möglichst bild- und kontextreiche Visualisierung, die eine Validierung der Systemgestalt durch die betroffenen Interessengruppen erleichtert.

Das System wird als Akteur in den Arbeitskontext aufgenommen und seine Rolle anhand typischer Konversations- und Kooperationsszenarien dokumentiert. Die zur Darstellung der Aktivitätsmuster verwendeten Aktivitätsdiagramme werden dabei möglichst bildhaft angelegt, um die Antizipation und Validierung der zukünftigen Arbeitssituation durch die Anwender zu erleichtern. Sie besitzen den Charakter von "Storyboards", die die einzelnen Arbeitsschritte der zukünftigen Arbeitswelt möglichst bildhaft und kontextreich vorstellen.[12]

Die Fiktion übersetzt die in der Kontextanalyse und der Systemgestaltung definierten abstrakten Aktivitäts- und Konversationsmuster (Drehbuch) in konkrete und anschauliche Handlungsstränge der zukünftigen Arbeitspraxis, die der Vorstellungskraft zuträglicher sind und damit eine Validierung der Systemgestaltung auf Modellebene ermöglichen. Die Einzelaktivitäten werden dabei kontextreich "ausgeschmückt", indem auch alle aktivitätsbegleitenden Interaktionen, Interaktionsmedien und -partner – u.U. auch bildhaft – notiert werden.

[12] Der Begriff "Storyboard" entstammt aus der Filmbranche und beschreibt ein skizzenhaftes bzw. Comic-ähnliches Skript für die filmische Umsetzung einzelner Abschnitte eines Drehbuchs. Es erfordert eine erste Transformation der beschriebenen Handlung in die Bildersprache nebst einer Festlegung der notwendigen Kameraeinstellungen und -fahrten; Zielgruppe sind oft auch "fachfremde" Kunden oder Mitarbeiter, die sich mit der Materie des "Filmemachens" wenig auskennen.

Alternativ zu einer expliziten statischen Dokumentation der Fiktion durch die Entwickler in Vorbereitung auf eine formale Validierung dieser Dokumente kann natürlich auch ein Workshop durchgeführt werden, in der die neue Arbeitsumgebung nicht explizit dokumentiert sondern in Form von Rollenspielen bzw. "Walk Throughs" dynamisch erarbeitet und diskutiert wird. Diese Abkürzung bietet sich an, wenn die Anzahl der Interessengruppen überschaubar ist und Vertreter dieser Interessengruppen für eine entsprechende Zeitspanne zur Verfügung stehen.

Im Allgemeinen ist jedoch eine explizite Dokumentation der zukünftigen Arbeitsweise auch hier unabdingbar, weil sie das eigentliche Ziel der Entwicklung anschaulich vor Augen führt und im Besonderen bereits auch Validierungsszenarien zur Abnahme des fertigen Softwaresystems liefert. Als Nebeneffekt gewinnt man bereits Dokumente, die als Basis für die Erstellung von Bedienungsanleitungen verwendet werden können, so dass entsprechende Aktivitäten (Erstellung einer Online-Hilfe, eines Handbuch, Ausarbeitung von Schulungskonzepten etc.) bereits parallel zum Entwicklungsprozess eingeleitet werden können.

Zum Vorgehen

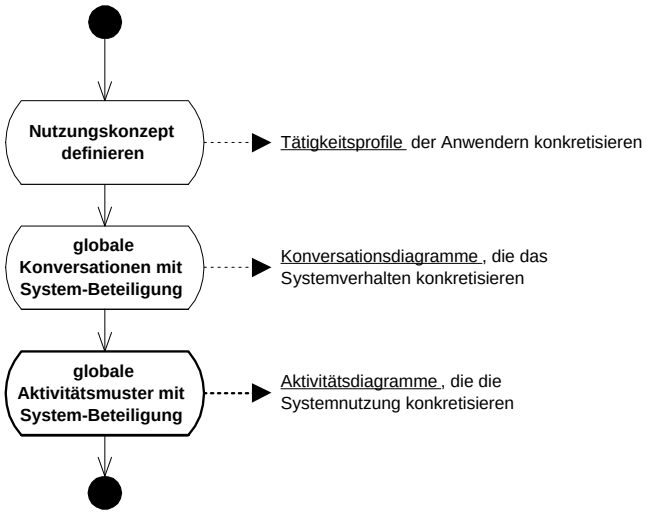


Abbildung 5.17.: Vorgehen in der Beschreibung der Fiktion

Abbildung 5.17 stellt das Vorgehen in der Fiktionsphase dar:

> Im ersten Schritt wird ein konkretes Nutzungskonzept erarbeitet, indem die Tätigkeitsprofile aller Anwender des Systems lokal konkretisiert und in Form von zusammenhängenden Anwendungsfällen dokumentiert werden.
>
> In nachfolgenden Schritten werden typische Konversations- und Aktivitätsmuster bzw. -szenarien beschrieben, die die Rolle des Softwaresystems in der konkreten Arbeitspraxis dokumentieren.

Ausgangspunkt für die Auswahl geeigneter Aktivitätsmuster ist das Kooperationsmodell der Kontextanalyse, das nun ausschnittsweise um Aktivitäten und Interaktionen des Softwaresystems erweitert und beispielhaft konkretisiert wird, indem typische und charakteristische

Systembenutzungssequenzen dokumentiert werden.

In vielen Fällen, so auch bei Apvel, wird das IS nicht die globalen Arbeitsabläufe umfassend verändern sondern die Arbeitssituation von Anwendern eher auf der Ebene lokaler Aktivitäten beeinflussen. Oft werden darum die Ergebnisse der Kontextmodellierung weiter Bestand haben und die dort modellierten Aktivitätsmuster unverändert bleiben können. Beschränkt man sich bei der Kontextmodellierung darüberhinaus auch auf eine eher abstrakte Ebene, so ist die Erstellung eines neuen Modells zur vollständigen Darstellung der zukünftigen Anwendungsumgebung ("Soll-Kontextmodell") meist wenig aufwendig.

Eine umfangreiche Umgestaltung und Neumodellierung der in der Kontextanalyse erstellten Modelle ist allerdings notwendig, sobald das Softwaresystem weitreichendere Auswirkungen auf die betriebliche Arbeitsorganisation besitzt. Solche Wirkungen können Kommunikationskanäle besitzen, die vom Softwaresystem verändert oder neu realisiert werden, indem das System beispielsweise neue Kommunikationsmedien, neue Kommunikationsinhalte und -formen einführt.

Unterstützt ein System desweiteren auch komplexe Kooperationen mehrerer Akteure, so wird es hier mit großer Wahrscheinlichkeit auch neue Interaktionsmuster zwischen den beteiligten Akteuren begründen und damit deren Arbeitsorganisation stark beeinflussen. Ein Beispiel für solche Systeme sind kollaborative Softwaresysteme (z.B. CSCW-Systeme), die primär gerade die globale Arbeitsorganisation verändern. In diesem Fall ist eine Iteration der Kontextanalyse sinnvoll, um für die Fiktion bzw. die nachfolgende Validierung der Systemgestalt zum Einen den zukünftigen Arbeitskontext auch im weiteren Umfeld des neuen Softwaresystems zu kennen und zum Anderen typische Systemnutzungsszenarien in dieser neuen Anwendungsumgebung identifizieren und beschreiben zu können.

Zur Modellierung der zukünftigen Systemnutzung beschränkt sich die Fiktion nicht nur auf eine globale Darstellung des übergreifenden Zusammenspiels der beteiligten Akteure, sondern sie verdeutlicht auch lokale Systemnutzungssequenzen auf der Tätigkeitsebene realer Systemanwender, die das System im Rahmen eigener zielorientierter Aktivitäten einsetzen (konkrete "Use Cases" im UML-Sinne). Zu diesem Zweck werden im nächsten Schritt nun auch die Tätigkeitsprofile aller zukünftigen Anwender des Systems konkretisiert und ein Nutzungskonzept erarbeitet, welches nicht nur den regulären "Betrieb" aus Anwendersicht verdeutlicht sondern auch den Umgang mit dem System in Ausnahmefällen skizziert.

Während beim Nutzungskonzept die Dokumentation des praktischen funktionsbezogenen Umgangs mit dem System im Vordergrund steht, befindet man sich in der Modellierung der akteurübergreifenden Kooperationsmuster auf einer höheren Abstraktionsebene, die die einzelnen Aktivitäten der beteiligten Akteure in einen größeren Arbeitszusammenhang stellt. KoA$_{\text{L}}$A kommt damit auch einer Forderung von Kari Kuutti nach, die eine Beschreibung der Systemnutzung im größeren Rahmen übergreifender Arbeitsprozesse einfordert: "We need application scenarios (use cases) embedded in context scenarios – something that has not yet been very well explicated" [Kuu95].

Bewährtes Mittel zur Darstellung der Nutzungssituation sind dabei Szenarien, die von Anwendern natürlichsprachlich in Textform beschrieben werden. KoA$_{\text{L}}$A ermöglicht die Beschreibung von akteurübergreifenden Aktivitätsmustern oder -szenarien sowohl in grafischer als auch semiformaler Textform. Laut Paech sind Szenarien oft ein guter Ersatz für ein aufwendiges Prototyping ([Pae00] S.205). Parallel zur Modellierung und der szenarischen Darstellung

der neuen Anwendungsumgebung können natürlich aber auch Prototypen aus Papier (paper mockups [GK91]) eingesetzt werden, um das Vorstellungsvermögen zu unterstützen, indem die neuen Arbeitsprozesse von den Anwendern auch bildlich vergegenwärtigt oder physisch durchgespielt werden können.

Die Qualität des Softwaresystems und damit seine Praxistauglichkeit wird von den Projektbeteiligten anhand der erstellten Modelle beurteilt. Da diese Modelle weitgehend semiformal und unvollständig sind, kann diese Qualität im Besonderen nicht durch Metriken berechnet oder in Simulationen untersucht werden, sondern erfolgt durch die "menschlichen Leser" der Modellierungsdokumente.

5.6.1. Nutzungskonzept

Ziel: Skizzierung der rollenspezifischen Systemnutzung

In diesem Abschnitt werden nun die Tätigkeitsprofile der in der Systemgestaltung (s. Abschnitte 5.5.2 und 5.5.5) definierten Nutzer-Rollen bzw. der in der Kontextanalyse (s. Abschnitt 5.4.5) beschriebenen realen Anwender-Akteure konkretisiert, indem ihre reale Arbeitstätigkeit beschrieben wird, wie sie sich nach der Einführung des Systems darstellt. Der Schwerpunkt der Modellierung liegt dabei auf der konkreten Nutzung des Systems, auch in Sonderfällen (s. Abschnitt 5.4.6). Nachdem die systemspezifischen Anteile der in Abschnitt 5.5.2 identifizierten Systemkooperationen bereits in Form des Systemprofils konkretisiert wurden, werden nun auch die Aktivitäten der Systemanwender konkretisiert, um damit ihren Anteil an der Kooperation deutlich zu machen. Die Aktivitätsbeschreibungen stellen eine Art "Bedienungsanleitung" dar, die alle relevanten Tätigkeiten der System-Anwender beschreiben und damit die Art und Weise des geforderten Umgangs mit dem System deutlich machen. Die dargestellten Aktivitäten umfassen neben dem systemspezifischen Nutzungsverhalten auch alle sonstigen lokalen Tätigkeiten der Nutzer. Desweiteren wird das Tätigkeitsprofil von Systemnutzern um Randbedingungen ergänzt, die ihnen durch die Systemgestaltung auferlegt werden.

Zur Modellierung

Die neuen Tätigkeitsprofile der Anwender werden vor Allem textuell dokumentiert. Im Folgenden finden sich als Beispiel Ausschnitte aus dem neuen Tätigkeitsprofil des Sachbearbeiters:

```
AKTEUR          Sachbearbeiter

ANFORDERUNG     PrüfungsergebnisseBerücksichtigen
Der Sachbearbeiter hat sicherzustellen, dass nur solche Studenten zum
Praktischen Jahr zugelassen werden, die auch ihr 2. Staatsexamen bestehen.
Die Ergebnisse der Prüfung sind hierfür in Erfahrung zu bringen und eventuell
betroffene Anträge noch vor der Durchführung der Nachverteilung zu löschen.

ANFORDERUNG     NachrangigkeitBestimmen
Der Sachbearbeiter hat sich darum zu kümmern, dass Anträge von Studenten, die
einem Praktischen Jahr unentschuldigt ferngeblieben sind, vor ihrer Zuteilung
als nachrangig markiert werden.
```

```
Das System weist zwar auf Solche Studenten hin, für die bereits im System ein
Antrag vorliegt. Die Entscheidung über die Nachrangigkeit hat jedoch
letztendlich der Sachbearbeiter zu treffen.
...

AKTIVITÄT       FachabteilungsPflege
                Pflege von Krankenhaus-Abteilungen
ABLAUF          Der Sachbearbeiter legt für jedes an einem Krankenhaus
                gelehrte Ausbildungsfach eine eigene Fachabteilung an
AUSNAHME        AbteilungsDoppel
                Als Sonderfall der Fachabteilungs-Pflege kann es notwendig
                sein, für ein bestimmtes Ausbildungsfach zwei Abteilungen
                einzupflegen, da diese aufgrund ihrer räumlichen Trennung auch
                in der Zuteilung getrennt behandelt werden.
                Im Fall der Universitätklinik trifft das z.B. die Abteilungen
                Psychiatrie (Günzburg), Neurologie (Dietenbronn) und
                Anästhesie (RKU-Ulm), die so auch direkt an der Klinik in Ulm
                existieren.
                In diesem Fall werden vom Sachbearbeiter für die auswärtige
                Abteilung jeweils eine zweite Fachabteilung angelegt, die zur
                Unterscheidung in Klammern hinter dem Fachnamen noch den Ort
                der Abteilung aufführt.

AKTIVITÄT       NeuesPJVorbereiten
                Aktualisierung aller Stammdaten vor Erfassung der Anträge
INTERAKTION     PflegtStammdaten >> System
                Diese abstrakte Interaktion des Sachbearbeiters schließt alle
                Aktionen zur Pflege der Stammdaten ein: Anlegen, Lesen, Ändern
                und Löschen von Praktischen Jahren, Krankenhäusern und
                Fachabteilungen
AUFTRAG         PJKopieren >PJ> System
ABLAUF          Zur Vorbereitung eines neuen Praktischen Jahres kopiert der
                Sachbearbeiter normalerweise das Praktische Jahr, welches ein
                Jahr vorher zur gleichen Jahreszeit stattfand, als Vorlage und
                passt es nachträglich an die aktuellen Bedingungen des
                laufenden Jahres an. Da sich die Krankenhäuser und deren
                Kapazitäten für die Herbst- bzw. Frühjahrs-Praktika wenig
                ändern, ist dies effizienter als das Anlegen eines neuen
                Praktischen Jahres von Grund auf.
                Parallel hierzu informiert sich der Sachberbeiter bei den
                Krankenhäusern bzw. bei der Studienkomission über den
                aktuellen Kapazitätsbedarf bzw. die aktuelle reale Auslastung
                und passt die eingepflegten Vertragskapazitäten in Absprache
                mit den Krankenhäusern und ihren Fachabteilungen entsprechend an.

AKTIVITÄT       Zwischenverteilung
                Zur Handhabung externer und verspäteter Bewerbungen nach der
                Hauptverteilung
INTERAKTION     BewerbenSich <Antrag[*]< ExterneStudenten
INTERAKTION     BewerbenSich <Antrag[*]< Studenten
AUFTRAG         Zwischenverteilung >PJ> Apvel
AUFTRAG         ZuteilungsbescheideGenerieren
                >Student[*]: Zuteilungsbescheid[*]> Apvel
ABLAUF          Auch nach der Durchführung der Hauptverteilung bewerben sich
                noch Studenten der Universität Ulm oder anderer Universitäten
                für ein Praktisches Jahr in Ulm. Um diese Studenten zu
                berücksichtigen, pflegt der Sachbearbeiter ihre Anträge
```

```
                sukzessive oder gesammelt ein und führt ab und an eine
                Zwischenverteilung durch, sobald eine bestimmte Anzahl an
                Anträgen eingegangen ist. Allen Studenten wird daraufhin ein
                Zuteilungsbescheid zugeschickt mit der Option, dem
                Zuteilungsergebnis zu widersprechen und am Nachrückverfahren
                teilzunehmen.

AKTIVITÄT       Krankheitsfälle
                Zur Handhabung von Krankheitsfällen
BEDINGTDURCH    Sonderfall:Krankheit
INTERAKTION     MeldetErkrankung <Student< Krankenhaus
ABLAUF          Wenn ein Student im laufenden Praktischen Jahr krank wird, so
                wird das PJ-Büro direkt von den Lehrkrankenhäusern darüber
                informiert. Der Sachbearbeiter hat dann im Apvelsystem solche
                Zuteilungen zu Ausbildungsfächern zu löschen und damit als
                nicht-besucht zu werten, in denen der Student die erlaubte
                Anzahl an Fehltagen überschritten hat. Sobald sich der Student
                wieder an das PJ-Büro wendet, wird er vom Sachbearbeiter von
                Hand seinen fehlenden Fächern zugeteilt. Dies geschieht
                komplett unabhängig vom üblichen Antrags- und
                Verteilungsverfahren!

AKTIVITÄT       VerspäteteAnträge
                Zur Handhabung verspäteter Anträge
BEDINGTDURCH    Sonderfall:Verspätungen
INTERAKTION     StelltPJAntrag <Antrag< Student
INTERAKTION     SchicktZuteilungsBescheid >Zuteilungsbescheid> Student
ABLAUF          Anträge Ulmer Studenten sind noch bis zu einer Woche nach
                Start eines Praktischen Jahres zu berücksichtigen. Da das
                Medizinstudium ein NC-Fach ist, haben Ulmer Studenten einen
                gesetzlichen Anspruch auf einen Praktikumsplatz.
                Trifft ein Antrag nach dem Stichtag der Bewerbung ein, wird
                der Antrag vom Sachbearbeiter eingepflegt, als nachrangig
                gekennzeichnet und zusammen mit anderen verspäteten und
                externen Anträgen im Rahmen einer Zwischenverteilung
                berücksichtigt.
                Trifft der Antrag jedoch nach dem Nachrückverfahren ein, so
                wird die Zuteilung des Studenten vom Sachbearbeiter von Hand
                vorgenommen.
```

Im Beispiel wird die abstrakte Kooperation “Datenpflege” aus der Sicht der Anwender teilweise konkretisiert. Die Aktivität “Neues PJ vorbereiten” beschreibt und konkretisiert beispielsweise alle Aktivitäten des Sachbearbeiters, die zur Vorbereitung eines neuen Praktischen Jahres durchgeführt werden müssen. Hierbei wird im Besonderen das konkrete Vorgehen bei der Anlage eines neuen Praktischen Jahres im Apvel-System beschrieben.

Die Aktivität “Zwischenverteilung” stellt hingegen die entsprechenden verwendeten Systemfunktionen in ihren konkreteren Nutzungszusammenhang. In der Ablaufbeschreibung finden sich Hinweise, wann, in welchem Kontext und in welcher Reihenfolge die Systemfunktionalitäten gebraucht und benutzt wird.

Zur Analyse der Arbeitsorganisation ist laut Paech das genaue Verständnis der Interaktion zwischen Nutzern und Softwaresystem bei der Durchführung der Arbeitsaufgaben notwendig. Hierbei sind die Besonderheiten wichtiger als das normative Verhalten. Insbesondere ist zu untersuchen, wie die Nutzer mit Ausnahmen in den Arbeitsprozessen umgehen ([Pae00],

S.205). Aus diesem Grund werden in KoA$_{\text{L}}$A bei der Darstellung der zukünftigen Arbeitssituation auch Nutzungskonzepte für Sonderfälle konkretisiert, die z.B. "Übersetzungen" in die Begriffswelt des Informationssystems notwendig machen.

Jedes IS definiert einen festen Handlungsrahmen, innerhalb dessen sich ein Anwender bewegen kann. Das IS "kennt" eine bestimmte Begriffswelt und ermöglicht dem Benutzer bestimmte fest definierte Möglichkeiten zur Interaktion und/oder Manipulation dieser "Welt". Er entwickelt ein bestimmtes mentales Modell, um sich in der neuen künstlichen Welt zurechtzufinden (siehe [Dut94]). Da sich die Wirklichkeit ständig ändert, ist dieses Modell aber nie vollständig befriedigend, d.h. die zur Verfügung stehenden "Ausdrucksmittel" (Systemdaten, Funktionen, etc.) reichen in der Praxis nie für alle realen Anwendungsfälle aus, so dass der Anwender gezwungen ist zu "improvisieren". Er muss daher zwangsläufig kreativ sein, um mit den fest definierten und damit "beschränkten" Mitteln des gegebenen Softwaresystems dennoch zum Ziel zu kommen. Besonders problematisch ist es natürlich, wenn es sich um ein IS handelt, das selbst Kontrolle über seine Benutzer ausübt (WFMS, Einsatzplanung, SAP etc.) und der Benutzer gezwungen ist, sich an ein "unpassendes" oder gar falsches Modell anzupassen.

Mit dem Einsatz eines neuen Softwaresystem stößt man also unweigerlich auf Situationen, die die eigentliche Funktionalität des Softwaresystems übersteigen. Da es oft unmöglich oder aus wirtschaftlichen Gründen unvertretbar ist, ein System so zu bauen, dass neben den Hauptnutzungsszenarien auch alle Sonderfälle in gleichem Maße berücksichtigt werden, ist es wichtig, bereits bei der Gestaltung des Systems auf seine Flexibilität zu achten. Besonders für alle bereits bekannten oder vorhersagbaren Sonderfälle sind daher Nutzungskonzepte zu erarbeiten, wie mit dem entworfenen System in diesen Fällen umzugehen ist. Aus ökonomischen, arbeitspsychologischen oder politischen Gründen kann es dabei durchaus sinnvoll und gewollt sein, ein Softwaresystem bewusst möglichst klein, flexibel und verständlich zu halten, auch wenn die konkrete Nutzung des Systems dann einige Einarbeitungszeit verschlingen kann.

Im Apvel-Beispiel gibt es beispielsweise in der Universitätsklinik zwei verschiedene Örtlichkeiten, an denen das Fach Psychiatrie gelernt werden kann. Sowohl in Ulm als auch in Günzburg existiert eine entsprechende Abteilung, die in der Verteilung getrennt zu berücksichtigen ist. Da in der Systemgestaltung von Apvel keine explizite Unterscheidung zwischen Ausbildungsfach und der ausbildenden Abteilung getroffen wurde und diesbezüglich pro Krankenhaus und Ausbildungsfach nur eine virtuelle Fachabteilung verwaltet wird, müssen für diesen Ausnahmefall im Apvel-Softwaresystem nun zwei unterschiedliche Wahlfächer angelegt werden, "Psychiatrie" und "Psychiatrie (Günzburg)".

Eine solche "Ausnahmebehandlung" wird im Beispiel in einem "AUSNAHME"-Abschnitt der Aktivität "FachabteilungsPflege" des Sachbearbeiters dokumentiert. Der Ausnahme-Abschnitt "AbteilungsDoppel" definiert implizit sowohl eine Sonderfall-Bedingung, die zu einer Abweichung vom normalen Verhalten führt, als auch eine entsprechende Aktivitäts-Aggregation: die Sonderfall-"Behandlung". Ausnahme-Abschnitte stellen damit eine Art Kurzschreibweise für die explizite Definition einer entsprechenden Bedingung und einer aggregierten Aktivität dar.

Weitere Kandidaten für die Dokumentation einer Nutzung des Systems sind alle im Abschnitt 5.4.6 identifizierten Sonderfälle. Im Apvel-Beispiel finden sich hierzu eine Beschreibung der Tätigkeiten des Sachbearbeiters, wie sie in den Sonderfällen von verspäteten Bewerbungen oder Erkrankungen von Praktikumsstudenten notwendig werden.

Zu vollständigen Beschreibung der Tätigkeitsprofile der betroffenen Anwender gehören allerdings nicht nur Beschreibungen ihrer Aktivitätsbereiche in normalen oder auch in speziellen Sachverhalten, sondern auch eine Definition aller Anforderungen, die das Softwaresystem selbst an seine Nutzer stellt. Hier sind alle Vereinbarungen zu thematisieren, die den praktischen Systembetrieb begleiten. Nachdem in den vorherigen Abschnitten der Systemgestaltung vor allem die Nutzerrechte bzw. die Systemleistungen dokumentiert wurden, werden nun auch die Nutzerpflichten bzw. Nutzerleistungen dokumentiert. Jeder Nutzer besitzt eigene spezifische Aufgaben (der Datenpflege, der regelmäßigen Aktualisierung, der Konsistenzüberprüfung, der regelmäßigen Wartung etc.), die mit dem erfolgreichen Betrieb des Softwaresystems einhergehen und nun in Form von Bedingungen bzw. Anforderungen modelliert werden, die der System-Akteur an seine Anwender stellt.

Im Apvel-Projekt ergibt sich durch die gewählte Systemgestaltung z.B. an den Sachbearbeiter die Anforderung, vor Eingabe eines neuen Antrags von Hand zu überprüfen, ob der Student sich evtl. bereits in einem früheren PJ angemeldet hat und seiner zugewiesenen Stelle unentschuldigt ferngeblieben ist. Ist das der Fall, so ist der Antrag vom Anwender als nachrangig zu markieren. Eine weitere Anforderung ergibt sich an den Sachbearbeiter in Bezug auf die Berücksichtigung von Prüfungsergebnissen. Sobald die Ergebnisse der Prüfungen zum 2. Staatsexamen vorliegen, muss der Sachbearbeiter alle Anträge von Studenten per Hand löschen, die diese Prüfung nicht bestanden haben. Dies ist notwendig, um die Gesetzmäßigkeit der Zuteilung sicherzustellen und die Chancen von Studenten im Nachrückverfahren zu verbessern.

Zum Vorgehen

Im letzten Abschnitt der Systemgestaltung wurde in Form des Systemprofils bereits konkretisiert, welche Rolle das Softwaresystem selbst in den Systemkooperationen übernimmt, die zuvor im Abschnitt 5.5.2 skizziert wurden. Nach der Dokumentation der Softwaresystemanteile werden hier in Form des Nutzungskonzepts nun auch die Anwenderanteile identifiziert und näher beschrieben. Die Kooperationen des Systemkontextmodells werden hier aus Nutzersicht soweit verfeinert, dass die Aktivitäten der Systemanwender weitgehend klar werden.

In der Darstellung des Nutzungszusammenhangs werden hierbei im Besonderen auch möglichst viele Sonderfälle betrachtet und ihre Behandlung beschrieben. Sonderfälle können dabei in unterschiedlicher Form Eingang in die Modellierung finden: als textuell oder grafisch beschriebenes Aktivitätsmuster, als Aktivitäts-Spezialisierung oder als entsprechend stereotypisierte Bedingung. Beispiele für nutzungsspezifische Sonderfälle aus dem Apvel-Projekt:

- die Hauptverteilung und die Nachverteilung werden wiederholt, wenn nicht allen Ulmer Studenten ein Platz zugeteilt werden konnte. Vorher werden in Absprache mit den Lehrkrankenhäuser die Ausbildungskapazitäten einzelner (beliebter) Abteilungen erhöht.
- auch wenn man nach einer Hauptverteilung schon ein oder mehrere Zwischenverteilungen durchgeführt hat, kann es passieren, dass man die Hauptverteilung wiederholen muss (Verfahrensfehler, Änderungen in den Kapazitäten bzw. Krankenhaus- und Abteilungsdaten). Bei einer Hauptverteilung gehen jedoch auch die Zwischenverteilungen wieder verloren, so dass sie im Anschluss an die endgültige Hauptverteilung wiederholt werden müssen.

- im Falle ein oder mehrerer PJ-Studenten mit Teilzeitstellen muss die Auslastung der Abteilungen von Hand nachgeprüft werden, da diese Studenten einzelne Plätze länger als ein Praktisches Jahr lang belegen, diese Belegung vom System aber in der Kapazitätberechnung nicht mitberücksichtigt wird. Das Zuteilungsergebnis muss daher vom Sachbearbeiter möglicherweise noch nachbearbeitet werden. Alternativ sind vor der Verteilung die Abteilungskapazitäten PJ-bezogen zu erniedrigen.
- einzelne Kapazitäten von Fachabteilungen können sich organisationsbedingt auch nach der Veröffentlichung der Hauptverteilungsergebnisse noch ändern. Die betroffenen Studenten werden hierzu per Hand neu verteilt und über die Änderung informiert. Die restlichen Zuteilungsergebnisse bleiben unangetastet.

Das Vorgehen in der Erstellung des *Nutzungskonzepts* richtet sich im Allgemeinen nach folgenden Leitfragen:

Wie verändern sich die Aktivitäten der Akteure?

Welche Interaktionen zwischen Anwender und System finden statt? Welche Aktivitäten sind davon betroffen bzw. werden ausgelöst? Wie sieht der lokale Anteil von Akteuren konkret aus, die an Kooperationen mit dem System beteiligt sind? Welche (neuen) Interaktionen werden initiiert?

Im Beispiel: Die Vorbereitung eines neuen Jahres, die Abwicklung der Bewerbung, die Durchführung der Zuteilung etc.

Welche Anforderungen stellt das Softwaresystem an seine Anwender?

Welche Ansprüche stellt das System an die Akteure? Muss das Systemwissen z.B. von jemandem regelmäßig aktualisiert werden? Wie gelangt das Wissen ins System? Wie und von wem wird dieses Wissen verarbeitet bzw. ausgewertet?

Im Beispiel: Der Sachbearbeiter hat die Nachrangigkeit der Studenten per Hand einzustellen. Außerdem ist das Prüfungsergebnis vom Sachbearbeiter derart zu berücksichtigen, dass er irrelevant gewordene Anträge per Hand löscht.

Wie gestaltet sich die Zusammenarbeit mit dem System in Sonderfällen?

Welche Sonderfälle sind bei der Systemnutzung zu erwarten? Wie wird mit dem System in Sonderfällen der Anwendungswelt gearbeitet? Wie geht man mit dem System in Fällen um, für die das System eigentlich nicht ausgelegt ist?

Im Beispiel: Was passiert bei Krankheitsfällen, verspäteten Anträgen etc.?

5.6.2. Konversationsszenarien

Ziel: Skizzierung zukünftiger Interaktionsmuster

Zur Beurteilung der Praxistauglichkeit eines IS reicht es nicht aus, sich nur die lokale Nutzungssituation der einzelnen Anwender anzusehen. Darüber hinaus müssen auch die akteurübergreifenden Aktivitätsbereiche untersucht werden, die sich mit dem Einsatz des neuen Softwaresystems ergeben bzw. deren Ablauf sich durch diesen Einsatz verändern. Ziel dabei

ist es, die Rolle des neuen Softwaresystems in Bezug auf die systemischen Aktivitätsbereiche im Umfeld des Softwaresystems herauszuarbeiten.

Eine gute Möglichkeit zur Bewertung der Praxistauglichkeit besteht in der Untersuchung von charakteristischen Interaktionsabläufen. Sich oft wiederholende Dialoge stellen den Kontext der zu beobachtenden praktischen Arbeitshandlungen in der Regel sehr kompakt und prägnant dar. Modelliert man solche Dialoge in Form von Konversationsdiagrammen und ergänzt sie um Angaben zur Systemnutzung, erhält man übersichtliche Modellierungs-Artefakte zur umfassenden Bewertung der neuen Nutzungssituation.

Mit der Identifikation von Einzelinteraktionen werden als Nebeneffekt zusätzlich auch die Brüche im globalen Arbeitsfluss analysiert, welche wiederum Rückschlüsse auf die Effizienz der dargestellten Arbeitssequenz bzw. -prozesses als Ganzes zulassen.

Die Kommunikationsmodellierung ist umso wichtiger, je mehr Einfluss der Einsatz des neuen IS auf die Arbeitspraxis und damit auf die Kommunikationspraxis besitzt. Wenn ein Werkzeug nur bestimmte lokale Arbeiten automatisiert bzw. unterstützt (Bsp: Datenverwaltung und Konsistenzsicherung per Rechner statt per Hand; Verteilung per Rechner statt per Hand), aber am globalen Ablauf und den zu beobachtenden Kommunikationsmustern insgesamt nichts ändert, müssen in der Regel keine neuen Konversationen modelliert werden. In diesem Fall nimmt man die in Abschnitt 5.4.2 identifizierten Konversationen und reichert sie um konkrete Systemnutzungsaktivitäten oder -interaktion an.

Falls sich jedoch die Arbeitsorganisation umfassend ändern sollte, ist in der Vorbereitung auf die Fiktionsphase evtl. eine Iteration der Kontextmodellierung (siehe Abschnitt 5.4) notwendig. In der Fiktion werden dann diejenigen Szenarien in Bezug auf die Systemnutzung konkretisiert, welche für die Verdeutlichung der Systemrolle den größten Nutzen versprechen.

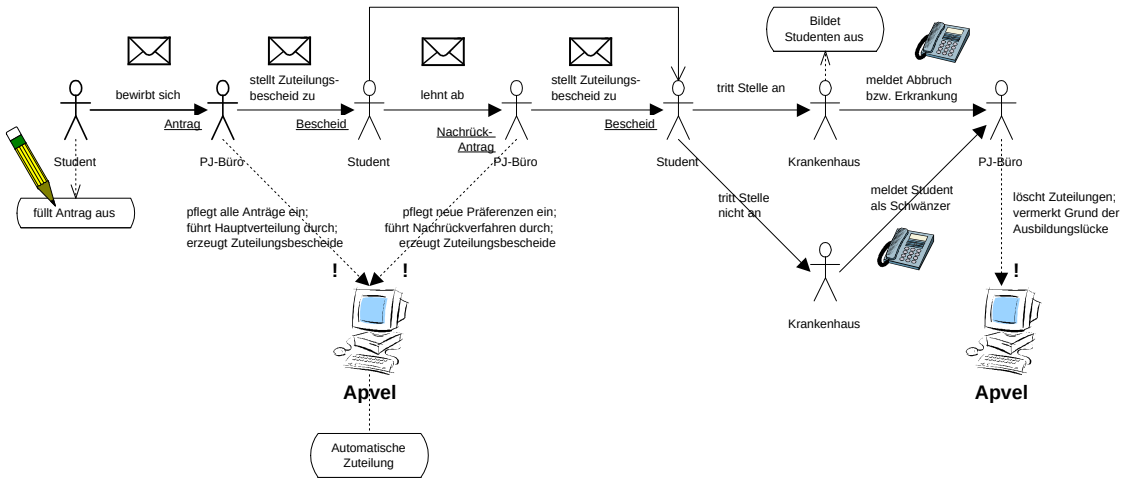


Abbildung 5.18.: Fiktives Konversationsdiagramm für den Bewerbungsablauf

Abb. 5.18 stellt die Systemnutzung im Kontext des Kommunikationsverlaufs im Rahmen einer normalen Bewerbung dar:

Kommunikationspartner sind dabei ein bestimmter Student, das PJ-Büro und das Krankenhaus, dem der Student zugeteilt wird. Annotiert werden die einzelnen Interaktionen jeweils durch eine bildhafte Repräsentation des benutzten Kommunikationsmediums, während die Akteure jeweils um lokale Aktivitätsbeschreibungen ergänzt werden.

Beispielsweise füllt der Student einen Antrag aus, bevor er ihn per Briefpost an das PJ-Büro einschickt. Als weitere Aktivitäten der Akteure werden im Diagramm auch alle Interaktionen mit dem System notiert, die im Rahmen des dargestellten Interaktionsverlaufs auftreten können.

Treffen im PJ-Büro Bewerbungs-Anträge für das aktuelle Praktische Jahr ein, pflegt das PJ-Büro diese Anträge in das Apvel-System ein. Sobald der Stichtag erreicht wird, gibt das PJ-Büro an das Apvel-System den Auftrag zur automatischen Zuteilung.

Nachdem auch alle Zuteilungsbescheide vom System generiert wurden, schickt das PJ-Büro den entsprechenden Bescheid an den Studenten per Brief u.s.w.

Die Modellierung wird von folgenden Leitfragen begleitet:

Welche Konversationen spiegeln die Rolle des Systems besonders gut wieder?

Diese Konversationen dienen als Vorlage für die fiktiven Konversationsszenarien. Sie stammen in der Regel aus der Kontextanalyse (Abschnitt 5.7)

Im Beispiel: Das Bewerbungsverfahren.

Welche Kommunikationsformen werden mit dem System neu eingeführt?

Hier ist evtl. eine Neumodellierung des Kontexts notwendig. Führt zu einer Iteration der Kontextanalyse.

Beispiel: Die Bewerbung und Bekanntgabe über die Webkomponente von Apvel stellt eine völlig neue Art der Kommunikation zur Verfügung, die eigens auszuarbeiten ist.

An welcher Stelle wird mit dem System interagiert?

Welche Akteure interagieren wie mit dem System? Welche Aufträge werden erteilt? Wie und wann interagiert das System evtl. mit Akteuren seiner Umgebung?

Im Beispiel: Bei der Eingabe der Anträge, dem Start der Zuteilung etc.

Welche Tätigkeiten führen die Akteure im Laufe der Kommunikation aus?

Welche Tätigkeiten werden in Reaktion auf eine vorhergehende Interaktion oder in Vorbereitung auf eine nachfolgende Interaktion durchgeführt?

Im Beispiel: Der Student füllt den Antrag aus, das Krankenhaus bildet die Studenten aus, etc.

Welche Medien werden bei der Kommunikation benutzt?

Die Annotation der verwendeten Medien dient der kontextreichen Bebilderung der einzelnen Interaktionen. Diese Notation kann auch zur Modellierung systemunterstützter Interaktionsbeziehungen verwendet werden.

Im Beispiel: Telefon, Briefpost, etc.

5.6.3. Aktivitätsszenarien

Ziel: Skizzierung zukünftiger Aktivitätsszenarien

Eine weitere Möglichkeit zur Untersuchung der Praxistauglichkeit eines Softwaresystems ist die Darstellung und Diskussion von akteur- oder einzeltätigkeitsübergreifenden Aktivitätsmustern und -szenarien. Die neue Arbeitsumgebung wird nun auch unter dem Blickwinkel neuer umfassender Handlungsmuster beleuchtet, die durch den Einsatz des neuen Softwaresystems eingeführt werden und in Form von Aktivitätsmustern möglichst bildhaft und kontextreich dokumentiert werden. Die zukünftigen Nutzer sollen darüber in den Stand versetzt werden, sich über die Auswirkungen des neuen Softwaresystems bereits vorneweg ein Bild machen zu können. Die dargestellten Aktivitätsbereiche sollen Antworten auf die Frage geben, wie die neue Anwendungsumgebung konkret aussehen würde, wenn der im Abschnitt 5.12 ausgestaltete Systemakteur fertig realisiert und eingesetzt werden würde. Sie stellen die Systemaktivitäten explizit in den Kontext von Arbeitsabläufen des umgebenden Anwendungsbereichs und dienen den betroffenen Interessengruppen damit zur Validierung der Systemgestaltung und als Diskussionsgrundlage für den weiteren Gestaltungsprozess.

Da man die Aktivitäten des zukünftigen Anwendungsbereichs aufgrund ihrer Vielfalt niemals vollständig erfassen und beschreiben kann, muss man sich in der Darstellung auf die charakteristischsten Aktivitätsszenarien beschränken, möchte man trotzdem ein möglichst umfassendes Bild des zukünftigen Anwendungsbereichs zeichnen. In der Darstellung der Aktivitätsszenarien beschränkt man sich desweiteren auf spezifische Blickwinkel, um nur die Ausschnitte des Aktivitätsbereichs zu modellieren, die im gewählten Kontext Relevanz besitzen.

Zur Modellierung

Die charakteristischsten Aktivitätssequenzen der neuen Arbeitswelt werden in Form von erweiterten Aktivitätsdiagrammen veranschaulicht (siehe Kap. 4.4.3), die akteurübergreifende Aktivitätsmuster oder -szenarien themenorientiert darstellen.

Erweiterte Aktivitätsdiagramme ordnen sowohl lokale als auch kooperative Handlungen und Interaktionen zwischen Akteuren zu logischen Aktivitätsabläufen an, die jeweils ein bestimmtes Thema der Anwendungsumgebung beleuchten. Diese Aktivitätsmuster werden alternativ oder zusätzlich auch textuell erläutert. Praxisorientierte Ablaufbeschreibungen haben sich im Requirements Engineering bereits sehr bewährt, da Menschen gerne in Abläufen denken, Geschichten erzählen und in Beispielen denken [Car95].

Abb. 5.19 zeigt ein erweitertes Aktivitätsdiagramm, das alle Aktivitäten des PJ-Büros bezogen auf die Organisation und Durchführung eines Praktischen Jahres in Beziehung setzt und die Verwendung des Softwaresystems in diesem Kontext deutlich macht:

Bei der Dokumentation von Aktivitätsszenarien sind hauptsächlich Aktivitäten von Nutzern des zukünftigen Softwaresystems interessant. Das Beispiel-Diagramm beschreibt hierzu alle Aktivitäten, die vom einzigen Nutzer des Apvel-Systems, dem PJ-Büro, normalerweise im Laufe der Organisation, Abwicklung und Abrechnung eines Praktischen Jahres durchgeführt werden. Aktivitäten, an denen das PJ-Büro selbst nicht beteiligt ist, wie z.B. die Verhandlung der Ausbildungskapazitäten, die Durchführung des PJ und die Buchführung, dienen nur zur weiteren Beschreibung des Aktivitätskontexts und werden im Diagramm gestrichelt dargestellt.

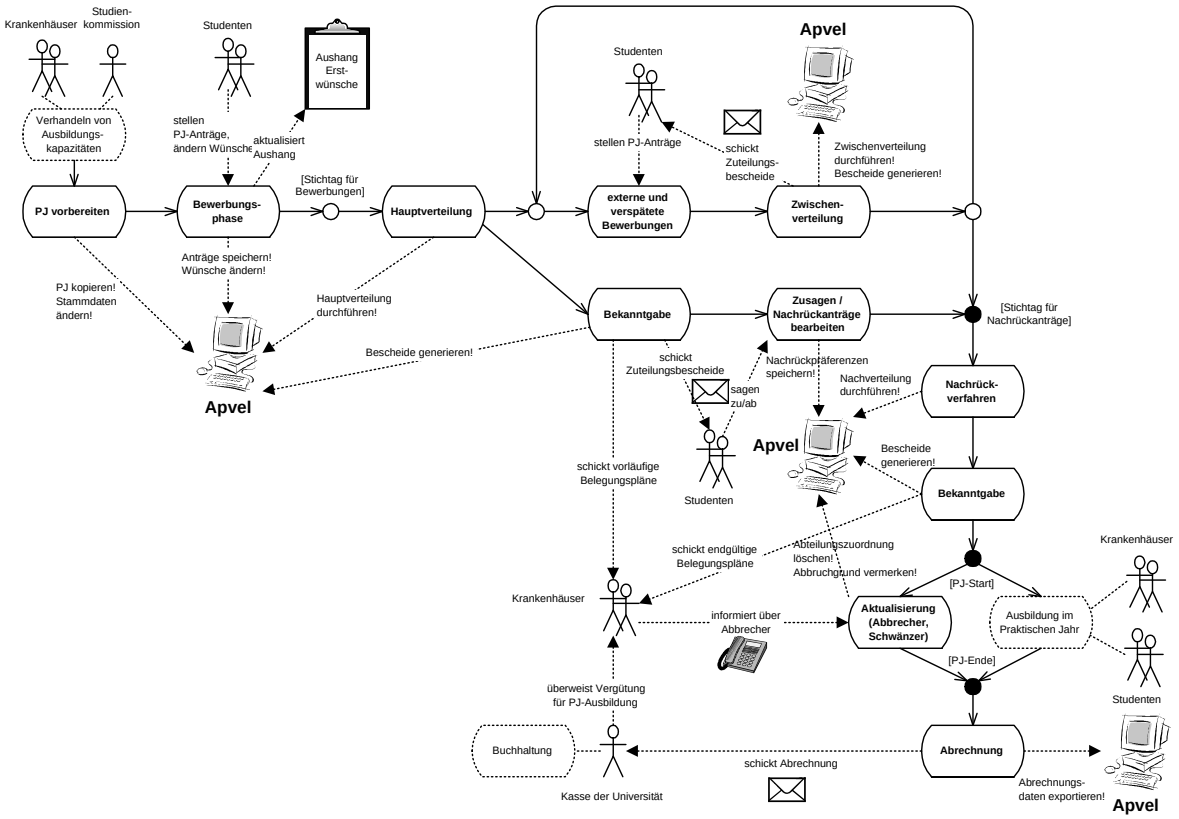


Abbildung 5.19.: Aktivitätsdiagramm für die Tätigkeiten des PJ-Büros

Zur Vorbereitung des Praktischen Jahres kopiert das PJ-Büro normalerweise die Stammdaten (Krankenhäuser, Abteilungen und Kapazitäten) des vergangenen Jahres und aktualisiert sie aufgrund der Ergebnisse der Verhandlungen mit den akademischen Lehrkrankenhäusern.

Mit Beginn der Bewerbungsphase treffen nach und nach schriftliche Bewerbungen von Medizinstudenten ein, die sukzessive ins System eingepflegt werden. Ab und zu wird dabei der Bewerberaushang aktualisiert, der angibt, welche Krankenhäuser von wieviel Studenten gewünscht werden. Da zusätzlich auch die Losnummern der Studenten mit angegeben werden, können sich die neuen Bewerber bereits ausrechnen, welche Chance ihr Erstwunsch besitzt und dementsprechend ihre Wünsche ändern.

Sobald der Stichtag für das Ende der Bewerbung verstrichen ist, lässt das PJ-Büro vom Apvel-System die Hauptverteilung durchführen, informiert alle Studenten durch generierte Serienbriefe über ihre Zuteilung und schickt den Lehrkrankenhäusern einen vorläufigen Belegungsplan.

Treffen daraufhin Absagen ein, so werden die Nachrückwünsche entsprechend ins System eingepflegt und nach Erreichen des Stichtags ein Nachrückverfahren durchgeführt.

Parallel zur Bekanntgabe und der Bearbeitung von Nachrückanträgen bzw. Zusagen treffen weiterhin verspätete oder externe Bewerbungen ein. Sobald hier eine entsprechende Anzahl zusammengekommen ist, werden auch diese ins System eingepflegt, als nachrangig markiert und gesammelt in einer Zwischenverteilung zugeteilt.

Das System erlaubt hier die Ausführung einer beliebigen Anzahl von Zwischenverteilungen, die jeweils alle noch nicht verteilten Studenten zuteilt. Auch die externen und verspäteten Bewerbungen haben die Chance, einen Antrag für das Nachrückverfahren zu stellen.

Nach der Bekanntgabe der Zuteilung treten die Studenten ihre Stellen an und das Praktische Jahr beginnt. Kommt es während des Jahres krankheitsbedingt oder aus anderen Gründen zum Abbruch oder Unterbrechung der Ausbildung, löscht das PJ-Büro im System alle betroffenen Abteilungszuordnungen und vermerkt den Abbruchgrund beim Studenten.

Am Ende des PJ lässt sich das PJ-Büro vom System die relevanten Abrechnungsdaten als Textdatei ausgeben und schreibt für jedes Krankenhaus eine entsprechende Kassenanweisung. Die Kasse der Universität überweist daraufhin die berechneten Geldbeträge an die jeweiligen Krankenhäuser.

Zum Vorgehen

Erweiterte Aktivitätsdiagramme dienen der Darstellung zentraler Aktivitätsbereiche aus einer agentenübergreifenden Sicht, die helfen soll, Zusammenhänge der Systembenutzung zu verdeutlichen. Es stellt eine Art Mischung bzw. Anreicherung der in der Kontextanalyse dargestellten Aktivitätsmuster mit kontextbezogenen Ausschnitten aus den in der Systemgestaltung kreierten Systemkooperationen bzw. Systeminteraktionen dar. Der Schwerpunkt liegt auf der beispielhaften kontextreichen Dokumentation der zentralen Abläufe und der damit verbundenen Interaktion mit dem System.

Aktivitätsdiagramme dienen nicht der Spezifikation von logisch/zeitlichen Reihenfolgen von Einzelschritten zur Erreichung eines bestimmten Ziels, sondern der systemischen Beschreibung des Arbeitskontexts aus verschiedenen Blickwinkeln bzw. reduziert auf bestimmte "Themengebiete", um sich in eine neue Arbeitswelt hineindenken zu können.

Mögliche Themengebiete, die zur Auswahl der darzustellenden Aktivitätsbereiche dienen können, sind beispielsweise die folgenden:

- ein bestimmter Akteur (wie z.B. das PJ-Büro in Abb. 5.19): In welchen Interaktionsbeziehungen steht der Akteur im Verlauf seiner allgemeinen Arbeitspraxis zum IS bzw. in welche Systemdialoge ist er verstrickt? Wie ist der Akteur durch das neue IS betroffen?
- ein bestimmtes Informationsobjekt des Systems und sein Lebenslauf: Untersuchung des Informationsflusses: Ursprung, Gebrauch und Senke. Beispiel: Rolle der Abteilungskapazitäten. Wer legt sie fest, wann werden sie geändert, wo werden sie gebraucht?
- ein bestimmter Vorgang: Welche Rolle spielt das System in einem bestimmten akteurübergreifenden Aktivitätsmuster (z.B. Geschäftsprozess)? In der Darstellung werden sowohl Interaktionen zwischen Akteuren als auch deren Handlungen kombiniert dargestellt. In Abb. 5.19 wird beispielsweise die Abwicklung der Organisation eines Praktischen Jahres beschrieben.
- ein bestimmter Sonderfall (im Apvel-Projekt könnte man hier z.B das Verfahren zur Behandlung verspäteter Bewerbungen umfassend darstellen)
- eine bestimmte Handlung eines Akteurs: Wie wird das IS von einem bestimmten Akteur innerhalb einer bestimmten Handlung verwendet? Wird die Handlung unterstützt oder behindert? Wie sieht es mit der gewonnenen Effizienz aus? Welche Aktivitäten führt der Akteur, welche das System, in welcher Reihenfolge aus? Im Apvel-Projekt könnte man hier z.B das Verfahren zur Durchführung des Nachrückverfahrens konkretisieren, indem man den Ablauf von sowohl system- als auch nutzerbezogenen Einzelaktivitäten beschreibt.
- eine typische Konversation: Was ist die Rolle des IS mit Bezug auf die Standardkonversationen? Welchen Platz nimmt es dort ein? Im Apvel-Projekt könnten hier beispielsweise

alle Aktivitäten und ihre Abfolge dokumentiert werden, die im Verlauf des Bewerbungsverfahrens eines externen Studenten zu beobachten sind.

Das Vorgehen in der Modellierung richtet sich nach folgenden Leitfragen:

Welche Aktivitätsmuster spiegeln die Rolle des Systems besonders gut wieder?

Welche Aktivitätsmuster werden durch das neue Softwaresystem am besten unterstützt, welche sind am meisten betroffen? Wie sehen die systemischen Handlungsräume konkret aus? Leitbild der Modellierung ist die Darstellung von Handlungsmöglichkeiten, nicht deren Festlegung! Untersucht werden hierzu sowohl lokale Handlungen von Akteuren als auch Kooperationen mehrerer Akteure.

Im Beispiel: Der vollständige organisatorische Ablauf zur Durchführung eines einzelnen Praktischen Jahrs aus der Sicht des PJ-Büros.

Welche Interaktionen mit dem System werden dabei durchgeführt?

Wo wird Systemwissen gepflegt? Welche Interaktionen des Systems beeinflussen die Einzelaktivitäten? Welche Aufgaben werden an welcher Stelle an das System delegiert? Welche Informationsflüsse zwischen System und Anwendern sind zu beobachten?

Im Beispiel: PJ kopieren, Anträge einpflegen, Abrechnungsdaten exportieren etc.

Welche Tätigkeiten sind von der Systemgestaltung am meisten betroffen?

Wie arbeiten die verschiedenen Nutzer mit dem Softwaresystem? Welche Tätigkeitsabläufe werden durch die gewählte Systemgestalt festgelegt? Was sind die typischen globalen Nutzungssequenzen?

Im Beispiel: Die konkreten Aktivitäten zur Verteilung der Studenten (ist im Beispieldiagramm eingebettet: Unterscheidung von Haupt-, Zwischen- und Nachverteilung).

Welche Interaktionen mit anderen Akteuren finden dabei statt?

Inwieweit ist die Kommunikation mit anderen Akteuren durch den Systemeinsatz betroffen? Welche Interaktionen können bzw. müssen wann stattfinden?

Im Beispiel: Die Kommunikation mit dem Studenten, dem Krankenhaus, der Kasse, etc.

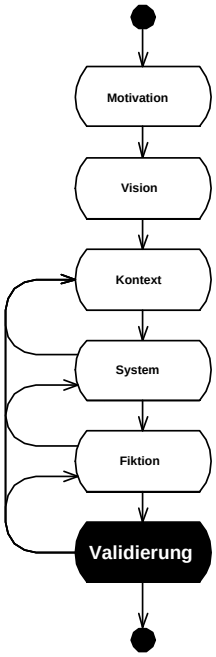
5.6.4. Zusammenfassung

Ziel der Fiktion ist eine kontextreiche Dokumentation der Rolle des neuen Softwaresystems. Diese Rolle wird zum Einen lokal durch akteurspezifische Nutzfälle, zum Anderen durch globale akteur- bzw. einzeltätigkeitsübergreifende Systemnutzungsszenarien dokumentiert.

Die Fiktion beschränkt sich in der Darstellung auf Ausschnitte der bereits in der Kontextanalyse modellierten Anwendungsumgebung, die sie nun mit Bezug auf die Systemnutzung verfeinert bzw. ergänzt und konkretisiert.

Die Fiktion ist damit ein spezialisiertes Kontextmodell, das die fachlichen Arbeitsprozesse im Umfeld der Nutzung des neuen Softwaresystems in den Vordergrund stellt und damit die Rolle des Systems in seiner globaleren Umgebung deutlich macht.

5.7. Validierung



Ziel: Überprüfung der Systemgestalt

In der Validierungsphase bewerten die Interessengruppen aufgrund der erstellten Modelle gemeinsam die Qualität des Gestaltungsergebnisses. Sowohl das IS selbst als auch seine Auswirkungen in der Arbeitsumgebung werden aufgrund der Ergebnisdokumente der Gestaltungs- und Fiktionsphase zusammen mit dem Kunden diskutiert, analysiert und validiert.

Neben einer Überprüfung auf vollständige Berücksichtigung aller Anforderungen impliziert die Validierung auch eine Beurteilung der Praktikabilität, des Kosten/Nutzen-Verhältnisses, der Risiken und eine Erwägung von Lösungsalternativen. Zur Beurteilung der Qualität einer gefundenen Lösung gehört desweiteren auch eine Untersuchung der Gesamttätigkeit der Systemnutzer und anderer Betroffener nach Maßgabe der humanen Arbeitsgestaltung (s. z.B [Uli94]) bzgl. des Entscheidungsspielraums, der Kommunikationssituation, den psychischen und physischen Belastungen, den zeitlichen Spielräumen, der Entscheidungsfreiheit, etc.

Die Validierung durch die Interessengruppen kann auf verschiedene Weise erfolgen, beispielsweise

- an Hand eines formales Review der Ergebnisdokumente durch die verschiedenen Interessengruppen unabhängig von den Entwicklern,
- durch Präsentation der Ergebnisse durch die Entwickler und anschließender gemeinsamer Diskussion bzw. Brainstorming-Workshop,
- durch einen gemeinsamen Walk Through der dokumentierten fiktiven Arbeitsprozesse, beispielsweise durchgeführt in Form eines Rollenspiels bzw. einer Simulation,
- evtl. unterstützt durch Mock Ups ([EK91]) oder funktionelle Prototypen ([BG91])

Bei der Evaluation der Ergebnisse einer vorausgehenden Gestaltungs- und Fiktionsphase werden unweigerlich neue Ideen zur Umgestaltung artikuliert. Sind diese so weittragend, dass sie

nicht spontan in den bisherigen Entwurf integriert werden können, kann eine neue Iteration des Gestaltungsprozesses notwendig werden. Sowohl die Kontextanalyse, die Systemgestaltung als auch die Fiktion sind dann potentiell erneut durchzuführen, bevor ein neuer Versuch der Validierung erfolgen kann.

Wird das dokumentierte Systemprofil hingegen in seiner dokumentierten Form von allen Interessengruppen akzeptiert, ist das Ziel des ersten Abschnitts der Softwareentwicklung und damit auch das Ende des Anwendungsgebiets von KoA$_{L}$A erreicht: als Ergebnis des Kontextdesigns liegt nun in Form des System-Akteurs und seines Interaktions- bzw. Tätigkeitsprofils eine Definition der Anforderungen an das zukünftige IS vor. Im nächsten Abschnitt der Softwareentwicklung, der "Softwarespezifikation" gilt es nun, ein konkretes Softwaresystem zu definieren, das diese Anforderungen erfüllt.

5.8. Der Übergang zur Softwarespezifikation

Ziel: Entwurf der technischen Systemgestalt

Nachdem mit KoA$_{L}$A nun die allgemeine Motivation für das Entwicklungsprojekt (="business requirements") und das konkrete "Problem" – d.h. das gewünschte Informationssystem aus Sicht der Anwender und im Hinblick auf seine Einbettung in den Arbeitskontext – definiert wurde (="user requirements"), ist nun eine geeignete softwaretechnische Lösung zu finden und exakt zu spezifizieren (="software requirements").

Für den letzten Schritt der Spezifikation bieten sich dabei aktuelle Softwareanalysemethoden wie z.B. RUP oder Fusion an ([BR97], [CAB^{+}93]). Um einen fließenden und nutzerorientierten Übergang von der bisher anthropomorph dokumentierten Idee eines technischen Akteurs zur konkreten Ausgestaltung eines technischen Softwaresystems zu erleichtern, bietet es sich an, in der Spezifikation sogenannte Konzeptionsmuster zu verwenden (siehe [Zue98]). Ein Konzeptionsmuster ist ein Muster, welches zur Interpretation und Gestaltung von tatsächlichen oder antizipierten Anwendungssituationen und Softwaresystemen verwendet werden kann. Züllighoven unterscheidet im sogenannten "WAM"-Ansatz beispielsweise die Konzeptionsmuster "Werkzeug", "Automat" und "Maschine", um Systemkomponenten in einer verständlichen, intuitiven und kompakten Weise darzustellen, die gleichzeitig eine Validierung durch die Anwender unterstützt.[13]

Die Ergebnisse der KoA$_{L}$A-Modelle stellen hierzu umfassende Vorarbeiten dar, so dass der Aufwand für eine weitere Konkretisierung der Anforderungen in Form der Softwarespezifikation weitgehend reduziert ist.

KoA$_{L}$A dokumentiert aus kommunikations- und damit anwendungsorientierter Sicht neben den vom System durchzuführenden Aktivitäten auch alle seine – möglicherweise auch sehr abstrakt spezifizierten – Interaktionen, an denen es teilnimmt bzw. teilnehmen kann. Diese beinhalten sowohl die Begriffswelt als auch alle "Aufgaben", die Anwender an das IS stellen.

[13] Konzeptionsmuster dienen dazu, sich in Nutzungssystemen zu orientieren. Sie erläutern, wie Nutzungssituationen von Anwendern interpretiert werden können bzw. sollen und stellen eine Begrifflichkeit dazu bereit. Sie dienen dazu, die Anwendungswelt mit zukünftigen Benutzern zu erfassen und zu interpretieren. Konzeptionsmuster sind ausreichend nahe am Software-Entwurf, um einen leichten Übergang hierzu zu ermöglichen.

In der Softwarespezifikation wird nun konzeptionell die äußere technische Systemgestalt spezifiziert, d.h die konkreten vom System zu verwaltenden – wenn auch nur konzeptionell konkretisierten – Datenelemente, die zu realisierenden technischen Funktionen wie auch die konkret zu implementierende Benutzerschittstelle (Layout, Komponenten der Oberfläche, Einzelinteraktionen etc.).

In der Spezifikationsphase ist zu entscheiden, wie die spezifizierten Interaktionen und abstrakten Kooperationen eines Benutzers mit dem System nun konkret ausgestaltet werden. Eine Interaktion ist eine besondere Art der Kooperation: Alle Beteiligten (Sender und Empfänger) haben ihren eigenen Anteil dazu beizutragen. Die genaue Aufgabenverteilung, d.h. z.B. wie etwas wann genau angewählt, ausgegeben oder eingegeben werden muss, ist noch offen. Erst mit dem Entwurf der Benutzeroberfläche und auch mit dem Entwurf der abstrakten statischen Datenstruktur (konzeptionelles Datenmodell) wird der genaue Ablauf der Interaktion festgesetzt. Bisher waren nur die Randbedingungen (Anforderungen, Gesetze, Vereinbarungen etc.) und das Ziel der Interaktion unter Benutzung der standardisierten Begrifflichkeit beschrieben. Nun geht es um die konkrete Ausgestaltung bzw. das Medium der Interaktion und die Repräsentation der kommunizierten Inhalte.

Alle Konzepte, die von Interaktionen oder Funktionen des IS betroffenen sind, sind Kandidaten für das vom System zu verwaltende Wissen. Dieser Ausschnitt der Begriffswelt kann somit zum konzeptionellen Datenmodell verfeinert bzw. transformiert werden (z.B. ER-Modellierung bzw. Klassenmodelle), das nun konkrete Definitionen der Entitäten unter Zuhilfenahme von elementaren Datentypen durchführt. Bei dieser Transformation wird die Anzahl der Attribute nun möglichst auf das notwendigste reduziert und ihre Redundanz weitgehend eliminiert. Redundante, d.h. berechenbare Aspekte werden in der Regel zu funktionalen Anforderungen an das zu realisierende System. Alle Beziehungen werden explizit spezifiziert und in ihren Kardinalitäten bestimmt. Virtuelle Konzepte, die allein durch die Oberfläche vermittelt werden, wie z.B. abstrakte Sichten oder temporäre Daten, finden hingegen Eingang in die Spezifikation der Benutzerschnittstelle und werden beispielsweise zu Boundary-Objekten nach RUP[JBR99].

Abb. 5.20 zeigt als Beispiel die Entity-Klassen eines Analysemodells nach RUP.

Bei der Abbildung der in der Systembegriffswelt verwendeten Konzepte auf ein konzeptionelles Datenmodell fallen eine Reihe von Entwurfsentscheidungen.

Ein paar Beispiele aus dem Apvel-Projekt:

- Abteilungen bekommen ein ”Reserviert”-Flag spendiert, damit der Benutzer selbst festlegen kann, welche Abteilungen in der Zwischenverteilung (für verspätete und externe Bewerber) ausgelassen werden sollen, damit ”brave” Ulmer in der Nachverteilung noch eine Chance bekommen
- Der Antrag bekommt ein Attribut ”nachrangige Behandlung”, damit die Benutzer selbst entscheiden und festlegen können, wer nachrangig verteilt werden soll
- Der Antrag bekommt eine ”Fachfolge” spendiert, in der der Benutzer für bestimmte Anträge die Reihenfolge der Fachart festlegen kann (Alleinerziehende etc.)
- Krankenhäuser und Abteilungen bekommen ein ”Aktiv”-Flag spendiert, damit der Benutzer festlegen kann, dass sie in der aktuellen Zuteilung nicht mehr benutzt werden

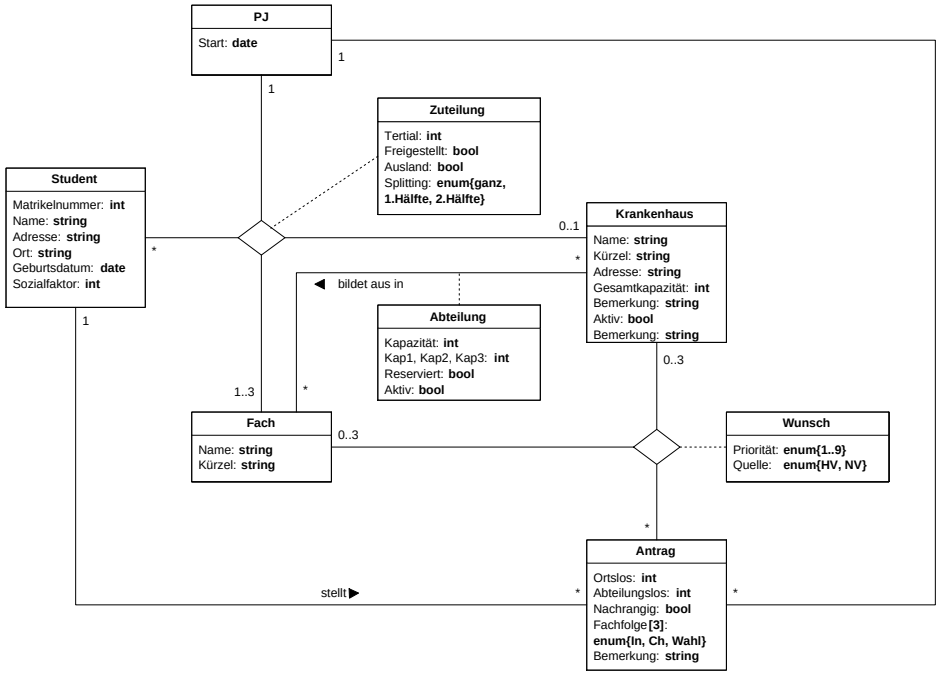


Abbildung 5.20.: Ein Konzeptionelles Datenmodell für das Apvel-System

sollen (statt eine echte Historisierung für die Stammdaten einzuführen!)

- Anträge und Krankenhäuser bekommen ein Bemerkungsfeld (nützlich, um z.B. die Auslandsaufenthalte genauer zu vermerken)
- Die ganzen Beziehungen werden konkretisiert und nun mit Attributen und Kardinalitäten versehen:
  - Studenten, die unterbrechen, werden in ihrer Zuteilung mit ”freigestellt” markiert; diejenigen, die Teile im Ausland verbringen, mit ”Ausland”
  - Kapazitäten der Abteilungen können tertialspezifisch ”getrimmt” werden
- Was ist das initiale Wissen des Systems? Sind ausgezeichnete Beispielinstanzen (“Universitätsklinik Ulm”, “Innere Medizin”, “Chirurgie” etc.) bereits vordefiniert oder müssen sie vom Anwender eingepflegt werden?
- Aus Gründen der Vereinfachung der semesterübergreifenden Verwaltung von Studenten, die an mehreren PJ teilnehmen bzw. sich bewerben, wird nun auf Ebenen des Datenmodells doch noch explizit zwischen Student und seinem Antrag bzw. seinen Anträgen unterschieden.

Die Systemaktivitäten werden in Form von Systemfunktionen bzw. Objekt-Methoden spezifiziert, die normalerweise Auswirkungen auf den durch das konzeptionelle Datenmodell modellierten Systemzustand besitzen und nun auch in ihrem Verhalten (Reihenfolge der Benutzung) und vor allem ihrer Funktionalität vollständig definiert werden. Geeignete Verfeinerungen der

Funktionen werden durchgeführt, um die SW-Spezifikation zu präzisieren bzw. zu formalisieren.

Die wesentlichste gestalterische Leistung der Softwarespezifikationsphase ist allerdings im Entwurf der Benutzerschnittstelle zu sehen, die nun vollständig zu spezifizieren ist. Im Falle von grafischen Benutzeroberflächen sind hier nun alle verwendeten Formulare in ihrer spezifischen Ausgestaltung zu definieren, d.h. es ist ihr grafisches Layout festzulegen, die verwendeten grafischen Komponenten und ihr Interaktionsmöglichkeiten bzw. -verhalten etc.

Die bereits mit KoA$_{\text{L}}$A identifizierten Kommunikationsmuster zwischen Anwendern und dem ausgestalteten IS bilden eine passende Grundlage für die verfeinerte Beschreibung von typischen Systemnutzungssequenzen, z.B. in Form von konkreten Use-Cases, die auch als Ausgangspunkt für die weitere Analyse und Verfeinerung der Softwareanforderungen Verwendung finden können (siehe z.B. [JBR99]). Hier kann z.B. spezifiziert werden, wie sich ein konkreter Anwendungsfall mit Bezug zu den realen Dialogen und anderen Oberflächenkomponenten der Benutzeroberfläche realisieren lässt, welche Interaktionen konkret notwendig sind, welche Aktivitäten ausgelöst werden und welche Dialogwechsel durchgeführt werden. Die abstrakten Ablaufbeschreibungen des Nutzungsprofils werden nun also im Bezug auf eine reale Benutzerschnittstelle verfeinert und konkretisiert.

Mit der technischen Ausgestaltung eines konkreten Softwaresystems mittels der Spezifikation der Benutzerschnittstelle, des Systemzustandsraums basierend auf einem konzeptionellen Datenmodell und der formalen Festlegung aller Systemfunktionen wird *eine* konkrete softwaretechnische Lösung beschrieben, die die Rolle des in KoA$_{\text{L}}$A entworfenen Systemakteurs ausfüllen soll. Es wird ein konkretes IS spezifiziert, das die zukünftige Arbeitspraxis möglich machen soll, wie sie als Gestaltungsergebnis von KoA$_{\text{L}}$A in der Phase des Kontextdesigns entworfen worden ist.

Das Ergebnis der Softwarespezifikation kann dabei auch eine Iteration der Kontextgestaltung notwendig machen. Sieht der Oberflächenentwurf in unserem Fallbeispiel beispielsweise vor, das die Liste der Sozialfaktoren frei pflegbar sein soll, so wird hierdurch das Wissen des Systems erweitert. Neues Systemwissen hat jedoch unweigerlich Einfluss auf die Systemfunktionalität, da das neue Wissen auch ins System gelangen bzw. aktualisiert werden muss etc. Diese Auswirkungen sind in einer erneuten Iteration der Kontextanalyse, Gestaltung und Fiktion zu berücksichtigen.

Einen ganz eigenen Entwurfsbereich bei der Softwarespezifikation stellt die Überlegung dar, welche zukünftigen Änderungen an Systemkomponenten bzw. -eigenschaften eingefordert werden könnten. Es steht im Raum, welche zukünftigen Entwicklungen vorwegzusehen sind. Zusätzlich zu den in Form von Sonderfällen definierten bekannten möglichen Ereignissen (z.B. Euro-Umstellung, wahrscheinlichen gesetzliche Änderungen etc.), auf die im Nutzungskonzept und damit in der Systemgestaltung bereits Rücksicht genommen wurde, können hier vorausschauend auch spezifische Konfigurationsmöglichkeiten im System eingebaut werden, die weitgehende Flexibilität auch in solchen Fällen erwarten lassen, die noch nicht klar abzugrenzen sind. Im Projekt Apvel kann es hier beispielsweise sinnvoll sein, die Pflichtfächer dynamisch festlegen zu können, um so z.B. dem Fall vorzugreifen, dass in einer zukünftigen Studienordnung für das Praktische Jahr an Stelle der Chirurgie nun die Anästhesie zum Pflichtfach gemacht wird.

Im Anschluss an die Spezifikation der Software wird das Softwaresystem programmtechnisch

entworfen, d.h. in seiner internen technischen Gestalt bzw. “Architektur” festgelegt (Datenstrukturen; Datenbanktabellen; Komponenten, Module oder Klassen; Dialoge bzw. Formulare; Ablaufstrukturen z.B. in Form von Objekt-Interaktionsdiagrammen etc.) und im Anschluss – zumindest prototypisch – realisiert und in Betrieb genommen.

Mit dem Verstehen und dem Umgang mit dem System werden sich die eingeführte Begrifflichkeit und die zur Verfügung stehenden Bedienkonzepte etablieren und weiterentwickeln, etablierte Handlungsmuster werden sich anpassen und mit unvermeidlichen graduellen Veränderungen im organisatorischen Umfeld werden sich mit der Zeit auch wieder neue Ideen zur Umgestaltung entwickeln. Es beginnt ein neuer Zyklus im Verstehen und Gestalten, der sich in neuen Revisionen der Organisations-, Kooperations und Wissensmodelle niederschlagen und damit zu einer neuen optimierten Systemgestalt führen wird. Das Ergebnis der Entwicklung ist gleichzeitig ihr Anfang.

6. Zusammenfassung

Die vorliegende Arbeit stellt eine völlig neue modellbasierte Vorgehensweise zur partizipativen und systemischen Gestaltung betrieblicher Informationssysteme im Kontext ihrer praktischen Anwendungsumgebung vor. Sie beschreibt einen möglichen Weg zur *Entwicklung* von Anforderungen, der in dieser Form einzigartig ist.

Die dargestellte Vorgehensweise trägt den Namen "Koala" und gibt Antwort auf eine Frage, die im Umfeld von Softwareentwicklungsprojekten oft gehört wird – zumal etablierte Analyse-Methoden sie in der Regel weitgehend vernachlässigen : "Wie kommt man eigentlich zu den Anforderungen, die man zu spezifizieren hat?"

Die vorliegende Arbeit

- ergänzt die etablierten Vorgehensmodelle um eine neue ausgezeichnete Phase der Kontextgestaltung, die noch vor der Softwaregestaltung, d.h. der Spezifikation des zu realisierenden Softwaresystems angesiedelt ist
- definiert eine eigene spezifische und iterative Vorgehensweise für diesen Gestaltungsprozess
- stellt neu entwickelte Modellierungstechniken zur Verfügung, um die Anwendungswelt systemisch, d.h. ganzheitlich zu modellieren und so dem Diskurs bzw. der partizipativen Gestaltungstätigkeit zugänglich zu machen
- spezifiziert neuartige diskursunterstützende und modellunabhängige Diagrammtypen und Modellierungskonzepte zur anwenderfreundlichen grafischen und/oder textuellen Dokumentation von Gestaltungsmodellen
- gibt einen nahtlosen Übergang zur Phase der Softwarespezifikation an, die mit herkömmlichen objektorientierten Entwicklungsmethoden durchgeführt werden kann
- definiert ein neues Rahmenwerk zur Einordnung von Methoden und Techniken in Bezug zur behandelten Gestaltungsebene (Handlungs-, Nutzungs- oder Softwaresystem), Gestaltungsphase (Motivation, Vision, Kontextanalyse, Systemgestaltung, Fiktion oder Softwarespezifikation) und Gestaltungsbereich (Wissen, Organisation oder Kooperation)

Während klassische Softwareentwicklungsmethoden stark von der Entwicklungssicht beeinflusst sind und sich auf die möglichst formale Modellierung des zu entwickelnden Softwaresystems beschränken, erweitert KoA$_{\text{L}}$A die betrachtete Gestaltungsebene auf das umfassende soziotechnische System. KoA$_{\text{L}}$As Modelle dienen nicht der formalen Spezifikation technischer Artefakte sondern sind vor allem Mittel und Medium der Kommunikation bzw. Gestaltung.

KoA$_{\text{L}}$A hilft dabei, die Anforderungen an das Informationssystem möglichst frühzeitig und in enger Beziehung zum Anwendungskontext zu klären, um so die Akzeptanz und Passgenauig-

keit des Softwaresystems in der Arbeitspraxis sicherzustellen, ohne teure Prototyping-Zyklen oder Wartungsaufwände in Kauf nehmen zu müssen.

Die folgenden Abschnitte fassen das Vorgehen und die Modellierungstechnik von KoA$_L$A noch einmal zusammen und stellen wichtige Merkmale heraus, die diese Vorgehensweise von existierenden Entwicklungsansätzen unterscheiden. Abschliessend werden einige potentielle Kritikpunkte diskutiert und es wird ein Ausblick auf zukünftige Weiterentwicklungsmöglichkeiten gegeben.

6.1. Das Vorgehen

Ausgehend von aktuellen Kooperationsbeziehungen werden durch eine Analyse der typischen Konversations- und Aktivitätsmuster die Tätigkeitsprofile aller Akteure aufgedeckt. Nach einer Bestimmung der zukünftigen Nutzergruppen und deren Anforderungen wird das IS als neuer technischer Akteur mit eigenem Tätigkeitsprofil und spezifischem Interaktionsverhalten entworfen und auf Modellierungsebene in diesen Arbeitskontext eingepasst. Durch Modellierung und Diskussion der sich neu ergebenden zentralen Aktivitätsmuster wird die gewählte Gestaltung des IS validiert und iterativ überarbeitet.

KoA$_L$A geht von der Prämisse einer möglichen konstruktivistischen Modellbildung a priori aus, welche die optimale Gestaltung eines konkreten IS zum Ziel hat. KoA$_L$A verfolgt dabei einen systemischen Ansatz, der nicht eine objektive Abbildung oder Analyse des Organisationskontexts versucht, sondern diesen aus subjektiver Sicht partizipativ mit den Betroffenen zusammen iterativ entwirft und beschreibt. Ziel dabei ist es, zuerst aus Praxissicht ein optimal "passendes" Informationssystem zu skizzieren, bevor in der folgenden Phase das Softwaresystem entwicklungsorientiert bzw. objektivistisch konkretisiert wird, indem die "technischen" Anforderungen analysiert und möglichst formal spezifiziert werden.

KoA$_L$A ist ein integrativer Ansatz zur Gestaltung von Informationssystemen, der diese Tätigkeit als iterativen Prozess begreift und ihn in einen größeren Rahmen – die Organisationsentwicklung – stellt. Die Idee der Einbeziehung des Organisationskontexts in die Gestaltung eines IS ist nicht neu. So beschreibt [WK78] bereits die Notwendigkeit der benutzerorientierten Systemgestaltung, die nicht nur eine aufgabenlogische Sichtweise einnimmt, sondern auch das soziale Verhalten konkreter Systemnutzer von Anfang an in den Gestaltungsprozess miteinbezieht. Stichhaltige Vorgehensweisen, die diese Idee in konkrete methodische Schritte umsetzen, sind allerdings rar oder bleiben sehr schwammig. Zur Diskussion siehe Kap. 3.

Den Weg zu den Anforderungen sieht KoA$_L$A im Gegensatz zu den meisten "zeitgenössischen" Ansätzen, die von einer Anforderungs-Analyse (zerlegend, konkretisierend) sprechen und dementsprechend den klassischen rational-analytischen Ansatz vertreten, primär als kreativen Gestaltungsprozess, der in einem systemischen Umfeld stattfindet (integrierend, abstrahierend). Anforderungen werden nicht nur "erhoben" und analysiert, sondern partizipativ entwickelt und automatisch mit Bezug zum Arbeitszusammenhang validiert. Damit verbessert sich die Qualität bzw. die Akzeptanz des Entwicklungsergebnisses, da in der Anforderungsanalyse nun auch der praktische Arbeitszusammenhang methodisch modelliert, untersucht und diskutiert wird, nicht nur das Softwaresystem selbst. KoA$_L$A fördert und expliziert den kreativen Gestaltungsprozess, den jede Anforderungsanalyse inhärent impliziert.

Gestaltung wird als Prozess angesehen, nicht als Produkt, das mit einer bestimmten Methode "herzustellen" ist. Der Gestaltung zugrundegelegt ist dabei ein Qualitätsbegriff, der die Effektivität, d.h. die Relevanz, Eignung und Adäquatheit der Softwarelösung in den Vordergrund stellt, und nicht deren Effizienz oder formale Korrektheit. Im Gestaltungsprozess spielen insbesondere Handlungsspielräume, Kommunikations- und Kooperationsmöglichkeiten, Qualifizierung und Eigenverantwortlichkeit eine Rolle, da Arbeitseffizienz insbesondere von der Fähigkeit der Menschen abhängt, wie sie mit Ausnahmesituationen in komplexen Situationen zurechtkommen [Sac95]. Christiane Floyd fordert in diesem Zusammenhang eine notwendige Paradigmenverschiebung: von der Produktorientierung, die als Ergebnis der Entwicklung das System selbst sieht, zur Prozessorientierung, die als eigentliches Ergebnis der Entwicklung die veränderten Arbeitsprozesse ansieht, in deren Umfeld das neue System benutzt wird [Flo92].

Britta Schinzel und Karin Klein [SK01] unterscheiden daher folgerichtig auch drei mögliche Richtungen einer notwendigen Professionalisierung der Informatik als Wissenschaft:

mathematisch formal: Das Computerprogramm als mathematischer Ausdruck, dessen Korrektheit nachweisbar ist. Der Informatiker als Mathematiker.

ingenieurwissenschaftlich: Das Softwaresystem als technische Funktionseinheit, die im Sinne der Preistauglichkeit eine optimale Problemlösung darstellt. Der Informatiker als technischer Problemlöser.

gesellschaftlich kritisch: Softwaresysteme als adäquate, alltagstaugliche Werkzeuge und Arbeitsumgebung für Menschen. Der Informatiker übernimmt gesellschaftliche Verantwortung.

KoA$_{\text{L}}$A ist dem dritten Bereich zuzuordnen.

Aktuelle Softwareentwicklungsmethoden berücksichtigen den Prozess der Gestaltung von Anforderungen nur sehr unzureichend. Sie antizipieren in der Regel eine klare Problemstellung, die mathematisch/wissenschaftlichen Methoden zugänglich ist. Wenn sie den organisatorischen Kontext überhaupt berücksichtigen, so behandeln sie diesen Bereich ebenso wie ein technisches und damit formal fassbares System.

6.2. Die Modellierung

Der Gestaltungsprozess geht vom Bereich menschlichen Handelns aus, von der Anwendungswelt. Zur Darstellung dieses Kontexts setzt KoA$_{\text{L}}$A in großem Umfang Beschreibungsmodelle ein, die sowohl grafisch als auch freitextlich dokumentiert werden.

KoA$_{\text{L}}$A stellt drei Modell- und fünf Diagrammarten zur Verfügung, deren zugrundeliegenden orthogonalen, d.h. modell- und diagrammübergreifend verwendbaren Modellierungskonzepte in ihrer Anzahl weitgehend minimiert und in ihrer Form und Bedeutung weitgehend an UML angelehnt sind.

Die relevante konzeptionelle Begriffswelt des Anwendungsbereichs wird im *Wissensmodell* abgebildet. Das *Organisationsmodell* dient der Dokumentation der statischen Organisationsstrukturen und trennt dabei die Beschreibung des Aufbaus der Organisation in Form der Organisationsstruktur von der Beschreibung des dynamischen Potentials seiner Akteure in

Form der Tätigkeitsprofile. In der Modellierung der dynamischen Kooperationsbeziehungen schließlich unterscheidet KoA$_L$A zwischen einer statischen Sicht, der Kooperationsstruktur, und zwei unterschiedlichen dynamischen Sichten, den Konversations- und Aktivitätsmustern.

Die Modelle stellen primär dar, wie mit dem System gearbeitet werden kann, nicht wie das System selbst arbeitet oder wie mit ihm gearbeitet werden soll. Die Beschreibung der Problemwelt geschieht dabei nicht in einer maschinen- sondern in einer menschenorientierten Weise. Insbesondere wird dabei das neue Softwaresystem anthropomorph als menschenähnlicher Akteur mit eigenem Aufgabenbereich und Interaktionsverhalten modelliert, wodurch die Kommunikation zwischen Entwickler und Interessengruppen verbessert und die konzeptionelle Evaluation erleichtert wird. Bei der Modellierung der Arbeitspraxis wird ein deskriptiver Ansatz verfolgt, bei dem vor allem der Kontext der beobachtbaren Aktivitäten im Vordergrund steht, also die Ziele, Wünsche und Anforderungen von selbstgesteuerten menschlichen Akteuren und ihre gegenseitigen Kommunikationsbeziehungen und Verpflichtungen.

In den vorherrschenden entwicklungsorientierten Methoden, die zur Modellierung menschlicher Aktivitätssysteme Verwendung finden, herrscht hingegen eher eine operationale Sichtweise vor, die Menschen als fremdgesteuerte ausführende Funktionseinheiten modelliert (Menschen als "Objekte", die nach einem festen Schema "Funktionen" ausführen).

Die erstellten Modelle stellen Materialien des Gestaltungsdialogs dar, die fortlaufend weiterbearbeitet werden. Die Modelle sind also keine verifizierbaren Abbilder von Realitäten, sondern gemeinsam konstruierte Artefakte, die den Abstimmungsprozess strukturieren, befruchten und dokumentieren sollen. Der Gestaltungs/Evaluationszyklus beginnt bereits auf Modellebene und nicht erst auf Realisierungsebene (z.B mittels explorativer oder evolutionärer Prototypen).

KoA$_L$As Modellierung des technischen Softwaresystem-Akteurs ermöglicht einen nahtlosen Übergang zur Softwarespezifikation, da sowohl UML-ähnliche Modellierungskonzepte verwendet werden, als auch die Abstraktionsebene so gewählt ist, dass der nachfolgende Softwareentwurf einen weitgehend natürlichen Konkretisierungsschritt darstellt.

6.3. Abgrenzung

In den folgenden Abschnitten werden nun ergänzend zu Kapitel 3 einige direkt konkurrierende Entwicklungsansätze bzw. Techniken diskutiert und KoA$_L$A gegenübergestellt.

Business Modeling

Die Geschäftsprozess-Modellierung (z.B. mit Hilfe von Prozessketten nach Aris [Sch99a]) bzw. die Modellierung von Anwendungsumgebungen in etablierten Entwicklungsansätzen (z.B. in Form des "Business Modeling" nach RUP [BR97]) beschäftigen sich wie KoA$_L$A auch mit der Modellierung von Arbeitskontexten.

Aufgrund des vorherrschenden rational-analytischen Leitbildes basiert die Modellierung in diesen Ansätzen im Allgemeinen auf rein "technischen" Konstrukten der objektorientierten

Entwicklungswelt wie z.B. Use-Cases, Klassen, Objekten, Zustandsautomaten, Interaktionsdiagrammen etc. In der Darstellung von Kommunikationsvorgängen konzentrieren sich diese Ansätze eher auf Syntax und physikalische Struktur der Kommunikationsbeziehungen als auf deren Inhalt (siehe z.B. [AHL92]).

KoA$_L$A bietet hingegen Konstrukte "höherer Ordnung", die dem soziotechnischen Charakter einer Organisation angemessener sind. Zur Modellierung von Organisationen stehen Konzepte wie z.B. Ziele, Aktivitätsbereiche, Begriffe oder Konversationen zur Verfügung. KoA$_L$A ersetzt die konkrete Prozessbeschreibung durch eine Spezifikation der verfolgten Ziele bzw. der gegebenen Aufträge (Motivation) und ihres Kontexts (z.B. in Form von Interaktionsbeziehungen und Erwartungshaltungen).

Nach Paech werden in der Geschäftsprozessmodellierung typischerweise vor allem exemplarische und deterministische Aktivitätsfolgen dokumentiert [Pae00]. Eine abstraktere oder umfassendere Darstellung wie z.B. die Definition von Aktivitätstypen oder die Beschreibung vollständigerer Aktivitätsbereiche unterbleibt oft. Nicht ohne Grund, da ein deterministischer analytischer Ansatz zur Definition von Verhalten sozialer Systeme immer nur Stückwerk bleiben muss.[1] KoA$_L$A ist hier ein umfassenderer Ansatz, der auch höhere Abstraktionsebenen, Auslassungen und Nichtdeterminismen erlaubt.

In den Ereignisprozessketten (bestehend aus Funktionen, Ereignissen, Verknüpfungsoperatoren) von Aris werden die Funktionen immer lokal einzelnen Akteuren zugeordnet [Sch99a]. In der Beschreibung von Aktivitätsfolgen ist dies nach Paech jedoch höchstens auf der Ebene einzelner Arbeitsschritte möglich und sinnvoll ([Pae00], S.115). Die ereignisgesteuerten Prozessketten sind damit wie auch die meisten anderen verbreiteten diagrammatischen Modellierungsmethoden (wie z.B. die Aktivitätsdiagramme von UML) eher für die Modellierung des Nutzungssystems als für die echte Geschäftsprozessmodellierung geeignet.

KoA$_L$A modelliert menschliche Aktivität, indem die zentralen Kooperationsbeziehungen dokumentiert werden. Hauptaugenmerk liegt dabei auf den Variationsmöglichkeiten, die nur aus dem inhärenten Kontext verstanden werden können. Menschliche Akteure besitzen einen freien Willen und gehen daher als "irrationale", d.h. nichtdeterministische, Elemente in die Modellierung ein. Oberstes Gebot ist die Dokumentation von Aktivitätskontexten. Die eigentlichen Tätigkeiten sind nur in zweiter Linie interessant, und nur dann, wenn sie Ausgangspunkt für die Festlegung weiterer Kontexte sind (z.B. Systemaktivitäten auslösen). Die Hauptknoten des Aktivitätsdiagramms (Aktivitäten, Interaktionen) stellen folgerichtig Ereignisse dar, die auftreten können, aber nicht müssen.

Die Geschäftsprozessmodellierung hingegen konzentriert sich auf die präskriptive Dokumentation zentraler wiederkehrenden Aktivitätsmuster ("Workflows") mit dem Ziel, diese möglichst weitgehend infrastrukturell zu unterstützen bzw. zu automatisieren. Dazu gehört neben der Identifikation und Beschreibung aller Einzelaktivitäten vor allem eine möglichst formale Festlegung des Kontrollflusses ("Wann wird wohin verzweigt?"). Hier spielt die Kommunikation – als notwendiger Datenfluss – nur eine untergeordnete Rolle. Oberstes Gebot ist die möglichst stringente, d.h. deterministische Festlegung von Aktivitätsabläufen. Die Geschäftsprozessdiagramme stellen folgerichtig die Aktivitäten und deren logische Abfolge in den Vordergrund, wie sie von zuständigen Akteuren durchzuführen sind (Arbeitsanweisungen, Algorithmen).

[1] neuere Ansätze der Geschäftsprozessmodellierung weichen den deterministischen analytischen Ansatz auf und unterstützen die spontane Abweichung vom vordefinierten Prozess

Bei der Geschäftsprozessmodellierung sind die definierten Prozesse die Essenz und das Ergebnis der Anstrengung. Im Business Process Engineering werden aktuelle Aktivitätsmuster (Geschäftsprozesse) identifiziert, modelliert, analysiert, simuliert oder optimiert. Ziel ist ein Prozessredesign zur Optimierung von "Leistungskennziffern".

In KoA$_L$A hingegen bilden diese Aktivitätsmuster nur den Denkrahmen für die kreative Ausgestaltung des System-Akteurs als neuer Organisationseinheit. KoA$_L$A bietet mit den Aktivitätsmustern einen strukturierten Ausgangspunkt für den kreativen Gestaltungsprozess an, anstatt damit ein Endergebnis möglichst formal und vollständig beschreiben zu wollen. Im Vordergrund steht nicht der globale betriebswirtschaftliche Nutzen eines bestimmten Geschäftsprozesses, sondern das lokale motivationale Aktivitätspotential und seine Realisierungsmöglichkeiten, d.h. der direkte Nutzungskontext des neuen Systems. Hauptaugenmerk von KoA$_L$A liegt nicht auf dem globalen Ablauf ("top/down-Ansatz"), sondern der lokalen Optimierung der Gebrauchstauglichkeit von Softwaresystemen ("bottom/up-Ansatz").

Use-Cases

Das Konzept des textuell beschriebenen Anwendungfalls – genannt "use case" – hat mittlerweile in viele Entwicklungsansätze erfolgreich Einzug gehalten.

Ein Use-Case stellt eine bestimmte Art der Softwaresystemnutzung dar, also eine bestimmte – meist zielorientierte – lineare Sequenz von Einzeltätigkeiten, die in einem fachlich logischen Zusammenhang stehen und die sich durch einen spezifischen Verlauf von Interaktionen mit dem System auszeichnet.

Ein Use-Case lässt sich auch in KoA$_L$A modellieren, und zwar ohne dass hierfür ein eigenes Modellierungskonzept eingeführt werden muss: ein Use-Case wird als besonderer Fall einer Aktivität modelliert. Das Konzept der Aktivität geht jedoch über den sehr eng gefassten Abbildungsraum eines Use-Cases hinaus. Eine Aktivität nach KoA$_L$A stellt einen allgemeinen Aktivitätsbereich dar und definiert darum einen bestimmten Typ bzw. eine bestimmte Kategorie von Aktivitätssequenzen – alle, die zum definierten Aktivitätsziel führen. Mit entsprechender Konkretisierung bzw. imperativer Beschreibung einer KoA$_L$A-Aktivität kann natürlich auch ein Use-Case beschrieben werden. Der mögliche Gegenstandsbereich einer Aktivität jedoch ist weit größer als der eines Use-Case: Die Menge aller Use-Cases ist in der Menge aller KoA$_L$A-Aktivitäten enthalten.[2]

Das Kontextdiagramm zur Darstellung der Systemkooperationen in der Systemgestaltungsphase ist mit einem abstrakten "Use Case Model" von UML vergleichbar. Hierbei treten jedoch an die Stelle der Use-Cases, die jeweils eine konkrete Aktivitätsfolge definieren, nun abstrakte Aktivitätsbereiche, deren Realisierung noch weitgehend offen gelassen werden kann.

"Essential use cases" (s. [CL99], abstrakte Kommunikationssequenzen ohne festgelegte Benutzerschnittstelle) lassen sich hingegen in Form von Aktivitätsspezialisierungen oder -verfeinerungen modellieren und entweder textuell als Ablaufbeschreibungen oder grafisch als Systemkonversationen dokumentieren.

[2] KoA$_L$As Aktivitäten beschreiben einen "Meta-Use-Case" (Metatyp-Ebene). Ein Aktivitätsmuster kann verwendet werden, um eine bestimmte Ausprägung eines solchen Metatyps zu beschreiben ("Use-Case", Typ-Ebene) oder aber in Form eines realen Szenariums zu konkretisieren (Instanz-Ebene).

Teile von "concrete use cases" (konkrete Systemnutzungssequenzen auf der Basis einer festgelegten Benutzerschnittstelle) lassen sich im Nutzungskonzept in der Konkretisierung der Aktivitäten auf Anwenderseite finden, auch wenn diese Nutzungsweisen zum großen Teil ausschließlich in Form abstrakter Nutzfälle, also als "essential use cases" beschrieben werden.

Die Aktivitätsbeschreibungen der Anwender beschreiben dabei nur als "Nebeneffekt" auch Sequenzen von Interaktionen zwischen System und Anwender. Hauptziel der Modellierung ist die kontextreiche Beschreibung der Arbeitswirklichkeit der Akteure. Die Systemnutzungssequenzen werden dabei nicht aus einer zielorientierten Entwicklungssicht heraus für die Darstellung der Nutzungsschnittstelle ausgewählt und erstellt ("wie wird eine bestimmte Systemfunktion planmäßig bewerkstelligt?"), sondern aus der Sicht der spontanen Aufgabenerfüllung oder Sonderfallbehandlung der Anwender heraus.

Mit dem Nutzungskonzept soll keine Benutzerschnittstelle des Systems spezifiziert oder eine präskriptive Bedienungsanleitung erstellt werden, sondern die Nutzungssituation der Anwender umfassend dargestellt und konkretisiert werden. Die Haltung bei der Beschreibung der Aktivitäten – Darstellung der Arbeitspraxis – ist dabei qualitativ eine andere als bei der Erstellung von allgemeinen Use-Cases zur Definition von planmäßigen Nutzungssequenzen als Teil einer Softwarespezifikation.

Contextual Design[BH98]

Contextual Design (abgekürzt "CD") ist wie KoA$_L$A auch ein anwenderzentrierter Prozess für das Design von Softwaresystemen, basierend auf einer genauen Analyse der Arbeitszusammenhänge, in denen das Softwaresystem eingesetzt werden soll.

CD beschreibt ein stark praxisorientiertes Vorgehen zur schrittweisen Ausgestaltung interaktiver Systeme. Ausgehend von Einzelinterviews ("contextual inquiry"), in denen die Entwickler die reale Alltagspraxis der Anwender kennenlernen, und Teamsitzungen zur gemeinsamen Interpretation und Konsolidierung der gewonnenen Erkenntnisse werden zunächst eine Reihe von Modellen erstellt, die die allgemeine Arbeitssituation der anvisierten "Kunden" verallgemeinern. CD kennt hierfür fünf verschiedene Modellarten, die zur Darstellung der Arbeit aus unterschiedlichen Perspektiven verwendet werden können:

- das "flow model" – vergleichbar mit KoA$_L$As Kontextdiagramm – zur Darstellung von Individuen bzw. Rollen mit Zuständigkeiten und den Kommunikationsflüssen zwischen ihnen, die jeweils mit einer Beschreibung der Interaktion und den transportierten Artefakten annotiert werden.
- dem "sequence model", das zur Beschreibung einzelner sich wiederholender Arbeitssequenzen einzelner Personen dient
- dem "artifact model" zur Beschreibung aller eingesetzten physikalischen Hilfsmittel, die die beobachteten Personen in ihrer täglichen Arbeit verwenden (z.B. Exceltabellen, Formulare, Pläne, Kalender, Pinnwände etc.)
- dem "cultural model" zur Beschreibung des vorherrschenden kulturellen Kontexts, z.B. in Form von Bedürfnisse des Einzelnen, gegenseitiger Erwartungen und Einflüsse, gemeinsamer Werte und politischer Vorgaben bzw. Randbedingungen

- dem "physical model" zur Darstellung der Arbeitsplätze in ihren räumlichen Dimensionen und Anordnungen. Das Diagramm bildet skizzenhaft die relevanten Räume, Tische, Wege, benutzte Werkzeuge/Hardware/Software/Artefakte etc. an ihren richtigen Positionen und realen Verhältnissen ab.

Ausgehend von dieser Darstellung der Arbeitsumgebung werden gemeinsam mit den Anwendern Ideen und Optionen zur Umgestaltung der bisherigen Arbeitsprozesse diskutiert und in der Folge Visionen erarbeitet, wie sich die Arbeitspraxis nach Einführung eines neuen Softwaresystems darstellen könnte. Die verschiedenen Visionen werden in freien Brainstorming-Sitzungen mit den Anwendern zusammen in Form einer "flow model"-artigen, sehr informellen Zeichnung bildhaft skizziert und schrittweise zu einer gemeinsamen Vision der neuen Arbeitsumgebung konsolidiert, bevor im darauffolgenden Systemdesign dann sukzessive nach dem konzeptionellen Entwurf der Benutzeroberfläche in Form des "user environment design" dann auch die reale Benutzeroberfläche Gestalt annimmt.

Im Gegensatz zu KoA$_{L}$A liegt bei CD der Schwerpunkt in der Festlegung und Beschreibung der einzelnen Arbeitsschritte. CD definiert zwar auch eine Reihe von Diagrammtypen, die zur Dokumentation und Beschreibung von Arbeitssituationen verwendet werden können. Deren Ausdrucksmächtigkeit, Vollständigkeit und formale Konsistenz fallen jedoch weit schwächer aus als bei KoA$_{L}$A. KoA$_{L}$As Aktivitätsdiagramme dienen z.B. nicht nur der Ablaufbeschreibung von Tätigkeiten einzelner Personen, sondern auch der akteurübergreifenden Beschreibung ganzer Aktivitätsbereiche. CD "flow models" dokumentieren zwar die Statik der Kommunikationsbeziehungen, aber es gibt dort kein Modell um wie in KoA$_{L}$As Konversationsdiagrammen auch deren Dynamik auszudrücken. Das "cultural model" findet sich darüberhinaus in KoA$_{L}$A weit umfassender in Form der Tätigkeitsprofile wieder, die nicht nur den kulturellen Kontext aufführen, sondern ihn auch mit den betreffenden Verantwortlichkeiten der Akteure in Verbindung setzen.

Der Prozess von CD ist darauf angelegt, den Weg von der Umgebungsanalyse zum Entwurf einer Softwarelösung zu beschreiben. Er sieht dabei keine expliziten Gestaltungsiterationen vor. Die erstellten Modelle dienen allein der Modellierung der aktuellen lokalen Arbeitspraxis betroffener Personen.

KoA$_{L}$As Modelle sind hingegen darauf abgestimmt, mit ihnen auch den Sollzustand modellieren zu können, also die zukünftige Arbeitspraxis kooperierender Personen global zu beschreiben. Auf Grundlage dieser Diagramme wird neben der schrittweisen Ausgestaltung und Dokumentation des Softwaresystemprofils auch eine modellbasierte Evaluation der gewählten Softwaresystemform ermöglicht. CD sieht hier nur den üblichen Weg vor, das Design mittels "storyboards" skizzen- und bildhaft zu beschreiben und per "paper mockups" oder Prototypen zu validieren.

Zusammenfassend lässt sich sagen, dass CD einen informellen, tätigkeitsorientierten und sequentiellen Entwicklungsprozess festlegt, während KoA$_{L}$A eine offene Vorgehensweise definiert, in welcher ein iterativer Gestaltungsprozess mittels eines umfassenden Modellierungs-Instrumentariums unterstützt wird.

OASE[Pae00]

OASE ("Objekt- und Aufgabenorientierte Software-Entwicklung") ist ebenso wie KoA$_{\text{L}}$A eine modellbasierte Methode zur schrittweisen Ausgestaltung von Softwaresystemen unter Einbeziehung ihrer Umgebungszusammenhänge.

OASE unterscheidet in der Modellierung das Anwendungssystem (Nutzer und Betroffene der Anwendungsumgebung und deren Arbeitspraxis), das Nutzungssystem (direkte Systemnutzer und das Softwaresystem in ihrem Zusammenspiel) und das eigentliche Softwaresystem (Benutzerschnittstelle und Anwendungskern). Jedes dieser Systeme wird dabei durch drei Teilmodelle beschrieben. Die *externe Aufgabenspezifikation* (vergleichbar mit der Phase der Motivierung nach Kap. 2.5) definiert die nach außen sichtbaren Rollen des Systems bzw. seiner wichtigsten Komponenten – je nach Systemtyp sind das menschliche Akteure, Nutzer, das zu entwickelnde Softwaresystem selbst oder auch die einzelnen Softwaresystemkomponenten – und deren Ziele bzw. Aufgabenbereiche. Die *interne Aufgabenspezifikation* (vergleichbar mit der Phase der Zielsetzung nach Kap. 2.5) modelliert die Daten des Systems und dessen interne emergenten Aktivitätsprozesse unter Einbeziehung etwaiger Daten- und/oder Kontrollflüsse. Der "Entwurf" (vergleichbar mit der Phase der Planung nach Kap. 2.5) schließlich verfeinert die vorhandenen bzw. definiert neue Systemkomponenten, indem er ausgehend von den beschriebenen Aktivitäten bestimmte Zuständigkeitsbereiche herauslöst und sie zusammen mit den betroffenen Datenelementen einzelnen Systemkomponenten zuweist. Die in der externen Aufgabenspezifikation definierten abstrakten Aufgabenbereiche des Systems werden im Entwurf also zu konkreten Aufgaben und Diensten von Systembestandteilen verfeinert.

Ausgehend von den Unternehmenszielen (bzw. Aufgaben des Anwendungssystems) werden in der Modellierung bzw. Gestaltung des Anwendungssystems die Aktivitäten des Unternehmens und das Anwendungsdatenmodell definiert, bevor abschließend die Akteure (Nutzer, das Softwaresystem etc.) und deren Zuständigkeiten bzw. Rollen abgeleitet werden. In der Modellierung des Nutzungssystems werden zunächst die Nutzerrollen identifiziert und nach Analyse und Gestaltung aller Arbeitsprozesse, die die Softwaresystemnutzung einschließen, werden schließlich die Aufgaben des Softwaresystems und seine Daten spezifiziert und in Form von Systemrollen gruppiert. Zur Modellierung der Benutzerschnittstelle wird schließlich ein Nutzungsentwurf durchgeführt, der zur Definition von Nutzungsszenarien, Datensichten und Dialogabläufen führt. Abschließend wird der Anwendungskern konkretisiert, indem ausgehend von den Systemdaten und Aktivitätsfolgen Komponenten des Systems in Form von Rollen mit eigenen Zuständigkeitsbereichen spezifiziert werden.

OASE definiert einen konsequenten Verfeinerungs- und Konkretisierungsprozess, der sowohl auf der Systemebene greift und hier von dem abstrakten Anwendungssystem zum Entwurf des Nutzungssystem und der Definition des konkreten Softwaresystems führt, als auch lokal auf der Ebene der Modellierung der einzelnen Systeme Anwendung findet. OASE beschreibt so einen analytischen und konstruktiven Gestaltungs-, Entwurfs- und Verfeinerungsprozess, der schrittweise von der Analyse der Anwendungsumgebung über die Definition der Anforderungen an das Softwaresystem zur Spezifikation und dem Entwurf des Softwaresystems führt.

OASE bietet ein sehr komplexes Vorgehensmodell, das zahlreiche Techniken, Notationen und Beschreibungsformen existierender Methoden integriert. Die Methode versucht zwar, einheitli-

che Konzepte für die Modellierung der drei unterschiedlichen Systeme zu verwenden (zugrunde liegt ein mathematisches Systemmodell) und die Einzelschritte bei der Modellierung dieser drei Systeme möglichst orthogonal zu gestalten (von Aufgabenbereichen über die Analyse der Aktivitäten zur Definition von Rollen und deren Diensten). Die Komplexität des Prozesses jedoch, die Anzahl der Modellierungsschritte und die Anzahl, Vielfalt und Umfang der verwendeten Beschreibungsmittel setzen der praktischen Anwendung in der Realität Grenzen.

OASE stellt desweiteren einen eher konventionellen rational-analytischen Ansatz dar, der den Anwendungsbereich mit den gleichen Mitteln zu beschreiben sucht, die sich auch zur Modellierung von technischen Systeme bewährt haben. OASE versucht, ausgehend von Aufgaben des Anwendungssystems schrittweise sequentiell die Aufgaben des Softwaresystems abzuleiten bzw. durch Analyse der beobachteten Aktivitäten das System konstruktiv in kleinere Einheiten zu detaillieren. Dabei stützt sich OASE hauptsächlich auf die Modellierung von Prozessen incl. Daten und Kontrollflüssen. Kommunikation wird nur in Form von Nachrichten modelliert.

KoA$_L$A beschreitet hier einen iterativen und ganzheitlicheren Weg, der den gestalterischen Aspekt der Entwicklung in den Vordergrund stellt. Das Anwendungssystem wird nicht normativ aufgabenorientiert analysiert und spezifiziert, sondern in seinem Verhalten möglichst interaktions- und darum nutzerorientiert beschrieben, um den gestalterischen Abstimmungs- und Entwicklungsprozess optimal zu unterstützen. Aus diesem Grund basiert der Ansatz hauptsächlich auf der Untersuchung und Ausgestaltung der Kommunikationsbeziehungen. Der menschliche Akteur und seine Interaktions- und Kooperationsbeziehungen stehen primär im Vordergrund. Diese werden nicht analytisch abgleitet oder konstruiert, sondern interaktiv umgestaltet. Die frei kombinierbaren Beschreibungsmittel sind darum qualitativ und quantitativ möglichst minimiert.

Verglichen mit OASE wird das System nicht konstruiert, sondern in seinem Verhalten beschrieben, indem seine Komponenten (menschliche oder technische Akteure) studiert und ihr Zusammenspiel möglichst vielfältig dargestellt werden, um das emergente Systemverhalten möglichst einfach absehen und diskutieren zu können. Das Softwaresystem wird nicht als Maschine mit eigenen Daten und Prozeduren (Dienste) modelliert, sondern als Akteur mit eigenem “Stellenprofil”, der in vielfältigen Kooperations- und Interaktionsbeziehungen steht.

Zusammenfassend kann man sagen, dass für KoA$_L$A eher der partizipative Gestaltungsprozess im Vordergrund steht, während OASE versucht, ein eher ingenieurwissenschaftliches Konstruktionsverfahren zu definieren.

6.4. Kritik

Es lassen sich einige kritische Punkte identifizieren, die einem erfolgreichen Einsatz von KoA$_L$A im Wege stehen könnten und nun im Folgenden entkräftet werden sollen:

Komplexität der Modellierung

KoA$_L$A führt eine neue komplexe Modellierungssprache ein, die den Erfolg einer Kontextgestaltung fragwürdig erscheinen lassen. Sowohl Entwickler als auch Anwender müssen zusätz-

lich zu den in der Softwarespezifikation eingesetzten Darstellungsmitteln (beispielsweise den zahlreichen Diagrammen von UML) auch noch die Modellierungssprache von KoA$_{L}$A lernen.

Gegenargumente:

- Die Komplexität wird weitgehend minimiert, indem sich die Modellierungskonzepte von KoA$_{L}$A incl. ihrer grafischen Darstellung an Konzepte von UML anlehnen und diese Konzepte im Gegensatz zu UML flexibel und diagramm- bzw. modellunabhängig kombinierbar sind. [3]
- Durch die Orthogonalität der Konzepte sind die Diagramme flexibel an den spezifischen Modellierungskontext anpassbar, wodurch auch die Vorgehensweise leicht projektspezifisch abgewandelt werden kann.
- Die Vorgehensweise definiert keinen bis ins kleinste einzuhaltenden Entwicklungsprozess, sondern einen generischen Handlungsrahmen, der projektspezifisch angepasst bzw. vereinfacht werden kann, z.B. indem nur einzelne Modelle bzw. Diagramme verwendet oder komplexere Zusammenhänge rein textuell dokumentiert werden etc.
- Da Modellelemente alternativ grafisch und/oder textuell dokumentiert werden können, besteht eine große Flexibilität in der projektspezifischen Auswahl und Verwendung der eingesetzten Beschreibungsmittel. Diese können dem Projekt bzw. seinen Beteiligten angepasst werden.

Zusätzlicher Aufwand

Das von KoA$_{L}$A eingeforderte Kontextdesign als eigenständige neue Phase der Softwareentwicklung, die eine explizite Modellierung und modellbasierte Umgestaltung des Anwendungskontexts vorschreibt, erfordert zusätzliche Aufwände. Diese Aufwände erscheinen auf den ersten Blick sehr hoch.

Gegenargumente:

- Fehler der Softwareentwicklung sind umso "billiger" zu beheben, je früher sie im Entwicklungsprozess auffallen. Der Anforderungsanalyse kommt hierbei eine Schlüsselbedeutung zu. [Boe84]
- Die Alternativen zum Einsatz von KoA$_{L}$A sind das evolutionäre oder explorative Prototyping bzw. die nachgeordnete Anpassung bzw. Erweiterung der Funktionalität (in der "Wartungsphase"), falls eine solche Nachbesserung überhaupt ökonomisch vertretbar ist. Die Kosten dieser Alternativen sind nicht vernachlässigbar bzw. übersteigen im Falle eines Misserfolgs – z.B. wenn das fertige System aufgrund seiner "Unangemessenheit" bzw. "Inkompatibilität" mit der sozialen Anwendungsumgebung keine Akzeptanz erhält – die Kosten von KoA$_{L}$A um ein Vielfaches, ganz abgesehen davon, dass diese Vorgehensweisen auch eher die potentiellen globalen Optimierungsmöglichkeiten einer Kontextgestaltung übersehen.

[3] Aktivitäten finden sich sowohl im Tätigkeitsprofil als auch in Aktivitätsmustern und Kontextdiagrammen. Interaktionen werden in fast allen Diagrammen verwendet. Tätigkeitsprofile können auch in Kontext- oder Konversationsdiagrammen Eingang finden etc.

- Die Aufwände zum Verständnis der Anwendungswelt fallen auch in allen anderen aktuellen Anforderungsanalyse-Methoden an. KoA$_L$A macht dieses Verständnis nur explizit und damit kommunizier- bzw. diskutierbar.
- Das Kontextdesign nach KoA$_L$A kann im Umfang projektspezifisch minimiert werden (z.B. kann in Fällen, in denen das Tätigkeitsprofil des Softwaresystems bereits weitgehend feststeht, die Kontextmodellierung mit der Fiktionsphase zusammenfallen)
- Teilbereiche der Modelle können direkt als Vorlage für Modelle der Softwarespezifikation Verwendung finden, die dort nur noch weiter zu konkretisieren sind.[4] KoA$_L$As Modelle sind damit zum Teil bereits Vorleistungen für die Spezifikation.
- Die Ergebnisse einer bereits durchgeführten Kontextmodellierung können für die Kontextgestaltung jedes anderen neuen Softwaresystems wiederverwendet werden, die im modellierten Arbeitsbereich Anwendung finden sollen.

Die neue Anwendungssituation ist nicht wirklich fassbar

Wirklichkeit ist prinzipiell nicht vollständig durch Modelle fassbar. Daher sind auch KoA$_L$As Fiktions-Modelle keine realistischen Beschreibungen der neuen Arbeitspraxis, so dass die Möglichkeit einer realistischen Einschätzung der gestalterischen Qualität eines Softwaresystems fragwürdig erscheint.

Gegenargumente:

- KoA$_L$As Modelle stellen diese Wirklichkeit als Fiktion dar. Eine solche Leitlinie ist für jedes Entwicklungsvorhaben notwendig, um die Aktivitäten der Projektteilnehmer zu koordinieren.
- Die Modelle selbst sind konzeptionelle Artefakte, die als Kommunikationsmedium eingesetzt werden, um die Diskussion und Abstimmung des Entwicklungsziels zu ermöglichen. Sie sollen keine Wirklichkeit korrekt spezifizieren, sondern das gemeinsame Ziel kontextreich beschreiben, um den Abstimmungsprozess zu fördern. Die Gestaltungsqualität wird nicht objektiv analysiert, sondern von jedem Projektbeteiligten subjektiv eingeschätzt.
- KoA$_L$As Modelle erlauben zwar keine vollständige Evaluation der Systemqualität, sie ermöglichen diese jedoch näherungsweise, und das im Gegensatz zu vielen etablierten Entwicklungsmethoden noch vor der realen Entwicklung und dem praktischen Einsatz des Systems.

Kontext- und Softwaregestaltung sind nicht trennbar

Die Gestaltung eines soziotechnischen Nutzungssystems ist von der Gestalt des eingebetteten Softwaresystems abhängig und umgekehrt. Beim Kontextdesign müssen die technischen Gestaltungsoptionen berücksichtigt werden, die die mögliche Funktionalität und Interaktivität

[4] Die Systembegriffswelt kann z.B. bereits als ein abstraktes konzeptionelles Objektmodell aufgefasst werden, das nur noch weiter zu konkretisieren ist. Die Systemkooperationen sind Kandidaten für die Dokumentation von Use-Cases, die an Hand der konkreten Benutzerschnittstelle noch verfeinert werden müssen etc.

des neuen Softwaresystems vorgeben bzw. einschränken. Ein Kontexdesign kann damit nicht ohne das Vorwissen um die technische Realisierung des anvisierten Systemprofils erfolgen.

Gegenargumente:

- Kontext- und Softwaredesign finden auf zwei getrennten Abstraktionsebenen statt. Das Softwaresystem und seine Gestalt ist sehr wohl auch bereits in der Kontextgestaltung präsent, und zwar als technischer Akteur mit eigenem Wissensmodell, Tätigkeitsprofil und Interaktionspartnern.
- Im Softwaredesign wird nur die technische Ausgestaltung des IS-Akteurs konkretisiert. Diese technische Gestalt – insbesondere die Benutzerschnittstelle – hat weniger Einfluss auf die globale Arbeitsorganisation, als vielmehr auf die lokale Tätigkeit der System-Anwender. Die Funktion (das Ziel) einer Interaktion ist Gegenstand der Kontextmodellierung und kann unabhängig von ihrer konkreten Ausgestaltung im Softwaredesign behandelt werden.
- Kontext- und Softwaregestaltung folgt nicht zwingend sequentiell aufeinander, sondern kann durchaus auch überlappend stattfinden. Parallel zur Modellierung und Gestaltung des Nutzungssystems, kann z.B. bereits die Konkretisierung des Informationssystems in Form eines "Paper-Mock-Ups" erfolgen, um den Abstimmungsprozess über die zukünftige IS-Rolle zu fördern.
- Kontext- und Softwaregestaltung sind zwei Tätigkeitsbereiche, die unterschiedliche Qualifikationen benötigen und auch darum als eigenständige Tätigkeitsbereiche behandelt werden sollten. Während die Kontextgestaltung sehr viel mit Organisations- und Arbeitsgestaltung zu tun hat und damit einen stark interdisplinären Charakter besitzt, ist die Softwaregestaltung eher den Software-Ingenieuren vorbehalten, die technische Randbedingungen, Gestaltungsmöglichkeiten und bewährte Problemlösungsmuster kennen.
- Aus Gründen des Projektmanagements ist es sehr nützlich, diese beiden Gestaltungsbereiche z.B. in Form getrennter Meilensteine auseinanderzuhalten, um den sozialen Abstimmungsprozess über die globale Rolle des Systems von der Diskussion um Detailaspekte seiner konkreten Realisierung zu trennen und damit handhabbarer zu machen.

6.5. Ausblick

Abschließend werden im folgenden Abschnitt noch einige Optionen der Weiterentwicklung von KoA$_{\text{L}}$A beleuchtet:

Integration existierender Ansätze KoA$_{\text{L}}$A ist eine offene Vorgehensweise, in die weitere bewährte Techniken eingebettet werden können. Es wäre noch zu untersuchen, in welcher Form diese mit dem größten Erfolg integriert werden können.

Im Kap. 3 werden eine Reihe konkurrierender Ansätze diskutiert. Ein Teil dieser Vorgehensweisen bzw. Techniken lässt sich natürlich auch in KoA$_{\text{L}}$A integrieren. Beispielsweise wären hier die Vorgehensweisen des "Participatory Design" [NS92, GK91] oder Workshopmethoden

wie z.B. USTM ([MFKH90, Mac97]) zu nennen, die gut in den Gestaltungs- und Fiktionsphasen von KoA$_L$A verwendet werden können, oder die "Hierarchical Task Analysis" nach [CSA89], die auch gute Dienste in der Erstellung der Tätigkeitsprofile der Kontextanalyse leisten könnte etc.

Konkretisierung der Teilschritte Momentan sind die einzelnen Phasen von KoA$_L$A hauptsächlich durch Leitfragen strukturiert. Es ist zu untersuchen, in wieweit diese noch durch geeignete Vorgehensweisen oder Techniken innerhalb der Phasen konkretisiert werden können. Beispielsweise in der Phase der Fiktion durch eine geeignete Organisation von Workshops oder anderen Diskussions- und Abstimmungsverfahren.

KoA$_L$A als "agile" Methode[5] Auch in revolutionären schlanken Entwicklungsansätzen wie z.B. dem "eXtreme Programming" [Bec00] oder dem evolutionären Prototyping ist vor der ersten Ausbaustufe eine Verständigung über das zu lösende Anwendungsproblem notwendig.

Obwohl die Möglichkeit einer der Realisierung vorausgehenden vollständigen und fehlerfreien Softwarebeschreibung in diesen Ansätzen stark angezweifelt wird, ist es dennoch notwendig bzw. sinnvoll, sich über den Anwendungskontext und die zukünftige Rolle des neuen Softwaresystems klar zu werden.

Um ein System zu erhalten, das mehr als ein lokales Optimum darstellt, ist ein umfassendes Problemverständnis notwendig, dass nicht alleine durch "encouraging hacking in Annäherungszyklen" erzielt werden kann.

KoA$_L$A beschreibt eine sehr umfangreiche allgemeine Vorgehensweise. Je nach konkretem Anwendungskontext könnten jedoch auch spezialisierte "leichtgewichtige" Anwendungsszenarien entwickelt werden, die z.B. nur die notwendigsten Modelle ausarbeiten bzw. einzelne Gestaltungsschritte auf das Minimum reduzieren. Zur Strukturierung der möglichen Anwendungskontexte derartiger "Light"-Versionen von KoA$_L$A könnten beispielsweise Jacksons "Problem Frames" dienen ([Jac94a]).

Evaluation KoA$_L$A wurde bisher erfolgreich für die Entwicklung kleinerer Informationssysteme für die Verwaltung der Universität Ulm eingesetzt, die zwar einen großen Kreis indirekt betroffener Akteure zu berücksichtigen haben, jedoch in der Regel nur ein bis drei direkte Systemanwender besitzen. Die Entwicklungsprojekte wurden dabei von studentischen Hilfskräften, Diplomanden oder auch Studenten im Studiengang Informatik (30 bis 50 parallele Projekte im Team) durchgeführt. Auch wenn die Erfahrungen mit KoA$_L$A in diesen Projekten äußerst ermutigend waren, steht eine Evaluation in größeren Projekten noch aus.

Insbesondere wären hier auch vergleichende Studien interessant, die dasselbe Projekt mit unterschiedlichen, konkurrierenden Vorgehensweisen durchführen und die Bewertung in enger

[5] Die "Agile" Softwareentwicklung entwickelt sich im Fachgebiet des Softwareengineering momentan zu einem neuen Schlagwort für "leichtgewichtige" hocheffiziente Vorgehensmodelle, Methoden und Techniken, die sich im Projektverlauf flexibel und kontinuierlich an wechselnde Anforderungen anpassen lassen. Die Vorgehensweise ist im Allgemeinen evolutionär, inkrementell, selbstorganisierend und emergent. Anforderungen, Technologien, Vorgehensweisen können im Projektverlauf ständig wechseln bzw. gewechselt werden.

interdisziplinärer Zusammenarbeit mit Psychologen, Arbeitswissenschaftlern und Betriebswirten vornehmen.

Übergang zur Spezifikation Der Übergang zu objektorientierten Methoden der Softwarespezifikation wird von KoA$_{\text{L}}$A wohl am besten unterstützt. Der Übergang zu anderen Methoden wie der strukturierten Analyse oder die Möglichkeit der Einbettung in revolutionäre Ansätze wie z.B. dem "Extreme Programming" ist noch zu untersuchen.

Ausweitung des Gestaltungsbereichs Der hauptsächliche Anwendungsbereich von KoA$_{\text{L}}$A ist die Ausgestaltung von Nutzungssystemen im direkten Umfeld neuer Informationssysteme. Es wäre jedoch auch denkbar, KoA$_{\text{L}}$A im umfassenderen Bereich der organisatorischen Gestaltung von allgemeinen Handlungssystemen wie z.B. Abteilungen oder ganzen Unternehmen einzusetzen. Es stellt sich die Frage, wie KoA$_{\text{L}}$A hier helfen könnte bzw. verändert oder erweitert werden müsste.

Werkzeugunterstützung Jede Methode ist nur so gut wie das Werkzeug, dass sie unterstützt. Zur Zeit existiert ein Werkzeug zur Analyse, Aufbereitung und Qualitätssicherung der semiformalen Dokumente der Projektdatenbank, genannt "xSeu". Dieses Werkzeug führt syntaktische und semantische Überprüfungen an den Dokumenten durch, stellt ein hypertextbasiertes Autorensystem für die Ansicht bzw. Bearbeitung der in diesen Dokumenten definierten Modellelemente zur Verfügung und erlaubt neben der Generierung anderer Dokumentationsformen wie Html, Latex, Rtf auch einen nahtlosen Übergang zur Softwarespezifikation bis hin zur Codegenerierung (zur Zeit können Java- oder Delphi-Klassenrümpfe generiert werden).

Als zukünftige Erweiterung wäre eine Integration der grafischen Darstellungsformen von KoA$_{\text{L}}$A wünschenswert. Hierbei wäre von besonderem Interesse, die "xSeu" zu einem intuitiven grafischen Editor für partizipative Settings auszubauen, der die werkzeugunterstützte Modellierung mit den Anwendern zusammen oder vielleicht sogar autonom durch die Anwender selbst erlaubt.

Anhang

A. Referenz der Modellierungssprache

Ein Anforderungserhebungsprozess wird nach [DvLF93] immer von Domänen-Modellen und Strategien geleitet. Eine Strategie ist dabei ein besonderer Weg der Traversierung des Metamodellgraphs, um seine diversen Knoten und Verbindungen zu instantiieren – sprich: eine bestimmte Vorgehensweise, das oder die Domänenmodelle schrittweise aufzubauen.[1]

Der in Kap. 5 beschriebenen "Strategie" von KoA$_L$A soll daher nun ergänzend auch eine Beschreibung des Metamodells von KoA$_L$A folgen. Dabei werden sowohl Syntax als auch Semantik aller Modellierungskonzepte präzise und knapp definiert.

Die Notation ist Diagramm- und Modell-unabhängig, so dass Modellierungsebenen und Diagrammelemente flexibel kombinierbar sind.

Metamodell

Die abstrakte Syntax aller Modelle von KoA$_L$A wird an Hand eines Metamodells definiert. Die Darstellung des Metamodells lehnt sich an die Metamodellierungssprache an, die auch zur Spezifikation von UML verwendet wird (siehe z.B. [UML03]).

Daran anschließend wird die Semantik der einzelnen Modellierungskonzepte in natürlicher Sprache definiert. Zur Definition der Konzepte ist im Vergleich zum Metamodell von UML nur ein sehr kleines und kompaktes Modell notwendig.

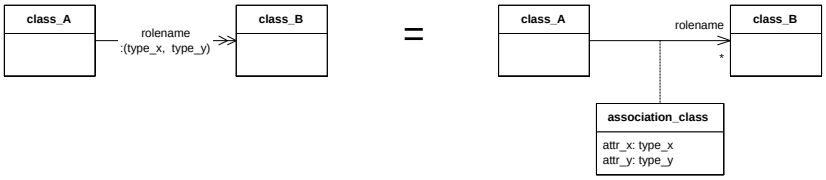


Abbildung A.1.: Gegenüberstellung der Darstellung von Beziehungen in KoA$_L$A (links) und UML (rechts)

Als Sprache zur Definition des Metamodells wird UML verwendet, erweitert um eine in Abb. A.1 dargestellte Kurzschreibweise: Eine gerichtete Assoziation mit einfachem Pfeil trägt implizit die Kardinalität "Eins" während ein Doppelpfeil für die Kardinalität "Null oder beliebig viele" steht. Der Rollenname einer Assoziation kann weiterhin ein Ergebnistupel besitzen. Die

[1] Jeder Strategieschritt ist dabei aus feineren Schritten aufgebaut wie z.B. Fragebogenaktionen, Durchführung von Interviews, Verifikationen gegenüber Metaconstraints, Taktiken zur Auswahl/Bewertung von Alternativen, Wiederverwendung analoger Domain Models etc.

Elemente dieses Tupels stellen Assoziationsattribute dar. Ist kein eigener Attributname angegeben, so ergibt sich der Name aus dem Namen des Typs. Als weitere Vereinfachung in der Darstellung von Attributen kann der Attributname von einem Fragezeichen gefolgt werden. Dies ist mit der Typisierung des Attributs als Wahrheitswert gleichzusetzen.

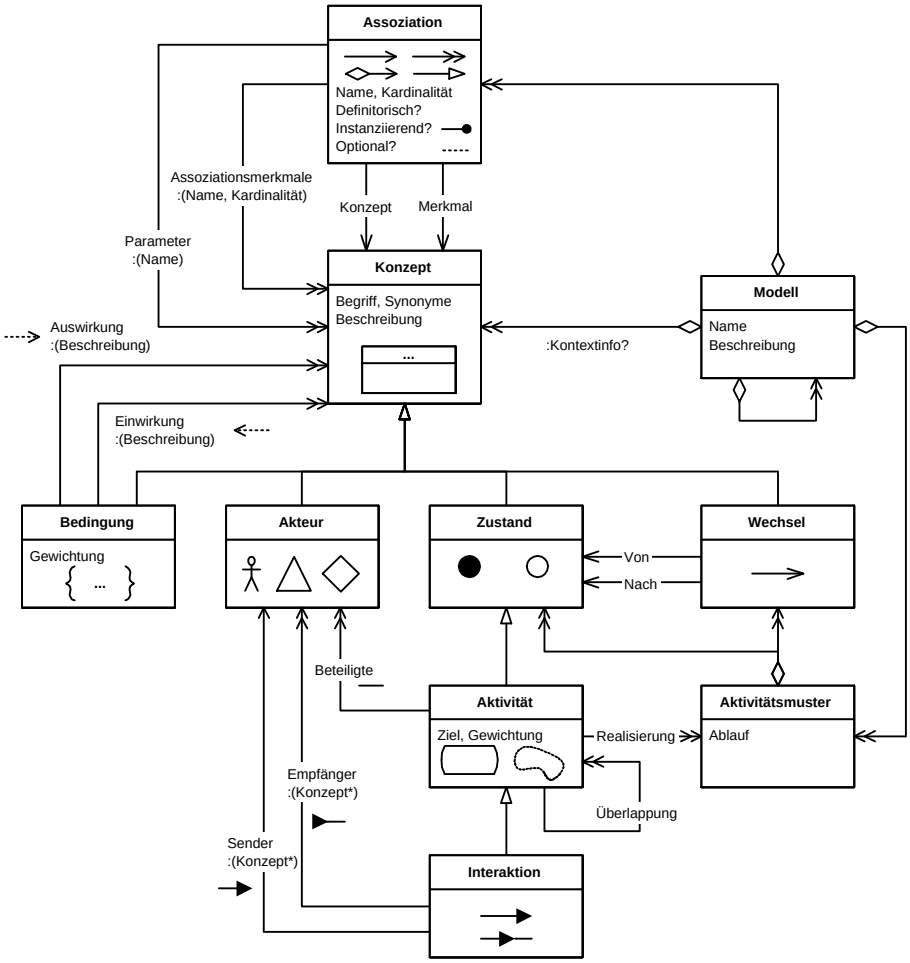


Abbildung A.2.: Das gemeinsame Metamodell aller KoA$_L$A-Modelle

In Abb. A.2 wird das Metamodell von KoA$_L$A dargestellt. Zusätzlich zu den Klassen, Beziehungen und Attributen finden sich in diesem Diagramm auch die grafischen Symbole wieder, mit denen die jeweiligen Modellkonzepte in den Diagrammen dargestellt werden.

Im Folgenden wird nun auf die einzelnen Konzepte und ihre grafische Notation eingegangen. Jedes Modellierungselement wird dabei nach folgendem Schema definiert:

Name des Modellierungskonzepts

Verwandte Begriffe: Begriffe, die das gleiche oder ähnliche Konzepte bezeichnen.

Textuelle Definition der Semantik.

Erläuterung weiterer Aspekte des Modellierungskonzepts:

Attribute	70
Beziehungen	Beziehungen zu anderen Modellierungskonzepten.
Darstellung	Textuelle Erläuterung der grafischen Darstellung. Beispiele für die textuelle Repräsentation findet sich in Kap. B.
Stereotypen	Sofern vorhanden eine Sammlung existierender Stereotypen (ähnlich zu UML). Stereotypen werden entweder mit einem ausgezeichneten Icon dargestellt oder textuell annotiert (in doppelten spitzen Klammern gesetzt: "<<stereotyp>>").
UML	Sofern möglich, eine informelle Abbildung bzw. ein Vergleich mit Metaklassen der UML.

A.1. Modell

Verwandte Begriffe: Package.

Ein Modell fasst eine Menge von Modellelementen zu einer konzeptionellen Einheit zusammen und stellt für diese einen Namensraum dar.

Ein Modell dient der Darstellung spezifischer Sachverhalte aus einem bestimmten Blickwinkel und kann aus aggregierten Teilmodellen aufgebaut sein.

KoA$_{\text{L}}$A kennt folgende Modell- bzw. Teilmodellarten:

- Wissensmodell
- Organisationsmodell
  - Kooperationsstruktur
  - Tätigkeitsprofil
- Kooperationsmodell
  - Kooperationsstruktur
  - Aktivitätsmuster
  - Konversationsmuster

Attribute

Name	Eindeutiger Name des Modells
Beschreibung	Textuelle Definition und Erläuterung des Modells

Beziehungen

Ein Modell aggregiert Modelle, Konzepte, Aktivitätsmuster oder Assoziationen:

Modelle	Alle enthaltenen Submodelle
Konzepte	Alle aggregierten Modellelemente. Dabei wird zwischen den Konzepten unterschieden, die eigentlicher Inhalt des Modells sind und solchen, die nur zur Verdeutlichung des Kontexts dienen, also Kontextinformationen darstellen (wie z.B. die Interaktionen, die in einem Tätigkeitsprofil oder einem Aktivitätsdiagramm angegeben werden; oder die Aktivitäten, die in einem Konversationsdiagramm angegeben werden).
Assoziationen	Alle aggregierten Assoziationen
Aktivitätsmuster	Alle aggregierten Aktivitätsmuster

Grafische Notation

Ein Modell wird textuell in der Projektdatenbank (siehe Kap. B) oder grafisch in Form von Diagrammen (siehe Kap. 4) definiert. Zur Definition eines Modells werden in der Regel mehrere Dokumente bzw. Diagramme benutzt.

KoA$_{\text{L}}$A kennt fünf verschiedene Diagrammarten:

- Konzeptdiagramm
- Kontextdiagramm
- Profildiagramm
- Konversationsdiagramm
- Aktivitätsdiagramm

Bezug zum UML-Metamodell

Vergleichbar mit Model.

A.2. Konzept

Verwandte Begriffe: Begriff, Kategorie, Typ, Klasse, Gattung, Art.

Ein Konzept stellt eine gedankliche Einheit oder Erkenntnisform dar, die eine bestimmte Kategorie von konkreten oder abstrakten Gegenstände oder Phänomenen umschreibt (siehe hierzu auch [Rei91]).

Die von einem Konzept umfasste Menge von Elementen wird auch als Extension des Konzepts bezeichnet. Diese Menge kann auch intensional über ihre gemeinsamen Merkmale spezifiziert werden (siehe hierzu auch die "Formale Begriffsanalyse" [GW96]).

Im Folgenden wird als Synonym von "Konzept" auch oft das Wort "Begriff" benutzt.

Attribute

Begriff	Eindeutiger Name des Konzepts
Synonyme	Ein Begriff kann mehrere Synonyme besitzen
Beschreibung	Textuelle Definition und Erläuterung des Konzepts bzw. des Begriffs, seine fachliche Semantik

Beziehungen

Assoziationen	Ein Begriff kann durch Rückgriff auf andere Begriffe intensional definiert werden

Grafische Notation

Wird wie eine UML-Class als Rechteck dargestellt. Synonyme werden durch "=" getrennt hinter dem Konzeptnamen aufgeführt.

Handelt es sich bei dem Begriff um ein reines Instanz-Konzept, d.h. um ein Konzept, dessen Merkmale alle instanziierend und nicht definitorisch sind und das selbst bei anderen Konzepten nur in instanziierenden Merkmalen verwendet wird, so kann das auch grafisch ausgedrückt werden, indem entweder das Rechteck mit einer dicken Linie gezeichnet und/oder der Konzeptname unterstrichen dargestellt wird.

Die grafische Kennzeichnung von Instanzmerkmalen, d.h. von nicht-definitorischen Merkmalen, durch Unterstreichen des Merkmalsnamens oder durch einen ausgefüllten kleinen Kreis am Beginn ausgehender Beziehungen kann dann auch entfallen.

Bezug zum UML-Metamodell

Vergleichbar mit einer Vereinigung von Class, DataType und Instance.

KoA$_{L}$A hebt die Trennung von Klassen- und Datentypspezifikation auf und erweitert die Typisierungsebenen in Richtung Metamodellierung. Mit KoA$_{L}$As Konzepten sind nicht nur Typ und Instanz-Ebene darstellbar, sondern auch Metatypen (Typen von Typen etc.). UML kennt zwar bereits rudimentäre Hilfsmittel wie den Classifier-Stereotypen <<metaclass>> und <<powertype>>, jedoch keine Typen höherer Ordnung.

A.3. Assoziation

Verwandte Begriffe: Beziehung, Aspekt, Eigenschaft, Merkmal.

Eine Assoziation definiert eine semantische Beziehung zwischen zwei begrifflichen Konzepten. Jede solche Beziehung definiert ein Konzeptmerkmal.

Ein Begriffsmerkmal kann parametrisiert sein und selbst wiederum weitere (Assoziations-)-Merkmale besitzen.

Attribute

Name	Eindeutiger Name des Merkmals
Kardinalität	Bei typisierenden, d.h. nicht-instanziierenden Beziehungen die Anzahl der möglichen Beziehungspartner (Instanzen)
Beschreibung	Textuelle Definition und Erläuterung des Merkmals
Instanziierend?	Instanziierende Assoziationen weisen einem Konzeptmerkmal einen Wert zu (Bsp: “Bäume sind grün”), nicht-instanziierende Assoziationen einen Typ (Bsp: “Bäume besitzen eine bestimmte Farbe”).
Definitorisch?	Definitorische Assoziationen weisen den Elementen der Konzeptextension (den “Instanzen”) Merkmale zu, nicht-definitorische dem Konzept selbst (dem “Typ”)
Optional?	Ein Merkmal ist entweder optional (stellt z.B. eine “default”-Belegung dar, bezeichnet eine temporäre Assoziation oder dient einer prototypischen Konzeptdefinition) oder es ist obligatorisch (stellt z.B. eine konstante, statische, allgemeingültige Aussage über das Konzept dar)

Beziehungen

Konzept	Das Konzept, dem die Assoziation ein Merkmal zuordnet
Merkmal	Das Konzept, das die Assoziation einem anderen Konzept als Merkmal zuordnet
Parameter	Eine Assoziation kann parametrisiert sein. Parameter besitzen einen Namen und haben ein bestimmtes Konzept als Typ
Assoziationsmerkmale	Eine Assoziation selbst kann auch wieder Merkmale besitzen. Solche Merkmale besitzen neben einem Typ optional wiederum Namen und/oder Kardinalitäten

Grafische Notation

Eine Merkmalsdefinition kann rein grafisch oder aber auch nur textuell erfolgen.

Zur grafischen Dokumentation eines Begriffsmerkmals wird zwischen dem zu definierenden Begriff ("Quellbegriff") und dem das Merkmal definierenden Begriff ("Zielbegriff") eine gerichtete Verbindungslinie gezogen, die in der Nähe des Zielbegriffs den Merkmalsnamen aufführt (entspricht Assoziationsrolle in UML).

Alternativ dazu wird das Merkmal rein textuell innerhalb des Rechtecks der Begriffsdefinition in der Form "<Name>: <Typ> = <Wert>" notiert. Sowohl Typ als auch Wert sind dabei optionale Angaben. Wenn nur der Name angegeben ist, so gibt dieser Name gleichzeitig auch den Typ an. Der Typ wird in Form eines Konzeptnamens angegeben. Der Wert in Form einer aus Konzepten aufgebauten, beliebig strukturierten Zeichenkette.

Je nach Qualität der Assoziation sind in der Darstellung folgenden Fälle zu unterscheiden:

Qualität	Grafische Darstellung	Textuelle Darstellung
Typisierend oder *instanzierend*	Der Pfeil endet im Zielbegriff mit einer Pfeilspitze oder mit einem kleinen gefüllten Kreis.	Der Zielbegriff befindet sich nach dem ":"-Zeichen (gibt einen Typ an) oder nach dem "="-Zeichen (stellt einen Wert dar).
Deskriptiv oder *definitorisch* (Typ- oder Instanzmerkmal)	Der Pfeil beginnt am Quellbegriff mit oder ohne kleinem gefüllten Kreis. Falls der Quellbegriff dick umrandet oder unterstrichen dargestellt wird, handelt es sich ausschließlich um deskriptive Assoziationen (der kleine Kreis kann entfallen).	Merkmalsname unterstrichen oder nicht.
Ein oder *mehrwertige* definitorische Assoziationen	Ein oder zwei Pfeile am Zielbegriff. Optional auch mit textueller Kardinalitätsangabe.	Beliebige Kardinalitätsangabe nach dem Zielbegriff in eckigen Klammern (Merkmal: Begriff[*]).
Parametrisiert oder nicht	Das Parameter-Tupel folgt direkt auf den Merkmalsnamen: "<Merkmal>(<Par1>: <Typ>, <Par2>: <Typ>, ...): ..." Ist kein Parametertyp angegeben, so ergibt sich der Typ aus dem Parameternamen.	
Attributiert oder nicht	Die Assoziationsmerkmale werden in einem Tupel aufgeführt, das direkt nach dem ":" der Merkmaldefinition folgt (noch vor dem Zielbegriff): <Merkmal>: (<M1>: <Typ>, <M2>: <Typ>, ...) ...	
Optional oder obligatorisch	Die Verbindungslinie ist gestrichelt oder durchgezogen.	Die Merkmalsdefinition steht in eckigen Klammern "[<Merkmal> ...]" oder nicht.

Zur Vereinfachung der grafischen Darstellung können zwei symmetrische Assoziationen auch in Form einer einzelnen Verbindungslinie notiert werden, die optional in der Nähe der Zielbegriffe die entsprechenden Merkmalsbeschreibungen aufweisen.

In der textuellen Definition von Konzept- oder Assoziationsmerkmalen und von Parametern stehen folgende Kurzschreibweisen zur Verfügung:

- "<Name>?" steht für "<Name>: Wahrheitswert"
- "<Name>!" steht für "<Name>: Wahrheitswert = Wahr"

Stereotypen

Aggregation Das Merkmal ist konzeptioneller oder physischer Bestandteil eines Konzepts. Wird als nicht ausgefüllte Raute am Beginn der Verbindungspfeils dargestellt. Wird normalerweise nur für definitorische Merkmale verwendet. Alternativ zur Notation als Verbindungslinie kann das aggregierte Konzept jedoch auch grafisch innerhalb des Darstellungselements des Aggregat-Begriffs notiert werden. Eine textuelle Darstellung dieses Assoziationstyps ist nicht vorgesehen.

Spezialisierung Temporäre (Rollenbeziehung, "may-be-a") oder statische Spezialisierung bzw. Instanziierung ("is-a").

Spezialisierungen werden mit einem Verbindungspfeil mit leerer Dreiecksspitze gekennzeichnet und sind normalerweise definitorisch. Da alle Spezialisierungs-Beziehungen instanziierend sind, entfällt die entsprechende Markierung am Ende des Dreiecks.

Will man eine deskriptive Beziehung von einer konkreten Ausprägung zu einem Konzept ausdrücken ("is-instance-of"), so wird der Beginn des spezialisierenden Verbindungspfeils mit einem ausgefüllten Kreis in der Nähe der Instanz markiert. Diese Markierung ist überflüssig, wenn der Instanzbegriff mit einem dicken Rand versehen oder der Instanzname unterstrichen dargestellt ist.

Handelt es sich um eine Rollenbeziehung ("temporäre" Generalisierung eines Begriffs), so stellt man den Spezialisierungspfeil gestrichelt dar.

Bezug zum UML-Metamodell

Vergleichbar mit einer Vereinigung von Association, AssociationClass, Generalization und Feature.

In KoA$_{\text{L}}$A wird keine Unterscheidung getroffen zwischen der Zuordnung eines Konzepts als Konzeptmerkmal und der Zuordnung eines Konzepts in Form einer benannten Beziehung. Spezialisierungen und Aggregationen stellen desweiteren ausgezeichnete Formen einer Assoziation dar.

A.4. Akteur

Ak·teur: Handelnder (frz. acteur "Schauspieler", lat. agere "handeln, betreiben")
[Wahrig, Deutsches Wörterbuch]

Verwandte Begriffe: Organisationseinheit.

Aktive oder passive, menschliche oder technische Organisationseinheit, die als Bestandteil einer Organisation in deren zielgerichteten Kooperations- und Handlungssystemen eine bestimmte Rolle ausfüllt (Beispiel für Organisationen: Unternehmen, Abteilung, Projektgruppe).

Beziehungen

Aktivitäten	Alle Aktivitätsbereiche, an denen der Akteur teilnimmt.
Interaktionen	Kommunikative Kooperationen, die der Akteur selbst auslöst ("Sender") oder an denen er als Interaktionspartner teilnimmt ("Empfänger").

Grafische Notation

Wird als Männchen-Icon dargestellt.

Stereotypen

Agent Aktives technisches System. Nichtautonomer Akteur ohne freien "Willen". Wird als Dreieck dargestellt. Optional kann es auch durch ein Icon dargestellt werden, das den Agenten bildlich zeigt.

Ist entweder ein *Automat*, also ein maschineller Akteur, der selbstgesteuert aktiv werden und Interaktionen auslösen kann (z.B. andere Softwaresysteme, Roboter, Einsatzplanungssysteme) oder aber ein *Werkzeug*, d.h eine maschinelle Organisationseinheit, die Interaktionen nur fremdgesteuert "vermittelt", d.h. Befehle ausführt (Bsp: Telefon, Fax, Kopierer, Drucker etc.).

Agent: Vertreter; Vermittler; Beauftragter
[Wahrig, Deutsches Wörterbuch]

Gegenstand Passiver materieller ("tangible") "Gegen"-"Stand" von Interaktionen, kann selbst aber keine solchen durchführen. Wird als auf eine Ecke gestelltes Quadrat dargestellt. Optional kann er auch durch ein Icon dargestellt werden, das den Gegenstand bildlich zeigt.

Ist entweder *Material*, d.h ein veränderliches materielles Element, welches Akteure, Werkzeuge oder Automaten "bearbeiten" können (wie z.B. Briefe, Dokumente, Datenbestände, Aktenarchiv) oder ein *Einrichtungsgegenstand*, d.h ein statische materielle

Organisationseinheit mit klarem Zweck, die aber nicht selbst aktiv werden kann (Bsp: Gebäude, Raum, Briefkasten, Postfach, Schwarzes Brett, Eingangskorb).

Gruppe Eine Gruppe von Akteuren (z.B. "die Studenten", "die Wähler") kann selbst auch eine "virtuelle" Organisationseinheit darstellen und daher als eigenständiger Akteur auftreten. Wird dargestellt als zwei überlappende Icons.

Bezug zum UML-Metamodell

Vergleichbar mit Actor.

A.5. Bedingung

Verwandte Begriffe: Constraint, Randbedingung, Anforderung.

Ein Sachverhalt, der Einfluss auf den Ablauf von Aktivitäten nimmt.

Er stellt z.B. eine Voraussetzung dar, eine Folge, eine Konsistenz- oder Vertragsbedingung, die eine gerade laufende Aktivität beeinflussen können etc. Jede Bedingung kann verschiedene Ursachen (Quellen) und Wirkungen (Senken) besitzen. Beispiele: Akteur stellt eine Anforderung (er definiert eine "Rand"-Bedingung) oder besitzt eine bestimmte Motivation, die zu bestimmten Aktivitäten führen oder sie beeinflussen; eine Aktivität kann zu einem Problem führen oder durch ein bestimmtes Problem motiviert werden etc.

Attribute

Beschreibung	Textuelle Beschreibung des Sachverhalts
Gewichtung	Kennzeichnung der Bedeutung der Bedingung: '++': äußerst wichtig; '+': wichtig; 'o' oder '': neutral; '-': vernachlässigbar, '--': unwichtig.

Beziehungen

Einwirkungen	Alle Konzepte (z.B. Akteure, Bedingungen oder Aktivitäten), die auf die Bedingung einwirken. Die Art der Wirkungen kann optional jeweils in einem Freitext beschrieben werden.
Auswirkungen	Alle Konzepte, auf die die Bedingung einwirkt. Die Art der Wirkung kann optional jeweils in einem Freitext beschrieben werden.

Grafische Notation

Der beschreibende Text der Bedingung wird als Freitext in geschweiften Klammern dargestellt, der mit gestrichelten einfachen "Wirkungs"-Pfeilen jeweils mit seinen Quellen und Zielen verbunden ist.

Alternativ kann auch ein gestrichelter Pfeil mit einem oder mehreren Bedingungs-Texten in geschweiften Klammern verwendet werden, der direkt Quelle und Ziel verbindet (z.B. Akteur stellt Anforderung an anderen Akteur oder eine bestimmte Aktivität beeinflusst oder motiviert eine andere Aktivität).

Ein Wirkungspfeil zwischen Bedingungen kann auch zwei Spitzen besitzen und einen beliebigen Text ("Konflikt", "+", "-", etc.) oder ein Symbol (Fragezeichen, Blitz, Ausrufezeichen etc.) tragen, um systemische Wirkungsflüsse auszudrücken. Dies kann insbesondere dazu benutzt werden, Interessenkonflikte explizit zu machen – indem man z.B. entsprechende in Widerspruch stehende Bedingungen (Randebedingungen, Anforderungen oder Motive) mit einem entsprechenden Wirkungspfeil verbindet und mit einem Blitzsymbol annotiert.

Die Gewichtung der Bedingung wird als textuelle Annotation am rechten oberen Rand des Textblocks der Beschreibung vermerkt.

Stereotypen

Motiv Ein Motiv ist eine Bedingung, die Aktivitäten des Akteurs verursacht und deren Ursprung der Akteur selbst ist

Problem Ein Problem ist ein Sonderfall eines Motivs: Problematische Tatbestände, die durch eine Aktivität entstehen können bzw. kausale Ursache für eine Aktivität sind. Kann auch durch ein Ausrufezeichen annotiert werden.

Sonderfall Sonderfälle sind Bedingungen, die eine Abweichung von der Regel darstellen und daher im Gestaltungsprozess gesondert berücksichtigt werden müssen. Sie werden auch als "Ausnahmen" bezeichnet. Sonderfälle führen in der beeinflussten Aktivität in der Regel zu einer expliziten "Ausnahmebehandlung". Kann auch durch ein Fragezeichen-Icon annotiert werden.

Anforderung Eine Anforderung ist eine Bedingung, die Aktivitäten des Akteurs verursacht oder beeinflusst und deren Ursprung ein externer Akteur ist

Vereinbarung Eine Vereinbarung ist eine von mehreren Akteuren eingegangene gegenseitige Verpflichtung, ein sogenannter "social contract". Alle Vertragspartner werden zugleich als Quelle und als Ziel dieser Bedingung gekennzeichnet. Als Sonderfall kann eine Vereinbarung auch einseitig und selbstgesteuert eingegangen werden ("Selbstverpflichtung"). Kann entweder durch zwei sich überlappende kleine Kreise oder durch ein Icon annotiert werden, das zwei sich schüttelnde Hände darstellt.

Voraussetzung Eine Aktivität, ein Zustand oder ein Wechsel kann an bestimmte Eingangsbedingungen geknüpft sein. Solche Bedingungen können statt in geschweiften auch in eckigen Klammern notiert werden: "[Guard-Bedingung]".

Bezug zum UML-Metamodell

Vergleichbar mit einer Vereinigung von Constraint, Comment und Guard.

A.6. Zustand

Verwandte Begriffe: Zeitpunkt, Ereignis.

Ein Zustand markiert einen bestimmten Stand der Kooperation. Zustände finden sich in der Modellierung von KoA$_L$A nur innerhalb von Konversations- und Aktivitätsdiagrammen.

Jeder eingehende Zustandswechsel führt zur "Aktivierung" des Zustands. Von den ausgehenden Zustandswechseln wird jeweils nur genau einer durchgeführt (keine Nebenläufigkeit; "ODER-Knoten").

Ein Zustand mit entsprechender Voraussetzungs- bzw "Guard"-Bedingung kann auch zur Modellierung eines Ereignisses verwendet werden (z.B. "[Wahltag]").

Beziehungen

Wechsel	Ein Zustand hat keinen oder beliebig viele Eingänge wie Ausgänge, die zu Zuständen und/oder Aktivitäten führen.

Grafische Notation

Als nicht ausgefüllter Kreis.

Stereotypen

Synchronisationspunkt Ein Zustand ist ein Synchronisationspunkt, wenn ausgehende Zustandswechsel nur dann durchgeführt werden können, sobald alle eingehenden Wechsel aufgetreten sind. Desweiteren sind dann auch alle ausgehenden Zustandswechsel voneinander unabhängig, d.h. können parallel stattfinden (Nebenläufigkeit, "UND-Knoten"). Wird als ausgefüllter Kreis dargestellt.

Bezug zum UML-Metamodell

Vergleichbar mit State.

A.7. Aktivität

Verwandte Begriffe: Kooperation, Aktivitätsbereich, Tätigkeit, Handlung.

Zielgerichtete Tätigkeit, an der ein oder mehrere Akteure (Kooperation) beteiligt sind. Kann imperativ (textuell oder grafisch in Form von möglichen Aktivitätsmustern) oder deklarativ (textuell beschriebenes Ziel) spezifiziert werden.

Bezogen auf Aktivitätsdiagramme stellt eine Aktivität einen Spezialfall eines Synchronisationspunkts dar (siehe *Zustand*). Vor der Durchführung der Aktivität müssen alle eingehenden Zustandswechsel eintreffen. Nach der Aktivität können alle ausgehenden Zustandswechsel parallel stattfinden.

Attribute

Ziel	Beschreibung des Ziels der Aktivität
Beschreibung	Optionale Beschreibung des Verlaufs der Aktivität. Eine Aktivität kann in ihrem Ablauf eventuell in Form einer Vereinbarung formal spezifiziert werden, normalerweise immer nach einem bestimmten Muster ablaufen oder auch rein situationsbedingt ablaufen und daher eventuell auch gar nicht weiter definierbar sein. Die Beschreibung kann mehrere mögliche Ablaufvarianten aufführen.
Gewichtung	Siehe hierzu auch Abschnitt A.5. Kennzeichnung der Bedeutung der Aktivität: '++': äußerst wichtig; '+': sehr wichtig; 'o' oder '': neutral; '-': vernachlässigbar.; '--': unwichtig

Beziehungen

Beteiligte	Die beteiligten Akteure werden durch eine durchgezogenen Linie mit der Aktivität verbunden.
Realisierungen	Kann in Form mehrerer alternativer Beispiel-Aktivitätsmuster konkretisiert werden, die potentielle Verläufe der Aktivität dokumentieren.

Überlappung	Eine Aktivität kann sich mit beliebig vielen anderen Aktivitäten überlappen, d.h. zum Teil überschneiden. Beispiel: Die im Profildiagramm eines bestimmten Akteurs angegebenen Interaktionen, die zu Aktivitäten des Akteurs hin oder von diesen wegführen, sind Kooperationen mit anderen Akteuren, die mit der Aktivität des Akteurs "einhergehen". Diese Kooperationen enthalten jedoch auch Anteile der Interaktionstätigkeit der Interaktionspartner. Die Beziehung von lokaler Aktivität des Akteurs und gemeinsamer Interaktions-Kooperation ist darum keine Aggregation, sondern eine Überlappung.
Sonderfälle	Jede Aktivität kann in Abweichung von normativen Situationen beliebig viele Spezialisierungen besitzen. Die auslösenden Bedingungen können als Sonderfall-Bedingungen der spezialisierten Aktivitäten dokumentiert werden.
Teilaktivitäten	Ausgezeichnete Teilaktivitäten können als aggregierte Aktivitäten modelliert werden.

Grafische Notation

Eine Aktivität wird wie ein UML-ActionState als Rechteck mit abgerundeten Seiten dargestellt. Alle beteiligten Akteure werden mit einem einfachen Strich ohne Pfeilspitzen mit der Aktivität verbunden.

Zur Notation von Teilaktivitäten:

Sollen aggregierte Teilaktivitäten durch grafisches Enthaltensein ausgedrückt werden, kann die äußere Aktivität auch durch eine gestrichelte "Freihand"-Linie notiert werden. Alle Akteure der übergeordneten Aktivität sind dann implizit auch Akteure aller aggregierten Teilaktivitäten, die jedoch zusätzlich noch neue Akteure auszeichnen können.

Wird eine normale Aggregationsbeziehung (mit Raute) zur Notation verwendet, so "erbt" diese Aktivität auch alle beteiligten Akteure. Wenn die Teilaktivität mit eigenen Akteuren notiert wird, so stellen diese Akteure die alleinigen Beteiligten der Teilaktivität dar.

Bezug zum UML-Metamodell

Im UML-Metamodell gibt es keine direkte Entsprechung für eine KoA_LA-Aktivität. Eine KoA_LA-Aktivität mit *einem* Beteiligten stellt eine Definition eines SubactivityState-Typs dar. In UML gibt es eine solche Metaebene nicht, es existiert kein "SubactivityStateType".

Im direkten Vergleich mit der UML-Nomenklatur sollte man daher KoA$_L$As Aktivitäten evtl. lieber als "Aktivitätsbereiche" bezeichnen, da sie zwar deskriptiv über ihre Ziele oder auch imperativ über Beispiel-Aktivitätsmuster näherungsweise definiert werden, jedoch im Unterschied zu UML's Activities keine konkrete Tätigkeitsausprägung spezifizieren. Aktivitätsbereiche definieren vielmehr Handlungsziele, die über eine Menge alternativer realer Aktivitätssequenzen erreicht werden können.

Eine Aktivität mit *mehreren* Beteiligten stellt analog die Definition eines UseCase-Typs dar.

Eine Aktivität im Sinne von KoA$_L$A umfasst alle gemeinsamen Tätigkeiten, die zur Erreichung eines Aktivitätsziels führen können ("Metatyp"-Ebene). Ein bestimmter Handlungsplan (UseCase = Aktivitätsmuster) umschreibt eine von vielen möglichen Umsetzungen und definiert damit eine bestimmte Untermenge ("Typ"-Ebene) aller möglichen zielorientierten Tätigkeitssequenzen. Ein Szenarium beschreibt hingegen den konkreten Ablauf einer bestimmten realen Tätigkeitssequenz ("Instanz"-Ebene).

A.8. Interaktion

Verwandte Begriffe: Dialog, Kommunikation.

Eine Interaktion ist eine Kooperation mehrerer Akteure, die von einem bestimmten Akteur initiiert wird, und dem Zweck der Ausübung einer bestimmten Wirkung auf die Interaktionspartner dient. Sie umfasst alle interaktionsbezogenen Aktivitäten der Partner, die während dieser Kooperation auftreten.

Eine Interaktion stellt eine Kooperation dar, die in den Modellen von KoA$_L$A nicht mehr weiter in lokale Aktivitäten von Akteuren verfeinert, sondern höchstens in Einzelinteraktionen zerlegt wird[2]. Eine Interaktion stellt damit die "reinste" Form einer Kooperation dar, nämlich eine Aktivität, die keine getrennten Aktivitätsanteile einzelner Akteure mehr subsumiert und sich vollständig nur im Zusammenspiel aller Beteiligten realisieren kann. Eine gemeinsame *Aktivität* hingegen kann immer sowohl in Anteile lokaler Aktivitäten (Arbeitsteilung!) als auch in Interaktionen (Kommunikation!) weiter zerlegt werden, die die Kooperation erst ermöglichen.

Attribute

Ziel	Die pragmatische Funktion der Interaktion.
Beschreibung	Optionale Beschreibung des Verlaufs der Interaktion

[2]Wenn eine solche Verfeinerung dennoch gewünscht wird, so kann man die Interaktion als Kooperation – also als Aktivität – modellieren. Einer Zerlegung einer Interaktion wie z.B. "Student schickt Hauptantrag per Post an das PJ-Büro" in die Einzelhandlungen "Wähler steckt ausgefülltes Bewerbungsformular in einen Briefumschlag, frankiert und adressiert ihn etc." und "PJ-Büro besorgt Post, öffnet Umschlag und liest Inhalt etc." steht dann nichts mehr im Wege.

Beziehungen

Sender	Der die Interaktion initiierende Akteur.
Empfänger	Alle Akteure, die als Adressaten der Interaktion auftreten.
Beteiligte	Akteure, die in der Interaktion nur als Kommunikationsmittel, Medium oder Werkzeug benutzt werden.

Grafische Notation

Interaktionen werden als Verbindungspfeile mit ausgefüllter Spitze dargestellt, die vom Initiator ausgehen und zu einem oder mehreren Interaktionspartnerm führen.

Ein Interaktionspfeil kann mit mehreren Zusatzinformationen annotiert werden:

Ziel: Die pragmatische Funktion der Interaktion wird als textuelle Beschriftung des Pfeils notiert.

Inhalt: Die in der Kommunikation vermittelten Konzepte (Objekte, Begriffe, Dokumente etc.) werden in der Nähe der Rezipienten in Form unterstrichener Begriffsnamen vermerkt.

Während einer Interaktion können Inhalte immer in beide Richtungen, d.h. zu einem der Interaktionspartner, aber auch zum Initiator zurück "fließen". Um Interaktionen und Konzepte physikalischer Beschaffenheit zu verdeutlichen können auch Pictogramme verwendet werden. Die in der Kommunikation thematisierten Begriffe können dabei auch als "Zugriffspfad" ("<Begriff>.<Merkmal>.<Merkmal> ...") angegeben werden.

Partner: Im Falle einer 1:1-Kommunikation führt ein Interaktionspfeil direkt vom Initiator zum Empfänger. Im Falle einer 1:n-Kommunikation führt ein Strich vom Initiator zur Mitte einer Seite der gefüllten Dreiecksspitze, während die gegenüberliegende Spitze des Dreiecks mit allen Empfängern verbunden wird. Ist der Initiator nicht eindeutig festzulegen (echte Kooperationen, Dialoge, n:1-Kommunikationen) oder will man das nicht, wird diese Beziehung nicht als Interaktion, sondern als Kooperation modelliert. Handelt es sich um eine "interne" bzw. reflexive Kommunikation (Erkenntnisprozesse, Ereignisse), so bleibt die Stelle des Interaktionspartners frei.

Resourcen: Akteure, die in der Interaktion nur als Kommunikationsmittel, Medium oder Werkzeug benutzt werden, werden mit einer der beiden restlichen Dreiecksseiten verknüpft, die nicht mit dem Initiator in Verbindung stehen.

Randbedingungen: Als normale Aktivität kann eine Interaktion auch bestimmten Randbedingungen unterliegen. Interessant sind hier z.B. die Häufigkeit, die zeitliche Dauer, die zeitlichen Abstände einer Interaktion etc. Diese Randbedingungen lassen sich als Bedingung modellieren. Auf diese Weise lassen sich im Diagramm optional auch Gestalt und Form der Interaktion festlegen. Bedingungen von Interaktionen werden notiert, indem sie (in geschweiften Klammern gesetzt) direkt neben dem Interaktionspfeil vermerkt werden.

Stereotypen

Auftrag Interaktion, die einen Auftrag übermittelt. Die Interaktion wird mit dem Auftrag gefolgt von einem Ausrufezeichen beschriftet und optional in doppelte Hochkommazeichen gesetzt (Bsp: "Hauptverteilung durchführen!"). Alternativ kann auch das ausgefüllte Dreieck mit dem Ausrufezeichen versehen werden.

Bezug zum UML-Metamodell

Subsumiert MESSAGE und STIMULUS, ist jedoch viel "breiter" angelegt. Kann auch vollständige Kooperationsbereiche modellieren, ohne deren Ablauf zu definieren.

A.9. Aktivitätsmuster

Verwandte Begriffe: Handlungsplan, Prozess, Use-Case.

Aktivitätsmuster beschreiben mögliche Aktivitätssequenzen.

Sie beschreiben in der Regel einen systemischen Aktivitätsbereich, d.h. eine Sequenz von – möglicherweise auch nebenläufigen – Aktivitäten, die sich typischerweise oft wiederholt.

Als Sonderfall kann ein Aktivitätsmuster auch eine streng sequentielle und deterministische zeitliche Abfolge von Einzelaktivitäten darstellen – ein Szenarium.

Attribute

Ablauf	Textuelle Erläuterung des Ablaufs des Aktivitätsmusters

Beziehungen

Wechsel	Ein Aktivitätsmuster besteht aus einer beliebigen Menge von Zustandswechseln
Zustände	Die Aktivitäten und Zustände des Aktivitätsmusters.

Grafische Notation

Wird als Aktivitäts- oder Konversationsdiagramm dargestellt.

Eine Konversation ist ein Aktivitätsmuster, das nur aus Interaktionen besteht. Jedes Konversationsdiagramm kann in ein Aktivitätsdiagramm transformiert werden, das nur Interaktions-Kooperationen mit jeweils zwei Beteiligten kennt und die Akteure des Konversationsdiagramms durch entsprechende Zustände (Aktivitätspotentiale) ersetzt.

Bezug zum UML-Metamodell

Vergleichbar mit einer Vereinigung von ActivityGraph, Interaction und UseCase.

A.10. Wechsel

Verwandte Begriffe: Zustandsübergang, Kontrollfluss.

Eine Wechsel kennzeichnet den Übergang von einem Zustand bzw. einer Aktivität in einen neuen Zustand bzw. eine neue Aktivität.

Beziehungen

Von	Zustand oder Aktivität, von der aus gewechselt wird
Nach	Zustand oder Aktivität, zu der gewechselt wird.

Grafische Notation

Einfacher Verbindungspfeil zwischen Zuständen oder Aktivitäten.

Ein solcher Wechsel kann auch zusammen mit einem “Datenfluss” einhergehen. Ein solcher Fluss ist äquivalent zu zwei “internen” bzw. anonymen Interaktionen (die Aktivität, von der aus gewechselt wird besitzt eine ausgehende Interaktion ohne Interaktionspartner; die Aktivität, zu der gewechselt wird besitzt eine eingehende Interaktion ohne Initiator).

Bezug zum UML-Metamodell

Vergleichbar mit Transition.

B. Beispiele für die semiformale textuelle Repräsentation

Der Vollständigkeit halber wird nun noch eine Auswahl semiformaler Dokumenttypen der Projektdatenbank vorgestellt, die benutzt werden können, um die vorgestellten KoA$_L$A-Diagramme zu erläutern oder die Modelle von KoA$_L$A rein textuell zu definieren.

Die verwendete Syntax ist dabei als Beispiel zu verstehen, wie man die verschiedenen Modellelemente von KoA$_L$A alternativ oder ergänzend zur grafischen Notation auch in textueller Form beschreiben kann.

Für die im Folgenden vorgestellten Dokumenttypen wurde eine Softwareentwicklungsumgebung geschaffen ("xSeu"), mit der man

- diese Dokumente syntaktisch verifizieren kann
- verschiedene Prüfungen zur Qualitätssicherung (Konsistenz, Vollständigkeit etc.) durchführen kann
- die Dokumente in Form eines Hypertexts aufbereiten, flexibel einsehen und bearbeiten kann
- verschiedene Ausgabeformate (html, latex, rtf) erzeugen kann
- Rümpfe für die spezifizierten Klassentypen in verschiedenen Programmiersprachen (Delphi, Java) generieren kann

Ein späterer Ausbau dieser Umgebung um grafische Editoren zur integrierten Erstellung entsprechender KoA$_L$A-Diagramme ist angedacht.

Die rein textuelle Darstellungsweise – überwiegend strukturierter natürlichsprachlicher Fließtext – hat sich in Projekten vor allem deshalb bewährt, weil sie oft keine ausführliche Einführung in die verwendete Darstellungsform erforderlich macht. Ihr Inhalt erschließt sich damit sofort auch "unbedarften" Anwendern und kann von Ihnen leicht validiert bzw. korrigiert werden.

Auch aus Entwicklersicht hat sich dieses Format in der täglichen Praxis der Dokumentation und Fortentwicklung der KoA$_L$A-Modelle bewährt; nicht zuletzt auch deshalb, weil neben den Dokumenten auch das Dokumentationsschema selbst leicht änderbar und flexibel erweiterbar ist.

Diagramme hingegen besitzen ihren großen Vorteil in der kompakten Darstellung von Zusammenhängen und zeigen ihre Stärke vor allem in gemeinsamen Arbeitstreffen, die mittels dieser grafischen Artefakte gut fokussiert werden können.

B.1. Wissensmodell

Im Folgenden nun zunächst ein Beispiel für ein DIC-Dokument, indem die Begriffswelt von Apvel textuell dokumentiert wird:

```
BEGRIFF         PraktischesJahr = PJ
BESCHREIBUNG    Ein Praktisches Jahr ist ein 48-wöchiger Ausbildungsabschnitt,
                der von Medizinstudenten im dritten klinischen
                Studienabschnitt absolviert werden muss. Ein Praktisches Jahr
                wird in drei 16-wöchigen zusammenhängenden Studienabschnitten
                durchgeführt (Ausbildungsfolge), den Tertialen.
ASPEKT          Zeitraum
                Fast ein ganzes Jahr
ASPEKT          Tertiale: Tertial[3]
                besteht aus drei Tertialen zu je 16 Wochen
ASPEKT          Antragsteller: Student[*]
                Für ein PJ bewerben sich in der Regel bis zu 200 Studenten

BEGRIFF         Krankenhaus = Lehrkrankenhaus = Krk = Ausbildungsort = ALK
                Ein akademisches Lehrkrankenhaus, das an die Universität
                angeschlossen ist. Die wirklich zur Verfügung stehenden Krk
                und deren Fächerangebot sind Gegenstand der jährlich
                stattfindenden Vertragsverhandlungen und ändert sich darum von
                PJ zu PJ:
                Biberach:     Anästhesie, Gynäkologie (nur zum Eingangstermin
                              Herbst), Orthopädie, Urologie;
                Göppingen:    Anästhesie, Gynäkologie, Orthopädie, Pädiatrie;
                Heidenheim:   Anästhesie, Gynäkologie, Neurologie,
                              Pädiatrie, Radiologie, Urologie;
                ...

ASPEKT          Name
ASPEKT          Vertragskapazität
                Bezeichnet die Anzahl der Ausbildungsplätze, die ein
                Krankenhaus zur Verfügung stellt.
ASPEKT          Adresse
                Notwendig für Anschreiben zur Anmeldung zum 3. Staatsexamen
                (Etiketten). Diese gehen an die Arbeitsstelle.

BEGRIFF         Klinikum
                Das Universitäts-Krankenhaus in Ulm
BEISPIELVON     Krankenhaus

BEGRIFF         Student = Medizinstudent = Stud
                Ein Student der Medizin, der Anwärter auf ein Praktisches Jahr
                ist, also einen Antrag auf Zuteilung einer Praktikumsstelle
                gestellt hat.
ASPEKT          Name
ASPEKT          Antrag[*]
                Die eingereichten Anträge. Im Normalfall nur ein Antrag.
                Ein Student kann aber eine Zuteilung ablehnen und sich für
                folgende PJ neu bewerben
ASPEKT          Matrikelnummer
ASPEKT          Sozialfaktor
                Die Wünsche eines Studenten werden abhängig von seiner
                sozialen Situation unterschiedlich berücksichtigt.
                1: Schwerbehindert
```

```
                2: Familie am Ausbildungsort
                3: Wohnsitz mit Ehegatten am Ausbildungsort
                4: Härtefall
                5: Keine Bevorzugung
ASPEKT          KrkLos
                Losnummer für die Auswahl des Wunschkrankenhauses.
                Wird für jeden Antragsteller nach dem Zufallsprinzip
                festgelegt und bestimmt die Chancen der Zuteilung des
                Krankenhauses.
ASPEKT          AbtLos
                Losnummer für die Auswahl der Wunschabteilung.
                Wird für jeden Antragsteller nach dem Zufallsprinzip
                festgelegt und bestimmt die Chancen der Zuteilung der
                Wunschabteilung.
ASPEKT          KrkPräferenz:(Priorität) Krankenhaus[0..3]
                Ein geordnete Wunschliste von bis zu drei Krankenhäusern.
                Erstes Krankenhaus ist das eigentliche Wunschkrankenhaus.
                Dieser Wunsch ist der sogenannte Hauptantrag. Die anderen sind
                die sogenannten "Hilfsanträge".
ASPEKT          AbtPräferenz(krk: Krankenhaus):(Priorität) Abteilung[0..3]
                Die in den bevorzugten Krankenhäusern gewünschten Abteilungen.
                Wunschliste ist geordnet. Erste Abteilung ist
                Lieblingsabteilung. Gewünscht werden hier nur das
                Wahlfach. Innere Medizin und Chirurgie sind sowieso
                Pflichtprogramm.
                Zusätzlich zu den Abteilungen des Krk können jetzt neu auch
                beliebige Abteilungen des Ulmer Klinikums gewünscht werden.
ASPEKT          Tertialvorgaben:
                (Tertial, Auslandsaufenthalt?, frei?) Fachart [0..3]
                Bis zu zwei Tertiale können im Ausland verbracht werden.
                Desweiteren kann ein Student auch vom Tertial freigestellt
                werden. Beispielsweise wegen Schwangerschaft oder weil er das
                entsprechende Fach schon in einem anderen PJ abgeleistet hat.
ASPEKT          Reihenfolgewunsch
                Ausgesuchte Studenten dürfen sich wünschen, in welchen
                Tertialen sie die unterschiedlichen Fächer ableisten wollen.
                Momentan sind das nur Alleinerziehende/Schwangere. Präferenzen
                ergeben sich z.B. aus folgenden Bedingungen:
                Innere Medizin: relativ unblutig -> Infektionsgefahr minimiert
                Chirurgie: Stressig (Arbeitszeiten)
                Anästhesie: Betäubungsgase sind für Ungeborene unzuträglich
ASPEKT          Zuteilung(Tertial):(Krk) Abt
                Pro Tertial ist genau eine Abt eines bestimmten Krk zugeteilt
```

B.2. Organisationsmodell

Als Beispiel der textuellen Beschreibung eines Tätigkeitsprofils nun ein Ausschnitt aus dem AGT-Dokument zur Dokumentation des Profils eines Wahlleiters (Projekt “Apvel”):

```
AGENT           Wahlleiter
BESCHREIBUNG    Person, die die gesamte Wahl, deren Vorbereitung und deren
                Auszählung kontrolliert und überwacht. Insbesondere obliegt
                dem Wahlleiter die technische Vorbereitung und Durchführung
                der Wahlen.
                Dezernat V Abteilung 2: Innerer Dienst
```

```
                Grüner Hof

ANFORDERUNG     Auslegungstermin
                Das Wählerverzeichnis muss bis spätestens 29 Tage vor
                der Wahl durch die Wahlleitung öffentlich aufgelegt werden.
                Dieser Termin ist unbedingt zu beachten, da sonst die ganze
                Wahl verschoben werden muss!

ANFORDERUNG     Öffentlichkeit
                Die Ermittlung und Feststellung der Abstimmungs- und
                Wahlergebnisse hat öffentlich zu erfolgen

ANFORDERUNG     Abstimmungszeitraum
                Die Wahl soll innerhalb eines Semesters
                durchgeführt werden. Die Abstimmung innerhalb der
                Vorlesungszeit.

AUFGABE         Wahlorganisation
                Durchführung von Wahlen
=               Sicherung | (Ankündigung, Bekanntmachung_Wahlvorschläge) |
                (Wählerverzeichnisorganisation, Auslegung,
                Berichtigung, Abschluss) |
                Aushang_Kandidaten | Entscheidet_Notwendigkeit |
                Legt_Wahlverfahren_fest | Gibt_Briefwahlunterlagen_heraus |
                Nimmt_Briefwahl_entgegen |
                !Abstimmungsausschuss.Ermittelt_Wahlergebnis*,
                Gibt_Ergebnis_bekannt

AUFGABE         Ankündigung
                Die Wahl ist spätestens am 35. Tag vor der Wahl öffentlich
                bekannt zu machen.
=               Ausschreibung

AUFGABE         Wählerverzeichnisorganisation
                Ist verantwortlich für die Erstellung des Wählerverzeichnisses
                Die Form ist in Paragraph 6(2) genauestens festgelegt
=               Erstellt_Wählerverzeichnis

HANDLUNG        Ausschreibung
                Gibt Wahl öffentlich bekannt.
                Incl. Räume, Termine, Amtsperiode, Sitze etc.
PARTNER         Aushang
=               !Drucker.Druckt

HANDLUNG        Stammdatenpflege
BESCHREIBUNG    Stammdatenpflege, Änderungen von Datenbeständen
=               LegtWahlAn | ÖffnetWahl | LöschtWahl,
                LegtWählergruppeAn | LöschtWählergruppe |
                LegtGremiumAn | LöschtGremium
VARIANTE        WahltypBekannt
                Falls die gleiche Art von Wahl bereits durchgeführt wurde,
                kann man die Stammdaten aus einem alten Wahlgang kopieren
=               KopiertWahl

HANDLUNG        BeruftWahlhelfer
PARTNER         Wahlhelfer, System
BESCHREIBUNG    Der Wahlleiter beruft die Wahlhelfer, die dann bei der Wahl
```

```
                selbst, bei der Auszählung o.ä. helfen. Wahlhelfer können
                keine Unterstützer oder Kandidaten sein.
                Sie werden möglichst auf freiwilliger Basis verpflichtet.
=               Wahlhelfer.WahlhelferHinzufügenÄndern|
                [Wahlhelfer.WahlhelferLöschen]

HANDLUNG        Erstellt_Wählerverzeichnis
                Kümmert sich um die Erstellung des Wählerverzeichnisses
                (Beantragung bei der __AGENT.Verwaltung), beurkundet
                dessen Richtigkeit und Vollständigkeit (Paragraph 6(4)), führt
                die Auslegung durch, macht diese bekannt (kann
                auch zusammen mit der Ausschreibung erfolgen),
                beurkundet Tag und Art der Bekanntmachung am Ende des
                Verzeichnisses, berichtigt auf Anfrage/Anträge hin Fehler
                und schließt es am 15.Tag endgültig ab
PARTNER         Verwaltung
                Wählerverzeichnis organisieren
PARTNER         Wähler
                Berichtigungen
PARTNER         Aushang
                Auslegung
PARTNER         System
                Import der Datei
=               !Verwaltung.Wählerverzeichnis, Import.OK,
                Berichtigt_Wählerverzeichnis
AUSNAHME        Klinik
                Da die Klinik seit neuestem eine eigene Personaldatenverwaltung
                durchführt, müssen die relevanten Daten dort möglichst
                frühzeitig angefordert werden.

HANDLUNG        Berichtigt_Wählerverzeichnis
                mit Datum und Unterschrift.
=               Wählerverzeichnis.PersonHinzufügenÄndern |
                Wählerverzeichnis.PersonAusWählerverzeichnisLöschen
AUSNAHME        Inkonsistenz
                Eine Berichtigung des Wählerverzeichnisses kann bis kurz vor
                der Wahl erforderlich sein. Falls dadurch Inkonsistenzen
                entstehen (Kandidat eines Wahlvorschlags ist nicht mehr
                wahlberechtigt), wird der Anwender daraufhingewiesen und die
                Berichtigung nicht durchgeführt. Die Berichtigung wird erst
                akzeptiert, wenn der Anwender die gemeldeten Inkonsistenzen
                beseitigt hat.
HANDLUNG        LegtWählerverzeichnisseAusUndGibtDiesBekannt
PARTNER         Kandidat
BESCHREIBUNG    -> Paragraph 7 (1),(2)
=               !Drucker.Druckt

HANDLUNG        Ermittelt_Endergebnis
                Organisiert Auszählung und berechnet Endergebnis
                (Verteilung der Sitze) entsprechend dem angewendeten
                Wahlvefahren (-> Paragraph 29 und 31, deHondt'sches oder
                Mehrheitswahl)
PARTNER         Abstimmungsausschuss, System
=               !Abstimmungsausschuss.Ermittelt_Wahlergebnis,
                GibtAuszähllistenEin,
                Ergebnis.WählergruppenwahlAuswahl
AUSNAHME        Würfeln
```

```
                Es gibt Fälle, in denen von Hand gelost werden muss,
                welcher Wahlvorschlag einen Sitz erhält oder welcher Kandidat
                gewählt ist. Hier erhält der Anwender selbst die Möglichkeit,
                das Ergebnis der Auslosung einzugeben. Auch Rückblickend ist
                erkenntlich, welche Kandidaten aufgrund eines Losentscheids
                einen Sitz erhalten haben!

HANDLUNG        Gibt_Ergebnis_bekannt
                Öffentlich und den gewählten Kandidaten. Siehe Paragraph 32.
                Erfolgt innerhalb 14 Tage kein Einspruch gilt die
                Wahl von den Gewählten als angenommen.
PARTNER         Wähler, Aushang, Kandidat, Drucker
                werden per Aushang oder Brief informiert
=               Grauwal.Ergebnisbekanntmachung
```

B.3. Kooperationsmodell

Abschließend nun ein Beispiel für ein PRJ-Dokument aus dem Projekt "Grauwal", das neben Erläuterungen zum Projektkontext hauptsächlich Beschreibungen zentraler Aktivitätsmuster beinhaltet:

```
PROJEKT         Grauwal
BESCHREIBUNG    Steht für _Gr_emien_A_ssistenzsystem zur D_u_rchführung von
                _W_ahlen. System zur Vorbereitung, Durchführung und Auszählung
                bzw. Dokumentation der Ergebnisse verschiedener
                universitätsweiter Wahlen (Universitätsrat, Senat,
                Fakultätsrat) und verwaltungsinterner Wahlen (Personalrat,
                Frauenvertreterin, Behindertenvertreter...)

MOTIVATION      Grauwal soll das Dezernat "Innerer Dienst" (V/2)
                bei der Durchführung von Gremienwahlen unterstützen.
                Insbesondere steht dabei der Wunsch des Kanzlers im
                Hintergrund, dass Wahlwiederholungen aufgrund von
                Verfahrensfehlern durch eine geeignete
                Systemunterstützung möglichst verhindert werden.

VISION          Ein Assistenzsystem, das sowohl die Vorbereitungs- als
                auch die Durchführungsphase von Wahlen an der Uni möglichst
                gut unterstützt.

VORGANG         Wahl
                Der Ablauf einer Wahl ist im Wahlgesetz konkret festgelegt.
                Hierbei sind insbesondere die einzuhaltenden Fristen unbedingt
                einzuhalten, da die Wahl sonst ungültig ist.
                Nachdem der Wahltermin feststeht, wird die Wahl
                ausgeschrieben.
                Mit Bekanntgabe der eingereichten Wahlvorschläge beginnt auch
                die Auslegung des Wählerregisters und die Organisation der
                eigentlichen Wahl. Nach der Abstimmung in den Wahllokalen wird
                das Ergebnis ausgezählt, öffentlich bekanntgegeben und noch
                einmal überprüft.
=               Rektor.Veranlasst_Wahl,
                Wahlleiter.BestimmtWahltermin, Ausschreibung,
                Wahlleiter.Entscheidet_Notwendigkeit,
                [ Ankündigung,
```

```
                  (( Vorbereitung, Durchführung) ||
                   ( BriefwahlBewerbung, Briefwahl)*),
                  Auszählung
                ],
                (Veröffentlichung || Kontrolle)

VORGANG         BriefwahlBewerbung
                Wähler kommt persönlich vorbei oder stellt schriftlich
                einen Briefwahlantrag. Die Adressdaten des Wählers werden mit
                den in Grauwal bereits vorhandenen (Import!) verglichen und
                sein Briefwahlwunsch vermerkt. An einem Tag X wird dann
                einmalig für alle Briefwähler ein "Wählerschein"-Serienbrief
                generiert, der dann zusammen mit den PASSENDEN Stimmzettel(n)
                und Wahlumschlägen eingetütelt und per Post verschickt wird.
                Hierzu muss für jeden Briefwähler extra überprüft werden, für
                welche Gremien er wahlberechtigt ist (ist evtl. im Feld
                Bemerkungen des importierten Wählerverzeichnisses
                eingeschränkt), diese Wahlmöglichkeit im Wahlschein per Hand
                angekreuzt werden und der passende Stimmzettel eingetütelt
                werden. Der Wählerschein beinhaltet die Adresse, die im
                Klarsichtfenster des Briefumschlags sichtbar ist.
PARTNER         Wähler, Wahlleiter, System
=               Wahlleiter.Gibt_Briefwahlunterlagen_heraus
VARIANTE        SofortigeAbstimmung
                Es kommt auch vor, dass Briefwähler persönlich vorbeikommen
                und sofort wählen. Hier kommt es also auf eine Möglichkeit an,
                die Wahlunterlagen auch einzeln für einen Wähler schnell
                erstellen zu können

VORGANG         Briefwahl
                Nachdem der Wähler seine Wahlunterlagen per Post bekommen hat,
                kreuzt er auf seinen Stimmzetteln seine gewählten Kandidaten
                an, steckt diese in die passenden Wahlumschläge und schickt
                sie per Post an die Wahlleitung. Am Auszähltag werden diese
                Wahlbriefe geöffnet und noch VOR der eigentlichen Auszählung
                in die passenden Urnen geworfen, in denen bereits alle
                "ZuFußWähler"-Stimmzettel in den passenden Wahlumschlägen ihr
                Dasein fristen.
=               Wähler.TätigtBriefwahl, Wahlleiter.Nimmt_Briefwahl_entgegen
PARTNER         Wähler, Wahlleiter

VORGANG         Ausschreibung
                Die Wahl selbst wird bekanntgegeben und zur Einreichung von
                Wahlvorschlägen aufgefordert. Sowohl Termin der Wahl, als auch
                der Termin des Auslegung des Wählerregisters wird
                angekündigt.
=               Wählerverzeichniserstellung ||
                Wahlleiter.LegtWählerverzeichnisseAusUndGibtDiesBekannt

VORGANG         Wählerverzeichniserstellung
BESCHREIBUNG    Das Rechenzentrum liefert der Wahlleitung die gesamte Liste
                der Wahlberechtigten mit entsprechender Zuordnung zu Fakultät
                und Wählergruppe. Danach stellt der Wahlleiter das
                Wählerverzeichnis auf.
PARTNER         Verwaltung, Wahlleiter, System
=               Verwaltung.Erstellt_Wählerverzeichnis,
                Wahlleiter.Erstellt_Wählerverzeichnis
```

```
VARIANTE        Personalratswahlen
                Bei Personalratswahlen muss das Register evtl. vom Wahlleiter
                selbst erstellt werden!

VORGANG         Ankündigung
                Die Wahlvorschläge und der genaue Ort und Termin der Wahl wird
                veröffentlicht.
PARTNER         Wahlleiter, System
=               Wahlleiter.Bekanntmachung_Wahlvorschläge

VORGANG         Vorbereitung
BESCHREIBUNG    Vor der eigentlichen Wahl müssen im Wesentlichen
                das gültige Wählerverzeichnis  und die beschlossenen
                Wahlvorschläge vorliegen und das notwendige Wahlverfahren
                festgelegt sein.
                Die Wahlvorschläge werden von Kandidaten & ihren Unterstützern
                beim Wahlleiter eingereicht
=               ( Unterstützer.UnterschreibtWahlvorschlag*,
                  Kandidat.Bestaetigt_Kandidatur,
                  Unterstützungsgruppe.Vorschlag_einreichen,
                  Wahlleiter.Stammdatenpflege
                )
                ||
                ( Wahlleiter.BearbeitetWahlvorschläge,
                  Wahlausschuss.Beschließt_Wahlvorschläge,
                  Wahlleiter.DrucktStimmzettel )
                ||
                (Wähler.Informiert_sich_über_Wahl |
                 Wähler.Beantragt_Berichtigung |
                 Unterstützungsgruppe.Erkundigt_sich |
                 Verwaltung.Informiert_über_Änderungen)*
                || Wahlleiter.BeruftWahlhelfer
                || Wahlleiter.Technische_Vorbereitung
PARTNER         Wahlleiter, Unterstützer, Kandidat, Wahlausschuss, Wähler,
                Verwaltung, System
AUSNAHME        KandidatAlsUntertützer
                Kandidaten können auch selbst als Unterstützer auftreten.
                In diesem Fall ist der Kandidat oft auch Einreicher des
                Wahlvorschlags
=               Unterstützer.UnterschreibtWahlvorschlag*,
                Unterstützungsgruppe.Vorschlag_einreichen
AUSNAHME        WahlvorschlagÄndern
                Ein Wahlvorschlag kann bis zum Stichtag beliebig
                nachgebessert werden (z.B. auch bei Formfehlern)
=               Unterstützungsgruppe.Vorschlag_ändern*

VORGANG         Durchführung
                Die eigentliche Wahl wird von den Wahlhelfern durchgeführt.
                Neben der Abstimmung im Wahllokal, kann jeder Wähler auch eine
                Briefwahl beantragen und durchführen.
                Die Auszählung selbst findet im Allgemeinen einen Tag nach der
                Wahl in einem eigenen Raum statt, wird von der Wahlleitung
                beaufsichtigt und von dieser zusammen mit allen Wahlhelfern
                durchgeführt.
=               Wahlleiter.Legt_Wahlverfahren_fest,
                Wahlleiter.Technische_Vorbereitung,
                Abstimmungsausschuss.Wahl_Ermöglichen,
```

```
                Abstimmung,
                Abstimmungsausschuss.Ermittelt_Wahlergebnis

VORGANG         Auszählung
PARTNER         Wahlleiter, Wahlhelfer, System
BESCHREIBUNG    Beschreibung des Verfahrens der Auszählung:
                Die Auszählung aller Wahlen des Wahltags findet
                hochschulöffentlich statt.
                Hierzu treffen sich alle Abstimmungsausschüsse (Wahlhelfer)
                - im Folgenden "Auszählgruppe" genannt - zusammen mit der
                Wahlleitung in einem eigenen Raum, in dem vom Fahrdienst schon
                alle verschlossenen Urnen abgestellt wurden.
                Jede Auszählgruppe ist nun für die Auszählung aller
                Stimmzittel aller Urnen eines bestimmten Wahllokals
                (normalerweise ein Lokal pro Fakultät) zuständig.
                Zunächst werden alle Briefwahl-Briefe geöffnet und deren
                Inhalt (Briefwahlumschläge) in die zugehörigen Urnen geworfen
                (Jeder Wähler ist durch seine Fakultätszugehörigkeit genau
                einem Wahllokal zugeordnet und steht auch dort im
                Wählerverzeichnis!).

                Nun werden von den Auszählgruppen zunächst die
                Wählergruppenwahlen ausgezählt, welche ein
                Fakultätsübergreifendes Gremium wählen (Hochschulrat
                oder Senat).

                Mit Hilfe von Hilfszähllisten (Strichlisten) wird hier
                - die Anzahl der Stimmen pro Kandidat
                - die Anzahl ungültiger Stimmen
                - die Gesamtsumme der gültigen Stimmen (pro Wahlvorschlag?)
                - die Summe der Fehlstimmen (Stimmen, die nicht gegeben
                  wurden; diese Zahl wird nicht weiter ausgewertet und kann
                  daher auch entfallen).
                bestimmt und zusammen mit
                - Anzahl Wahlberechtigte (ersichtlich aus dem
                  Wählerverzeichnis)
                - Anzahl Wähler (Summe aller abgehakten wirklichen Wähler aus
                  dem Wählerverzeichnis incl. Briefwähler)
                - Anzahl gültiger Stimmzettel
                - Anzahl gültiger Stimmen
                - der Wahlbeteiligung (Anzahl Wähler / Anzahl Wahlberechtigte)
                in die "Niederschrift des Abstimmungsausschusses über die
                Ermittlung des Abstimmungsergebnisses" der Wählergruppenwahl
                übertragen, von allen Mitgliedern der Auszählgruppe
                unterschrieben und zusammen mit den ausgefüllten Auszähllisten
                zur Wahlleitung gebracht.
                Im Regelfall reichen zur Bestimmung der Stimmen pro Kandidat
                ein bis zwei Auszähllisten. In Sonderfällen (Studentenwahlen)
                sind mehrere Auszähllisten notwendig, so das ein eigenes Blatt
                zur Berechnung der Stimmensummen für die Kandidaten notwendig
                wird.

                Dies machen die Auszählgruppen auch weiterhin ohne
                Grauwal-Unterstützung per Hand!

                Die Wahlleitung fasst nun die Ergebnisse aller Auszählgruppen
                bezogen auf ein Gremium (Hochschulrat oder Senat) zusammen
```

```
                (Summenbildung), wertet die Zahlen aus (d'Hontsches
                Auswertungsverfahren) und erstellt die "Niederschrift des
                Wahlauschusses über die Feststellung des Wahlergebnisses".

                Im Folgenden werden dann nach dem gleichen Muster die
                Fakultätsratswahlen ausgezählt und entsprechende
                Niederschriften erstellt.

                Genau die Zusammenfassung bzw. Auswertung der
                Auszählgruppenergebnisse und die Erstellung der "Niederschrift
                des Wahlauschusses" wird von Grauwal unterstützt bzw.
                durchgeführt.
=               Abstimmungsausschuss.Ermittelt_Wahlergebnis,
                Wahlleiter.Ermittelt_Endergebnis

VORGANG         Abstimmung
                Wähler (= Wahlberechtigten) können in ihrem nach Wählergruppe
                und Fakultät zugeordneten Wahlraum wählen. Der Wähler betritt
                den Wahlraum und legitimiert sich bei den Wahlhelfern (per
                Studentenausweis oder per Abgleich mit dem Wählerverzeichnis).
                Dabei wird insbesondere überprüft, ob er schon gewählt hat
                (persönlich oder per Brief) und seine Wahl vermerkt (um die
                Anzahl der Umschläge in der Wahlurne zu kontrollieren, werden
                am Ende des Abstimmungszeitraums alle Wahlberechtigten
                gezählt, die auch tatsächlich gewählt haben). Daraufhin erhält
                er Stimmzettel und Wahlumschläge für alle Wahlen, zu denen er
                berechtigt ist. Im geheimen Kämmerlein (Wahlkabine) kommt es
                nun zum Äußersten: der Wahl. Danach darf er nun seine
                Briefelchen in die passenden Urnen stecken. Wenn er nun alles
                richtig gemacht hat, wird er mit einem Mohrenkopf belohnt ;-)
PARTNER         Wahlhelfer, Wähler
=               Wähler.Wählt

VORGANG         Veröffentlichung
BESCHREIBUNG    Der Wahlleiter gibt das Ergebnis der Wahl öffentlich bekannt.
PARTNER         Wahlleiter, System
=               Wahlleiter.Gibt_Ergebnis_bekannt

VORGANG         Grauwalnutzung
                Das allgemeine Nutzungszenario von Grauwal sieht
                folgendermaßen aus:

                1. Gremien, Wählergruppen und Checkliste erfassen oder von
                   letztem Wahlgang kopieren; Wahltermin eingeben;
                2. Import aller Wahlberechtigten;
                3. Wahlvorschläge eingeben; Bekanntmachung der Wahlvorschläge;
                4. Stimmzettel generieren; Wahlhelfer mit Stimmzetteln,
                   Wahlumschlägen, Wählerverzeichnissen bestücken
                5. Wahl durchführen lassen (Wahlhelfer);
                6. Auszählungsergebnisse einpflegen (neu genannte Kandidaten
                   übernehmen);
                7. Minderheitenschutz per Hand vornehmen;
                8. Wahlergebnis dokumentieren (lassen) und veröffentlichen

                Parallel hierzu finden folgende Aktivitäten statt:
                o System informiert/erinnert an aktuelle Termine
                  (Startbildschirm oder Statuszeile);
```

```
                o Benutzer hakt erledigte Aktivitäten in der Checkliste ab
                o System unterstütz die Aquise Wahlhelfer über
                  Serienbrieffunktionalität für das Anschreiben
                o Briefwähler können zu beliebigen Zeiten erfasst werden.
                  Deren Wahlumschläge werden bis zur Auszählung aufbewahrt
PARTNER         Wahlleiter, System
=               Wahlleiter.Stammdatenpflege,
                Wahlleiter.Erstellt_Wählerverzeichnis,
                Wahlleiter.BearbeitetWahlvorschläge,
                Wahlleiter.GibtAuszähllistenEin,
                Wahlleiter.TrägtKandidatenNach,
                Wahlleiter.Ermittelt_Endergebnis,
                Wahlleiter.Gibt_Ergebnis_bekannt

VORGANG         Wählerverzeichnis
                Zusammen oder bereits vor der Ausschreibung einer Wahl wird
                das aktuelle Wählerregister und die notwendigen
                Wählerverzeichnisse sowohl in schriftlicher Form als Ausdruck,
                als auch in Form einer Datei mit festdefinierter Datenstruktur
                beim URZ beantragt und von diesem an die Wahlleitung
                ausgeliefert.
                Die Importdatei enthält eine komplette Liste aller
                Wahlberechtigten. Sie führt die drei ursprünglichen Listen
                (aus Klinik-SAP, Uni-SVA, Prüfungsamt-HISSOS) einheitlich
                zusammen!
                Das Wählerregister spielt durch alle Phasen der Wahl hindurch
                eine ausgezeichnete Rolle:
                1. Beantragung der Wahlberechtigtendatei, des ausgedruckten
                   Wählerverzeichnisses und des ausgedruckten
                   Wählerregisters beim URZ
                2. Import der Datei
                3. Optional: Doppelbereinigung anhand eines vom URZ erhaltenen
                   Auszugs von Mehrfachbeschäftigungsverhältnissen, (Doppel
                   werden auch von Grauwal selbst gefunden; Menüoption), die
                   dann in Grauwal und in den Ausdrucken per Hand
                   eingearbeitet werden (löschen, ändern, durchstreichen etc.)
                4. Wahlvorbereitung
                   - Auslegung des Wählerregisters, [Korrektur (Wähler können
                     Änderung beantragen)]
                   - Wahlvorschläge aufgrund Wählerregister prüfen (Kandidaten
                     müssen wahlberechtigt sein)
                   - Briefwähler aufgrund Wählerregister überprüfen.
                     Serienbrief (z.B. für Abstimmungskarte)
                5. Wählerverzeichnis plus Register an die Wahllokale liefern
                6. Wahl durchführen
                7. Wahlergebnis als Serienbrief an die Kandidaten schicken
                   (aufgrund Adressdaten im Wählerregister)
PARTNER         Wahlleiter, Verwaltung, System
VARIANTE        Handarbeit
                Die Listen der Verwaltung sind von ihrer Struktur her noch
                nicht standardisiert. Sie liegen im Allgemeinen als Exceldatei
                vor, die noch "zu Fuß" zu vereinheitlichen sind, da die Quelle
                der drei Dateien völlig unterschiedlich sind (Klinik, RZ oder
                Studiensekretariat)

VORGANG         Bewerbung
                In der Ausschreibung ruft die Wahlleitung zum Einreichen von
```

```
                Wahlvorschlägen auf. Treffen diese termingerecht ein, haben
                sie eine Chance auf die offizielle Zulassung durch den
                Wahlauschuss und finden so ihren Weg auf die Bekanntgabe der
                Wahlvorschläge, die Stimmzettel (Wahllokal, Briefwahl) und die
                Veröffentlichung des Wahlergebnisses.

                Ablauf:
                Ausschreibung,
                Diskussion, Kandidatensuche, Unterstützersuche,
                Wahlvorschlag definieren und einreichen,
                Zulassung, Bekanntgabe der Wahlvorschläge, Wahl,
                Mitteilung des Erfolgs/Misserfolgs nach der Wahl
PARTNER         Wahlleiter, Unterstützer, Kandidat, System
AUSNAHME        Inkorrekt
                Sind Wahlvorschläge inkorrekt, erhält die Unterstützergruppe
                Gelgenheit zur Nachbesserung. Findet diese nicht termingerecht
                statt, so fällt der Vorschlag unter den Tisch.
                Als Eingangsdatum wird allerding der Tag des Eintreffens der
                ersten version beibehalten!
AUSNAHME        Rückzug
                Wahlvorschläge können auch zurückgezogen werden.
                Allerdings nur bis zum Stichtag der Einreichung von
                Wahlvorschlägen.

VORGANG         WahlhelferAquise
                Wahlhelfer werden von der Wahlleitung akquiriert, in Grauwal
                erfasst (ensprechender Wähler im Wählerverzeichnis bekommt ein
                "Häkchen"), geschult (div. Anschreiben), in den Wahllokalen
                (munitioniert mit Wählerverzeichnis, Wahlumschlägen,
                Stimmzetteln, Urnen, Wahlordnung etc.) und abschließend bei
                der Auszählung eingesetzt.
                Um die Organisation zu vereinfachen, kann in Grauwal eine
                Liste aller Wahlhelfer in Form einer csv-Datei generiert
                werden, um damit diverse Serienbriefe zu generieren.
PARTNER         Wahlleiter, Wahlhelfer

VORGANG         UrnenHandling
                Die Wahlurnen werden vom Fahrdienst in die verschiedenen
                Wahllokale geliefert und während der Abstimmung von den
                Wahlhelfern unter Verschluss gehalten (wenn nicht
                beaufsichtigt). Alle Schlüssel werden vom Wahlleiter zentral
                verwaltet. Zur Auszählung landen alle gefüllten und
                abgeschlossenen Urnen dann im Auszählraum, wo sie zuerst mit
                den Briefwahlumschlägen befüllt und dann zur Auszählung
                komplett entleert werden.
PARTNER         Wahlleiter, Fahrbereitschaft, Wahlhelfer

VORGANG         WordFernsteuerung
                Diverse Word-Dokumente werden von Grauwal erzeugt, indem
                Word automatisch gestartet wird, ein neues Dokument erstellt
                und mit Inhalten gefüllt wird, die sich zum Teil aus Vorlagen
                (per Hand vom Benutzer erstell- und pflegbare Word-Dokumente)
                ergeben und zum Teil programmgesteuert generiert werden
                (Kandidaten-Tabellen von Wahlvorschlägen).

                Die generierten Dokumente bleiben dann auf dem
                Arbeitsplatzrechner im geöffneten Zustand stehen und können
```

	vom Benutzer nun per Hand weiterbearbeitet, gespeichert und/oder ausgedruckt werden.
PARTNER	System, Word
VORGANG	Serienbrief Diverse Funktionen von Grauwal generieren eine csv-Datei mit einer Liste von Datensätzen, die dann per Hand (oder automatisch angestoßen) zur Generierung von Serienbriefen dienen können, indem sie vom Word-Benutzer als Datenquelle ausgewählt werden können.
PARTNER	System, Word

Literaturverzeichnis

[Ach00] Achatz, Klaus: *Systematische Modellbildung für Prozessautomatisierungssysteme.* Doktorarbeit, Universität Ulm, Juli 2000.

[AF95] Avison, D. E. und G. Fitzgerald: *Information Systems Development: Methodologies, Techniques and Tools.* McGraw-Hill Company, London, 1995.

[AHL92] Auramaki, Esa, R. Hirschheim und K. Lyytinen: *Modeling Offices Through Discourse Analysis: The SAMPO Approach.* The Computer Journal, 35(4):342–352, 1992.

[Ale98] Alexander, Ian: *A Co-operativ Task Modeling Approach to Business Process Understanding.* In: *ECOOP Workshop on Object-Oriented Business Process Modeling*, 1998.

[Ale99] Alexander, Ian: *Migrating Towards Co-operative Requirements Engineering.* Computing and Control Engineering Journal, February, 1999.

[Alf85] Alford, M. W.: *SREM at the Age of Eight: The Distributed Computing Design System.* IEEE Computer, 18(4):36–46, 1985.

[ALL88] Auramaki, Esa, Erkki Lehtinen und Kalle Lyytinen: *A Speech-Act-Based Office Modeling Approach.* ACM Transactions on Office Information Systems, 6(2):126–152, 1988.

[AMP94] Anton, A. I., W. M. McCracken und C. Potts: *Goal decomposition and scenario analysis in business process reengineering.* In: Wijers, G., S. Brinkkemper und I. Wasserman (Herausgeber): *Proceedings of CAiSE'94*, Band 811 der Reihe *Lecture Notes in Computer Science*, Seiten 94–104, Utrecht, The Netherlands, Juni 1994. Springer Verlag.

[And95] Andelfinger, Urs: *Diskursive Anforderungsanalyse und Validierung – Ein Beitrag zum Reduktionsproblem bei Systementwicklungen in der Informatik.* Doktorarbeit, Technische Hochschule Darmstadt, 1995.

[AWH90] Avison, D. E. und A. T. Wood-Harper: *Multiview: An exploration in information systems development.* Blackwell, Oxford, 1990.

[AWHVW98] Avison, D. E., A. T. Wood-Harper, R. T. Vidgen und J. R. G. Wood: *A further exploration into information systems development: the evolution of Multiview2.* Information Technology & People, 11(2):124–139, 1998.

[BBC+88] Baron, R., E. Beslmuller, F. Calmes, G. Charbonnel, D. Conrath, V. De Antonellis, G. De Petra, Ph. Dumas, V. Savolainen und C. Si-

MONE: *Office Support Analysis and Design (OSSAD), Final Report on Office Modeling, Language and OSSAD.* Technischer Bericht Esprit Project No. 28, IOT, Munich, 1988.

[BBHL94] BOEHM, B., P. BOSE, E. HOROWITZ und M.-J. LEE: *Software Requirements as Negotiated Win Conditions.* In: *Proceedings: 1st International Conference on Requirements Engineering*, Seiten 74–83. IEEE Computer Society Press, 1994.

[BCDS93] BLYTH, A. J. C., J. CHUDGE, J. E. DOBSON und M. R. STRENS: *ORDIT: A New Methodology to Assist in the Process of Eliciting and Modeling Organisational Requirements.* In: *Conference on Organizational Computing Systems*, Supporting Software Development Organizations, Seiten 216–227, 1993.

[BD93] BRÖHL, A. P. und W. DRÖSCHEL: *Das V-Modell.* Oldenbourg Verlag, München, Wien, 1993.

[Bea00] BROY, MANFRED und DIETER ROMBACH ET AL.: *Studie für das BMBF: Analyse und Evaluation der Software-Entwicklung in Deutschland.* http://www.dlr.de/IT/IV/Studien/evasoft_abschlussbericht.pdf, 2000.

[Bec00] BECK, KENT: *Extreme Programming Explained.* Addison Wesley Publishing Company, Upper Saddle River, NJ, USA, 2000.

[BG91] BØDKER, SUSANNE und KAJ GRØNBÆK: *Design in Action: From Prototyping be Demonstration to Cooperative Prototyping.* In: *Design at Work* [GK91], Kapitel 10, Seiten 197–218.

[BH77] BOSTROM, ROBERT P. und J. STEPHEN HEINEN: *MIS Problems and Failures: A Socio-Technical Perspective.* MIS Quarterly, September:17–32, 1977.

[BH98] BEYER, HUGH und KAREN HOLTZBLATT: *Contextual Design.* Morgan Kaufmann Publishers, 1998.

[Blu86] BLUMER, HERBERT: *Symbolic Interactionism: Perspective and Method.* University of California Press, 1986.

[Boe84] BOEHM, B. W.: *Software Engineering Economics.* In: *IEEE Transactions on Software Engineering (SE)*, Band 10/1, Januar 1984. Also published in/as: Prentice-Hall publishers, 1981.

[BR97] BOOCH, GRADY und JAMES RUMBAUGH: *Unified Method for Object-Oriented Development Version 1.0.* Rational Software Corporation, 1997.

[BR02] BROY, MANFRED und DIETER ROMBACH: *Software Engineering. Wurzeln, Stand und Perspektiven.* Informatik Spektrum, 25(6):438–451, 2002.

[BRLD94] BUBENKO, J., C. ROLLAND, P. LOUCOPOULOS und V. DEANTONELLIS: *Facilitating "Fuzzy to Formal" Requirements Modeling.* In: *Proceedings: 1st International Conference on Requirements Engineering*, Seiten 154–165. IEEE Computer Society Press, 1994.

[CAB+93] COLEMAN, D., P. ARNOLD, S. BODOFF, C. DOLLIN, H. GILCHRIST und P. JEREMAES: *O-O Development — The FUSION Method.* Prentice-Hall, Englewood Cliffs (NJ), USA, 1993.

[Car94] Carmichael, Andy: *Object Development Methods.* SIGS Books, Inc., New York, USA, 1994.

[Car95] Carroll, John M.: *Scenario-Based Design. Envisioning Work and Technology in System Development.* John Wiley & Sons, Inc., New York, 1995.

[CdAS89] Conrath, David W., Valeria de Antonellis und Carla Simone: *A Comprehensive Approach to Modeling Office Organization and Support Technology.* In: B. Pernici, A. A. Verrijn-Stuart (Herausgeber): *Office Information Systems: The Design Process*, Seiten 73–92. Elsevier Science Publishers B.V. (North Holland), 1989.

[Che81] Checkland, Peter: *Systems Thinking, Systems Practice.* John Wiley & Sons, Inc., New York, 1981.

[Che89] Checkland, Peter: *Soft Systems Methodology.* In: Rosenhead, Jonathan (Herausgeber): *Rational Analysis for a problematic world*, Seiten 71–119. John Wiley & Sons, Inc., New York, 1989.

[CJS92] Chandrasekaran, B., T. Johnson und J. W. Smith: *Task Structure Analysis for Knowledge Modeling.* Communications of the ACM, 33(9):124–136, 1992.

[CL99] Constantine, L. L. und L. A. D. Lockwood: *Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design.* Addison Wesley Publishing Company, 1999.

[CM90a] Cardenas, Alfonso F. und Dennis McLeod: *An Overview of Object-Oriented and Semantic Database Systems.* In: Cardnas, Alfonso F. und Dennis McLeod [CM90b], Seiten xvii–xxv.

[CM90b] Cardenas, Alfonso F. und Dennis McLeod: *Research foundations in object-oriented and semantic database systems.* Prentice-Hall, Inc., New Jersey, 1990.

[Coc00] Cockburn, Alistair: *Writing Effective Use Cases.* Addison Wesley Publishing Company, 2000.

[CSA89] Carey, M. S., R. B. Stammers und J. A. Astley: *Human-computer interaction design: The potential and pitfalls of hierarchical task analysis.* In: Diaper, D. (Herausgeber): *Task analysis for human-computer interaction*, Seiten 56–74. Ellis Horwood, 1989.

[CY90] Coad, Peter und Edward Yourdon: *Object Oriented Analysis.* Yourdon Press, 1990.

[Dah97] Dahme, Christian: *Systemanalyse menschlichen Handelns.* Westdeutscher Verlag, Opladen, 1997.

[Dav93] Davis, Alan M.: *Software Requirements. Object, Functions, and States.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[DBCS94] Dobson, J. E., A. J. C. Blyth, J. Chudge und R. Strens: *The ORDIT Approach to Organizational Requirements.* In: Jirotka, M. und J. Goguen

(Herausgeber): *Requirements Engineering: Social and Technical Issues*, Seiten 87–106. Academic Press, London, 1994.

[DHP+99] Deifel, B., U. Hinkel, B. Paech, P. Scholz und V. Thurner: *Die Praxis der Softwareentwicklung.* Informatik Spektrum, 22(1):24–36, 1999.

[Die91] Dietz, Jan L. G.: *A Communication Oriented Approach To Conceptual Systems Modeling.* In: *Dynamic Modeling of Information Systems*, Seiten 37–60. Elsevier Science Publishers B.V., North Holland, 1991.

[DR97] Dahme, Christian und Arne Raeithel: *Ein tätigkeitsorientierter Ansatz zur Entwicklung von brauchbarer Software.* Informatik Spektrum, 20:5–12, 1997.

[Dut94] Dutke, Dr. Stephan: *Mentale Modelle: Konstrukte des Wissens und Verstehens.* Verlag für Angewandte Psychologie, Göttingen, Stuttgart, 1994.

[DvLF93] Dardenne, Anne, Axel van Lamsweerde und Stephen Fickas: *Goal-directed requirements acquisition.* Science of Computer Programming, 20(1–2):3–50, April 1993.

[DW99] D'Souza, Desmond Francis und Alan Cameron Wills: *Objects, Components, and Frameworks with UML.* Addison Wesley Publishing Company, 1999.

[Eas88] Eason, K.: *Information Technology and Organizational Change.* Taylor & Franzis, London, 1988.

[EK91] Ehn, Pelle und Morten Kyng: *Cardboard Computers: Mocking-it-up or Hands-on the Future.* In: *Design at Work* [GK91], Kapitel 9, Seiten 169–195.

[Ell79] Ellis, C. A.: *Information Control Nets: A mathematical model of office information systems.* In: *Proc. of the 1979 ACM Conference on Simulation, Measurement and Modeling of Computer Systems*, Seiten 225–239, Boulder, Colorado, 1979. ACM Press.

[ER03] Endres, Albert und Dieter Rombach: *A Handbook of Software and Systems Engineering. Empirical Observations, Laws and Theories.* Pearson Education Limited, 2003.

[Fal98] Falkenberg, E. D. (Herausgeber): *A Framework of Information Systems Concepts.* Report of the IFIP WG 8.1 Task Group FRISCO, 1998.

[FDK] Floyd, Christiane, Yvonne Dittrich und Ralf Klischewski: *Social Thinking - Software Practice.* citeseer.nj.nec.com/floyd99social.html.

[Fit94] Fitzgerald, Brian: *The Systems Development Dilemma: Whether to Adopt Formalised Systems Development Methodologies or Not?* In: Baets, W. (Herausgeber): *Proceedings of Second European Conference on Information Systems*, Seiten 691–706, Holland, 1994. Nijenrode University Press.

[Fit96] Fitzgerald, Brian: *Formalized systems development methodologies: a critical perspective.* Information Systems Journal, 6(1):3–23, Januar 1996.

[FK98] Floyd, Christiane und Ralf Klischewski: *Modellierung - ein Handgriff zur Wirklichkeit.* In: Klaus Pohl, A. Schürr, G. Vossen (Herausgeber): *Modellierung '98 - Proceedings*, Nummer 6/98-I (März 1998) in *Technische Berichte*, Seiten 21–26. Universität Münster, 1998.

[FKN+92] Finkelstein, A., J. Kramer, B. Nuseibeh, L. Finkelstein und M. Goedicke: *Viewpoints: A Framework for Integrating Multiple Perspectives in Systems Development.* International Journal of Software Engineering and Knowledge Engineering, 1(2):31–58, 1992.

[Flo92] Floyd, Christiane: *Software Development as Reality Construction.* In: Floyd, Christiane, H. Züllighoven, R. Budde und R. Keil-Slawik (Herausgeber): *Software Development and Reality Construction*, Kapitel 3.2, Seiten 86–100. Springer Verlag, 1992.

[Fre90] Frese, Erich: *Organisationtheorie.* Betriebswirtschaftlicher Verlag Dr. Th. Gabler, Wiesbaden, 1990.

[FS90] Ferstl, O. K. und E. J. Sinz: *Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM).* Wirtschaftsinformatik, 32(6):566–581, 1990.

[GGJZ00] Gunter, Carl A., Elsa L. Gunter, Michael Jackson und Pamela Zave: *A Reference Model for Requirements and Specifications.* IEEE Software, May/June:37–43, 2000.

[GK91] Greenbaum, Joan und Morten Kyng: *Design at Work. Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.

[GL93] Goguen, Joseph A. und Charlotte Linde: *Techniques for Requirements Elicitation.* In: *Proceedings of the 1st International Symposium on Requirements Engineering*, Seiten 152–163, 1993.

[GM95] Gerstenmaier, Jochen und Heinz Mandl: *Wissenserwerb unter konstruktivistischer Perspektiver.* Zeitschrift für Pädagogik, 41(6):867–888, 1995.

[Goo94] Goorhuis, Henk: *Konstruktivistische Modellbildung in der Informatik.* Doktorarbeit, Universität Zürich, März 1994.

[Gör93] Görz, Günther: *Einführung in die künstliche Intelligenz.* Addison Wesley Publishing Company, 1993.

[Gra98] Graham, Ian: *Requirements Engineering and Rapid Development: An Object-Oriented Approach.* Addison Wesley Publishing Company, 1998.

[Gru90] Grudin, J.: *The computer reaches out: the historical continuity of interface design.* In: *Proceedings of CHI 90.* ACM Press, 1990.

[GW96] Ganter, Bernhard und Rudolf Wille: *Formale Begriffsanalyse—Mathematische Grundlagen.* Springer, Berlin/Heidelberg, 1996.

[Har87] Harel, D.: *Statecharts: A Visual Formalism for Complex Systems.* Science of Computer Programming, 8:231–74, 1987.

[Hei99] Heinrich, L. J.: *Grundlagen der Wirtschaftsinformatik.* In: *Informatik-Handbuch* [RP99], Kapitel F1, Seiten 1019–1034.

[Hes96] Hesse, Wolfgang: *Theory and practice of the software process - a field study and its implications for project management.* In: Montangero, C. (Herausgeber): *Software Process Technology*, LNCS 1149, 5th European Workshop, EWSPT 96, Seiten 241–256. Springer Verlag, 1996.

[Hes97] Hesse, Wolfgang: *Wie evolutionär sind die objekt-orientierten Analysemethoden? Ein kritischer Vergleich.* Informatik-Spektrum, 20(1):21–28, 1997.

[Hes98] Hesse, Wolfgang: *Vorgehensmodelle für objektorientierte Software-Entwicklung.* In: Kneuper, R. (Herausgeber): *Vorgehensmodelle für die betriebliche Anwendungsentwicklung*, Teubner Reihe Wirtschaftsinformatik, Seiten 110–135. B.G. Teubner Stuttgart, 1998.

[HH93] Hix, Deborah und H. Rex Hartson: *Developing User Interfaces: Ensuring Usability Through Product and Process.* John Wiley & Sons, Inc., 1993.

[HK89] Hirschheim, R. A. und H. K. Klein: *Four paradigms of information systems development.* Communications of the ACM, 32(10):199–216, 1989.

[HK90] Hull, Richard und Roger King: *A Tutorial on Semantic Database Modeling.* In: Cardenas, Alfonso F. und Dennis McLeod [CM90b], Kapitel 1, Seiten 1–33.

[HK99] Hitz, Martin und Gerti Kappel: *UML@Work.* dpunkt.verlag, 1999.

[HKL95] Hirschheim, R. A., H. K. Klein und K. Lyytinen: *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundation.* Cambridge University Press, 1995.

[HMW98] Hupe, Patrick, Florian Matthes und Holm Wegner: *Ein bruchloser Übergang von der Prozeßmodellierung zu kooperativen Software-Architekturen.* Technischer Bericht, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Germany, November 1998.

[Hoa85] Hoare, C. A. R.: *Communicating Sequential Processes.* Prentice Hall, Inc., 1985.

[Hol94] Holm, P.: *The COMMODIOUS Method - Communication Modeling as an Aid to Illustrate the Organisational Use of Software.* In: *6th Int. conf. on Software Engineering and Knowledge Engineering*, 1994.

[Hru98] Hruby, Pavel: *Structuring Specification of Business Systems with UML (with an Emphasis on Workflow Management Systems).* In: *Proc. OOPSLA'98 Business Object Workshop.* Springer, 1998.

[IH96] Iivari, Juhani und Rudy Hirschheim: *Analyzing Information Systems Development: A Comparison and Analysis of eigth IS Development Approaches.* Information Systems, 21(7):551–575, 1996.

[Jaa95] Jaaksi, Ari: *Object-oriented Specification of User Interfaces.* Software - Practice and Experience, 25(11):1203–1221, 1995.

[Jac94a] Jackson, Michael: *Problems, Methods and Specialization.* Software Engineering Journal, November:57–82, 1994.

[Jac94b] Jacobson, Ivar: *Object-Oriented Software Engineering.* Addison Wesley Publishing Company, 1994.

[Jac97] Jackson, Michael: *Problem Complexity.* In: *International Conference on Engineering of Complex Computer Systems*, Seiten 239–247, 1997.

[JBR99] Jacobson, Ivar, Grady Booch und James Rumbaugh: *The Unified Software Development Process.* Addison Wesley Publishing Company, 1999.

[JBS97] Jablonski, S., M. Böhm und W. Schulze: *Workflow-Management: Entwicklung von Anwendungen und Systemen - Facetten einer neuen Technologie.* Dpunkt-Verlag, 1997.

[Jef96] Jeffrey, H. Joel: *Addressing the Essential Difficulties of Software Engineering.* J. Systems Software, 32:157–179, 1996.

[JM02] Jarke, Matthias und Heinrich C. Mayr: *Mediengestütztes Anforderungsmanagement.* Informatik Spektrum, 25(6):452–464, 2002.

[Jon90] Jones, C. B.: *Systematic Software Development Using VDM.* Prentice Hall, Inc., 1990.

[JP88] Jarke, Matthias und Klaus Pohl: *Requirements Engineering in the Year 2001: On (Virtually) Managing a Changing Reality.* Technischer Bericht 95-03, Nature Report Series. Lehrstuhl für Informatik. Aachen, 1988.

[KC91] Knuth, R. A. und D. J. Cunningham: *Tools for Constructivism.* In: Duffy, T. M., J. Lowyk und D. H. Jonassen (Herausgeber): *Designing Environments for Constructive Learning*, Band 105 der Reihe *NATO ASI Series, Series F: Computer and System Sciences*, Seiten 163–188. Springer Verlag, 1991.

[KI89] Kling, Rob und Suzanne Iacono: *The Institutional Character of Computerized Information Systems.* Office: Technology and People, 5 1:7–28, 1989.

[KKB95] Kaptelinin, Victor, Kari Kuutti und Liam Bannon: *Activity Theory: Basic Concepts and Applications.* Lecture Notes in Computer Science, 1015:189–201, 1995.

[KL99] Kavakli, Vagelio und Pericles Loucopoulos: *Goal-driven Business Process Analysis Application in Electricity Deregulation.* Information Systems, 24(3):187–207, 1999.

[KM91] Kensing, Finn und Kim Halkov Madsen: *Generating Visions: Future Workshops and Metaphorical Design.* In: *Design at Work* [GK91], Kapitel 8, Seiten 155–168.

[KM96] Kop, Christian und Heinrich C. Mayr: *Objektorientierte Analyse und konzeptueller Vorentwurf.* Emisa-Forum, 1, 1996. Fachgruppentreffen '95: Requirements Engineering für Informationssysteme. Karlsruhe 12 - 13 Okt 95.

[Kos62] Kosiol, E.: *Organisation der Unternehmung.* Gablerverlag, 1962.

[Kov98] Kovitz, Benjamin L.: *Practical Software Requirements.* Manning, Greenwich, 1998.

[Kov01] Kovitz, Ben: *Talk at RE'01 Fifth IEEE International Symposium on RE, Aug. 27-31, 2001*, 2001.

[KS82] Kling, R. und R. Scacchi: *The Web of Computing: Computer Technology as Social Organization.* Advances in Computers, 21:2–90, 1982.

[KS96] Kotonya, G. und I. Sommerville: *Requirements Engineering with Viewpoints.* Software Engineering Journal, 11(1):5–18, 1996.

[Kuu95] Kuutti, Kari: *Work Processes: Scenarios as a Preliminary Vocabulary.* In: *Scenario-Based Design* [Car95], Kapitel 1, Seiten 19–36.

[Lar98] Larman, Craig: *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design.* Prentice Hall PTR, Upper Saddle River/NJ, 1998.

[Lef97] Leffingwell, Dean: *Calculating the Return on Investment from More Effective Requirements Management.* American Programmer, 10(4):13–16, 1997.

[LH87] Lyytinen, Kalle und Rudy Hirschheim: *Information systems failures - a survey and classification of the empirical literature.* Oxford Surveys in Information Technology, 4:257–309, 1987.

[LR] Lewis, C. und J. Rieman: *Task-Centered User Interface Design.* http://hcibib.org/tcuid/.

[Mac97] Macaulay, Linda A.: *Requirements Engineering (Applied Computing).* Springer Verlag, 1997.

[Mag97] Maguire, M.: *RESPECT User Requirements Framework Handbook.* Technischer Bericht RESPECT consortium report 5.1, European Community Telematics Aoolications Project TE 2010, 1997.

[Mar90] Martin, James: *Information Engineering: book 1, 2, and 3.* PH, ; ACM CR 9104-0255, 1990.

[May99] Mayhew, Deborah J.: *The Usability Engineering Lifecycle: A Practitioner's Guide to User Interface Design.* Morgan Kaufmann Publishers, 1999.

[MDJ+98] Michelis, G. De, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, M. Papazoglou, J. W. Schmidt und C. Yu E. Woo: *Cooperative information systems - a manifesto.* In: Papazoglou, M. und Schlageter (Herausgeber): *Cooperative information systems - trends and directions*, Seiten 315–363. New York: Academic Press, 1998.

[MFKH90] Macaulay, Linda A., Chris J. H. Fowler, Mark A. R. Kirby und Andrew Hutt: *USTM: A New Approach to Requirements Specification.* Interacting with Computers, 2(1):92–118, 1990.

[Mil89] Milner, R.: *Communication and Concurrency.* Prentice Hall, Inc., 1989.

[Mul79] Mullery, G. P.: *CORE: A Method for Controlled Requirements Specification.* In: *Proceedings of the 4th International Conference on Software Engineering*, Seiten 126–135. IEEE Computer Society Press, 1979.

[Mum93] Mumford, Enid: *The ETHICS approach.* Communications of the ACM, 36(6):82–82, Juni 1993.

[MV87] Maturana, H. R. und F. J. Varela: *Der Baum der Erkenntnis. Die biologischen Wurzeln des menschlichen Erkennens.* Goldman, 1987.

[ND86] Norman, D. A. und S. W. Draper (Herausgeber): *User Centered System Design: New Perspectives on Human Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Nie94] Nielsen, Jakob: *Usability Engineering.* Academic Press, 1994.

[NS92] Namioka, A. und D. Schuler (Herausgeber): *Participatory Design.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[Nur98] Nurcan, Selmin: *Main Concepts For Cooperative Work Place Analysis.* In: *Proceedings of the Telecooperation Conference of the 15th IFIP World Computer Congress*, Vienna, Austria, 1998.

[OBG+95] Oesterle, H., C. Brenner, C. Gassner, T. Gutzwiller und T. Hess: *Business Engineering. Prozess- und Systementwicklung.* Springer-Verlag, Berlin, 1995.

[Ort02] Ortner, Erich: *Sprachingenieurwesen. Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik.* Informatik Spektrum, 25(1):39–51, 2002.

[Pae00] Paech, Barbara: *Aufgabenorientierte Softwareentwicklung.* Springer Verlag, 2000.

[Pas94] Pasch, Jürgen: *Kooperative Software- Entwicklung im Team.* Springer Verlag, 1994.

[PC86] Parnas, David Lorge und Paul C. Clements: *A Rational Design Process: How and Why to Fake It.* IEEE Transactions on Software Engineering, 12(2):251–257, 1986.

[Per89] Pernici, B.: *Objects with Roles.* In: Tsichritzis, D. (Herausgeber): *Object Oriented Development*, Seiten 75–100. Universite de Geneve, Centre Universitaire D'Informatique, 1989.

[Pet62] Petri, C. A.: *Kommunikation mit Automaten.* Doktorarbeit, Universität Bonn, 1962.

[Pfl98] Pfleeger, Shari Lawrence: *Software Engineering: theory and practice.* Prentice Hall, Inc., 1998.

[Pre94] Pressman, Roger S.: *Software Engineering: A Practitioner's Approach.* McGraw-Hill, 1994.

[PS93] Pinci, V. O. und R. M. Shapiro: *Work Flow Analysis.* In: *Winter Simulation Conference*, Seiten 1122–1130, New York, Dezember 1993. ACM Association for Computing Machinery.

[PTA94] Potts, Colin, Kenji Takahashi und Annie Anton: *Inquiry-based Requirements Analysis.* IEEE Software, 11(2):21–32, 1994.

[Rap88] Rapoport, Anatol: *Allgemeine Systemtheorie: wesentliche Begriffe und Anwendungen.* Verlag Darmstädter Blätter, Darmstadt, 1988.

[Rat97] Rational Software Corporation: *The Unified Modeling Language, Version 1.1.* Technischer Bericht, Internet: http://www.rational.com/uml, 1997.

[Rau92] Rauterberg, Matthias: *Partizipative Modellbildung zur Optimierung der Softwareentwicklung.* In: Studer, Rudi (Herausgeber): *Workshop Informationssysteme und Künstliche Intelligenz*, Band 303 der Reihe *Informatik Fachberichte*, Seiten 113–128. Springer Verlag, 1992.

[RC95] Rosson, Mary Beth und John M. Carroll: *Narrowing the Specification-Implementation Gap in Scenario-Based Design.* In: *Scenario-Based Design* [Car95], Kapitel 10, Seiten 247–278.

[Rei91] Reimer, Ulrich: *Einführung in die Wissensrepräsentation.* B.G. Teubner Stuttgart, 1991.

[Rem89] Remer, Andreas: *Organisationslehre: eine Einführung.* de Gruyter, Berlin; New York, 1989.

[Rob97] Robertson, Suzanne: *Putting the RAD in Requirements: Accelerating past impediments.* In: *Proceedings of the EuroSTAR'97*, Edinburgh, 1997.

[RP99] Rechenberger, Peter und Gustav Pomberger: *Informatik-Handbuch.* Hanser Verlag, 1999.

[RUP98] *Online Manuals of the Rational Unified Process 5.5.* $RationalUnifiedProcess5.5/manuals/intro/im_keymc.htm$, 1998.

[Sac95] Sachs, Patricia: *Transforming Work: Collaboration, Learning, and Design.* Communications of the ACM, 38(9):36–44, September 1995.

[Sch88] Schäfer, G.: *Functional Analysis of Office Requirements: A multiperpective approach.* John Wiley & Sons, Chichester, 1988.

[Sch99a] Scheer, August-Wilhelm: *ARIS, Vom Geschäftsprozeß zum Anwendungssystem.* Springer Verlag, Heidelberg, 1999.

[Sch99b] Schefe, Peter: *Softwaretechnik und Erkenntnistheorie.* Informatik Spektrum, 22:122–135, 1999.

[Sea69] Searl, J. R.: *Speech Acts.* Cambridge University Press, Cambridge, 1969.

[Sho93] Shoham, Y.: *Agent-oriented programming.* Artificial Intelligence, 60(1):51–92, 1993.

[Sin99] Sinz, E. J.: *Architektur von Informationssystemen.* In: *Informatik-Handbuch* [RP99], Kapitel F2, Seiten 1035–1047.

[SK01] Schinzel, Britta und Karin Klein: *Qua vadis, Informatik?* Informatik Spektrum, 24:91–101, April 2001.

[SM93] Sutcliffe, A. und N. Maiden: *Bridging the requirements gap: policies, goals and domains.* In: *In Proceedings of the 7th International Workshop on System Specification and Design, 52-55.*, Seiten 52–55, 1993.

[SN87] Schönecker, H. G. und M. Nippa: *Computergestützte Organisationshilfen für die Praxis.* FBO-Verlag, Baden-Baden, 1987.

[Spi88] Spivey, J. M.: *Understanding Z: A Specification Language and Its Formal Semantics, Cambridge, UK.* Cabridge University Press, Cambridge, UK, 1988.

[SR99] Suzanne und James Robertson: *Mastering the requirements process.* Addison Wesley Publishing Company, 1999.

[SSK+84] Sirbu, Marvin, Sandor Schoichet, Jay S. Kunin, Michael Hammer und Juliet Sutherland: *OAM: An Office Analysis Methodology.* Behaviour and Information Technology, 3(1):25–39, 1984.

[ST89] Steinle, Claus und Michael Thewes: *Gestaltung der Büroarbeit durch computergestützte Kommunikationsanalysen: Merkmale, Vergleich und Praxiseignung.* TÜV Rheinland GmbH, Köln: E. Debusmann, S. Dworatschek, M. Mehrtens, 1989.

[Sta97] Stapelton, J.: *Dynamic Systems Development Method: The Method in Practice.* Addison Wesley Publishing Company, Harlow, Essex, 1997.

[Sta02] Staab, Steffen: *Wissensmanagement mit Ontologien und Metadaten.* Informatik Spektrum, Juni:194–209, 2002.

[Suc87] Suchman, Lucy A.: *Plans and Situated Actions.* Cambridge Universtiy Press, Cambridge, 1987.

[TLH+88] Turner, W. S., R. P. Langerhorst, G. F. Hice, H. B. Eilers und A. A. Uijttenbroek: *SDM: System Development Methodology.* North-Holland, Amsterdam, The Netherlands, 1988.

[Uli94] Ulich, Eberhard: *Arbeitspsychologie.* Schäffer-Pieschel Verlag Stuttgart, 1994.

[UML03] Object Modeling Group: *Unified Modeling Language Specification, Version 1.5*, März 2003.

[UP95] Ulrich, Hans und Gilbert J. B. Probst: *Anleitung zum ganzheitlichen Denken und Handeln. Ein Brevier für Führungskräfte.* Verlag Paul Haupt, Bern, Stuttgart, Wien, 1995.

[UU93] University, Lancaster und Manchester University: *Informing CSCW System Requirements.* Technischer Bericht ISBN 0-901800-29-5, Lancaster University ans Manchester University, 1993.

[WAP88] William A. Pasmore, Ph.D.: *Designing Effective Organizations: The Sociotechnical Systems Perspective.* Wiley Series On Organizational Assessment and Change. John Wiley & Sons, Inc., New York, 1988.

[Wat81] Watzlawick, P.: *Die erfundene Wirklichkeit.* Piper, München und Zürich, 1981.

[WB95] Wirfs-Brock, Rebecca: *Designing Objects and Their Interactions: A Brief Look at Responsibility-Driven Design.* In: *Scenario-Based Design* [Car95], Kapitel 13, Seiten 337–360.

[Web01] Weber, Herbert: *Quo Vadis Software Engineering.* Softwaretechniktrends, 21(1), 2001.

[WF87] Winograd, T. und R. Flores: *Understanding Computers and Cognition.* Addison-Wesley, Reading, Massachusetts, 1987.

[WGKI98] Wedekind, Hartmut, Günter Görz, Rudolf Kötter und Rüdiger Inhetveen: *Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik.* Informatik Spektrum, 21:265–272, 1998.

[Wie98] Wieringa, Roel J.: *A Survey of Structured and Object-Oriented Software Specification Methods and Techniques.* ACM Computing Surveys, 30(4):459 ff, 1998.

[Wie99] Wiegers, Karl E.: *Software Requirements.* Microsoft Press, 1999.

[Win96] Winkelmann, Rolf: *Softwareentwicklung.* Publicis MCD Verlag, Erlangen, 1996.

[WK78] Wollnik, Michael und Herbert Kubicek: *Elemente eines handlungsbezogenen Konteptes der benutzerorientierten Systemgestaltung.* In: Petri, Carl Adam (Herausgeber): *Ansätze zur Organisationstheorie rechnergestützter Informationssysteme*, Seiten 331–375. Oldenbourg, 1978.

[WS95] Wood, Jane und Denise Silver: *Joint Application Development.* John Wiley & Sons, Inc., New York, 1995.

[YS99] Yu, L. und B. F. Schmid.: *A conceptual framework for agent-oriented and rolebased work ow modeling.* In: Wagner, G. und E. Yu (Herausgeber): *Proc. of the 1st Int. Workshop. on Agent-Oriented Information Systems*, Seiten 94–104, 1999.

[Zav82] Zave, P.: *An Operational approach to requirements specification for embedded systems.* IEEE Transactions on Software Engineering, 8(3):307–320, 1982.

[Zis77] Zisman, M.D.: *Representation, Specification and Automation of Office Procedures.* Doktorarbeit, University of Pennsylvania, Warton School of Business, 1977.

[Zue98] Zuellighoven, Heinz: *Das objektorientierte Konstruktionshandbuch.* dpunkt Verlag, 1998.

Kurzfassung

Die vorliegende Arbeit stellt eine neue modellbasierte Vorgehensweise zur partizipativen und systemischen Gestaltung betrieblicher Informationssysteme im Kontext ihrer praktischen Anwendungsumgebung vor.

Sie beschreibt einen möglichen Weg zur *Entwicklung* von Anforderungen und gibt damit eine Antwort auf eine Frage, die im Umfeld von Softwareentwicklungsprojekten oft gehört wird – zumal etablierte Analyse-Methoden sie in der Regel weitgehend vernachlässigen: "Wie kommt man eigentlich zu den Anforderungen, die man zu spezifizieren hat?". Der Schwerpunkt des Interesses liegt damit auf einer Phase der Softwareentwicklung, die noch vor der eigentlichen Spezifikation des zu entwickelnden Softwaresystems angesiedelt ist.

Die vorgestellte Vorgehensweise trägt den Namen "Koala": **Ko**mmunikationsorientierte **A**nalyse und Gesta**l**tung von **A**rbeitskontexten. KoA$_{\text{L}}$A fasst die Entwicklung von Anforderungen an Informationssysteme als kreativen Gestaltungsprozess auf, der nicht nur das Softwaresystem betrifft, sondern auch die Anwendungsumgebung miteinbezieht.

Aufgrund der immer weiter steigenden Komplexität sowohl der zu entwickelnden Systeme selbst, als auch ihres organisatorischen und technischen Umfelds, gehört die erfolgreiche Gestaltung großer Softwaresysteme heute mit zu den schwierigsten technischen Herausforderungen. Mit der wachsenden Durchdringung von Unternehmen mit Informationstechnologie wird insbesondere die Entwicklung betrieblicher Informationssysteme immer mehr zu einem herausfordernden interdisziplinären Unterfangen, das viele andere Bereiche wie die Betriebswirtschaft, Arbeitswissenschaft, Soziologie und Psychologie berührt. Da entsprechende soziotechnische Fragen in den üblichen Softwareentwicklungsmethoden kaum thematisiert werden, führt das im Letzten zu dem Umstand, dass fertige Softwaresysteme in der Praxis oft keine oder nur unzureichende Akzeptanz finden und dementsprechend die Erfolgsquote von Softwareentwicklungsprojekten insgesamt relativ gering ist.

Der Softwareentwicklungsprozess beginnt in Wirklichkeit bereits mit der Entwicklung des zu lösenden Problems, noch bevor die eigentliche Problemlösung – das Softwaresystem – analysiert und spezifiziert werden kann. Die Entwicklung von Software muss als Gestaltungsaufgabe angesehen werden, die wesentlichen Einfluss auf die soziale Wirklichkeit nimmt. Obwohl sich diese Sichtweise mittlerweile durchzusetzen beginnt, sind entsprechende Softwareentwicklungsmethoden bisher nur in Ansätzen zu entdecken. Insbesondere fehlen hier pragmatische Methoden, die einen Softwareingenieur in seiner Alltagspraxis unterstützen und sich nahtlos in bestehende Entwicklungsprozesse einfügen. Dies ist die Motivation für die vorliegende Arbeit.

KoA$_{\text{L}}$A beschreibt eine Vorgehensweise zur Gestaltung betrieblicher Informationssysteme, deren Schwerpunkt in der kommunikationsorientierten Modellierung und partizipativen Um-

gestaltung realer Arbeitszusammenhänge liegt. Endprodukt des Gestaltungsprozesses ist das praxisorientierte Modell eines soziotechnischen Handlungssystems, das als Ausgangspunkt für die Spezifikation der technischen Anforderungen an das Softwaresystem verwendet werden kann.

Ausgehend von aktuellen Kooperationsbeziehungen werden durch eine Analyse der typischen Konversations- und Aktivitätsmuster die Tätigkeitsprofile aller Akteure aufgedeckt. Nach einer Bestimmung der zukünftigen Nutzergruppen und deren Anforderungen wird das Softwaresystem als neuer technischer Akteur mit eigenem Tätigkeitsprofil und spezifischem Interaktionsverhalten entworfen und auf Modellierungsebene in diesen Arbeitskontext integriert bzw. "eingepasst". Durch Modellierung und Diskussion der sich neu ergebenden zentralen Aktivitäts- und Konversationsmuster wird die gewählte Gestalt des Informationssystems validiert und iterativ überarbeitet.

Der Gestaltungsprozess geht vom Bereich menschlichen Handelns aus, von der Anwendungswelt. Zur Darstellung dieses Kontexts setzt KoA$_{\text{L}}$A in großem Umfang Beschreibungsmodelle ein, die sowohl grafisch als auch freitextlich dokumentiert werden. KoA$_{\text{L}}$A stellt drei Modell- und fünf Diagrammarten zur Verfügung, deren zugrundeliegenden orthogonalen, d.h. modell- und diagrammübergreifend verwendbaren Modellierungskonzepte in ihrer Anzahl weitgehend minimiert und in ihrer Form und Bedeutung weitgehend an UML angelehnt sind, um ihre Einführung zu erleichtern und ein einfaches Verständnis der Modelle zu gewährleisten.

Die relevante konzeptionelle Begriffswelt des Anwendungsbereichs wird im *Wissensmodell* abgebildet. Das *Organisationsmodell* dient der Dokumentation der statischen Organisationsstrukturen und trennt dabei die Beschreibung des Aufbaus der Organisation in Form der Organisationsstruktur von der Beschreibung des dynamischen Potentials seiner Akteure in Form der Tätigkeitsprofile. Im *Kooperationsmodell* schließlich unterscheidet KoA$_{\text{L}}$A zwischen einer statischen Sicht auf die Kooperationsbeziehungen in Form der Kooperationsstruktur, und zwei unterschiedlichen dynamischen Sichten in Form von Konversations- und Aktivitätsmustern.

KoA$_{\text{L}}$As Modellierung des technischen Softwaresystem-Akteurs ermöglicht schließlich einen nahtlosen Übergang zur Softwarespezifikation, da sowohl UML-ähnliche Modellierungskonzepte verwendet werden als auch die Abstraktionsebene so gewählt ist, dass der nachfolgende Softwareentwurf einen weitgehend natürlichen Konkretisierungsschritt darstellt.

KoA$_{\text{L}}$A ist aus Erfahrungen in industriellen Entwicklungsvorhaben enstanden und hat sich bereits in zahlreichen Entwicklungsprojekten für die Verwaltung der Universität Ulm bewährt, die von studentischen Hilfskräften, Diplomanden und Studenten im Studiengang Informatik durchgeführt wurden.