KIT REPORT 125

Abbild oder Konstruktion – Modellierungsperspektiven in der Informatik

---

Martin Fischer, Gernot Grube, Fanny-Michaela Reisin (Hrsg.)

Technische Universität Berlin
FR 6–10, Franklinstr. 28/29
10587 Berlin, Germany

fischli / gernot / reisin@cs.tu-berlin.de

Dezember 1995

Abstract

This report contains some results of a student project on modeling. We introduce the representation view and the construction view as two complementary views of modeling in computer science.
By interpreting basic philosophical and social science texts according to constructivistic approaches and approaches based on the so-called correspondence theory of truth, the two views are motivated, explained and related to some questions in computer science.
Then problems, modeling approaches and modeling methods in the field of software engineering and artificial intelligence are analyzed, interpreted and classified in terms of the complementary views of representation and construction.

Zusammenfassung

Der Bericht enthält Ergebnisse eines Studienprojektes zum Thema Modellierung. Abbild- und Konstruktionsperspektive werden als komplementäre Sichten auf informatische Modellierung eingeführt.
Diese Perspektiven werden anhand der Interpretation geisteswissenschaftlicher Grundlagentexte zu korrespondenztheoretischen und konstruktivistischen Ansätzen motiviert, erläutert und zu informatischen Fragestellungen in Beziehung gesetzt.
Anschließend werden Probleme, Modellierungsansätze und Modellierungsmethoden aus Software-Engineering und KI im Lichte der Abbild- und Konstruktionsperspektive analysiert, interpretiert und eingeordnet.

Vorwort

“Modellbildung ist zentral in allen Arbeitsfeldern der Informatik. Wir betrachten Modellierung als Oberbegriff für die Begriffe Repräsentation, Design, Konstruktion, die wie Modellierung eine Doppelbedeutung als Tätigkeit, Vorgang, Prozeß und als Produkt, Ergebnis, Gegenstand haben. Dieser Gegensatz wird in dem Wortpaar Programm - Programmierung deutlich, das wir ebenfalls als Spezialfall von Modellierung auffassen. Unser Interesse gilt dabei stärker dem Modellieren, Programmieren, etc. als etwa den Programmen als Gegenständen und deren formaler Transformation.”[1]

Im Rahmen eines von den Herausgeber/innen betreuten Studienprojektes im Sommersemester 1994 und Wintersemester 1994/95 im Studiengang Informatik an der TU Berlin haben wir versucht uns der Frage nach informatischer Modellierung auf drei Ebenen zu nähern:

“1. Aus theoretischer Sicht stellen sich Fragen nach den Begriffen Modell und Modellierung.
2. Aus einer disziplinären Sicht wollen wir Modellierungsansätze aus verschiedenen Bereichen der Informatik und aus anderen Disziplinen kennenlernen und einordnen.
3. Anhand praktischer Aufgaben, die mit Modellierungswerkzeugen bearbeitet werden, erproben die Teilnehmer/innen ihr gewonnenes Modellierungsverständnis und reflektieren ihre Erfahrungen.”[2]

Im ersten Teil des Projektes standen theoretische Sichten auf das Verhältnis von Modell und Realität im Vordergrund. Dabei kristallisierten sich zwei Perspektiven auf Modell und Modellierung heraus. Diese von uns Abbild- und Konstruktionsperspektive genannten Sichten werden in Teil I dieses Bandes eingeführt und erläutert.

In Teil II gehen wir auf den geistesgeschichtlichen Hintergrund dieser Perspektiven anhand korrespondenztheoretisch orientierter und konstruktivistischer Autoren ein und versuchen Brücken zu informatischen Fragestellungen zu schlagen.

[1] Zitat aus der Ankündigung der Lehrveranstaltung ‘Projekt: Modellierung’.
[2] Ebenfalls zitiert aus der Ankündigung der Lehrveranstaltung.

Diese theoretischen Fragestellungen wurden im ersten Teil des Projektes konfrontiert mit praktischen Erfahrungen beim Modellieren. Die Teilnehmer/innen modellierten Aspekte eine Bibliothek, wobei eine Gruppe die Programmiersprache Smalltalk und eine zweite Gruppe eine KI-Sprache zur Wissensrepräsentation verwendete. Dabei wurde bewußt auf eine Vorstudie verzichtet, um gerade den Prozeß der Erstellung erster Modelle in den Blick zu bekommen. Die Ergebnisse dieser praktischen Aufgabe werden in diesem Band nicht explizit thematisiert, jedoch spiegeln sich die Erfahrungen, die die Autorinnen und Autoren bei dieser praktischen Arbeit gemacht haben, in den Texten in vielfältiger Weise wider.

Im zweiten Teil des Projektes wurden Konzepte und Methoden aus der Softwareentwicklung und der Künstlichen Intelligenz mit Hilfe der im ersten Teil des Projektes entwickelten Perspektiven auf das ihnen zugrunde liegende Modellierungsverständnis hin untersucht. Diese Arbeiten finden sich in Teil III dieses Bandes.

Für die Softwareentwicklung werden zum einen die historische Entwicklung von Methoden der Anforderungsanalyse am Beispiel von Anforderungsbeschreibungssprachen (Requirement Definition Languages) betrachtet. Grundkonzepte der informatischen Modellierung werden anhand von Texten von John Backus, David L. Parnas und Peter Naur untersucht. Zum dritten wird der partizipative Ansatz PETS (Partizipation, Evolution und Transparenz bei der Softwareentwicklung) in Abgrenzung zu einer produktionsorientierten Sicht, dem Phasenmodell, beleuchtet.[3]

Für die Künstliche Intelligenz werden Methoden zur Unterstützung des Wissensakquisitionsprozesses, die Methodologie KADS (Knowledge Analysis and Design Support) und der Ansatz “sloppy modeling”, untersucht. Dem sind allgemeinere Bemerkungen zum Wissensbegriff in der Künstlichen Intelligenz vorangestellt.

Teil IV enthält ein Resümee der Arbeit, sowie einen Ausblick auf weitergehende Fragestellungen, die sich im Anschluß an die hier vorgestellten Überlegungen ergeben.

Wir möchten Steffen Barthel, Helga Chelli, Katrin Gaßner, Ingo Herpolsheimer, Raffi Mezaduryan, Omar Mzee und Jutta Romberg unseren Dank aussprechen, deren Beiträge im Rahmen von praktischer Arbeit, Vorträgen und Diskussionen im Projekt wesentlich zum Entstehen dieses Berichts beigetragen haben.

[3] Im Projekt haben wir auch objektorientierte Ansätze mit einbezogen, was sich aber in dem vorliegenden Band nicht widerspiegelt.

Inhaltsverzeichnis

Teil I

Einführung

Kapitel 1

Modellierung in der Informatik
Autor: Gernot Grube

1.1 Das Konzept der Modellierungsperspektiven

Modell und Modellbegriff

Der Begriff des Modells wird in der Informatik sehr vielseitig gebraucht, wenn man bedenkt, in wie vielen sehr unterschiedlichen Kontexten er dort Verwendung findet. Etwa sprechen wir vom Simulationsmodell, vom Analysemodell, vom Softwareentwicklungsmodell, vom Datenmodell, von der Programmspezifikation als Modell des Problems oder einer Aufgabenstellung, vom Semantikmodell, oder vom Benutzermodell. Die Sonderrolle des Modellbegriffs zeigt sich auch sofort, wenn wir ihn mit anderen zentralen Begriffen der Informatik vergleichen. Solche Begriffe, wie etwa Compiler, Programmspezifikation oder Syntax einer Programmiersprache, sind in ihren Gebrauchsmöglichkeiten gegenüber dem Modellbegriff erheblich eingeschränkter. Sie helfen gegebenenfalls ein Teilgebiet der Informatik, zum Beispiel das des Übersetzerbaus, gegenüber anderen Teilgebieten abzugrenzen, während der Modellbegriff in nahezu jedem Teilgebiet der Informatik gebräuchlich ist.

Minimale Explikation des Modellbegriffs

Um zu entscheiden, ob etwas, das den Titel Modell trägt, auch eines ist, oder ob es sich bei etwas, das nicht Modell genannt wird, doch um ein solches handelt, ist eine Explikation des Modellbegriffs nötig. In aller erster Näherung könnte eine Explikation folgendermaßen ausfallen: X ist ein Modell, wenn X ein Y abbildet, wobei Y das Original genannt wird. Nicht gerade überraschend aber entscheidend ist, daß das Modell Bestandteil einer zweistelligen Relation ist. Daher erwartet

man von einer guten Explikation, daß sie erstens näher bestimmt, wodurch sich die beiden Relata Modell und Original unterscheiden, und zweitens, um welche Relation es sich handelt. Es ist schwierig, eine gute Explikation von ausreichender Allgemeinheit zu finden. Etwa können sich die Relata dadurch unterscheiden, daß das Modell eine Skizze, ein Graph oder ein Programm ist und das Original ein „Ausschnitt der Wirklichkeit", Gegenstände oder Vorgänge. Sie können sich aber auch dadurch unterscheiden, daß das Modell etwas Konkretes und das Original ein abstraktes Konzept ist. Und es muß auch noch einen Unterschied geben, wenn es sich um ein Modell von einem Modell handelt. Herbert Stachowiak gibt in seiner sehr umfangreichen Arbeit zu einer allgemeinen Modelltheorie an, daß etwas immer für einen bestimmten Zweck Modell zu einem Original ist. So unterscheidet sich das Modell immer durch seinen Zweck vom Original. Das Modell ist nicht mit dem Original identisch, insofern es das Original für bestimmte Zwecke und es daher „verkürzt" abbildet.[1] Der Grundgedanke für eine nähere Bestimmung der Relation ist bei Stachowiak, daß es in irgendeinem Sinne eine Abbildung sein muß.[2]

Unter Modellierung können wir also einen Prozeß verstehen, durch den für bestimmte Zwecke eine Abbildung konstruiert wird.

Abbildungsperspektive

Der angeführte Explikationsversuch drückt eine bestimmte Modellvorstellung aus. Wir nennen diese Vorstellung die Abbildperspektive. Die Abbildperspektive legt zum einen fest, was ein Modell ist, und zum anderen, welchen Bedingungen ein Modellierungsprozeß unterliegt, damit durch ihn ein Modell konstruiert wird. Die Abbildperspektive sollte also Kriterien implizieren, an denen die Modellierung orientiert ist und nach denen etwas als Modell erkannt und bewertet werden kann. Diese Kriterien beruhen auf wichtigen Grundannahmen, die die Abbildperspektive enthält. So wird angenommen, daß die Relata unabhängig voneinander sind (Unabhängigkeitsannahme). Das heißt, etwa ein Sachverhalt, ein Original, zu dem wir ein Modell konstruieren, existiert auch ohne dieses Modell. Und es läßt sich ein Modell erfinden, zu dem wir zu einem bestimmten Zeitpunkt noch kein Original haben. Eine weitere Annahme ist die, daß Modell und Original miteinander verglichen werden können (Vergleichbarkeitsannahme). Beide Annahmen beziehen sich gleichermaßen auf die Relata und die Relation. Um beurteilen zu können, ob zwischen Modell und Original eine Abbildungsrelation besteht, muß es möglich sein, Modell und Original miteinander zu vergleichen. Der Vergleich zwischen Modell und Original erscheint sinnvoll, da das Original unabhängig vom Modell gegeben ist. Falls ein Vergleich zur Revision des Modells

[1] S. [Stachowiak 1973] zum sogenannten Verkürzungsmerkmal und zum pragmatischen Merkmal.

[2] S. [Stachowiak 1973] zum sogenannten Abbildungsmerkmal.

führt, bleibt das Original von dieser Revision unberührt.

Zwei sehr wichtige Kriterien der Abbildperspektive sind Korrektheit und Vollständigkeit. Bei der Konstruktion eines Modells ist zu beachten, daß das Modell korrekt und vollständig ist. Hier ist die Beziehung zwischen Modell und Original analog zur Beziehung zwischen einem Kalkül und seinen Anforderungen zu sehen. So wie vom Kalkül verlangt wird, daß er in bezug auf seine Anforderungen korrekt und vollständig ist, so wird von einem Modell in bezug auf sein Original verlangt, daß es korrekt und vollständig ist. Damit diese Kritierien angewendet werden können, müssen Unabhängigkeits- und Vergleichbarkeitsannahme erfüllt sein. Es ist das Modell, das gegebenenfalls nicht korrekt und unvollständig ist. Ein unvollständiges Modell ist keine Abbildung des Originals. Weil das Original unabhängig vom Modell gegeben ist und dieses mit jenem verglichen werden kann, läßt sich feststellen, ob ein Modell korrekt und vollständig ist, oder ob es noch einmal revidiert werden muß. Einerseits spezifizieren die beiden Kriterien die Relation, indem sie festlegen, daß es sich nur dann um eine Abbildung handelt, wenn das Modell korrekt und vollständig ist. Andererseits spezifizieren sie die Relata, indem sie voraussetzen, daß diese die beiden Grundannahmen erfüllen.

Beispiel und Gegenbeispiel

Nehmen wir an, die Aufgabenstellung lautet für ein Softwareentwicklungsteam, ein Datenmodell zu entwickeln, das einen Güterbahnhof abbildet, zu dem Zweck, die Signal- und Weichenschaltung durch ein Steuerungsprogramm zu unterstützen. Dann muß die Modellierung irgendwo ihren Ausgang nehmen. Der Ausgangspunkt könnte eine Vorstudie sein, die der Auftraggeber zur Verfügung stellt. Die Vorstudie ist zum Beispiel ein Prosatext mit Tabellen und Diagrammen, der den Güterbahnhof beschreibt. Sehr wichtig ist es, daß dieses Dokument seinen Gegenstand eindeutig und unmißverständlich beschreibt. Indem sich das Entwicklungsteam auf die Vorstudie stützt, wird unter bestimmten Zwecken, vor allem der Anforderungen des Auftraggebers, ein Datenmodell konstruiert. Die Vorstudie spielt die Rolle des Originals.

In dem Beispielfall sind die Grundannahmen erfüllt, so daß die Kriterien für die Entwicklung und Bewertung des Datenmodells angewendet werden können. Das Original existiert unabhängig vom Modell, und sofern das Original tatsächlich unmißverständlich und eindeutig ist, ergeben sich auch beim Vergleich zwischen ihm und dem Modell keine prinzipiellen Schwierigkeiten. Der Vergleich zwischen der Konstruktion des Datenmodells und dem Original muß vorgenommen werden, um festzustellen, ob das Modell korrekt und vollständig ist. Da die Vorstudie unabhängig vom Modell gegeben ist, muß das Datenmodell etwa wegen Unvollständigkeit revidiert werden bis es die Sachverhalte der Vorstudie, also die

Bedingungen des Güterbahnhofs wirklich abbildet. Damit die Kriterien angewendet werden können, kommt es auf die Beschaffenheit des Originals an. Wenn die Vorstudie eine Ungenauigkeit enthält oder an einer Stelle mißverständlich ist, dann ergibt sich dort ein Interpretationsspielraum und es ist nicht mehr klar zu entscheiden, ob es sich bei dieser oder jener Interpretation um eine korrekte und vollständige Abbildung handelt.

Also verlangen diese Kriterien, daß die Grundannahmen erfüllt sind, und das bedeutet, daß das Original bestimmte Eigenschaften besitzen muß. Es zeigt sich hier auch ein Zusammenhang zwischen den Grundannahmen, den Eigenschaften des Originals und den Kriterien. Während den Kriterien die Grundannahmen zugrundeliegen, hängt die Erfüllung der Grundannahmen von bestimmten Eigenschaften des Originals ab. Für den Vergleich mit dem Modell muß das Original unabhängig vom Modell und präzise gegeben sein. Jedenfalls scheint es nicht sinnvoll, die Kriterien anzuwenden, wenn die Unabhängigkeitsannahme zwar in irgendeinem Sinne erfüllt sein müßte, aber das Original eigentlich nicht die Eigenschaften besitzt, die den Vergleich mit dem Modell erlauben.

Betrachten wir nicht die Vorstudie sondern den Güterbahnhof als Original und die Vorstudie selbst als Modell, so wäre die Unabhängigkeitsannahme scheinbar noch erfüllt, jedoch nicht die Vergleichbarkeitsannahme. In welchem Sinne ist der Güterbahnhof eindeutig und unmißverständlich, damit ein Modell mit ihm verglichen werden könnte? Die Vorstudie bildet für die Modellierung einen Ausgangspunkt. Sie legt ein Original fest. Was ist über ein Original, hier über einen Güterbahnhof, zu sagen, bevor es nicht irgendwie festgelegt, nicht modelliert ist?[3] Ein Original, das als Bezug für eine Modellierung in der Abbildperspektive in Frage kommt, ist selbst ein Modell. Und unter diesen Umständen ist es zweifelhaft, ob wenigstens die Unabhängigkeitsannahme erfüllt ist, wenn der Güterbahnhof und nicht die Vorstudie als Original betrachtet wird. Offenbar weist „der Güterbahnhof“ nicht die Eigenschaften auf, die er als Original besitzen müßte, damit die beiden Grundannahmen erfüllt sind (und zwar für die Zwecke einer Modellierung).

Was wird aus der skizzierten Vorstellung zu Modell und Modellierung, der

[3] In dem Studienprojekt *Modellierung* an der TU (SS94 und WS94/95) gab es Arbeitsgruppen, die Teile einer Bibliothek modellieren sollten, ohne dafür auf eine Vorstudie zurückzugreifen. Die Entwicklerinnen konnten das Modell von einer oder mehreren existierenden Bibliotheken bilden. Da auf eine Vorstudie (auf ein Original mit bestimmten Eigenschaften) verzichtet wurde, ergaben sich auf der Suche nach dem Original „Bibliothek“ große Schwierigkeiten. Die Protokolle und Stellungnahmen der Arbeitsgruppen vermitteln den Eindruck, als hätte es für jeden Entwickler mindestens ein eigenes Original gegeben. Man brauchte einen Ausgangspunkt, ein Original, für die zweckmäßige Modellierung einer Bibliothek. Und es war nicht viel damit gewonnen, eine existierende Bibliothek zum Original zu erklären. Es war daher zunächst nötig, diesen Ausgangspunkt zu modellieren. Das Original mußte modelliert werden, damit das aufgabengemäße Modell einer Bibliothek auf ein Original bezogen werden konnte. Solche Erfahrungen legen die Vermutung nahe, daß es nicht möglich ist, ohne Modellierung ein Original zu haben.

Abbildperspektive, wenn die Grundannahmen nicht erfüllt sind? Die Unabhägigkeits- und Vergleichbarkeitsannahme liegen ja der Abbildperspektive und den von dieser implizierten Kriterien zugrunde.

Konstruktionsperspektive

Die Abbildperspektive läßt sich durch eine andere Vorstellung ersetzen, die die beiden Grundannahmen nicht notwendig voraussetzt. Wir haben diese andere Vorstellung in Anlehnung an die Auffassungen des radikalen Konstruktivismus und ausgehend von Ansätzen in der Wissenssoziologie die Konstruktionsperspektive genannt. In dieser Perspektive existieren Original und Modell nicht unabhängig voneinander, sondern konstituieren sich gegenseitig. Es wird davon ausgegangen, daß jedes Original immer schon das Ergebnis einer Modellierung ist. Ist in der Abbildperspektive Modellierung die Konstruktion des Modells, so ist in der Konstruktionsperspektive Modellierung ebenso die Konstruktion des Originals. Und in der Konstruktionsperspektive sind Modell und Original nur in dem Maße vergleichbar, in dem auch das Original das Ergebnis einer Modellierung ist.

Ein Kriterium der Konstruktionsperspektive, das anwendbar ist, ohne daß die Grundannahmen erfüllt sind, ist die Angemessenheit. Dieses Kriterium orientiert die Konstruktion eines Modells daran, inwiefern das Modell seinen Zwecken angemessen ist. Die Zwecke für die ein Modell konstruiert wird, sind jetzt entscheidend. Natürlich werden auch mit der Konstruktion eines Modells in der Abbildperspektive Zwecke verfolgt, aber das Korrektheits- und Vollständigkeitskriterium betreffen das Modell unabhängig von seinen Zwecken. Die Kriterien der Konstruktionsperspektive stützen sich nicht auf das Verhältnis zwischen einem Modell und einem Original, das die Grundannahmen erfüllt.

Erweiterte Explikation des Modellbegriffs

Damit die Explikation des Modellbegriffs die Konstruktionsperspektive ausdrückt, ist sie um den Gesichtspunkt zu erweitern, daß auch das Original immer für bestimmte Zwecke Original ist.[4] Diese Bedeutung der Zwecke in der Konstruktionsperspektive für das Modell sowie für das Original sollte sich auf die Kriterien auswirken, die diese Perspektive impliziert.

Tatsächlich muß die Anwendung des Angemessenheitskriteriums nicht auf die Zwecke beschränkt werden, denen das Modell unterliegt. Gehen wir davon aus, daß die Konstruktion des Modells nicht unabhängig von der Konstruktion des Originals ist und umgekehrt, so dürfte auch die Angemessenheit des Modells nicht unabhängig von der Angemessenheit des Originals sein und umgekehrt. Da das

[4]Das hieße in bezug auf Stachowiaks Merkmale, das Verkürzungs- und das pragmatische Merkmal auch auf das Original zu beziehen.

Original in der Konstruktionsperspektive nicht unabhängig gegeben ist, steht auch seine Angemessenheit zur Disposition. Unter Umständen zieht die Anwendung dieses Kriteriums eine Revision des Originals nach sich.

Der wesentliche Unterschied zwischen Abbild- und Konstruktionsperspektive zeigt sich an der Funktion, die das Original erfüllt. In der Abbildperspektive wird Modellierung auf den Prozeß beschränkt, der das Modell konstruiert. In der Konstruktionsperspektive ist auch das Original ein Ergebnis eines Modellierungsprozesses. Zwecke sind unter dieser Perspektive für Original und Modell relevant.

Der theoretische Hintergrund der Abbild- und Konstruktionsperspektive

Es gibt in der Informatik ein ziemlich verbreitetes Bild, das eine elementare Modellierung beschreibt. Links im Bild schwebt eine Wolke, das ist ein Stück Wirklichkeit, und rechts im Bild befindet sich ein Rechteck, das ist das Modell der Wirklichkeit. Dies Motiv steckt auch noch in komplexeren Bildern, die den Modellierungsvorgang subtiler darstellen. In diesem Bild sind die Modelle Konstruktionen, die sich auf die Wirklichkeit beziehen, die von den Konstruktionen unabhängig existiert. Je präziser unsere Modelle die Wirklichkeit abbilden, desto besser können wir mit ihrer Hilfe in der Wirklichkeit operieren. Die Aktivitäten der Informatikerinnen, die an diesem Bild orientiert sind, bestätigen es durch ihre Erfolgsrate. Es gibt allerdings eine ganze Reihe sogenannter *folk theories* oder *common sense theories*, die in praxi erfolgreich aber von einem wissenschaftlichen Standpunkt aus unannehmbar sind.[5]

Die Diskussion über den Wahrheitsgehalt des zitierten Bildes reicht weit zurück in die Theoriegeschichte und wird heute weitergeführt. Eine der problematischen Fragen lautet, ob nur das Modell eine der Wirklichkeit verpflichtete Konstruktion sei, oder ob nicht auch die Wirklichkeit selbst, die objektiven Tatsachen, ebenso als eine Konstruktion verstanden werden müßte. Diese Auseinandersetzung wird von einer Grundvorstellung über die Beziehung zwischen Welt und Modellen beherrscht, nach der unser Erkenntnisvermögen einem Spiegel vergleichbar ist und ein gutes Modell einem guten Spiegelbild entspricht. Der Spiegel und die Spiegelbilder (die Modelle) gehören in den Bezirk des Subjektiven und die Wirklichkeit, die es zu spiegeln gilt, gehört in den Bezirk des Objektiven. Über diese Grundvorstellung sind in der Philosophie,[6] der Psychologie[7] und der Soziologie[8] heftige Auseinandersetzungen geführt worden. Dabei handeln die Fragen,

[5] Empirische Untersuchen der Psychologen illustrieren diesen Sachverhalt.

[6] S. z.B. [Rorty 1981].

[7] S. z.B. [Watzlawik 1976].

[8] S. z.B. [Berger und Luckmann 1969] und vgl. das Kapitel *Gesellschaftliche Konstruktion der Wirklichkeit* in diesem Report.

um die es in den Auseinandersetzungen geht, von Erkenntnisprozessen. Statt von der Modellierung der Wirklichkeit wird von der Erkenntnis der Wirklichkeit gesprochen.

Geht man von Erkenntnisprozessen aus, und begreift man sie als Modellierungsprozesse, die sich auf die Wirklichkeit beziehen, so lassen sich in der Auseinandersetzung um die Beziehung zwischen Modell und Wirklichkeit drei Positionen unterscheiden. Wir nennen sie erstens die *Position der Abbildtheorie*, zweitens die *pragmatische Position* und drittens die *konstruktive Position*. In der Literatur lassen sich alle drei Positionen sehr gut identifizieren. Die Texte von Francis Bacon etwa sind ein Dokument, dem die Position der Abbildtheorie zugrunde liegt.[9] Die pragmatische Position in ihrer Wirkung auf ein Verständnis der Beziehung zwischen Modell und Wirklichkeit, kommt sehr klar in den Texten von William James zum Ausdruck.[10] Die konstruktive Position wird im Rahmen einer Symboltheorie von Nelson Goodman vertreten, dem es dabei besonders auf die Beziehung zwischen Symbol und Symbolisiertem ankommt.[11]

Die oben angesprochene Grundvorstellung zur Beziehung zwischen Welt und Modell paßt gut zur Position der Abbildtheorie, während die beiden anderen Positionen ohne weiteres als Kontrahenten der Abbildtheorie dargestellt werden können. So wird nach der Abbildtheorie angenommen, es existiere unabhängig von Erkenntnismodellen unsere Wirklichkeit. Diese Wirklichkeit, die die Rolle des Originals spielt, ist objektiv gegeben. Falls es gelingt, Wirklichkeit in einem Modell abzubilden, so haben wir ein richtiges Modell, eine wahre Erkenntnis. Nach der pragmatischen Position ist die Wirklichkeit nicht notwendig ein unbeeinflußbarer und objektiver Erkenntnisgegenstand. Unsere Modelle könnten beachtlichen Einfluß auf die Zusammensetzung der Wirklichkeit haben. Nach dieser Position ist es eine falsch gestellte Frage, ob unsere Modelle eine objektive Wirklichkeit abbildeten oder, ob sie diese erst konstruierten. Ein Modell ist *gut*, nicht weil es ein wahres Abbild der Wirklichkeit ist, sondern weil wir mit ihm erfolgreich in der Wirklichkeit operieren können. Die konstruktive Position lehnt die Annahme, daß es eine objektive Wirklichkeit für uns gibt, vollständig ab. Im Stile der Goodmannschen Ausdrucksweise heißt es nach dieser Position, daß Modelle die Wirklichkeit nicht abbilden, sondern erschaffen. Eine Wirklichkeit für sich genommen, die gegenüber den Modellen einen besonderen Objektivitätsgehalt hätte, gibt es nicht. Es zeigt sich in dem Für und Wider unter den drei Positionen, daß den eigentlichen Kontrast die abbildtheoretische und die konstruktive Position bilden, während die pragmatische Position beinahe in einer Vermittler-

[9] Vgl. hierzu das Kapitel *Wissenschaft als Abbildung der Realität* in diesem Report und das Kapitel *Bedeutung und Metasprache: Alfred Tarski*.

[10] S. z.B. [James 1977].

[11] Vgl. zur Position von Goodman in diesem Report das Kapitel *Symbolische Konstruktion der Wirklichkeit* und außerdem als Kommentar zur konstruktiven Position das Kapitel *Gesellschaftliche Konstruktion der Wirklichkeit*.

rolle erscheint, da nach ihr die ontologische Frage, die Frage nach dem Status der Wirklichkeit, in der sich die beiden anderen Positionen widersprechen, gar keine interessante Frage ist. Auf der anderen Seite ist es möglich, die wesentlichen Eigenschaften eines Modells, die ihm die pragmatische Position zuschreibt, auch unter den Eigenschaften zu finden, die Modelle nach der abbildtheoretischen oder der konstruktiven Position besitzen. So ist die zweckbestimmte „Verkürzung“ in der Abbildtheorie eine Eigenschaft der Modell-Original-Beziehung, die von der pragmatischen Position berücksichtigt wird. Für die konstruktive Position gilt, daß die Originale, die etwa mit Realitäten identifiziert werden, nicht willkürlich durch Modelle konstruiert werden können, sondern ihrerseits die Konstruktionen der Modelle „begrenzen“. Diese Eigenschaft der Modell-Original-Beziehung wird von der pragmatischen Position in den Vordergrund gerückt. Die pragmatische Position zeichnet sich gegebenenfalls dadurch aus, daß sie auf kontroverse Annahmen schlicht verzichtet. Der scharfe Kontrast in der Auseinandersetzung um die Beziehung zwischen Modell und Wirklichkeit besteht zwischen der abbildtheoretischen und der konstruktiven Position. In diesem Kontrast finden wir konkurrierende Auffassungen zum Modell und darüber, was ein Erkenntnis- beziehungsweise ein Modellierungsprozeß ist. Diese beiden Auffassungen betrachten wir als theoretischen Hintergrund der beiden Modellierungsperspektiven.

Die Modellierungskriterien

Die Kriterien haben einen außergewöhnlichen Stellenwert, da sie den Prozeß der Modellierung dirigieren und auf ihrer Grundlage entschieden werden kann, was ein Modell ist und von welcher Qualität es ist. Gibt es eine plausible Vorgehensweise, um Kriterien, die die Perspektiven implizieren, in einer Kriterienliste zusammenzustellen? Können wir erwarten, daß sich jedes Kriterium dieser Liste immer eindeutig einer der beiden Perspektiven zuordnen läßt?

Das Korrektheits- und das Vollständigkeitskriterium können offenbar eindeutig der Abbildperspektive zugeordnet werden, da sie in der Konstruktionsperspektive nicht anwendbar sind, wenn die Unabhängigkeits- und die Vergleichbarkeitsannahme nicht erfüllt sind. Weil für die Konstruktionsperspektive lediglich gilt, daß sie die Erfüllung dieser Annahmen *nicht notwendig* voraussetzt, ist es denkbar, daß die Annahmen erfüllt sind und zugleich die Konstruktionsperspektive eingenommen wird. Dann wären auch in dieser Perspektive beide Kriterien anwendbar.

Offenbar kann das Angemessenheitskriterium der Konstruktionsperspektive nicht vergleichbar eindeutig zugeordnet werden. Zwar bezieht es sich in dieser Perspektive auf Modell und Original, aber es muß sich nicht auf Modell und Original beziehen. Und wenn es lediglich auf das Modell angewendet wird, so kann es auch zu den Kriterien der Abbildperspektive gerechnet werden.

Werden also die Kriterien von den Perspektiven impliziert oder können sie je

nach Perspektive verschieden interpretiert werden? Vielleicht ließe sich sogar ein Korrektheitsbegriff definieren, der unabhängig von den Grundannahmen (Unabhängigkeits- und Vergleichbarkeitsannahme) gebraucht werden kann

Das Kriterium Verständlichkeit läßt sich offensichtlich auch in beiden Perspektiven anwenden. Die Intention, die mit diesem Kriterium verfolgt wird, daß eine Modellkonstruktion für die Modellbenutzerinnen verständlich sein soll, daß bei der Konstruktion besonders auf Verständlichkeit zu achten ist, ist mit der Abbild- und der Konstruktionsperspektive verträglich. In der Abbildperspektive wäre das Modell solange Revisionen zu unterwerfen, solange es nicht ausreichend verständlich ist. In der Konstruktionsperspektive wird das Modell aus den gleichen Gründen Revisionen unterworfen, aber es ist darüber hinaus möglich, daß auf dem Wege, ein verständliches Modell zu konstruieren, auch das Original revidiert wird. Da das Original in der Konstruktionsperspektive auch als Konstruktion betrachtet wird, und die Original- und Modellkonstruktionen immer ineinandergreifen, ist es naheliegend anzunehmen, daß die Verständlichkeit des Modells von der des Originals in starkem Maße abhängt, und eine Revision des Originals den gesamten Modellierungsprozeß auf dem Wege zum verständlichen Modell erleichtern könnte. Möglicherweise gibt es Kriterien wie etwa das der Verständlichkeit, die bei ihrer Anwendung in der Abbildperspektive nur eingeschränkt angewendet werden können.

Welcher Perspektive wir ein Kriterium zuordnen scheint von der Wirkung abzuhängen, die wir uns von seiner Verwendung für die Modellierung versprechen. Daher könnte es ein gangbarer Weg sein, Kriterien und Perspektiven zuzuordnen, und gemäß der Perspektiven eine Kriterienliste zusammenzustellen, wenn wir die Intention, die einem Modellierungskriterium zugrunde liegt, mit den Modellierungsmöglichkeiten verknüpfen, die eine Perspektive zuläßt. Diese Liste läßt sich dann in drei Schritten zusammenstellen. Der erste Schritt ist die Darstellung der Möglichkeiten, die die Perspektiven den Kriterien vorgeben. Der zweite Schritt ist, zu prüfen welches Modellierungskriterium, das in der Informatik gebraucht wird, welche Möglichkeiten beansprucht oder beanspruchen sollte. Im dritten Schritt sind neue Kriterien zu bestimmen, die die Perspektiven aufgrund der Modellierungsmöglichkeiten bieten.

Zunächst zu den Möglichkeiten, die der Anwendung der Kriterien in der Abbildperspektive zur Verfügung stehen. 1.) In der Abbildperspektive ist das Original fixiert (es ist eher eine Tatsache als eine Konstruktion) und das Modell eine Konstruktion, die während der Modellierung ständigen Änderungen unterliegt. Für die Modellierung ist das Original so etwas wie die letzte Instanz. 2.) Modellieren in der Abbildperspektive heißt, das Modell konstruieren, es solange verändern, bis es das Original abbildet. Die Qualitäten des Modells hängen davon ab, ob es bestimmte Anforderungen bezüglich des Originals erfüllt beziehungsweise das Original tatsächlich abbildet. 3.) In dieser Perspektive müssen Kriterien bestimmt werden, die abhängig vom gegebenen Original festlegen, wann eine Modellierung

mit einem guten Modell abgeschlossen ist. Die Kriterien sollten in der Objektivität des Originals verankert sein.

Demgegenüber zu den Möglichkeiten in der Konstruktionsperspektive: 1.) In dieser Perspektive können Modell und Original, letzteres als die Konstruktion eines ausgezeichneten Modells, Änderungen unterliegen. In dieser Perspektive verliert das Original den Status der letzten Instanz und zwar an die Modellierenden beziehungsweise sämtliche Personen, die in einem Zusammenhang mit der Modellierung stehen. 2.) Modellieren in dieser Perspektive heißt, Modell und Original solange zu konstruieren bis bestimmte Interessen der Modellierenden befriedigt sind. Die Qualitäten des Modells hängen direkt von den Interessen der Modellierenden und nur indirekt vom Original ab. 3.) In der Konstruktionsperspektive müssen Kriterien bestimmt werden, die in Rücksicht auf die Interessen der Modellierenden festlegen, auf welche Weise ein gutes Modell konstruiert werden kann. Auch der Abschluß einer Modellierung ist eine Sache der Interessenlage der Modellierenden. Die Kriterien sollten in der Interessenlage der Modellierenden verankert sein.

Die Kriterien, die entweder der Abbild- oder der Konstruktionsperspektive zugeordnet werden, sollten also der Verschiebung vom besonderen Status des Originals in der Abbildperspektive zum besonderen Status der Modellierenden in der Konstruktionsperspektive Rechnung tragen. In der erweiterten Explikation des Modellbegriffs für die Konstruktionsperspektive zeigt sich diese Verschiebung, indem dort auch das Original von Zwecksetzungen abhängig gemacht wird. Die Zwecke sowohl für das Modell als auch für das Original gehen auf die Interessen der Modellierenden zurück beziehungsweise sämtlicher Personen, die in einem Zusammenhang mit der Modellierung stehen. Diese Kriterien müssen bei diesen Interessen ansetzen und dürfen von der Revidierbarkeit von Modell und Original ausgehen.

1.2 Die Modellierungsperspektiven in der Informatik

Die Rolle der Modellierung in der Informatik

Die Überlegungen zum *Konzept der Modellierungsperspektiven* sollen zeigen, daß es Gründe gibt, zwei auseinanderliegende Vorstellungen zu Modell und Modellierung zu unterscheiden. Im folgenden gilt es, die Rolle der Modellierung in der Informatik zu skizzieren, denn wenn der Unterschied zwischen Abbild- und Konstruktionsperspektive folgenreich ist und Modellierung in der Informatik eine zentrale Rolle spielt, dann ist es interessant, die Folgen dieses Unterschieds für die Informatik zu untersuchen.

Ein Hauptgegenstand der Informatik sind Programme. Wenn man bedenkt, wie unterschiedlich Programme sein können, etwa in Abhängigkeit von ihrem Verwendungskontext oder ihrer Funktionalität, dann erscheint es ziemlich schwer, eine Definition des Programms zu geben, die solche Unterschiede zusammenfassend angemessen berücksichtigen kann. Die Definition müßte vielleicht so allgemein ausfallen, daß mit ihr für die existierenden Programme wenig gewonnen wäre. Eine Gemeinsamkeit aller Programme ist jedoch in unserem Zusammenhang beachtenswert, nämlich ihre Zeichenhaftigkeit. Programme sind Zeichen. Sie sollten immer eine Bedeutung haben. Wer mit Programmen zu tun hat, muß sich mit der Relation zwischen Programm und Bedeutung befassen.

Es sieht so aus, als spiegelten die zahlreichen Ausdrücke in der Informatik, die das Wort Modell enthalten, eine Tendenz wider, die Relation Programm und Bedeutung als eine Relation zwischen Modell und Original zu betrachten. In einigen Fällen liegt eine solche Betrachtung sehr nahe, zum Beispiel bei einem Programm, das einen Vorgang der Wirklichkeit simulieren soll, oder einem Programm, das das Modell für Expertenwissen sein soll. Für den allgemeinen Fall können wir versuchen, die Zeichenhaftigkeit der Programme mit der der Modelle zu identifizieren. Das Modell interessiert uns ja immer in bezug auf ein Original, wie uns ein Programm in bezug auf seine Bedeutung interessiert. Wenn wir Programme allgemein als Modelle betrachten, dann ist die Rolle der Modellierung in der Informatik beachtlich.

Es liegt nahe, einige Modellierungsvorgänge in der Informatik scharf gegeneinander abzugrenzen. Man denke etwa an die Modellierung einer Vorstudie, einer Programmspezifikation, eines Programmsystems, eines Compilers oder einer Benutzeroberfläche. Die Begründung aber dafür, daß Modellierung eine herausragende Rolle spielt, soll sich nicht auf die vielen konkreten Modellierungssituationen oder -aufgaben in der Informatik stützen, sonderen allgemeiner darauf, daß ein Hauptgegenstand der Informatik, die Programme, als Modelle und die Aktivitäten der Informatikerinnen, letztlich Programme zu entwickeln, als Modellierungen betrachtet werden können. Die Unterscheidung verschiedener Modellierungsvorgänge in der Informatik, die von dem Unterschied zwischen Abbild- und Konstruktionsperspektive ausgehen soll, wird sich allerdings gerade mit konkreten Modellierungsaufgaben zu befassen haben. Es ist denkbar, daß die Unterscheidung in Abbild- und Konstruktionsperspektive eine Grundlage bietet, um verschiedene Modellierungsvorgänge aufgrund verschiedener Modellierungsaufgaben gegeneinander abzugrenzen.

Wir nehmen an, daß es gute Gründe gibt, die Vorstellungen zu Modell und Modellierung in der Informatik generell in die Abbild- und die Konstruktionsperspektive zu klassifizieren, und daß diese *Klassifikation* bislang weder auf einer *analytisch–diagnostischen Ebene* noch auf einer *praktischen*, eher *orientierenden Ebene* ausreichend geleistet worden ist. Darüber hinaus haben wir die Vermutung, obwohl beide Vorstellungen in der Disziplin vorzufinden sind, daß beide

überwiegend unversöhnlich gegeneinander vertreten werden, wobei sich die Abbildperspektive deutlich besser als die Konstruktionsperspektive behauptet. Und wir vermuten, daß die Konstruktionsperspektive eine angemessene Beschreibung der Aktivitäten der Informatiker erlaubt.

Die Bedeutung des Perspektivenkonzepts für die Informatik

Wenn es eine wesentliche Differenz zwischen den beiden skizzierten Perspektiven gibt und Modellierungen in der Informatik eine herausragende Rolle spielen, dann interessieren uns die Konsequenzen dieser Differenz für Modellierungen in der Informatik. Es lassen sich die Konsequenzen für die Informatik in engere und weitere unterscheiden. Engere Konsequenzen sind solche, die unmittelbar die Modellierung betreffen, und weitere solche, die das wissenschaftliche Selbstverständnis der Disziplin betreffen.

Zunächst zu den Konsequenzen im engeren Sinne. (1) Es könnte sich herausstellen, daß Modellierungsprobleme, die beim Modellieren in der Abbildperspektive aufgetreten sind, beim Wechsel in die Konstruktionsperspektive nachvollziehbar und erklärbar werden. (2) Eventuell werden auf diese Weise ganze Problemkomplexe wie etwa die Softwarekrise oder das Scheitern der symbolverarbeitenden KI durchschaubar. (3) Es ist denkbar, daß in der Abbildperspektive der Einfluß der Modellkonstruktion auf das Original schlicht verschleiert bleibt. (4) Je nach Perspektive müßten Methoden und Werkzeuge unterschiedlich konzipiert sein, so daß die Eignung von Methoden und Werkzeugen von der Perspektive abhängt. (5) Es könnte sein, daß die eine Perspektive einer bestimmten Modellierungsaufgabe angemessener ist als die andere. (6) Und es lassen sich eventuell sogar verschiedene Modellierungssituationen unterscheiden, wobei in Modellierungssituationen, in denen die Grundannahmen erfüllt sind, die Abbildperspektive einzunehmen wäre und sonst die Konstruktionsperspektive. (7) Vielleicht läßt sich vor dem Hintergrund der beiden Perspektiven der Konflikt besser begreifen, der zwischen der Forderung nach präzisen (formalisierten) Modellen, um von kontingenten Interpretationen der Modellierenden unabhängig zu sein, und der anderen Forderung, Modelle partizipativ zu entwickeln, heftig hin und her geht.

Während die Konsequenzen im engeren Sinne noch offensichtlich von technischer Relevanz sind, da sie unmittelbar auf konkrete Modellierungen bezogen sind, haben die Konsequenzen im weiteren Sinne keine direkt technische Relevanz, da sie sich auf Grundsätze des wissenschaftlichen Arbeitens beziehen. (1) Stellt die Abbildperspektive die Modellierung als einen objektivierbaren Prozeß dar, der, wenn er wissenschaftlich voll entwickelt ist, unabhängig von bestimmten Interpreten beziehungsweise Modellierenden durchgeführt werden kann? Außerdem sollte der Modellierungsprozeß, wenn er objektiviert ist, auch automatisierbar sein. Demgegenüber stellt die Konstruktionsperspektive Modellierung

als einen Prozeß dar, in dem sich die Interessen der Modellierenden realisieren, und der daher nicht objektiviert werden kann, indem er von diesen Interessen abgelöst wird. (2) Eventuell liefern die beiden Perspektiven eine Interpretationsfolie, durch die ein „Grundlagenstreit“ in der Informatik durchschaubar und auflösbar wird. Der Streit zwischen denjenigen, die die Informatik gänzlich oder vor allem als Ingenieurwissenschaft verstanden wissen wollen, und denjenigen, die sie zu großen Teilen als eine Sozialwissenschaft verstehen. (3) Es könnte sein, daß es nur in der Konstruktionsperspektive möglich ist, die Aktivitäten der Informatiker, nämlich die Modellierungen, angemessen zu beschreiben. Diese Konsequenz wäre für die Informatik als Wissenschaft von erheblicher Bedeutung, auch wenn sie direkt keine praktischen Folgen nach sich zöge. (4) Der Bereich der sogenannten Neuen Medien könnte möglicherweise nur in der Konstruktionsperspektive erfaßt werden. Denn im Zusammenhang mit den informatischen Artefakten, die unter dem Begriff *Neue Medien* versammelt werden, stellt sich die Schwierigkeit, von Originalen zu sprechen, besonders deutlich.

Modellierungserfahrungen

Eine interessante Frage ist es, ob beim Lösen konkreter Modellierungsaufgaben Hinweise auf die Perspektiven zu entdecken sind. Eine solche Frage berührt den Zusammenhang zwischen praktischen Modellierungen und dem Konzept der Modellierungsperspektiven. In dem Studienprojekt *Modellierung*, aus dem dieser Report hervorgegangen ist, wurden einige konkrete Modellierungsaufgaben durchgeführt. Etwa beschäftigte sich eine Arbeitsgruppe (A1) damit, ein Modell des Ausleih- und Rückgabevorgangs in einer Bibliothek zu entwickeln, und zwar mit Hilfe des objektorientierten Werkzeugs Smalltalk80. Eine andere Arbeitsgruppe (A2) versuchte, ein Modell des Such- und Archivierungsvorgangs von Dokumenten in einer Bibliothek zu entwickeln, und zwar mit Hilfe des Werkzeugs BACK, das als Wissensrepräsentationssprache konzipiert wurde.

Keiner der Arbeitsgruppen stand eine Vorstudie zur Verfügung oder andere Richtlinien, die bestimmte Bedingungen für die Modellierung vorgegeben hätten. Daher waren die Arbeitsgruppen zunächst mit der Frage nach dem Original beziehungsweise nach einem geeigneten Ausgangspunkt für die Modellierung beschäftigt. Gezwungenermaßen begannen sie ihre Modellierung damit, ein Ausgangsmodell zu entwickeln.

Bei der Entwicklung ihrer Ausgangsmodelle machten die Arbeitsgruppen verschiedene Beobachtungen. A1 stellt fest, daß die Konzeption des Ausgangsmodells stark geprägt war durch Bedingungen des Werkzeugs Smalltalk80.[12] A2

[12]Hierzu heißt es im Projektbericht: „... bemerkenswert ist, daß wir bereits zu diesem frühen Zeitpunkt [Ausgangsmodell] die Umsetzung des Modells mit einem Programmierwerkzeug im Kopf hatten und eine Unterscheidung von Büchern und ihren tatsächlich im Regal vorhandenen

stellt fest, daß es enorme Schwierigkeiten gegeben hat, ein Original zu entdecken, bei dem die Modellierung ansetzen könnte.[13]

Beim Fortgang der Modellierung, die sich an die Ausgangsmodelle knüpfte, machten beide Arbeitsgruppen vergleichbare Beobachtungen. A1 beobachtete, daß ihr Ausgangsmodell zu wenig objektorientiert konzipiert war, daß es noch zu stark von einer prozeduralen Sicht geprägt war, um ohne weiteres in ein Modell mit Smalltalk80 transformiert werden zu können. Und A2 stellte fest, daß sie ihr Ausgangsmodell nicht ohne weiteres in ein BACK–Modell umsetzten konnte.[14]

Beide Arbeitsgruppen sahen die Notwendigkeit, *Zwischenmodelle* zu konstruieren, wobei A2 feststellte, daß die Konstruktion eines Zwischenmodells öfter zu Revisionen des Ausgangsmodells führt. Die Arbeitsgruppe beobachtete eine deutliche Wechselwirkung zwischen den Entwicklungen der verschiedenen Modelle.

Die Erfahrungen der Arbeitsgruppen an den konkreten Modellierungsaufgaben ermöglichen uns im Rückgriff auf die beiden Modellierungsperspektiven eine sehr pointierte Deutung. Das, was die Arbeitsgruppen als Modellierungsschwierigkeiten identifiziert haben, muß tatsächlich in der Abbildperspektive nicht aber in der Konstruktionsperspektive als Problem betrachtet werden. So entstehen in der Abbildperspektive enorme Schwierigkeiten, wenn es kein Original (Ausgangsmodell) mit bestimmten Eigenschaften gibt, so daß es die Unabhängigkeits- und Vergleichbarkeitsannahme erfüllt. In der Konstruktionsperspektive läßt sich dies dagegen nicht als ein Problem betrachten, das die Modellierung erschwert beziehungsweise ihr im Wege steht, da hier nicht von einem gegebenen Original ausgegangen wird, sondern die Konstruktion eines Ausgangsmodells (Originals) bereits wesentlicher Bestandteil der Modellierung ist.

Und die beobachteten Wechselwirkungen unter verschiedenen Modellen, die sich in Modell–Original–Beziehungen befinden, sind in der Konstruktionsperspektive ein wesentlicher Modellierungsvorgang, während solche Wechselwirkungen in der Abbildperspektive nicht gesehen werden beziehungsweise als äußerst problematische Manipulation am Original während der Modellierung bewertet werden müßten.

Exemplaren vornahmen. Diese Unterteilung ist ein rein durch unsere Programmiererfahrung entstandenes Konstrukt ...“

[13] Hier heißt es im Projektbericht: „Neben den Konsistenzproblemen waren auch noch Kommunikationsprobleme auffällig, die unter anderem durch die unklare Vorstellung des Zielmodells entstanden sind. Es wurde ständig zwischen verschiedenen Abstraktionsebenen, einer Bibliothek ganz allgemein und den für unsere Aufgaben relevanten Prozessen innerhalb einer Bibliothek, gewechselt.“

[14] In den Protokollen von A2 steht hierzu: „In der anschließenden Modellierungsphase hatten wir zunächst versucht, unsere Theorie [Ausgangsmodell/Original] direkt in BACK umzusetzen, was allerdings gescheitert ist, da die Kluft zwischen unserer Theorie und den Repräsentationskonzepten von BACK zu groß war.“

Modellierungskriterien in der Informatik

Vom Modell unterscheiden wir die Modellierung als denjenigen Prozeß, durch den ein Modell hergestellt wird. In das nähere Umfeld von Modell und Modellierung gehören unserer Ansicht nach Modellierungsmethoden und -werkzeuge, da die Methoden und Werkzeuge den Modellierungsprozeß ganz erheblich unterstützen. Um das Werkzeug einigermaßen sicher von der Methode zu unterscheiden, gehen wir davon aus, daß ein Werkzeug im Rahmen einer Methode benutzt wird (so läßt sich auch von einer spezifischeren Methode als von einem Werkzeug sprechen, die im Rahmen einer allgemeineren Methode verwendet wird).

Von der Modellierungs*methode* ließe sich noch das Modellierungs*konzept* unterscheiden. Der Unterschied beruht darauf, daß zwar ein Konzept die Modellierung wie auch eine Methode die Modellierung dirigiert, aber das Konzept gegenüber der Methode weniger spezifisch artikuliert ist.

Um Modellierung mit Hilfe des Perspektivenkonzepts sowohl auf einer analytisch–diagnostischen als auch einer praktischen und orientierenden Ebene zu klassifizieren, benötigen wir eine Strategie. Eine sinnvolle Strategie beruht auf den Modellierungskriterien, denn es zeigt sich, daß die Aufgabe, eine Analyse und Klassifikation der Modellierungskonzepte, Methoden und Werkzeuge durchzuführen, nur zu bewältigen ist, wenn der Umgang mit den Kriterien besser beherrscht wird. Gerade in dem Maße, in dem die Kriterien die Modellierungen leiten, sind sie es auch, die diese einer Klassifikation zugänglich machen. Dieser Sachverhalt bestätigt den außerordentlichen Stellenwert, den die Kriterien im Zusammenhang mit den beiden Modellierungsperspektiven haben.[15]

Es findet sich eine lange Liste möglicher Kriterien: Korrektheit, Vollständigkeit, Adäquatheit, Verifizierbarkeit, Voraussagbarkeit, Änderbarkeit, Erweiterbarkeit, Fehlertoleranz, Portierbarkeit, Funktionalität, Formalisierbarkeit, Kommunizierbarkeit, automatische Transformierbarkeit, Konsistenz, Benutzerfreundlichkeit, Aufgabenangemessenheit (Angemessenheit), Kompetenzerweiterung, menschengerechte Systemgestaltung, Modularität, Sicherheit, Transparenz, Robustheit, Wiederverwendbarkeit, Sozialverträglichkeit, Effizienz, Nützlichkeit, Redundanzfreiheit, schnelles Erstellen von Software, Verständlichkeit, Wartbarkeit usw. Die Kriterien sind oft nicht explizit definiert, und selbst wenn wir für alle aufgeführten eine Definition hätten, dürfte es schwierig sein, einige klar gegeneinander abzugrenzen. Wie etwa unterscheidet sich Sozialverträglichkeit von menschengerechter Systemgestaltung oder Konsistenz von Korrektheit oder Transparenz von Verständlichkeit? Es erscheint sinnvoll, die Liste potentieller Kriterien bis auf ein paar Kriterien zusammenzustreichen, deren Intentionen klar genug dargestellt und die deutlich gegeneinander abgegrenzt sind: Korrektheit,

[15] Die Arbeiten im Studienprojekt *Modellierung* zeigen die Bedeutung der Kriterien. Vgl. hierzu den Projektbericht.

Vollständigkeit, Funktionalität, Benutzerfreundlichkeit, Kommunizierbarkeit und Angemessenheit.

Um die Kriterien der kleinen Liste den Perspektiven zuzuordnen, greifen wir auf die Vorüberlegungen zurück, wonach die Intention eines Kriteriums zu bestimmen ist, damit es über diese mit den Möglichkeiten einer Perspektive verknüpft werden kann.

Sehen wir uns die Intention der ersten drei Kriterien an, so ist klar, daß es für sie auf die Beziehung zwischen Modell und Original (auch ausgezeichnetem Modell) ankommt, die Modellierenden oder andere Beteiligte, die mit der Modellierung oder dem Modell in Zusammenhang stehen könnten, spielen keine Rolle. Die Korrektheit eines Modells hängt nicht von den Interessen des Konstrukteurs ab, sie ist eine Sache des Modells. Ein Modell ist vollständig oder unvollständig, ganz gleich wer es entwickelt hat. Da diese Kriterien also unter anderem darauf abzielen, unabhängig von Modellierenden zu gelten, sind sie der Abbildperspektive zuzuordnen.

Das vierte und fünfte Kriterium beziehen sich ausdrücklich auf Personen. Zum einen sind es die Benutzerinnen, die wir als am Modell oder auch an der Modellierung Beteiligte betrachten, und zum anderen die Gruppe derjenigen, die über das Modell beziehungsweise für die Modellierung kommunizieren. Zu dieser Gruppe können Entwickler, Auftraggeberinnen und Benutzer gehören. Für diese Kriterien kommt es nicht primär auf eine Modell-Original-Beziehung sondern auf die beteiligten Personen an. Wegen der Möglichkeiten, die die Konstruktionsperspektive bietet, müssen diese Kriterien ihr zugeordnet werden. So wie das Angemessenheitskriterium gewöhnlich intendiert ist, muß es ebenfalls dieser Perspektive zugeordnet werden. Es scheint ja unmittelbar von den Interessen, entweder der Entwicklerinnen, der Auftraggeber oder Benutzerinnen abzuhängen, was als angemessen oder unangemessen gilt.

Modellierung, Methoden und Werkzeuge

In dem Studienprojekt *Modellierung* wurden in mehreren Arbeitsgruppen Modellierungskonzepte, -methoden und -werkzeuge mit Hilfe der Modellierungskriterien klassifiziert. Einige Ergebnisse sollen hier kurz angeführt werden.

Die Analyse von Texten von Backus und Parnas ergab, daß das Modellierungsverständnis dieser Autoren im ganzen der Abbildperspektive zuzuordnen ist. Während ein Text von Naur ein Modellierungsverständnis erkennen läßt, das der Konstruktionsperspektive zugeordnet werden muß. So hebt Backus hervor, daß unter den Gesichtspunkten Korrektheit und Funktionalität, seine Methode der Funktionalen Programmierung gegenüber anderen Methoden große Vorteile habe. Naur im Kontrast dazu stellt gerade allgemein anerkannte Ansprüche, wie etwa den Anspruch der Korrektheit, in Frage und rückt die Beteiligten an einem Soft-

wareentwicklungsprozeß in den Mittelpunkt, indem er die wesentliche Rolle der Intuition für die Modellierung herausarbeitet. Dadurch werden in Naurs Modellierungsperspektive Kriterien wie Korrektheit, Vollständigkeit und Funktionalität relativiert, während andere Kriterien, die näher an den Intuitionen beziehungsweise den Interessen der Beteiligten liegen, wie etwa Kommunizierbarkeit, in den Vordergrund rücken.[16]

Bei dem Versuch, Modelle der Softwareentwicklung wie etwa das Phasenmodell, das objektorientierte Modell oder das zyklische PETS–Modell zu klassifizieren, zeigt sich, daß der Unterschied zwischen Abbild- und Konstruktionsperspektive im Software Engineering sehr deutlich an der Rolle des Menschen, etwa der Systembenutzer oder auch anderer, die irgendwie am System beteiligt sind, zu erkennen ist. Ein Modellierungsansatz wie PETS, der die beteiligten Personen in den Vordergrund rückt und nicht ein modelliertes System, ein Modellierungsprodukt, ein Modell, wird der Konstruktionsperspektive zugeordnet. Während das Phasenmodell, jedenfalls in seiner idealen, konzeptionellen Gestalt, nur in der Abbildperspektive eine erfolgreiche Modellierung garantiert. Nur dann, wenn die Modelle einer abgeschlossenen Phase die Eigenschaften eines Originals haben, die in der Abbildperspektive verlangt sind, kann in der nächsten Phase modelliert werden, ohne daß hierdurch die abgeschlossene Phase in Frage gestellt werden müßte.[17]

Im Falle von Backus läßt sich behaupten, daß seine Perspektive, seine Methode und das von ihm vorgestellte Werkzeug sehr schön ineinander greifen.[18] Wie steht es mit der These, daß die Kombination von Perspektive, Methoden und Werkzeugen in der Regel derart konsequent sei? Die Erörterungen *Zur Geschichte der Requirement Definition Languages* läßt erkennen, daß es offenbar ohne weiteres möglich ist, etwa eine Methode, die in der Konstruktionsperspektive entwickelt wurde, für eine Modellierung in der Abbildperspektive zu verwenden. Oder es kann ein Werkzeug, das zur Unterstützung einer Modellkonstruktion in der Abbildperspektive entwickelt worden ist, auch in der Konstruktionsperspektive gebraucht werden. Letzten Endes determiniert nicht die Methode oder das Werkzeug die Modellierungsperspektive, sondern das subjektive Modellierungsverständnis aller an der Modellierung Beteiligten.[19] Wir ziehen hieraus den Schluß, daß es eine Aufgabe ist, die einigen Aufwand erfordert, Modellierungsperspektive, Methoden und Werkzeuge immer konsequent aufeinander abzustimmen.

Beim *sloppy modeling*, einem Konzept bzw. einer Methode zur Gestaltung wissensbasierter Systeme, wird davon ausgegangen, daß die Un-

[16] Vgl. hierzu das Kapitel *Grundkonzepte der Softwaremodellierung* in diesem Report.

[17] Vgl. hierzu das Kapitel *Partizipative Entwicklung von Softwaresystemen - PETS* in diesem Report.

[18] Vgl. das Kapitel *Grundkonzepte der Softwaremodellierung* in diesem Report.

[19] Vgl. hierzu das Kapitel *Requirement Definition Languages* in diesem Report.

abhängigkeitsannahme nicht erfüllt ist. Es gibt beispielsweise nicht ein Original der Wissensdomäne im Kopf der Expertin, das im Datenmodell abzubilden wäre. Hier wird von der Wechselwirkung unter verschiedenen Modellen, etwa dem der Wisseningenieurin und dem des Experten im Anwendungsbereich, ausgegangen. Dies ist die Wechselwirkung der Konstruktionsperspektive, durch welche sich Modell und Original gegenseitig konstituieren. Das heißt es sind Modell und Original revidierbar. Diese Auffassung verträgt sich zum Beispiel nicht mit dem Modellierungskonzept der schrittweisen Verfeinerung, die in einer Richtung verfolgt wird und rückwirkende Veränderungen nicht zuläßt. Da es keine unantastbaren Komponenten bei der Systementwicklung gibt (Ausgangsmodelle oder Originale), und sämtliche Komponenten zu irgendeinem Zeitpunkt neu konstruiert werden könnten, muß ein Werkzeug zur Methode sloppy modeling vor allem der Forderung nach Transparenz aller Daten und ihrer Entstehungsgeschichte entsprechen.[20]

Eine Analyse von *KADS,* wobei es sich ebenfalls um eine Methode zur Modellierung von wissensbasierten Systemen handelt, legt nahe, daß solche Modellierungen in der Konstruktionsperspektive stattfinden, da es kein gegebenes Original oder etwa im Vergleich mit anderen Modellierungssituationen beim Software Engineering keine präzise Spezifikation gibt, zu der ein Modell entwickelt werden könnte. Dieser Sachverhalt wirkt sich auch auf die Kriterienfrage aus. Das heißt, es lassen sich die Kriterien der Abbildperspektive gar nicht befriedigen, daher „muß auf Grund der fehlenden Spezifikation und dem Wunsch nach Kommunizierbarkeit des Modells der Anspruch auf Korrektheit und Vollständigkeit durch mehr oder weniger vage Minimalanforderungen ersetzt werden.“[21]

Sämtliche Analysen liefern Argumente und Belege dafür, daß die Abbildperspektive in der Informatik dominiert. Was aber folgt aus dieser Feststellung? Sie scheint sehr akademisch, wenn an die Differenz der beiden Perspektiven keine praktischen Folgen für die Modellierungen geknüpft sind.

1.3 Die Folgen der Abbild- und Konstruktionsperspektive für die Informatik

Praktische Relevanz der Konstruktionsperspektive

Daß wir in der Abbildperspektive modellieren, ist nicht sehr schwer nachzuweisen. Daß es theoretische Gründe gibt, der Abbild- eine Konstruktionsperspektive entgegenzusetzen ist im wissenschaftlichen Diskurs wiederholt vorgetragen worden. Es ist neu, die Differenz dieser beiden Perspektiven für die Informa-

[20] Vgl. hierzu das Kapitel *Sloppy Modeling* in diesem Report.
[21] Vgl. das Kapitel *Wissensmodellierung mit KADS* in diesem Report.

tik fruchtbar zu machen. Dafür aber ist die Voraussetzung, daß die Konstruktionsperspektive eine mit der Abbildperspektive vergleichbare Praxisrelevanz besitzt. Auf theoretischer Ebene lassen sich keine überzeugenden Gründe finden, die der Abbildperspektive eine größere Praxisrelevanz als der Konstruktionsperspektive bescheinigen würden. Offenbar ist der eigentliche Gesichtspunkt, unter dem die Abbildperspektive ihrer Konkurrentin in praxi einiges voraus hat, ihre enorm verbreitete Anwendung und eine dementsprechende Erfolgs- beziehungsweise Bestätigungsquote. Diese Verbreitung ist nicht sehr überraschend, da sie der Wissenschaftstradition zu verdanken sein dürfte, besonders in der Mathematik und den Naturwissenschaften, der sich die Informatik im allgemeinen verpflichtet fühlt.

Gibt es Informatikprojekte, die große Probleme aufgeworfen haben oder sogar gescheitert sind, bei denen sich die Probleme auf hartnäckige Modellierungen in der Abbildperspektive zurückführen lassen, sobald man einen Perspektivwechsel in die Konstruktionsperspektive vorgenommen hat?

Das Projekt der symbolischen KI in seiner ursprünglichen kompromißlosen Version kann als gescheitert betrachtet werden. In dem Konflikt zwischen dem Konnektionismus und der Symbolverarbeitung war ein wichtiges Argument der Konnektionisten, daß sie sich bei ihrem Projekt am Vorbild des realen menschlichen Gehirns orientieren würden. Ein ähnliches Argument wurde auch von den Symbolisten vorgetragen, denn sie, so ihr Argument, würden sich an den realen geistigen Prozessen des Menschen orientieren. Sie betrachteten die geistigen Prozesse, die sie für Symbolverarbeitungsprozesse hielten, und die Konnektionisten betrachteten die neurophysiologischen Prozesse. Das Argument der Konnektionisten kann wie folgt weiter ausformuliert werden: Die Symbolisten haben ein Modell der menschlichen Intelligenz konstruiert, das die Wirklichkeit nicht zutreffend abbildet. Sie haben ihre Methode und die Strukturen und Eigenschaften ihrer Werkzeuge auf die Wirklichkeit übertragen und zugleich unterstellt, daß dies objektiv den geistigen Vorgängen im menschlichen Gehirn entspräche. Wenn diese KI-Modellierungen unspektakulär wie andere auch in der Abbildperspektive erfolgten, dann ist man also bei diesen Modellierungen davon ausgegangen, daß unabhängig von ihren Konstrukten menschliche Intelligenz auf symbolverarbeitende Prozesse zurückzuführen ist. Indem diese Konzeption der “intelligenten Wirklichkeit“ fixiert wurde, waren den Modellierungen klare Grenzen gesetzt. Entscheidend ist hierbei, daß diese Grenzen nie zur Disposition standen, da es sich um Bedingungen handelt, die mit dem Original gegeben sind. Heute läßt sich sagen, daß das ursprüngliche symbolistische KI-Projekt an der Unterstellung dieses Originals oder Ausgangsmodells für die menschliche Intelligenz scheiterte. Wenn sich das KI-Projekt in dieser Art entwickeln mußte, bis es an dem Status des Originals scheitert, den ihm die Abbildperspektive zuschreibt, dann hätte es sich natürlich anders entwickeln können im Rahmen der Konstruktionsperspektive, in der alles, was Bestandteil der Modellierung ist, zur Disposition steht. Diese

Einschätzung motiviert die These, daß das Projekt erfolgreicher in der Konstruktionsperspektive verlaufen wäre, da man an entscheidender Stelle, beim Status des Originals, beweglicher gewesen wäre.

Erklärungen und Lösungen durch das Konzept der Modellierungsperspektiven

Andere größere Probleme bei der Softwareentwicklung führten dazu, daß von der Softwarekrise gesprochen wurde. Diese Krise läßt sich anhand einiger Konfliktlinien nachzeichnen. Eine solche Konfliktlinie markieren etwa die Autoren Backus, Parnas und Naur. Wir haben die Möglichkeit, diesen Konflikt als einen Konflikt zwischen Abbild- und Konstruktionsperspektive zu erklären. Die Autoren Backus und Parnas versuchen die Softwareprobleme konsequent im Rahmen der Abbildperspektive und ihrer Bedingungen zu lösen, während Naur einen Ansatz entwickelt, der für Modellierung, für die Softwareentwicklung, in der Konstruktionsperspektive eintritt. Wenn das Konzept unserer Modellierungsperspektiven eine Erklärung für den Konflikt liefert, dann interessiert uns vor allem, ob diese Erklärung auch Lösungen erkennbar macht.

Wir vermuten, daß das Konzept der beiden Perspektiven den Konflikt nicht nur erklärt, sondern auch einen Lösungsansatz nahelegt. Das Entscheidende für einen Lösungsansatz ist, das Verhältnis der beiden Perspektiven zueinander zu bestimmen. Offenbar gibt es unterschiedliche Modellierungssituationen. Während der Softwareentwicklung gibt es Modellierungsphasen, in denen in der Abbildperspektive modelliert werden sollte, da die Grundannahmen erfüllt sind, und es gibt Phasen, in denen nach der Konstruktionsperspektive modelliert werden sollte, da die Grundannahmen nicht erfüllt sind und auch nicht nachträglich erfüllt werden können, ohne daß Probleme auftauchen, die die Softwarekrise bestätigen würden. Wenn der Fehler darin liegt, zu versuchen, sämtliche Softwareprobleme entweder konsequent in der Abbild- oder der Konstruktionsperspektive zu lösen, dann ließe er sich beheben, indem zwischen beiden Perspektiven je nach Modellierungssituation abgewechselt würde. Ein Lösungsansatz würde darin bestehen, verschiedene Modellierungssituationen über das Konzept der beiden Modellierungsperspektiven präzise zu unterscheiden und abhängig von einer gegebenen Situation die entsprechende Perspektive zu wählen.

In ähnlicher Weise können wir versuchen, den Streit zum Selbstverständnis der Informatik als Wissenschaft zu erklären, der darüber entbrannt ist, ob es sich um eine reine Ingenieurwissenschaft oder zum Teil auch eine Sozialwissenschaft handelt. Sind Fragen der Ergonomie oder sozialer Implikationen solche, die an die Informatik zwar heranzutragen aber keinesfalls ihr direkt zuzurechnen sind? Wenn es gemäß der Definition der Konstruktionsperspektive Modellierungssituationen gibt, die diese Perspektive erfordern, dann können eine Reihe von Aspek-

ten, die in der Abbildperspektive überhaupt keine Rolle, aber in solchen Modellierungssituationen eine Rolle spielen, unmittelbar in der Modellierung in der Konstruktionsperspektive berücksichtigt werden. In dieser Perspektive spielen bei der Modellierung Komponenten eine Rolle, die nicht zur Informatik gerechnet werden, wenn diese lediglich Modellierung in der Abbildperspektive umfaßt.

Neues Modellierungskriterium ?

Als bedeutende Konsequenz der Modellierung in der Konstruktionsperspektive schält sich heraus, daß bei der Modellierung die Interessen der Modellierenden beziehungsweise aller an der Modellierung in irgendeinem Sinne Beteiligten wichtig sind. Diese Interessen, die in Bezug auf das Modell, das Original und die Modell–Original–Beziehung existieren, müssen explizit gemacht werden. Natürlich führt die Umsetzung dieser Forderung zu erheblich komplexeren Modellierungsprozessen. Aber gerade hierdurch sollten sich teure Modellierungssackgassen vermeiden lassen. Es ist zu prüfen, ob ein Kriterium zu bestimmen wäre, dessen Anwendung in der Konstruktionsperspektive die explizite Einbindung der Interessen sicherstellt. Vielleicht erlaubt ein solches Kriterium auch die sichere Diagnose derjenigen Interessenlage, die der Modellierung in der Abbildperspektive typischerweise zugrunde liegt. Die Diagnose der Interessen zusammen mit der Analyse der Modellierungssituation entscheidet letztlich über die geeignete Modellierungsperspektive.

Zur allgemeinen Anwendung des Konzepts der Modellierungsperspektiven

Aufgrund der Defizite, die die Modellierungen in der vorherrschenden Abbildperspektive aufweisen, haben wir unser Perspektivenkonzept in der Begrifflichkeit Abbild- und Konstruktionsperspektive entwickelt. Unter Umständen ist es möglich, die Terminologie zu neutralisieren, um eine Skizze schlicht für die „Modellierung in der Informatik“ zu beschreiben: 1. Modellieren ist ein Herstellungsbeziehungsweise Konstruktionsprozeß, durch den, indem verschiedene Zwecke verfolgt werden, Modelle entstehen. Den Zwecken unterliegen die Modelle, die Originale und die Modell–Original–Beziehungen. Sämtliche Zwecke sind in den Interessen der Modellierenden und aller an der Modellierung in irgendeinem Sinne Beteiligten verankert. 2. Verschiedene Modellierungen oder auch Modellierungssituationen gehen auf verschiedene Zwecksetzungen zurück. Letztlich entscheiden also die Interessen über die Definition einer Modellierungssituation. 3. Von der Definition der Modellierungssituation hängt es ab, welche Methoden und Werkzeuge die Modellierung lenken beziehungsweise unterstützen können. Zu beachten ist, daß dieser Skizze die erweiterte Explikation des Modellbegriffs zu-

grundeliegt, die eventuell als einen sehr geläufigen Sonderfall die minimale Explikation einschließt.

Teil II

Konstruktions- und Abbildperspektive

Eine kurze Vorbemerkung zu Teil II

In diesem Teil werden philosophische und sozialwissenschaftliche Texte in Hinblick auf die Gesamtfragestellung diskutiert.

Leser/innen, denen die Beschäftigung mit solchen Grundlagen in der Informatik eher überflüssig erscheint, mögen diesen Teil (vorerst) überblättern, und gleich mit Teil III fortfahren.

Wir haben die Autoren mit dem Ziel ausgewählt, einerseits die im Teil I eingeführten Modellierungsperspektiven zu erläutern und in einen geistesgeschichtlichen Rahmen einzuordnen. Zum zweiten vertreten die Autoren auch je eigene Positionen im Spannungsfeld von Abbild- und Konstruktionsperspektive, von denen aus wir jeweils versuchen, Brücken zu Fragestellungen der Informatik zu schlagen, was als Motivation oder Einstieg für die im Teil III gestellten Fragen nützlich sein mag.

Francis Bacon kann als klassischer Vertreter einer Abbildperspektive gelten. Gleichzeitigt verknüpft er diese Abbildsicht mit Grundannahmen über wissenschaftliche Methoden und über die Rolle der Wissenschaft, die wir aus heutiger Sicht am ehesten den traditionellen Ingenieurswissenschaften, und die Informatik versteht sich selbst weitgehend als eine solche, zuordnen können.

Alfred Tarki liefert eine in der Informatik, und zwar nicht nur für die formale Semantik, einflußreiche Variante einer Abbild- oder Korrespondenztheorie der Wahrheit.

Nelson Goodman vertritt eine konstruktivistischen Position. Insbesondere thematisiert er die Rolle von Sprachen oder genauer Aufschreibsystemen für die Konstruktion von Wirklichkeit. Seine These, nach der die Wahl der Modellierungswerkzeuge (z.B. formale Sprachen) die Art der Wirklichkeit, die damit konstruiert wird, wesentlich mit prägt, wird zwar in Teil III nicht explizit aufgenommen, sie bildet aber eine wichtige Hintergrundannahme unserer Arbeit.

Peter Berger und Thomas Luckmann beschreiben, wie Wirklichkeit sozial konstruiert und gleichzeitig als wahr und gegeben empfunden werden kann. In Bezug auf die Informatik ist das in zweierlei Hinsicht relevant. Zum einen ist die Herstellung informationstechnischer Artefakte selbst ein sozialer Prozess, in dem Wirklichkeit konstruiert (im Sinne von interpretiert und im Sinne von technisch hergestellt) wird. Zum zweiten können informationstechnische Artefakte, die von Menschen genutzt werden, selbst als (virtuelle) quasi-soziale Wirklichkeiten angesehen werden, und sie wirken möglicherweise auch als solche.

Kapitel 2

Wissenschaft als Abbildung der Realität: Francis Bacon
Autor : Martin Fischer

Francis Bacon (1561 - 1626) entwickelt in scharfer Abgrenzung von der scholastisch, aristotelischen Tradition und ohne auf bedeutende wissenschaftliche Erfindungen seiner Zeit Bezug zu nehmen[1] neue Grundlagen für die empirischen Wissenschaften. Das hier behandelte Werk: 'Novum Organum' (Neues Organon)[2] von 1620 ist der methodische Teil eines von Bacon geplanten aber unvollendeten Werkes zur vollständigen Erneuerung der Wissenschaften: 'Instauration Magno'.

Im folgenden will ich Bacons Konzeption von Wissenschaft, insbesondere sein Modell- und sein Methodenverständnis behandeln und im zweiten Abschnitt diese Aspekte in ihrem Bezug zur Informatik diskutieren.

2.1 Bacons Konzeption von empirischer Wissenschaft

Bacon beansprucht, eine "wahre Anleitung zur Interpretation der Natur", so der Untertitel des Neuen Organon, zu liefern. Er hat dabei keineswegs ein naives Verständnis bezüglich der Erkennbarkeit der Realität, sondern er diskutiert ausführlich die Einflüsse und Einschränkungen, er nennt sie Idole[3], denen menschliche Wahrnehmung und Erkenntnis unterliegt.

[1] Galilei 1564 - 1642, Kepler 1571 - 1630

[2] Soweit nicht anders angegeben beziehen sich die Referenzen in diesem Kapitel auf das 'Neue Organon', [Bacon 1620]. Die Zitierung erfolgt nach Buch (I oder II) und Nummer des Aphorismus.

[3] Idole bedeutet eigentlich Gespenster.

"Vier Arten von solchen Idolen halten den menschlichen Geist gefangen. (…) die erste Art soll als Idol des Stammes bezeichnet werden; die zweite als Idol der Höhle; die dritte als Idol des Marktes; die vierte als Idol des Theaters." (I 39) Als Idole des Stammes bezeichnet Bacon angeborene allgemein menschliche Eigenschaften. Dazu gehören etwa die Einschränkungen, denen die menschliche sinnliche Wahrnehmung unterliegt. Idole der Höhle sind individuelle Eigenarten von Menschen, die sowohl angeboren als auch erworben sein können. Idole des Marktes nennt Bacon kulturelle und soziale Grundansichten und -annahmen, die die Erkenntnis beinflussen. Er nennt hier zum Beispiel die Sprache, die durch ihre Begrifflichkeit eine Sicht auf die Welt vorgibt. Idole des Theaters sind schließlich solche, die sich in expliziten Theorien über die Welt äußern und tradieren. Hierunter fallen für Bacon vor allem die Lehrmeinungen und Dispute der von ihm scharf kritisierten scholatischen Philosophie, die den unvoreingenommenen Blick auf die Realität verstellen. (vgl. I 41-44)

So hellsichtig Bacon gegenüber den Einflüssen ist, denen menschliche Erkenntnis unterliegt, so überzeugt ist er gleichzeitig, daß diese Einschränkungen überwunden werden können und wahre Erkenntnis mit Hilfe wissenschaftlicher Methoden möglich ist: "Die Aufstellung der Begriffe und Sätze durch wahre Induktion ist gewiß das geeignete Heilmittel, die Idole abzuhalten und zu eliminieren;" (I 40)

Bacons Modellverständnis

Zur Klärung von Bacons Modellverständnis will ich die drei für den Modellbegriff[4] relevanten Aspekte: Status des Originals, Status des Modells und die Beziehung zwischen Original und Modell diskutieren.

Nach Bacon ist die Natur (von Gott) unabhängig vom Menschen gegeben. (II 15) Sie besteht aus kleinsten unabhängigen Körpern, die elementare Eigenschaften haben. Der Zusammenhang bzw. das Zusammenspiel der Körper und Eigenschaften folgt Gesetzen. (vgl. z.B. II 2, II 17) Die Welt oder Teile der Welt haben den Status des Originals, das vorgegeben und auf bestimmte Weise strukturiert ist.

Wissenschaftliche Erkenntnis liefert 'wahre' Modelle ('verum exemplar mundi' I 124), die analog zur Realität strukturiert sind. Modelle sollen schriftlich fixierbar sein (I 101). Bacon selbst bevorzugt halbformale Modellierungswerkzeuge, insbesondere Tabellen (I 102), und sieht mathematische Modelle als Idealfall von Genauigkeit und Klarheit an (II 8).

Die Beziehung zwischen Modell und Original ist für Bacon eine Abbildung. "Denn ich lege im menschlichen Geist den Grund zu einem Bild der Welt, wie

[4] siehe Teil I dieses Berichtes

sie vorgefunden wird und nicht wie sie die eigene Überlegung einem diktiert hat." (I 124) Er benutzt meist den Terminus 'exemplari' (I 120, I 124), was Abschrift, Abbild bedeutet, und auch 'imago' (I 120), also Bild, und 'modulus' (I 124).

Im Idealfall ist die Modell–Original–Relation bei Bacon nicht nur eine korrekte, sondern auch eine vollständige Abbildung. "Denn was würdig ist zu existieren, das ist auch wert erkannt zu werden, denn das Wissen ist das Abbild des Seins." (I 120) Er hält also sowohl eine Erkenntnis elementarer Entitäten, also eine partiell isomorphe Abbildung, als auch eine umfassende Erkenntnis der Natur, also eine totale Abbildung zwischen dem Original Welt und dem Erkenntnismodell, für möglich.

Bacons Methodenverständnis

Ich will hier nicht Bacons Methode im einzelnen beschreiben, sondern seine Beschreibung von und seine Anforderungen an methodisches Vorgehen darstellen.

Bacon sieht in Methoden Werkzeuge für die menschliche Erkenntnis. Wie es Werkzeugen, etwa Hebeln und Kränen, bedarf um einen Obelisken aufzustellen, so bedarf die menschliche Erkenntnisfähigkeit Methoden, um die Ergebnisse ihrer Tätigkeit zu verbessern. (vgl. Vorrede)

> "Weder die bloße Hand noch der sich selbst überlassene Verstand vermögen Nennenswertes; durch unterstützende Werkzeuge wird die Sache vollendet; man bedarf ihrer nicht weniger für den Verstand als für die Hand. Und so wie die Werkzeuge die Bewegung der Hand wecken und lenken, so stützen und schützen in gleicher Weise die Werkzeuge des Geistes die Einsicht." (I 2)

Ebenso wie bei der Aufstellung eines Obelisken die Kraft eines einzelnen Menschen hinter einem systematischen werkzeuggestützten Vorgehen zurücksteht, ist im Rahmen der Erkenntnis der Welt und beim Erfinden neuer Artefakte die Genialität des einzelnen Denkers einem methodengestützten Vorgehen des durchschnittlichen Menschen weit unterlegen. (I 122)

Bacon fordert erstens ein systematisches Befragen der Natur, was sich etwa in 'vollständigen' Fragekatalogen und Tabellen äußert (II 11-13), und zweitens eine systematische Entwicklung von Experimenten zur Überprüfung von Theorien. (I 99) Er verfolgt dabei eine 'bottom up' Methode, d.h. er schreitet von der Erkenntnis der Einzeldinge und ihrer Eigenschaften zu Zusammenhängen und Gesetzen fort. (z.B. I 36)

Die richtige Erkenntnis der Natur und ihrer Gesetze führt dann, sofern man systematisch die Einzelteile wieder verknüpft, quasi automatisch nicht nur zu Experimenten, sondern auch zu neuen nützlichen Erfindungen; "was bei der Betrachtung als Ursache erfaßt wird, dient bei der Ausführung als Regel." (I 3)

Zusammenfassend ergibt sich folgendes Bild der Baconschen Methodik zur Er-

forschung von Natur und technischen Erfindungen:

> "Größeres hingegen ist von dem neuen Licht der aus den Einzeldingen nach festen Methoden und Regeln angeleiteten Kernsätzen zu erhoffen, welche wieder auf neues Einzelne hinführen und es bezeichen. Denn der Weg geht nicht in einer Ebene, sondern es geht bergauf und bergab, zunächst bergauf zu den Grundsätzen , bergab dann zu den Werken." (I 103)[5]

2.2 Bezug zur Informatik

Was ist eigentlich ingenieursmäßig?

Der Begriff 'ingenieursmäßiges Vorgehen' taucht in der Informatik an vielen Stellen auf. Er wird meist dazu verwendet, bestimmte Modellierungsmethoden zu begründen, und wird keineswegs einheitlich verwendet.[6] Bacon scheint mir hier eine recht gute Explikation dessen zu liefern, was mit dem Begriff gemeint ist.

Die antike Tradition (Platon, Aristoteles) hatte Kunst (das heißt hier vor allem Handwerk und Technik) und Wissenschaft (Philosophie) als Gegensatz gesehen. Die Begriffspaare: Handeln und Herstellen in der Ethik und wahr und nützlich in der Erkenntnistheorie kennzeichnen diesen Gegensatz, wobei Philosophie - ebenso wie das Handeln gegenüber dem Herstellen - deswegen übergeordnet ist, weil sich ihre Zwecke auf sich selbst richten, im Gegensatz zur Technik, die ihre Zwecke außerhalb ihrer selbst hat.

Bacon bricht mit dieser Tradition, indem er wissenschaftliche Erkenntnis in den Dienst des technischen und, was für ihn gleichbedeutend ist, sozialen Fortschritt stellt. "Das wahre und rechtmäßige Ziel der Wissenschaften ist kein anderes, als das menschliche Leben mit neuen Erfindungen und Mitteln zu bereichern." (I 81)[7] Damit kann er als theoretischer Begründer der Ingenierswissenschaften angesehen werden.

Bacon fordert ein streng methodisches Vorgehen, das unabhängig von der Genialität Einzelner ebenso neue Erkenntnisse und neue Erfindungen hervorbringt, wie deren Qualität sichert. Das kann aus meiner Sicht als gemeinsamer Nenner der verschiedenen Verwendungen von ingenieursmäßig hervorgehoben werden.

Er favorisiert weiterhin ein stark formalisiertes Vorgehen und eine möglichst for-

---

[5] vgl. auch (I 110) und zu Experimenten (I 99)

[6] Der Begriff wird z.B. von Backus und Parnas beansprucht. Vgl. Elis und Freitag in Teil 3 dieses Bandes.

[7] vgl. auch Instauration Magna: Einteilung des Werkes: "wenn die Lehrsätze einmal richtig entdeckt sind werden sie ganze Scharen von Werken mit sich führen (…)" [Bacon 1620, S. 55] "daß (…) Hilfe für den Menschen und ein Stamm von Erfindern hervorgehen mögen, welche die Not und das Elend der Menschen zumindest teilweise mildern und besiegen." [Bacon 1620, S. 51]

male schriftliche, im Idealfall mathematische Darstellung, was ebenfalls typisch für Methoden in der Informatik ist, die als ingenieursmäßig verteidigt werden.

Mit Bacons Annahmen über die Struktur der Realität stellt sich die Entwicklung (Konstruktion) neuer Artefakte mit wissenschaftlicher Methode wie folgt dar: Im ersten Schritt erfolgt eine methodengeleitete Erkenntnis der Entitäten in der Welt und der Gesetze ihres Zusammenspiels. Artefakte entstehen dann im zweiten Schritt durch eine ebenfalls methodengeleitete Neukombination von Elementen unter Berücksichtigung der Gesetzmäßigkeiten. Diese Sicht deckt sich mit dem (zumindest traditionellen) ingenieurswissenschaftlichen Selbstverständnis, nach dem sich technische Entwicklungen im Rahmen der Eigendynamik der Wissenschaft selbst vollzieht und weitgehend unabhängig von sozialen, ökonomischen und politischen Einflüssen erfolgt.

Für Bacons Position ist es weiterhin grundlegend, daß technischer Fortschritt die Unterwerfung der Natur zum Ziel hat[8], und daß das gleichbedeutend mit einer Verbesserung der Lebenssituation der Menschen ist. Hier kann man sich fragen, inwieweit sich Ingenierswissenschaften, zu denen sich auch die Informatik zählt, heutzutage von dieser Grundannahme der Kontrollierbarkeit natürlicher und sozialer Systeme wirklich verabschiedet haben.

Bacons Vorstellung der Realität

Bacon sieht die Welt als strukturiert an. Es gibt elementare Gegenstände, elementare Eigenschaften und gesetzmäßige Zusammenhänge. Damit ist die Struktur der Welt analog zu mathematischen Strukturen, genauer Algebren, aufgebaut. Die Realität ist dann mit logischen Sprachen vollständig beschreibbar. Das deckt sich mit der in der Informatik verbreiteten Sicht, mit Spezifikationssprachen, die eine algebraische Semantik haben[9], die Realität adäquat beschreiben zu können.

Auch im Bereich objektorientierter Programmiernug und Modellierung kann man ähnliche Annahmen über die Struktur der Welt finden.

Bacons Methode und Modellierung in der Informatik

Bacon verfolgt eine ‘bottom up’ Methode bei der Modellierung, d.h. bei ihm bei der Erstellung von wahren Modellen der Realität. Informatische Methoden, insbesondere funktionale Ansätze, bevorzugen dagegen meist schrittweise Verfeinerung, und legen damit eine ‘top down’ Sicht nahe.[10]

[8] vgl. Instauratio Magna: Einteilung des Werkes: [Bacon 1620, S. 41]

[9] vgl. auch das folgende Kapitel über Tarski

[10] Eine Ausnahme bildet hier K. Morik, die im Rahmen ihres Methodenpluralismus vorsieht, von Beispielen auszugehen und mit Methoden des maschinellen Lernens automatisch Zusammenhänge zu erkennen. (Vgl. Birgit Schelm in Teil 3 dieses Bandes.)

Eine weitere wichtige Forderung Bacons, die systematische Entwicklung von Experimenten zur Überprüfung von Ergebnissen, ist in der Informatik ebenfalls wenig entwickelt. Hier stellen eher konstruktivistische Ansätze Ausnahmen dar, wobei das Testen nicht der Überprüfung der Wahrheit (Vollständigkeit und Korrektheit), sondern der Angemessenheit des Modells dient.

Festgestellt werden kann allerdings die Analogie von Bacons Vorgehensweise mit traditionellen Phasenmodellen für die Softwareentwicklung bzw. Systemanalyse. Erster Schritt ist die Analyse der Realität, so wie sie ist, d.h. die Identifikation der kleinsten Elemente und ihrer Eigenschaften sowie das Auffinden von Regelhaftigkeiten oder Gesetzmäßigkeiten (Ist-Analyse). Die Lösung des Problems besteht dann aus der systematischen Rekombination dieser Elemente und ihre teilweise Ersetzung durch technische Artefakte. Im Phasenmodell wird allerdings das angestrebte Ziel der Entwicklung im Rahmen konkreter Anforderungen definiert (Soll-Konzept), während bei Bacon anscheinend nützliche Artefakte quasi automatisch entstehen.

Beiden Ansätzen gemeinsam ist die streng lineare Abfolge der Phasen und ein streng geregeltes, möglichst mechanisches Vorgehen. “So muß das Werk gleichsam wie durch eine Maschine vorangetrieben werden.” (Vorrede) Darin kommt die Überzeugung zum Ausdruck, daß durch eine möglichst formale Vorgehensweise sowohl innerhalb der einzelnen Phasen als auch bei den Übergängen die Qualität der Ergebnisse gesichert ist.

Zusammenfassung

Bacon ist für uns interessant als Vertreter der Abbildperspektive. Er sieht die Einflüsse auf die menschliche Erkenntnis durch sinnliche Beschränkungen, individuelle Besonderheiten, Kultur und Sprache, theoretische Weltmodelle sehr klar, hält sie aber für überwindbar.

Bei Bacon ist die Abbildperspektive sehr eng mit einer ingenieurswissenschaftlichen Sicht verknüpft. Ziel wissenschaftlicher Forschung sind letztlich Werkzeuge. Methodisches Vorgehen und nicht Intuition oder Genialität stellen den Erfolg sicher. Die technische Entwicklung entsteht aus der Eigendynamik der Forschung und ist per se der Verbesserung der menschlichen Situation dienlich.

Wie im folgenden[11] noch näher erläutert werden wird, ist das Zusammenfallen von Abbildperspektive und traditioneller ingenieurswissenschaftlicher Ausrichtung, wie sie Bacon in seinem Programm wissenschaftlicher Erkenntnis entwickelt, nach wie vor prägend für große Teile der Disziplin Informatik.

[11] siehe insbesonder Melahat Elis und Michael Freitag in Teil 3 dieses Bandes

Kapitel 3

Bedeutung und Metasprache: Alfred Tarski Autor: Martin Fischer

Wir hatten die Abbildperspektive so charakterisiert, daß von einer gegebenen Realität ausgegangen wird, die erkennbar ist. So zugespitzt wird dies jedoch von kaum jemandem ernsthaft vertreten. Deshalb soll hier eine Variante der abbildorientierten Sicht vorgestellt werden, die in der Disziplin Informatik sehr einflußreich ist.

3.1 Tarskis Semantikkonzeption

Alfred Tarski (1901 - 1983) gilt als einer der einflußreichsten Logiker dieses Jahrhunderts. Er lieferte wesentliche Beiträge zur Semantiktheorie formaler Sprachen. Ich beziehe mich hier im wesentlichen auf seine frühen Arbeiten: "Der Wahrheitsbegriff in den formalisierten Sprachen"[1] sowie "Die semantische Konzeption der Wahrheit und die Grundlagen der Semantik."[2].

Tarski selbst versteht seine Arbeit als einen Beitrag zur Korrespondenztheorie der Wahrheit.[3] "Die *Semantik* ist eine Disziplin, die sich - grob gesprochen - *mit bestimmten Beziehungen zwischen Ausdrücken einer Sprache und den Gegenständen (oder 'Sachverhalten') befaßt, auf die die Ausdrücke sich 'beziehen'.*"[4]

Er versucht den Begriff Wahrheit und andere semantische Begriffe für formali-

[1] [Tarski 1935]
[2] [Tarski 1944]
[3] vgl. [Tarski 1944] S. 143 und [Tarski 1935] S. 265
[4] [Tarski 1944] S. 146

sierte Sprachen im allgemeinen zu präzisieren und für spezielle Sprachen zu definieren. Eine Definition für die Umgangssprache hält er für aussichtslos, weil sie unpräzise und mehrdeutig ist und sich verändert.[5] Zudem, und das ist aus logischer Sicht problematischer, enthalten natürliche Sprachen semantische Begriffe (Wahrheit, Erfülltsein, Bezeichnen, Folgerung, etc.), die auf alle Ausdrücke der Sprache selbst angewendet werden dürfen. In solchen Sprachen sind aber Antinomien, wie die des Lügners formulierbar, die zu Widersprüchen führen.[6] Das ist vom logischen Standpunkt aus gesehen fatal, da in einer Theorie, die einen Widerspruch enthält, alle Sätze als wahr gefolgert werden können.

Zur Definition semantischer Begriffe ist es deshalb notwendig zwischen der Sprache, für die man diese Begriffe definiert, Tarski nennt sie Objektsprache, und der Sprache, in der man diese Begriffe definiert, der Metasprache, zu unterscheiden.

Die Objektsprache darf keine auf sie selbst anwendbaren semantischen Begriffe enthalten. Die Metasprache hingegen muß drei Arten von Termini enthalten: Allgemein logische Termini wie ‘dann und nur dann wenn’, die nötig sind, um überhaupt Definitionen aufzuschreiben; Namen für alle Ausdrücke der Objektsprache, um über diese Ausdrücke reden zu können; die Objektsprache selbst oder Übersetzungen für alle Ausdrücke der Objektsprache, um über Gegenstände oder Sachverhalte in der Welt reden zu können.[7]

Teildefinitionen des Wahrheitsbegriffs haben bei Tarski die Form:

(W) *X* ist wahr genau dann, wenn *p*

wobei *X* der Name des objektsprachlichen Satzes und *p* seine metasprachliche Übersetzung ist.[8] *p* wird dabei so verwendet (in *suppositio formalis*[9]), daß über das gesprochen wird , auf das sich der metasprachliche Ausdruck bezieht.[10]

Sätze, wie der oben genannte, erregen nach Tarski “bezüglich der Klarheit ihres Inhaltes und der Korrektheit ihrer Form im allgemeinen keinen Zweifel (freilich unter der Voraussetzung, dass die Aussagen, die wir (…) für das Symbol ‘p’ einsetzen, keinen derartigen Zweifel erregen)”[11]. Um das wiederum formal zu behandeln bedarf es einer Meta-Metasprache. “Auf diese Weise gelangen wir zu einer ganzen Hierarchie von Sprachen.”[12]

Tarski entgeht dem Problem einer unendlichen Hierarchie von Metasprachen, in-

[5] vgl. [Tarski 1935] S. 277

[6] vgl. [Tarski 1935] S. 270f und [Tarski 1944] S. 151

[7] vgl. [Tarski 1944] S. 153

[8] Die (W)-Sätze eignen sich nur für endliche Sprachen als Teildefinitionen, für unendliche Sprachen sollen aber für alle Sätze der Objektsprache Sätze der Form (W) aus einer adäquaten Definition des Wahrheitsbegriffs folgen. (vgl. [Tarski 1935] S. 305f)

[9] In *suppositio formalis* bedeutet, daß über das, worauf sich die Ausdrücke beziehen gesprochen oder geschrieben wird, im Gegensatz zu in *suppositio materialis*, wo über die Ausdrücke selbst (wie auf den linken Seiten der Äquivalenzen) geschrieben wird.

[10] vgl. [Tarski 1944] S. 144

[11] vgl. [Tarski 1935] S. 270

[12] [Tarski 1944] S. 152

dem er sie bei einer als verstanden geltenden Sprache abbricht. Als solch eine Sprache sieht er offenbar die Sprache der Mathematik, genauer die der Mengenlehre, an. Er führt also semantische Begriffe für Objektsprachen auf mengentheoretische Begriffe wie Individuen, Mengen, Relationen und Funktionen zurück. Den Bezug dieser metasprachlichen Entitäten zur Realität scheint Tarski dabei für unproblematisch zu halten.

Kritik der Tarskischen Position

Für die Definition der Semantik formaler Sprachen ist diese Hergehensweise unproblematisch, sinnvoll und nützlich. So kann etwa der semantische Folgerungsbegriff unabhängig von syntaktischen oder technisch realisierten Ableitungsregeln definiert werden.

Kritisieren kann man Tarski dagegen, insofern er sein Semantikkonzept auch für angemessen hält, um damit empirische Theorien oder formalisierte Fragmente der Umgangssprache zu behandeln. Sofern es sich um formale Ausdrücke handelt, die sich auf gegebene Realität beziehen, stellt sich hier die Frage, wie diese Beziehung genau zu verstehen ist.

Tarski gelingt es zwar, die Definition semantischer Begriffe für komplexe Ausdrücke auf einfache Ausdrücke zurückzuführen. Für einfache Ausdrücke kann er semantische Beziehungen aber lediglich definieren, aber nicht erklären. Darüber hinaus enthalten die Definitionen selbst immer nur (meta-)sprachliche Ausdrücke für Gegenstände, Eigenschaften etc. aber keine Gegenstände selbst.

Hier trifft Tarski die an jede Korrespondenztheorie anzulegende Kritik, das letztlich ungeklärt bleibt, wie sprachliche und nichtsprachliche Entitäten genau miteinander in Beziehung gesetzt werden können. Auch bei Tarski bleibt ungeklärt, wie sich metasprachliche Ausdrücke auf Gegenstände oder Sachverhalte in der Welt beziehen.

3.2 Tarski und informatische Semantik

In der Informatik wird das Tarskische Semantikkonzept (meist ohne auf Tarski bezug zu nehmen) verwendet, um die Semantik von Sprachen (Programmiersprachen, Spezifikationssprachen, Wissensrepräsentationssprachen etc.) zu definieren. In modelltheoretischen Semantiken sind die Modelle Teil metasprachlicher Ausdrücke.

Als Metasprachen werden dabei mathematische, insbesondere mengentheoretische, und logische Sprachen verwendet. Die Bedeutung von funktionalen Programmen wird als Funktion, die von (algebraischen) Spezifikationen als Algebra beschrieben, um nur zwei Bespiele zu nennen. Auch operationale Semantiken

werden in bezug auf einen formalen Ableitungsbegriff, der als Relation dargestellt wird, mathematisch beschrieben.

Auch in der Informatik wird das Tarskische Konzept sowohl für Sprachen im allgemeinen verwendet, d.h. um semantische Eigenschaften unabhängig von konkreten Implementierungen zu definieren, als auch um die Bedeutung konkreter formaler Ausdrücke, z.B. eines Programms, zu bestimmen.

Wie wir das bereits bei Tarski selbst gesehen hatten, wird dabei ein Programm eben nicht mit realen Gegenständen und Sachverhalten in Beziehung gesetzt, sondern mit metasprachlichen, mathematischen Ausdrücken. Korrektheit, als im Rahmen einer korrespondenztheoretischen Sicht wesentliche Kriterium für die Güte von Modellen, wird in der Regel als Beziehung zwischen Programmen und mathematischen Entitäten, z.B. Funktionen, verstanden. Der Bezug der mathematischen Ausdrücke, oder wenn man will auch mathematischen Gegenstände, zu dem was sie bezeichnen, wird üblicherweise nicht thematisiert und als unproblematisch vorausgesetzt.

Auch die Annahme abstrakter Informationen[13] als derjenigen Entitäten, auf die sich (formal-)sprachliche Ausdrücke beziehen, hilft hier nicht weiter, da Informationen in diesem Zusammenhang so gedacht werden, daß sie analog zu syntaktischen (oder mathematischen) Entitäten strukturiert sind. So wird lediglich noch eine (abstrakte) Zwischenschicht zwischen Objektsprache und Realität eingeführt, die aber deren Verhältnis auch nicht weiter klärt, es sei denn, man nimmt von vorn herein schon eine analoge Struktur von Syntax und Welt an.

Auch die gängige Sicht, Korrektheit als Beziehung zwischen Software und formaler Spezifikation zu verstehen, ist Tarskisch zu interpretieren. Software ist ein objektsprachlicher Ausdruck, die Spezifikation die metasprachliche Übersetzung davon. Korrektheit bedeutet hier, daß eine korrekte Übersetzung vorliegt. Die Semantik der Spezifikation kann dann wiederum in einer mathematischen Metamatasprache beschrieben werden, deren Realitätsbezug, wie oben bereits dargestellt, wiederum als unproblematisch gilt.

3.3 Semantik ohne Wirklichkeitsbezug

Tarski, ebenso wie seine Nachfolger in der Informatik, versteht seine Theorie zwar selbst als eine befriedigende Präzisierung der Abbildsicht, aber auch er löst das Problem der Korrespondenz zwischen Sprache und Realität nicht.

Löst man sich dagegen von dieser korrespondenztheoretischen Intention, so kann man das Tarskische Konzept auch so interpretieren, daß es gerade erlaubt, objektsprachlichen Ausdrücken Bedeutung in einem abstrakten, metasprachlichen,

[13] vgl. das nach wie vor einflußreiche Lehrbuch von Bauer und Goos: [Bauer und Goos 1982] S. 3 und S. 44f

mengentheoretischen Raum zuzuordnen, ohne auf eine Beziehung zu einer gegebenen Realität zu rekurrieren.

Faßt man Computer nicht allein als Rechenmaschine auf, sondern schließt auch Ein- und Ausgabegeräte mit ein, so können mit Computern solche fiktiven mathematischen Welten in die Handlungswelt von Computernutzern zurückprojeziert werden. Das erlaubt gerade die Loslösung von widerständiger Realität und die Konstruktion beliebiger Welten (innerhalb von Komplexitäts- und technischen Grenzen, denen insbesondere die Ein- und Ausgabemodalitäten unterliegen) mit Hilfe von Computern und Programmen.

Was bei virtuellen Realitäten evident erscheint, daß neue Realitäten, die Menschen erleben und in denen Menschen handeln, mittels Computerprogrammen geschaffen werden, gilt möglicherweise für informationstechnische Artefakte im allgemeinen. Zumindest werden bei Systemen, die Menschen benutzen, neue Gegenstände geschaffen, neue Regeln des Umgangs mit diesen Gegenständen definiert, neue Kommunikationsmöglichkeiten bereitgestellt, etc. Durch die Einführung informationstechnischer Artefakte werden Arbeits-, Kommunikations- und Handlungsmöglichkeiten zumindest mehr oder weniger stark verändert.

Zusammenfassung

Tarski formuliert seinem eigenen Verständnis nach eine Korrepondenztheorie für semantische Begriffe. In der Informatik wird seine Theorie mit derselben Grundintuition übernommen.

Tarski trennt in Objektsprache, für die semantische Begriffe definiert werden, und Metasprache, in der diese Begriffe definiert werden. Eine Korrespondenz zwischen Objekt- und Metasprache ist offensichtlich, unklar bleibt, wie die Beziehung zwischen Metasprache und Realität zustande kommt. Für mathematische und logische Sprachen scheint Tarski ebenso wie seine informatischen Nachfolger, diese Beziehung für unproblematisch zu halten.

Man kann Tarskis Semantik aber auch so interpretieren, daß sie es ermöglicht, beliebigen formalsprachlichen Ausdrücken in abstrakten mathematischen Welten Bedeutung zuzuordnen. Mit Computern wird es dann möglich, unabhängig von widerständiger Realität (virtuelle) Welten herzustellen.

Gerade das Zusammenspiel zwischen korrespondenztheoretischem Selbstverständnis und der (formal-)sprachlichen und technischen Möglichkeit, neue Realitäten zu konstruieren, macht möglicherweise den Witz des Tarskischen Einflusses innerhalb der Informatik aus, der aus meiner Sicht kaum unterschätzt werden kann.

Kapitel 4

Symbolische Konstruktion von Wirklichkeit: Nelson Goodman
Autor: Ingo Preik

4.1 Die Theorie der Notation

Goodman formuliert in [Goodman 1973][1] den Begriff des Notationssystems. Dieser Begriff soll die ihm wesentlichen Eigenschaften einer Partitur erfassen. Darunter versteht Goodman die eindeutige Erfassung einer Klasse von Aufführungen (ein Werk) durch die Partitur und umgekehrt die eindeutige Bestimmtheit einer Partitur durch eine Aufführung, d.h. von einer gegebenen Aufführung komme ich zu einer eindeutig bestimmten Partitur.

Um den Begriff des Notationssystems zu definieren, führt Goodman Symbolschemata und Symbolsysteme ein. Dabei besteht ein Symbolschema[2] aus konkreten Marken und aus einer Relation, der Zeichenindifferenz, zwischen ihnen. Ein Zeichen ist dann die Abstraktionsklasse (oder Äquivalenzklasse) von Marken bezüglich der Zeichenindifferenz. Eine Marke ist die Realisation eines Zeichens, wenn sie zu der Abstraktionsklasse gehört, die dieses Zeichen konstituiert. Ein Symbolsystem[3] besteht aus einem Symbolschema, das mit einem Bedeutungsfeld korreliert ist. Die so gegebene Relation zwischen Marken und Bedeutungsfeld ist die Denotation oder Kompatibilitätsbeziehung. Die Kompatibilitätsklasse ist die Abstraktionsklasse bezüglich dieser Relation.

Ein Symbolschema ist ein Notationsschema, wenn es die Bedingungen der syntaktischen Disjunktivität (keine Marke gehört zu zwei Zeichen) und der syntaktischen Differenziertheit (es gibt ein Verfahren um die Zugehörigkeit einer Marke

[1] siehe insbesondere Kap. IV
[2] vgl. [Goodman 1973] S. 138ff
[3] vgl. [Goodman 1973] S. 150ff

zu einem Zeichen festzustellen[4]) erfüllt. Ein Symbolsystem ist ein Notationssystem, wenn es die Bedingungen der semantischen Disjunktivität (jedes Element des Bedeutungsfeldes wird nur von einem Zeichen denotiert), der semantischen Differenziertheit (es gibt ein Verfahren um die Denotation eines Objektes durch ein Zeichens festzustellen[5]) und der Unzweideutigkeit (jede Marke eines Zeichens hat die gleiche Denotation) erfüllt. An dieser Stelle läßt sich überhaupt erst eine Relation zwischen Zeichen und Elementen von der zwischen Marken und Kompatiblen ableiten.

4.2 Die neugeschaffene Wirklichkeit

Goodman beabsichtigt eine Untersuchung der Repräsentation.[6] Dazu führt er die Denotation als Kern der Repräsentation ein und erklärt sie als allein von den bildlichen Eigenschaften eines Symbols abhängig. Jedoch sei die Repräsentation eines Objektes durch ein Bild in keiner Weise von einer Ähnlichkeit zwischen Bild und Objekt abhängig, Repräsentation sei vielmehr ein recht willkürlicher Zusammenhang („beinahe alles kann für alles andere stehen“[7]). Ein solcher Zusammenhang kann durch ein Repräsentationssystem, also ein Symbolsystem, definiert werden.

So ist auch die Frage, inwieweit ein Bild realistisch ist, nicht von irgendeiner Ähnlichkeitsbeziehung abhängig, sondern wird von Goodman an ein Repräsentationssystem geknüpft. Ein Bild ist demnach realistisch, wenn dessen Repräsentationsweise dem gewohnten Repräsentationssystem entspricht. Goodman lehnt andere auf objektiv erscheinenden Kriterien fußende Definitionen ab, so z.B. die Definition, nach der ein Bild realistisch ist, wenn es möglichst viele Informationen über das dargestellte Objekt enthält. Realismus hängt nach Goodman von Gewohnheit ab.[8]

Das Sehen von Objekten beinhaltet nach Goodman zugleich die Herstellung eben dieser Objekte, denn ein Objekt wird immer nur mit einer Auswahl seiner Eigenschaften betrachtet. Ein Bild greift nun bestimmte Eigenschaften und Beziehungen zwischen Objekten heraus und stellt sie irgendwie organisiert dar. Diese sich

[4] Genaugenommen definiert Goodman endliche Differenziertheit etwas schwächer:

> “For every two characters k and k’ and every mark m that does not actually belong to both, determination either that m does not belong to k or that m does not belong to k’ is theoretically possible.”

(Vgl. [Goodman 1973] S. 143. Ich zitiere hier das Original, weil die Übersetzung in der hier verwendeten Ausgabe nicht korrekt ist.)

[5] Hier gilt analoges wie in Fußnote 4 zur syntaktischen Differenziertheit bemerkt. (vgl. auch [Goodman 1973] S. 159)

[6] Hier beziehe ich mich im wesentlichen auf Kap. I von [Goodman 1973].

[7] [Goodman 1973] S. 17

[8] vgl. [Goodman 1973], insbesondere S. 44 - 50

aus dem Repräsentationssystem ergebende Organisation definiert nun erst das eigentliche Objekt. Das heißt, die Natur ist ein Produkt der Kunst und Sprache.

Werden neue Eigenschaften und Beziehungen durch Bilder manifestiert, so können diese mit der Zeit das Repräsentationssystem verändern und somit einen echten Beitrag zur Erkenntnis liefern.

Zusammenfassend läßt sich die Auffassung Goodmans folgendermaßen darstellen: Ein Bild ist keine Abbildung eines Objektes, sondern eine die Wirklichkeit selbst konstruierende Schöpfung. Die Vorstellung von dem Bild als Abbildung eines Objektes scheitert nach Goodman schon an der Unmöglichkeit darzulegen, was ein Objekt überhaupt ist.

4.3 Bezug zur Informatik

Es stellt sich nun die Frage, wie die Begriffe Goodmans auf die Informatik angewandt werden können.

Sicherlich ist in einem Modellierungsmittel (Methode oder Werkzeug) ein Notationsschema zu sehen. Die nötigen formalen Bedingungen sind erfüllt, denn jede Marke, in diesem Falle ein Pixelbild des Quellcodes auf einem Ausdruck oder ein binärer Code auf einem Datenträger, lassen sich eindeutig und praktisch nachvollziehbar einem Zeichen der Programmiersprache zuordnen.

Wird nun zu einem Modellierungsmittel ein geeignetes Bedeutungsfeld korreliert, so entsteht ein Notationssystem. (Diese Korrelation kann auch als Modell-Original-Beziehung interpretiert werden.) Unter einem Bedeutungsfeld läßt sich die zu modellierende Wirklichkeit verstehen. Allerdings läßt sich eine Programmiersprache als solche nicht mit einem Notationssystem vergleichen. Schließlich ist in einem Notationssystem die Repräsentation eines Objektes vorgegeben, eine Programmiersprache dagegen ist lediglich ein Notationsschema. Die Bedeutung eines Zeichens muß erst während der Modellbildung festgelegt werden. Jedoch wird den meisten Modellierungsmitteln eine Art Erklärung beigegeben, z.B. wird bei der objektorientierten Programmierung erklärt, was unter einem Objekt zu verstehen ist. Dieses Verstehen eines Modellierungsmittels ließe sich nun als Bezug zu einem Bedeutungsfeld deuten, so daß ein Modellierungsmittel zusammen mit dessen Verständnis als ein Notationssystem zu sehen wäre.

Legt also ein gegebenes Verständnis eines Modellierungsmittels die Repräsentation der Wirklichkeit doch fest, und erfüllt es die gleiche Funktion wie eine Partitur? Dazu dürfte die zu modellierende Welt innerhalb eines gegebenen Verständnisses nur auf eine Weise modelliert werden. Dies ist aber so nicht einsichtig. Daß verschiedene Personen unterschiedliche Modelle bilden, um eine gegebene Aufgabe zu lösen, ist eine alltägliche Erfahrung.[9] An dieser Stelle

[9] Auch bei der praktischen Modellierung (einer Bibliothek) im Rahmen des Studienprojektes

ließe sich allerdings einwenden, daß jedes dieser Modelle im Grunde eine andere Wirklichkeit modelliert. Denn jedes dieser Modelle enthält andere Objekte, andere Eigenschaften und bzw. oder andere Beziehungen zwischen den Objekten. Wird nun diese konstruierte Wirklichkeit als Bedeutungsfeld angesehen, so wird dadurch ein Notationssystem definiert. Nimmt man eine Dreiteilung der Modellierungsgegenstände an, nämlich in zu modellierende Realität, formulierte bzw. konstruierte Wirklichkeit und Formalisierung durch die Zeichen des Modellierungsmittels, so besteht das Notationssystem aus der Verbindung der beiden letzten Teile. Der erste Teil, die Realität ist mit Goodman allenfalls als gewohnheitsmäßig konstruierte Wirklichlichkeit, etwa im Rahmen des Symbolsystems der natürlichen Sprache, zu verstehen, und nicht als eine von Symbolisierung unabhängige Realität.

Inwiefern lassen sich nun Goodmans Aussagen über Notations- und Symbolsysteme in der Informatik interpretieren? Nach Goodman hängt das Erkennen eines Gegenstandes sowie der Gegenstand selbst vom gewohnten Repräsentationssystem des Erkennenden ab. Übertragen auf die Informatik heißt dies somit, daß ein Modellierungsmittel die Entstehung des ihm zuzuordnenden Bedeutungsfeldes beeinflußt, daß also eine Programmiersprache bzw. deren Verständnis die Formulierung bzw. Konstruktion der Wirklichkeit prägt. Dies mag vorsichgehen, indem bei dem Versuch, die durch das Modellierungsmittel und dessen Verständnis gegebene Organisation der Zeichen in der Wirklichkeit wiederzufinden, eben diese Wirklichkeit so organisiert und damit auch neu erschaffen wird.

Der wirklichkeitsbildende Charakter eines Modellierungsmittels hängt nicht von der Programmiersprache, sondern von deren Verständnis ab, da die Programmiersprache an sich eben nur ein Notationsschema darstellt und somit keinen Bezug zu einem Bedeutungsfeld enthält. Folgt aber dieses Verständnis und damit der Bezug nun unmittelbar aus der Syntax der Programmiersprache? Dies ist wohl im allgemeinen zu verneinen. Verschiedene Personen haben unter Umständen unterschiedliche Arten des Verständnisses einer Sprache.[10]

Goodmans Aussagen über den Realismus eines Bildes lassen sich ebenso im Kontext der Informatik wiederfinden. Der Annahme, ein bestimmtes Repräsentationssystem produziere realistische Bilder, läßt sich in der Informatik die Auffassung gegenüberstellen, eine bestimmte Programmiersprache sei der natürlichen Weltsicht besonders nahe. Die Adäquatheit eines Modellierungsmittels ist ebenso relativ zu einem Repräsentationssystem, wie der Realismus eines Bildes. Dies stimmt mit der Beobachtung überein, nach der von seiten verschiedener Programmierschulen die jeweilige Sprache als besonders adäquat gegenüber der Wirklichkeit angesehen wird, denn gerade das Bild von der Wirk-

ergaben sich trotz der Verwendung des gleichen Modellierungsmittels eine Reihe unterschiedlicher Modelle.

[10] Die Erfahrungen im Studienprojekt bestätigten dies.

lichkeit wird von der Sprache beeinflußt. Adäquatheit wäre also auch hier nur eine Sache der Gewohnheit.

Wird nun Kreativität, also das Entdecken bzw. Erschaffen neuer Eigenschaften von Objekten und der Beziehungen zwischen ihnen, durch ein neues Modellierungsmittel begünstigt, bzw. ist sie sogar nur mit Hilfe neuer Modellierungsmittel möglich? Läßt sich ein Modellierungsmittel mit seinem Verständnis als ein Notationssystem auffassen, so ist dies sicherlich der Fall, da in einem Notationssystem die Repräsentation festgelegt ist und somit kein Raum für Kreativität bleibt, schließlich ist die Aufzeichnung einer gegebenen Aufführung in Form einer Partitur auch kein Akt der Kreativität. Dies gilt in diesem Sinne allerdings nur, sofern es die Beziehung zwischen der konstruierten Wirklichkeit und dem Formalismus betrifft. Bei der Erschaffung der konstruierten Wirklichkeit jedoch ist ebenso wie bei der Komposition eines Musikstückes Kreativität nötig. Wenn allerdings bei der Konstruktion dieser Wirklichkeit die Strukturen des Modellierungsmittels verwendet werden, so liegt die Kreativität als das Erschaffen solcher Strukturen wohl eher in der Erschaffung der Modellierungsmittel. Allerdings ist durch die Auswahl verschiedener Eigenschaften der Realität und der Zuordnung dieser in die durch das Modellierungsmittel gegebenen Strukturen noch Raum für Kreativität geblieben. Insgesamt liegt also die Kreativität bei gegebenem Modellierungsmittel nur in der Beziehung zwischen Realität und konstruierter Wirklichkeit und auch hier nicht in der Struktur dieser Wirklichkeit, denn diese wird durch das Modellierungsmittel gegeben, sondern im Verständnis dieser Struktur.

Kapitel 5

Gesellschaftliche Konstruktion der Wirklichkeit: Peter L. Berger und Thomas Luckmann
Autorin: Birgit Schelm

In ihrem Buch „Die gesellschaftliche Konstruktion der Wirklichkeit“ [Berger und Luckmann 1969] begreifen Peter Berger und Thomas Luckmann gesellschaftliche Wirklichkeit als ein Resultat von ständigen Wechselwirkungen zwischen den Individuen einer Gesellschaft und dem, was diese Individuen als gesellschaftliche Wirklichkeit wahrnehmen. Sie gehen also nicht davon aus, daß das, was wir als Wirklichkeit empfinden, eine absolute Wirklichkeit ist, die so auch ohne die Menschen existiert, sondern begreifen diese Wirklichkeit als das Ergebnis von interaktiven, sozialen Prozessen. Phänomene, die wir als Wirklichkeit wahrnehmen, können durch solche Prozesse entstehen, in ihrer Existenz weiter gefestigt, aber auch verändert werden.

Der Kern ihres vor einem soziologischen und philosophischen Hintergrund geschrieben Textes ist, das Wissen in der Alltagswelt und die Entstehung dessen, was die Individuen einer Gesellschaft als objektive gesellschaftliche Wirklichkeit wahrnehmen, in einem wissenssoziologischen Rahmen zu betrachten.

5.1 Entstehung von gesellschaftlicher Wirklichkeit

Die Prozesse, durch die als objektiv empfundene gesellschaftliche Wirklichkeit entsteht, bezeichnen Berger und Luckmann als *Institutionalisierungsprozesse.* Obwohl Institutionalisierungsprozesse als interaktive Prozesse zu verstehen sind, unterteilen Berger/Luckmann diese in verschiedene Stadien, um das Entstehen

von gesellschaftlicher Wirklichkeit besser veranschaulichen zu können.

Aus der direkten Interaktion von Individuen entstehen bestimmte Handlungsabläufe oder Verhaltensweisen. Diese Handlungsabläufe oder Verhaltensweisen werden zur Routine, wenn die beteiligten Individuen diese Handlungsmuster als sinnvoll[1] und diese Handlungsmuster immer häufiger anwenden. Dieser Vorgang, bei dem Handlungsmuster zur Routine werden, wird auch als *Habitualisierung* bezeichnet. Auch wenn bereits habitualisierte Tätigkeiten nicht immer wieder von neuem auf ihre Sinnhaftigkeit hin überprüft werden müssen, so behalten sie doch ihren sinnhaften Charakter bei. „Habitualisierte Tätigkeiten behalten natürlich ihren sinnhaften Charakter für jeden von uns, auch wenn ihr jeweiliger Sinn als Routine zum allgemeinen Wissensvorrat gehört, zur Gewißheit geworden und dem Einzelnen für künftige Verwendung zuhanden ist." ([Berger und Luckmann 1969] S. 57)

Der nächste Schritt zur Institutionalisierung solcher Handlungsmuster ist die Typisierung derselben. Unter Typisierung verstehen Berger/Luckmann dabei das Zuordnen von Individuen zu Kategorien. Diese Kategorien prägen unsere Wahrnehmung von Individuen allerdings schon von vorneherein. Die Kategorien werden als Typen bezeichnet und sind Bestandteil der gesellschaftlichen Wirklichkeit. Diese Typisierung ist allerdings umso empfänglicher für Veränderungen, je direkter der Kontakt zu der jeweiligen Person ist. Verhaltensweisen der Person, die nicht mit der Zuordnung zu einem bestimmten Typ übereinstimmen, führen dann sehr schnell dazu, daß diese Zuordnung wieder revidiert wird. Diese Revision der Zuordnung wird allerdings umso schwieriger und unwahrscheinlicher, je größer die Distanz zur betreffenden Person ist.

Bei der Typisierung von habitualisierten Tätigkeiten werden bestimmte Handlungsmuster bestimmten Typen, d.h. Kategorien von Handelnden, zugeordnet und umgekehrt. Die Folge davon ist, daß von bestimmten Typen bestimmte Handlungsmuster erwartet werden bzw. erwartet wird, daß bestimmte Handlungsmuster von bestimmten Typen ausgeführt werden. Handlungen werden durch Handelnde erklärt und umgekehrt. Jede solche Typisierung ist bereits eine Institution. Kurz gesagt „Institutionalisierung findet statt, sobald habitualisierte Handlungen durch Typen von Handelnden reziprok typisiert werden." ([Berger und Luckmann 1969] S. 58).

Durch die im Lauf einer gemeinsamen Geschichte entstandenen wechselseitigen Typisierungen erhalten die Institutionen eine Historizität, und durch die Tatsache ihres Vorhandenseins kontrollieren Institutionen menschliches Verhalten.

Die Institutionalisierung vollendet sich selbst, wenn die entwickelten Handlungsmuster an Personen weitervermittelt werden, die bei der Entwicklung der jeweiligen Institution nicht beteiligt waren, z.B. also an nachfolgende Generationen.

[1] sinnvoll in Bezug auf das erzielte Ergebnis kann z.B. der Erhalt der eigenen Macht, aber auch die Vereinfachung bestimmter Arbeitsabläufe sein

Da die Erinnerung an die Notwendigkeit und Sinnhaftigkeit der Handlungsmuster nicht vorhanden ist, muß deren Notwendigkeit auf anderem Wege nachgewiesen werden. Da die Sinnhaftigkeit nur durch Legitimationsmechansimen nachgewiesen werden kann, entsteht die Gefahr, daß die Handlungsmuster nicht eingehalten werden, so daß auch noch Kontrollmechanismen nötig werden, um die Einhaltung der Handlungsmuster zu gewährleisten.

An diesem Punkt erlangen die institutionalen Handlungsmuster eine eigene objektivierte Wirklichkeit, „werden als über und jenseits der Personen, welche sie »zufällig« im Augenblick verkörpern, daseiend erlebt. Mit anderen Worten: Institutionen sind nun etwas, das seine eigene Wirklichkeit hat, eine Wirklichkeit, die dem Menschen als äußeres, zwingendes Faktum gegenübersteht“ ([Berger und Luckmann 1969] S. 62).

Jede weitere Vermittlung und Reflexion der neu geschaffenen Institutionen bewirkt auch wieder, daß diese Institutionen weiter gefestigt werden. Eine wichtige Rolle spielt dabei das Wissen von den Strukturen der sozialen Welt. Dieses Wissen wird während der Sozialisation der Individuen einer Gesellschaft als objektive Wahrheit an diese vermittelt und von ihnen ebenso internalisiert. Für Berger/Luckmann ist dieses Wissen „Verwirklichung im doppelten Sinne des Wortes: Erfassen der objektivierten gesellschaftlichen Wirklichkeit und das ständige Produzieren eben dieser Wirklichkeit in einem“ ([Berger und Luckmann 1969] S. 71). Dadurch machen sie deutlich, daß das Wissen der Individuen nicht nur den Individuen hilft, sich in der gesellschaftlichen Wirklichkeit zurechtzufinden, sondern auch dazu dient, diese gesellschaftliche Wirklichkeit zu produzieren.

5.2 Bezug zur Informatik

Institutionalisierungsprozesse in der Informatik

Institutionalisierungsprozesse, wie oben beschrieben, lassen sich auch in unterschiedlichen Ausprägungen innerhalb der Informatik wiederfinden, so läßt sich z.B. Modellierung bzw. Problemlösung als Institutionalisierungsprozeß betrachten.

Dabei wird bereits bei der Anforderungsanalyse ein erstes Modell entwickelt, in dem sich die Dinge widerspiegeln, die die Personen, die die Anforderungsanalyse erstellen, wahrnehmen und die ihnen wichtig erscheinen. Erweist sich dieses Modell als sinnvoll, z.B. in bezug auf die Aufgabenstellung, wird im allgemeinen im weiteren Lösungsprozeß auf dieses Modell zurückgegriffen. Der zu modellierende Vorgang wird nicht wieder von neuem untersucht. Basierend auf diesem ersten Modell wird also das nächste Modell entwickelt, auf das dann wiederum bei den nächsten Schritten vorrangig zurückgegriffen wird. Häufig wird dieses Phänomen

auch noch durch das Projektmanagement unterstützt, das z.B. vorsieht, sich immer nur auf das Modell der vorangegangenen Stufe zu beziehen und nicht wieder zum Ausgangsmodell zurückzukehren oder die verschiedenen Stufen in mehreren Zyklen immer wieder zu durchlaufen.

Eine auf diese Art und Weise gewonnene Lösung erscheint den am Lösungsprozeß beteiligten Personen sinnvoll, da sie die Sinnhaftigkeit der einzelnen Modelle immer wieder feststellen konnten und mußten, um in ihrem Lösungsprozeß voranzukommen. Personen, die an diesem Lösungsprozeß nicht beteiligt waren, erscheint eine so gewonnene Lösung, die sich ihnen z.B. in Form eines neuen Rechnerarbeitsplatzes oder einer neuen Software präsentiert, allerdings nicht unbedingt sinnvoll. Die Lösung hat für sie also einen Status angenommen, in dem sie nicht mehr als etwas Veränderliches erscheint, sondern (zumindest für Benutzerinnen[2], die an dem Entstehungsprozeß nicht beteiligt waren) einen unumstößlichen Fakt darstellt, der die Realität der Benutzerinnen und damit auch ihre Rolle in ihrem jeweiligen Arbeitszusammenhang z.T. erheblich verändert (z.B. in vielen Fällen der Neueinführung von Rechnerarbeitsplätzen). Das Rollenwissen, das die Benutzerinnen während ihrer sekundären Sozialisation (vgl. [Berger und Luckmann 1969] S. 148) internalisieren, und auf dessen Grundlage sie ihre bisherige Berufstätigkeit ausüben konnten, reicht für die durch die Einführung informatischer Artefakte veränderte Arbeitsrealität nicht mehr aus, so daß eine erneute Sozialisation nötig wird, in der die Benutzerinnen das für die veränderte Rolle nötige Wissen internalisieren. Auf diese Sozialisation der Benutzerinnen gehe ich im folgenden ein.

Sozialisationsprozesse in der Informatik

Berger/Luckmann unterscheiden zwei Stufen der Sozialisation: Die primäre Sozialisation und die sekundäre Sozialisation. Die primäre Sozialisation dient der Internalisierung der Alltagswelt und findet im Kindesalter statt, wobei die Eltern oder andere mit der Sozialisation betraute Personen dem Kind die gesellschaftliche Wirklichkeit, wie sie von ihnen wahrgenommen wird, vermitteln. Durch diese primäre Sozialisation wird der Mensch Mitglied der Gesellschaft (vgl. [Berger und Luckmann 1969] S. 141). Die sekundäre Sozialisation dagegen findet nach der primären Sozialisation statt und dient in erster Linie dem Erwerb von rollenspezifischem Wissen. Dazu gehört das „Sich–zu–eigen–Machen“ von rollenspezifischem Vokabular, d.h. „die Internalisierung semantischer Felder, die Routineauffassung und –verhalten auf einem institutionalen Gebiet regulieren“([Berger und Luckmann 1969] S. 149), sowie die „»stillen

[2]Wir betrachten im folgenden in erster Linie informatische Artefakte, die für gewerbliche Zwecke konstruiert werden und damit die Arbeitsgebiete der Benutzerinnen beeinflussen und verändern. Unter Benutzerinnen verstehen wir also Personen, die mit informatischen Artefakten im Rahmen ihrer Berufstätigkeit umgehen müssen.

Voraussetzungen«, Wertbestimmungen und Affektnuancen dieser semantischen Felder"(ebd.). Die hier betrachteten Sozialisationsprozesse lassen sich der sekundären Sozialisation nach Berger/Luckmann zuordnen.

Wie bereits erwähnt beschränke ich mich bei der Betrachtung der Sozialisation von Benutzerinnen informatischer Artefakte auf Benutzerinnen, die mit diesen Artefakten im Rahmen ihrer beruflichen, nicht informatisch bestimmten Tätigkeit zu tun haben. Die Entscheidung, solche Artefakte zu verwenden, wird selten von den Benutzerinnen selbst getroffen, sondern von anderen Personen ihres Arbeitsumfeldes, die aus der Sicht der Benutzerinnen meist eine Vorgesetztenposition inne haben. Die Legitimierung des Einsatzes solcher technischer Artefakte erfolgt häufig unter den Stichpunkten „technischer Fortschritt", Rationalisierung und Arbeitserleichterung, die so fast den Status einer symbolischen Sinnwelt im Sinne Bergers und Luckmanns erhalten. Auf der anderen Seite sind es aber genau diese Stichpunkte, unter denen technische Artefakte konstruiert und vor allem weiterentwickelt werden.

Betrachtet man nun die Sozialisation der Benutzerinnen, lassen sich grob zwei Kategorien von Benutzerinnen unterscheiden: Solche, die technischen Artefakten prinzipiell positiv gegenüberstehen, und solche, die technische Artefakte eher ablehnen.

Bei dieser speziellen Sozialisation, wie sie im Zusammenhang mit der Benutzung informatischer Artefakte anzutreffen ist, ist immer wieder zu beobachten, daß die Benutzerinnen sich selbst überlassen bleiben, also keine Schulung oder ähnliche Unterstützung in diesem Sozialisationsprozeß erfahren, obwohl sich diese Situation allmählich verbessert.

Bei Benutzerinnen, die technischen Artefakten positiv gegenüberstehen, ist die Bereitschaft zur Internalisierung eher gegeben, als bei der anderen Gruppe von Benutzerinnen. Zu beobachten ist, daß sie z.T. sogar mit großer Hingabe selbst völlig unzureichende Systeme auch ohne Unterstützung durch andere Personen internalisieren und dabei das nötige Vokabular verinnerlichen, das mit der Rolle von fortgeschritteneren Benutzerinnen dieser Artefakte verbunden ist. Vor allem beim Erlernen des Umgangs mit völlig unzureichenden, aber neuen Systemen werden aus Faszination und Hingabe zur Technik Dinge in Kauf genommen, die in anderen Bereichen des Lebens wegen ihrer Unbequemlichkeit nie akzeptiert würden.

Bei der anderen Gruppe von Benutzerinnen fehlt diese Euphorie, weswegen sie nicht oder nur kaum bereit sind, sich mit den ihnen vorgesetzten Artefakten „anzufreunden", vor allem wenn sie keine Unterstützung durch andere Personen erfahren. Die Folge davon ist, daß die oft versprochene Arbeitserleichterung nicht eintritt. Im Gegenteil, es wird, wo dies möglich ist, auf althergebrachte Arbeitsmethoden zurückgegriffen, da ansonsten zusätzlich Probleme bewältigt werden müssten, die ohne die Einführung dieser Artefakte nie aufgetreten wären. Die So-

zialisation dieser Benutzerinnen verläuft häufig erfolglos.

Allerdings hat die Tatsache, daß bei weitem nicht alle Benutzerinnen von technischen Artefakten den Sozialisationsprozeß erfolgreich durchlaufen haben, zur Folge, daß der Aspekt der Benutzerinnenfreundlichkeit in der Informatik eine zunehmende Rolle spielt, und daß die Notwendigkeit, Benutzerinnen von technischen Artefakten entsprechend zu schulen, um diesen Sozialisationsprozeß zu fördern, erkannt wurde. Dadurch sind für Informatikerinnen wiederum neue Aufgabengebiete entstanden: Zum einen das der Softwareergonomie, zum anderen das der Schulung.

5.3 Die Rolle der Sprache

Sprache und die Entstehung gesellschaftlicher Wirklichkeit

Am Beispiel der Institutionalisierungsprozesse wird deutlich, welche Mechanismen bei der Entstehung gesellschaftlicher Wirklichkeit zum Tragen kommen, und daß objektive Wahrheiten, die ja auch Bestandteil gesellschaftlicher Wirklichkeit sind, „nur“ Ergebnis solcher Prozesse sind. Für den Aspekt der Modellierung in der Informatik ist noch ein weiterer Punkt, den Berger/Luckmann im Zusammenhang mit der gesellschaftlichen Konstruktion von Wirklichkeit betrachten, und an dem zudem das Prinzip der Wechselwirkungen sehr deutlich wird, interessant: Die Sprache. Um auf die Bedeutung von sprachlichen Ausdrucksmitteln einzugehen, erläutere ich im folgenden zunächst, wie Berger/Luckmann Sprache begreifen und welche Eigenschaften und Bedeutungen sie Sprache zuschreiben, bevor ich auf die Bedeutung der Sprache in der Informatik eingehe.

Rein formal betrachten Berger/Luckmann Sprache als ein System aus vokalen Zeichen. Dieses Zeichensystem hat Objektcharakter, ist also etwas, das Individuen außerhalb ihrer selbst vorfinden, und dem sie sich in seiner Wirkung auf sie selbst nicht entziehen können. „Ich treffe auf sie (Sprache) als auf einen Tatbestand außerhalb meiner selbst, und ihre Wirkung auf mich ist zwingend.“ ([Berger und Luckmann 1969] S.40)

Als Symbole bezeichnen Berger/Luckmann sprachliche „Verweisungen“, durch die Wirklichkeitssphären überspannt werden. Wirklichkeitssphären sind dabei z.B. die Wirklichkeit der Alltagswelt, aber auch die Wirklichkeit spezieller Gruppen, wie etwa einer Sekte oder Religionsgemeinschaft, aber auch der Wissenschaftlerinnen einer Spezialdisziplin.

Die besondere Bedeutung, die der Sprache bei der gesellschaftlichen Konstruktion von Wirklichkeit zukommt, ist in ihrer Ablösbarkeit von der „vis–à–vis“ Situation begründet. Dadurch kann Sprache „Sinn, Bedeutung, Meinung […] vermitteln, die nicht direkter Ausdruck des Subjekts »hier und jetzt« sind“

([Berger und Luckmann 1969] S. 39). Daher kann Sprache dazu verwendet werden, Erfahrungen und Bedeutungen zu speichern und vor allem an nachfolgende Generationen zu vermitteln.

Sprache kann also dazu verwendet werden, „Phänomene zu »vergegenwärtigen«, die räumlich, zeitlich und gesellschaftlich vom »hier und jetzt« abwesend sind“([Berger und Luckmann 1969] S.41). Durch Sprache können also entfernte Wirklichkeitssphären zugänglich gemacht werden; Berger/Luckmann sprechen dann von einer symbolischen Sprache. In einer Wirklichkeitssphäre, die fern der Alltagswelt ist, z.B. in der Wissenschaft, dient Sprache dazu, Symbole zu bilden, aber auch diese wieder in die Alltagswelt zurückzuholen. Dabei werden die Symbole als objektiv wirkliche Fakten präsentiert.

Ein weiterer wichtiger Aspekt der Sprache ist ihre Fähigkeit, Sinnzonen bzw. semantische Felder herzustellen und diese andererseits aber auch durch Sprache wieder gegeneinander abzugrenzen. Denn erst mit Hilfe dieser Sinnzonen und semantischen Felder können biographische und historische Erfahrungen verarbeitet und vor allem weitervermittelt werden (vgl. [Berger und Luckmann 1969] S. 43).

Da also erst durch die Sprache die Weitervermittlung von Erfahrungen und Wissen über die Strukturen der sozialen Welt möglich ist, spielt die Sprache bei der Institutionalisierung und damit auch bei der gesellschaftlichen Konstruktion von Wirklichkeit eine zentrale Rolle.

Die Rolle der Sprache in der Informatik

Modellierung ist in der Informatik ohne Sprache nicht denkbar, da auf allen Stufen der (Software–)Modellierung sprachliche Ausdrucksmittel zum Tragen kommen. Das beginnt mit einer natürlichsprachlich formulierten Aufgabenstellung und findet seinen Abschluß in einem formal – programmiersprachlich beschriebenen Programm, das auf einem Computer ausgeführt wird. Dabei ist Sprache das Mittel, mit dessen Hilfe Modelle beschrieben und dadurch begreifbar werden. Da Sprache die Eigenschaft hat, Phänomene zu vergegenwärtigen, die räumlich, zeitlich und gesellschaftlich von der direkten „hier – und – jetzt“ Situation entfernt sind, ist es möglich, z.B. Arbeitsabläufe in einem Betrieb zu beschreiben und dadurch diese Arbeitsabläufe auch außerhalb der konkreten Arbeitssituation für Personen, die das entsprechende Arbeitsumfeld nicht kennen, begreifbar zu machen. Eine solche sprachliche Beschreibung kann aber das Wissen einer Person, die in dem beschriebenen Arbeitsumfeld tätig ist und die Arbeitsabläufe gut kennt, da sie sie selbst ausführt, nicht exakt wiedergeben. Sie ist vielmehr eine Konstruktion, die durch die Erfahrungen derjenigen Person, die diese Beschreibung erstellt, und durch deren Vorstellung von der Bedeutung der verwendeten Begriffe geprägt ist.

So ist bereits eine Aufgabenstellung eine Konstruktion, die aber nicht nur von

individueller Wahrnehmung, Erfahrung und Begriffsverständnis geprägt ist, sondern auch Intentionen beinhaltet, die durch die Sprache transportiert werden und die Wahrnehmung der Personen, die mit der Lösung der Aufgabe befaßt sind, beeinflussen.

Jedes Modell in der Informatik, das durch sprachliche Mittel beschrieben wird, stellt also bereits eine Konstruktion dar, auch wenn es sich dabei noch gar nicht um die endgültige Lösung handelt. Die sprachliche Beschreibung eines Arbeitsablaufes, wie sie z.B. Mitarbeiterinnen in einem Softwareprojekt in Form einer Vorstudie vorliegt, vermittelt durch die Fähigkeit der Sprache zur Objektivation, den Mitarbeiterinnen den Eindruck, die dort beschriebenen Arbeitsabläufe fänden tatsächlich genau so statt. Alle weiteren Wahrnehmungen der zu modellierenden Arbeitsabläufe sind dann bereits von dem Wissen, das eine solche Vorstudie liefert, geprägt.

Ein solches Zwischenmodell hat aber in seiner objektivierenden Wirkung zunächst nur Einfluß auf die mit der Modellierung betrauten Personen. Die fertige Implementierung des Modells, die in einer formalen Programmiersprache formuliert und in rechnerausführbaren Code übersetzt wurde, hat eine noch viel weitgreifendere Wirkung, die über die bloße Semantik der verwendeten Programmiersprache hinausgeht. Durch die Semantik der Programmiersprache wird festgelegt, wie der Computer arbeiten soll, auf dem das Programm zur Ausführung kommt. Ein Programm reguliert jedoch nicht nur den Computer, sondern auch den Arbeitsstil des Menschen, der mit dem Programm arbeitet. Diese Regulierung kommt allerdings nicht erst bei der Implementierung zustande, sondern wird bereits bei der Modellierung auf einer nicht–formalen Ebene festgelegt.

Das bedeutet also, daß selbst bei der Formulierung von Modellen mit Hilfe formaler (Programmier–)Sprachen mit einer formal festgelegten Semantik Realitäten konstruiert werden, die über diese rein formale Semantik hinausgehen und die die Wahrnehmung, Erfahrung und Meinung der Personen widerspiegeln, die am Modellierungsprozeß beteiligt waren.

Zusammenfassung

Die Darstellung der gesellschaftlichen Konstruktion der Wirklichkeit von Berger/Luckmann ist sehr umfassend und läßt den Bezug zur Informatik nicht immer unmittelbar erkennen, da sie in erster Linie die gesellschaftliche Wirklichkeit betrachten und weniger eine gegenständliche Wirklichkeit, wie die Natur, auf die sich etwa Bacon bezieht. Wenn man allerdings bedenkt, daß sich ein beachtlicher Teil der informatischen Anwendungen auf die Umgestaltung von Arbeitsprozessen bezieht, die auch als Institutionen im Sinne Berger/Luckmanns betrachtet werden können, so erschließen sich eine Fülle von Möglichkeiten, den Ansatz Berger/Luckmanns für die Informatik nutzbar zu machen.

Teil III

Modellierungansätze in der Informatik

Eine kurze Vorbemerkung zu Teil III

Die Texte, die in diesem Teil versammelt sind, unterscheiden sich zum einen aufgrund der verschiedenen Modellierungsansätze, die sie thematisieren, und zum anderen aufgrund der Argumentationsweisen, auf denen die Texte basieren. Die konzeptionelle Klammer, die die Texte zusammenhält, sind die Konstruktions- und die Abbildperspektive, die in Teil I eingeführt und in Teil II vertieft worden sind.

Die Texte in diesem Teil verwenden das Perspektivenkonzept, um Modellierungsansätze in der Informatik zu untersuchen. Das Perspektivenkonzept liefert eine Interpretationsfolie, die die Untersuchungen orientiert. Die Texte zeigen sehr deutlich, auf welch konsequente Weise dieses Konzept ein Verständnis der verschiedenen Modellierungsansätze ermöglicht. Außerdem ist es möglich, die unterschiedlichen Modellierungsansätze in diesem Rahmen nach klaren Kriterien aufeinander zu beziehen.

Florian Theißing untersucht die Geschichte einer Modellierungsmethode, nämlich der Requirement Definition Languages, und arbeitet heraus, daß es im wesentlichen zwei Strömungen gegeben hat, denen die Entwicklungsgeschichte dieser Modellierungsmethode folgte. Von diesen Strömungen läßt sich die eine der Abbild- und die andere der Konstruktionsperspektive zuordnen.

Die Autoren Melahat Elis und Michael Freitag vergleichen in ihrem Beitrag drei unterschiedliche Modellierungsansätze dreier einflußreicher Informatiker. Sie zeigen, daß John Backus und David L. Parnas Ansätze vertreten, die der Abbildperspektive zuzuordnen sind, während der Ansatz von Peter Naur sehr gut von der Konstruktionsperspektive aus zu verstehen ist.

Irina Leyde untersucht in ihrer Arbeit den Ansatz PETS (Partizipation, Evolution und Transparenz bei der Softwareentwicklung), indem sie ihn mit dem sogenannten Phasenmodell, einem *klassischen* Ansatz vergleicht. Im Gegensatz zum Phasenmodell, das der Abbildperspektive zuzuordnen ist, berücksichtigt der Ansatz PETS ausdrücklich die Personen, die an der Modellierung beteiligt sind. Die Berücksichtigung der Modellierenden ist ein Kennzeichen der Konstruktionsperspektive.

Jürgen Schöning und Birgit Schelm untersuchen zwei verschiedene Ansätze der Wissensmodellierung. Beiden Ansätzen liegt eine ähnliche Diagnose derjenigen Probleme zugrunde, die bei der informatischen Modellierung von Wissen entstehen. Und gerade diese Diagnose kann mit Hilfe des Perspektivenkonzepts gut systematisiert werden, da es für den Modellierungsansatz eine Rolle spielt, ob man eine Vorstellung von Wissensprozessen und Wissen zugrundelegt, die der Abbildperspektive oder der Konstruktionsperspektive zuzurechnen ist.

Kapitel 6

Zur Geschichte der Requirement Definition Languages
Autor: Florian Theißing

In diesem Kapitel möchte ich beschreiben, wie sich Techniken zur Beschreibung von Informationssystemen und der Anforderungen an Informationssysteme (auch genannt Systemanalysetechniken oder Requirement Definition Languages – RDL)[1] entwickelt haben, und welches Wirklichkeitsverständnis in sie eingegangen ist. In einem ersten Teil soll dargestellt werden, aufgrund welcher Problemstellungen die Entwicklung solcher Sprachen in Angriff genommen wurde und welche Lösungswege dabei beschritten wurden. Danach möchte ich etwas näher auf einige Sprachkonzepte und Techniken eingehen. Im dritten und letzten Teil möchte ich versuchen, das diesen Techniken zugrundeliegende Modellierungsverständnis herauszuarbeiten und zu dem historischen Kontext in Beziehung zu setzen.

Vorher will ich aber noch erklären, warum ich ein historisches Herangehen für sinnvoll halte, um zu Erkenntnissen über Modellierungsperspektiven in der Informatik zu kommen. Entwicklungsentscheidungen fallen nicht im luftleeren Raum, sondern werden von Menschen getroffen, die vielfältigen Einflüssen und Anregungen ihrer Umwelt ausgesetzt sind. Diese Einflüsse prägen auch die inhaltliche Gestaltung der entwickelten Techniken. Um zu verstehen, warum eine bestimmte Technik, ein Verfahren oder eine Methode in einer bestimmten Weise gestaltet wurde, kann es deshalb nützlich sein, zu untersuchen, wer die an der Entwicklung beteiligten Personen waren, welche Weltbilder oder ”Visionen” sie hatten, welche Probleme und welche Lösungsverfahren allgemein anerkannt waren, un-

[1] Diese Ausdrücke werden, obwohl sie eigentlich sehr Unterschiedliches zu bezeichnen scheinen, für dieselben Werkzeuge gebraucht. Auch ich werde diese Bezeichnungen beliebig verwenden.

ter welchen Bedingungen sich der Entwicklungsprozeß vollzog, usw. So kann ein historischer Ansatz zur Klärung der Frage beitragen, warum eine bestimmte Modellierungsperspektive Eingang in eine Methode oder ein Werkzeug gefunden hat.

6.1 Entstehungsbedingungen der RDL

Probleme entstehen

Mit der Verbreitung der Computer in den Unternehmen und mit der Entwicklung großer Anwendungssysteme für das Militär in den fünfziger Jahren entwickelte sich die Arbeitsteilung zwischen DV-SpezialistInnen und DV-NutzerInnen. Zwischen die NutzerInnen und den Computer trat die DV-Abteilung oder das Projektteam[2] mit CodiererInnen, ProgrammiererInnen und Systemanalytikern als Vermittlungsinstanz[3]. Die Fähigkeit, Programme für Computer zu entwickeln, die in den wissenschaftlichen Einrichtungen noch eine Art Zusatzqualifikation der WissenschaftlerInnen gewesen war, wurde nun zum eigenständigen Beruf und die DV-SpezialistInnen entwickelten ein berufliches Selbstverständnis, das sich von dem ihrer KollegInnen aus den Fachabteilungen unterschied. Aus diesem Umstand, daß sich das Wissen über und auch der Zugang zum Computer bei einem bestimmten Berufstand konzentrierte, entwickelten sich im Laufe der sechziger Jahre einige Probleme.

Mit der rasanten Verbreitung der Computer in den Betrieben in den sechziger Jahren stieg der Bedarf an ausgebildetem DV-Personal rapide an. Das Ausbildungssystem für DV-Berufe entwickelte sich aber nur langsam und erfahrene ProgrammiererInnen gab es, da der Computer noch eine sehr junge Entwicklung war, nur sehr wenige. ProgrammiererInnen wurden knapp. Um sich angemessen mit DV-Arbeitskräften versorgen zu können, zahlten die Unternehmen Gehälter, die oft als weit überzogen angesehen wurden. Software-Projekte gerieten in Schwierigkeiten, weil DV-Kräfte in Schlüsselpositionen die Firma für eine besser bezahlte Stelle verließen.

Da die Personalknappheit durch verstärkte Ausbildung nur sehr langsam beseitigt werden konnte, sollte die Produktivität der vorhandenen ProgrammiererInnen soweit wie möglich erhöht werden. Daraus ergab sich aber schon das nächste

[2] Die betrieblichen Anwendungen wurden zum allergrößten Teil in den Unternehmen selbst entwickelt, bei den avancierten Systemen des Militärs waren oft auch Fremdfirmen an der Entwicklung beteiligt.

[3] Die Unterteilung in Systemanalyse, Programmierung und Kodierung entstand im Rahmen der ersten großen militärischen Entwicklungsprojekte Mitte der fünfziger Jahre, insbesondere des SAGE-Systems, das für die Organisation von Systementwicklungsprojekten insgesamt Maßstäbe setzte.

Problem. Eine genauere Kontrolle der ProgrammiererInnen durch das Management, die als ein zentrales Mittel zur Produktivitätssteigerung angesehen wurde, schien aufgrund des besonderen Charakters der Programmierarbeit unmöglich. Programmieren galt als Kunst, deren Erfolg stark von der Intuition und Kreativität, wenn nicht gar Genialität der einzelnen ProgrammiererInnen abhängig gemacht wurde, Faktoren, die sich wirksamen Kontrollen entzogen.[4] Ein DV-Manager klagte Anfang der sechziger Jahre: "My programmers are using an English language compiler to write their programs, but looking over their work I still don't know what they are doing."[5]

Die Trennung von Anwendung und Entwicklung brachte noch ein weiteres Problem mit sich. Um zu brauchbaren Anwendungen zu kommen, mußten sich die EntwicklerInnen, die über das notwendige technische Wissen verfügten und die Beschäftigten der Fachabteilungen, die das Organisationswissen mitbrachten, miteinander verständigen. Dieser Kommunikationsprozeß gestaltete sich äußerst schwierig. Die EntwicklerInnen verstanden oft die Anforderungen der NutzerInnen nicht, die wiederum den technischen Jargon der EntwicklerInnen nicht verstanden. Die Folge war, daß DV-Abteilungen oft Programme entwickelten, die zwar ihren eigenen technischen Spezifikationen entsprachen aber den Anforderungen der Nutzer nicht genügten. ein DV-Manager hielt daher "the gap between the computer staff and the operating management" ([McCarn 1970] S. 24) für das Hauptproblem der Systementwicklung Ende der sechziger Jahre.

Die überzogene technische Orientierung der DV-Kräfte wurde als eine Ursache für die zunehmende Isolierung der DV-Kräfte und die daraus resultierenden Verständigungsschwierigkeiten angesehen. Tatsächlich hatten die DV-Kräfte, da die Programmierung in der Anfangsphase vor allem davon bestimmt war, die Beschränkungen der Hardware zu überwinden, ein vornehmlich technisches Selbstverständnis entwickelt. Auch die Ausbildung konzentrierte sich auf die Vermittlung technischen Wissens. Die SystementwicklerInnen waren deshalb in erster Linie an der Lösung technischer Probleme interessiert. Die Einbettung ihres Systems in einen sozialen und organisatorischen Gesamtzusammenhang und die damit verbundenen Probleme wurden nicht gesehen oder unterschätzt. Enid Mumford, eine Soziologin, die in den sechziger Jahren Untersuchungen über Probleme der Softwareentwicklung durchführte schrieb: "the real clash of values in our user firms arose because of an unwillingness to recognize the importance of the human factors operating in user areas"[6], und auf der ersten Konferenz zu Software-Engineering 1968 beschrieb ein Teilnehmer die Einstellung seiner Kol-

[4] Versuche, die Produktivität der ProgrammiererInnen beispielsweise nach der Anzahl der pro Tag geschriebenen Lines of Code zu beurteilen, wurden bald als untauglich eingestellt. Die Situation änderte sich erst mit der Entwicklung strukturierter Systementwicklungsmethoden, die durch die Strukturierung des Arbeitsprozesses eine verbindlichere Kontrolle der geleisteten Arbeit möglich machten.

[5] Hughes in [Greenberger 1962] S. 285

[6] [Mumford 1972] zitiert nach [Friedman und Cornford 1989] S. 230

legen: ”Many of the people who design software refer to users as ’they’ and ’them’. They are some odd breed of cats living there in the outer world, knowing nothing, to whom nothing is owed. Most of the designers of manufacturer’s software are designing, I think, for their own benefit - they are literally playing games. They have no conception of validating their design before sending it out, or even evaluating the design in the light of potential use.”[7]

Natürlich war ein großer Teil der SystementwicklerInnen anderer Meinung. Für sie war die Unfähigkeit der NutzerInnen, ihre Anforderungen verständlich offenzulegen, das eigentliche Problem. So meinte ein anderer Teilnehmer der Konferenz: ”Users are interested in systems requirements and buy systems that way. But that implies that they are able to say what they want. Most of the users aren’t able to. One of the greatest difficulties will be out of our field as soon as the users realize what kind of problem they have.”[8]

So waren im Laufe der Entwicklung des Computereinsatzes in den sechziger Jahren einige miteinander verwobene Problemfelder entstanden: das Problem der Knappheit von qualifiziertem DV-Personal, die Schwierigkeit, die Arbeit der ProgrammiererInnen zu kontrollieren, und zunehmende Kommunikationsprobleme zwischen den SystementwicklerInnen und den AnwenderInnen. Die Entwicklung der RDL war Teil der Anstrengungen, die unternommen wurden, um diese Probleme in den Griff zu bekommen. Es wurden verschiedene Lösungsansätze entwickelt, die, aus unterschiedlichen Gründen, die Entwicklung von RDL als sinnvoll und notwendig erscheinen ließen. Diese Lösungsansätze will ich in den beiden nächsten Abschnitten kurz darstellen.

Lösungsstrategie Automatisierung

Der Verknappung der DV-Arbeitskräfte versuchte man durch beschleunigte Automatisierung zu begegnen. Die Anpassung an die spezifischen Bedingungen der Hardware sollte möglichst weit automatisch erfolgen, so daß sich die ProgrammiererInnen weitgehend auf die Lösung des jeweiligen Anwendungsproblems konzentrieren konnten. Dadurch sollte einerseits der Einsatz weniger qualifizierten Personals möglich werden und andererseits die Produktivität der ProgrammiererInnen erhöht werden. Ein Wissenschaftler begründete beispielsweise die Entwicklung von höheren Programmiersprachen mit der Notwendigkeit, die knapper werdende menschliche Intelligenz durch die reichlich vorhandene ”maschinelle Intelligenz” zu ersetzen: ”It is becoming clear that the limiting factor in the growth of the use of machines is beginning to be mankind’s ability to instruct machines in complex activities and not the machine’s raw information-processing power.

[7] Smith in [Naur et al. 1968] S. 23

[8] Berghuis in [Naur et al. 1968] S. 23

Eine ähnlich Ansicht wird auch bei Parnas deutlich. Vgl. Melahat Elis und Michael Freitag in Teil 3 dieses Bandes.

Programs are already being produced at the rate of some 300,000,000 words per year .... These circumstances naturally produce an intense pressure to economize on human intelligence in programming by developing machine intelligence as a substitute, so as to trade what is plentiful for what is scarce."[9]

Auch für den Wissenschaftler R. Bosak, der sich sehr früh mit der Entwicklung von RDL befaßte, war die Steigerung der Effizienz von ProgrammiererInnen ein Antrieb zur Automatisierung der Programmierung: "In the area of programming language, developments will make it possible for programs to be written and modified more efficiently, both from the standpoint of man-hours and elapsed time required. It will make it possible for less highly trained personnel to program the computer."[10] Aber Bosak hatte auch schon ein zweites Ziel der Automatisierung im Blick, nämlich die vollständige Ersetzung der Arbeitskraft der Programmierer durch automatische Systementwicklung: "The ultimate is to remove the programmer entirely from the process of writing operational programs. In effect, the manager would write and modify his own programs."[11]

Dieses Ziel, das Systementwicklungspersonal durch Automatisierung vollständig überflüssig zu machen, nahm an Attraktivität in dem Maße zu, in dem die Probleme der Kontrolle über die Programmierarbeit und die Verselbständigung der DV-Abteilungen ins Bewußtsein der verantwortlichen Manager rückten. Ein Offizier der Airforce beispielsweise soll die Entwicklung von COBOL mit den Worten kommentiert haben: "Now we can take back command of the Air Force from those damned programmers."[12]

Zur vollständigen Automatisierung der Systementwicklung wurden bald auch frühe Phasen der Programmentwicklung in den Blick genommen: "Work in automatic coding has concentrated on getting rid of the coder. We have not yet tackled the job of automatic programming (...) one step further is automated systems design, and this too may be attempted within the next ten years."[13]

So gab es parallel zur Entwicklung der Programmiersprachen bald Versuche, Sprachen zu entwickeln, mit denen Anwender ihre Anforderungen an ein Informationssystem direkt, ohne den Umweg über die ProgrammiererInnen, dem Rechner mitteilen konnten. Diesem Zweck waren die Konzepte für "Nonprocedural languages" verpflichtet, Sprachen, mit denen AnwenderInnen beschreiben können sollten, *was* ein System leisten sollte, ohne angeben zu müssen, *wie* das System es leisten sollte. Dieser Ansatz wurde in der weiteren Entwicklung der

[9] Sayre in [Greenberger 1962] S. 274

Außerdem erhoffte man sich anfangs durch die höheren Programmiersprachen eine bessere Kontrolle der Arbeit der Programmierer, indem ihre Programme auch für Manager lesbar werden sollten.

[10] Bosak 1960 zitiert nach [Kraft 1977] S. 27

[11] Bosak 1960 zitiert nach [Kraft 1977] S. 27

[12] Hopper in [Greenberger 1962] S. 285

[13] Hopper in [Greenberger 1962] S. 272

RDL weiterverfolgt. Die Entwicklung von RDL war durch die Hoffnung motiviert, den Prozeß der Anwendungsentwicklung vollständig automatisieren zu können, um dadurch die Probleme der Knappheit und Kontrolle der ProgrammiererInnen zu beseitigen. Dieses Ziel beeinflußte die Entwicklung und inhaltliche Ausgestaltung von RDL wesentlich.

Lösungsstrategie Kommunikationsunterstützung

Eine andere Strategie zur Lösung der angefallenen Probleme bestand im Versuch, die Kommunikation zwischen AnwenderInnen und SystementwicklerInnen zu unterstützen.[14]Schon die Arbeitsteilung zwischen Systemanalytikern und ProgrammiererInnen sollte die Kommunikation zwischen AnwenderInnen und DV-SpezialistInnen verbessern helfen. Die Systemanalytiker sollten eine Scharnierfunktion zwischen den AnwenderInnen und den ProgrammiererInnen einnehmen. Dazu sollten sie sowohl mit den Problemen des Anwendungsbereiches als auch mit den Möglichkeiten der Computertechnik vertraut sein. In der Praxis teilten die Systemenalytiker allerdings allzuoft die technische Orientierung der DV-SpezialistInnen.

Ein anderer Ansatz bestand darin, die Systemanalytiker mit Werkzeugen auszustatten, die ihre Arbeit erleichtern sollten. Dazu gehörten auch RDL. Es gab unterschiedliche Ansichten, auf welche Weise RDL zur Lösung eines Kommunikationsproblems beitragen könnten. Friedmann und Cornford beschreiben eine Sichtweise der Systemanalytiker auf RDL: ”They assume that the major problems of requirement specification are that users are inconsistent and incomplete in the way they state their requirements. The analyst must therefore express those requirements in a language that will make inconsistencies apparent, and identify gaps.”[15]

War es aus dieser Perspektive das Ziel, die AnwenderInnen quasi der Schlamperei zu überführen, ging eine andere Sichtweise davon aus, daß es tatsächlich ein Sprachproblem zwischen AnwenderInnen und SystementwicklerInnen gibt. RDL sollten dazu dienen, eine gemeinsame Kommunikationsbasis zu schaffen, auf der Anforderungen für beide Seiten adäquat darstellbar waren und Mißverständnisse durch einen gemeinsamen, genau definierten Sprachvorrat vermieden werden konnten. So war neben dem Ziel der Automatisierung auch der Versuch, die Kommunikation zwischen SystementwicklerInnen und AnwenderInnen zu verbessern, (in dem einen oder anderen Sinne) ein Faktor, der die Entwicklung von RDL motivierte und prägte.

[14]Dieser Lösungsansatz spielte eine eher untergeordnete Rolle. Die am weitesten verbreitete Reaktion der DV-SpezialistInnen auf Komunikationsprobleme bestand darin, sie zu ignorieren.

[15][Friedman und Cornford 1989] S. 262

6.2 Entwicklung der RDL

In den fünfziger und sechziger Jahren arbeiteten sowohl Computerhersteller als auch wissenschaftliche Forschungseinrichtungen an der Entwicklung von Systemanalysetechniken und Requirement Definition Languages. Während die Computerhersteller dabei meistens auf bestehenden Verwaltungstechniken wie Formularen, Organisationsschemata oder Flußdiagrammen aufbauten, versuchten die Ansätze der wissenschaftlichen Einrichtungen eher Mathematik und formale Logik für ihre Entwicklungen fruchtbar zu machen. Im folgenden will ich einige dieser Methoden, die beispielhaft für bestimmte Entwicklungsrichtungen sind, näher vorstellen.

Eines der ersten Hilfsmittel zur Darstellung von Informationsflüssen waren Flußdiagramme oder Flowcharts. Flowcharts wurden zuerst von F. W. Taylor und seinen MitarbeiterInnen, dem Ehepaar Gilbreth entwickelt, um Materialflüsse innerhalb einer Fabrik darzustellen. In die Informatik übernommen wurden Flußdiagramme von v. Neumann und Goldstine, die Ende der vierziger Jahre Flußdiagramme als Repräsentation von Programmen vorschlugen. Statt der Verarbeitung von Material wurde in den Kästen die Verarbeitung von Information dargestellt, während die Pfeile den Kontrollfluß repräsentierten. Später wurden Flußdiagramme auch für die Darstellung des Informationsflusses in einem System benutzt. Statt des Kontrollflusses repräsentierten die Pfeile hier den Datenfluß. Die Verarbeitung wurde meist durch kurze Texte oder Bilder in den Kästen dargestellt. Eine formalisierte Darstellung der Verarbeitung gab es nicht.

Die ersten formalisierten Darstellungselemente gab es bei den Information Process Charts (IPC), einer Technik, die in den fünfziger Jahren entwickelt wurde. IPC bestand aus einer Kombination aus Flowcharts und Tabellen. In den Tabellen war je eine Zeile pro Operation vorgesehen. In bestimmten Spalten mußten die Informationsfelder, die von der Operation benutzt wurden, angegeben werden, während in einer anderen Spalte die Art der Operation angegeben werden mußte. Das eigentlich Neue war die Darstellung der Operationen in den Diagrammen: Gab es bei den älteren Techniken keine Vorschriften über die Darstellung der Art der Verarbeitung, so waren bei IPC bestimmte Verben genau spezifiziert und definiert mit dem Ziel, eine konsistente Verständigung zwischen allen NutzerInnen sicherzustellen.[16]

Accurately Defined Systems (ADS) von NCR ist beispielhaft für die Ansätze, die in den sechziger Jahren von den großen Computerherstellern entwickelt wurden. Neu an ADS war, daß es neben einem Werkzeug zur Beschreibung eines Informationssystems auch die Analysemethode bereitstellte. Auf einem Satz von fünf verschiedenen Formulartypen sollten Output, Input, Verarbeitung und zu speichernde Daten definiert werden. Ausgehend vom benötigten Output sollten alle

[16] vgl. [Couger und Knapp 1974] S. 47

Datenelemente durch den Verarbeitungsprozeß zurückverfolgt werden, um zu einer konsistenten Definition des Inputs, der notwendigen Verarbeitung und Speicherung zu kommen. Die Entwickler von ADS sahen den Nutzen von ADS in erster Linie in der Unterstützung der Kommunikation zwischen den verschiedenen am Entwicklungsprozeß beteiligten Gruppen: ”Everyone who earns his livelyhood in or about data processing installations has, as an occupational hazard, experienced the frustration of inadequate or misunderstood system definitions at one time or another. In each case, the problem reflects the professional viewpoints, the personalities, or the inadequacies of the individuals involved. But there are enough common denominators so that the problem can be typified: - Frequently, the person defining a system and the individual responsible for implementing it have different backgrounds and work within entirely different disciplines. For example, there are semantic differences in interpretation of terms and objectives, and differences in basic assumptions. - As a general rule, the person outlining the objectives of a system uses one set of criteria while the data processing man subscribes to another. - Definitions tend to be loosely written and freely interpreted. Gaps are filled in on the basis of seemingly logical assumptions which are inevitably interpreted by the parties involved (...). We needed a bridge between the different worlds of programmers and industry specialists so they could exchange concepts with clarity and understanding.”[17] Ende der sechziger Jahre wurde auch eine automatisierte Version von ADS entwickelt.

Ein einflußreicher und stark mathematisch orientierter Ansatz war die Information Algebra, die die Language Structure Group des CODASYL Development Comittees Anfang der sechziger Jahre entwickelte. Ziel war es, ”to extend the concepts of stating the relationships among data to all aspects of data processing. This will require the introduction of increased capability into compilers for translating this type of relational expression into procedural terms. Specifications of such functions as READ, WRITE, OPEN, MOVE, and much of the procedure control definition will be left to the compiler, thereby reducing the work of the system analyst. The analyst will specify the various sets of data and the relationships and rules of association by which these data are manipulated and classed into new and different sets of data, including the desired output.”[18] Durch Anwendung von Konzepten der Algebra sollte ein nichtprozeduraler Sprachansatz entwickelt werden, der als theoretische Grundlage für ”truly automatic programming systems”[19] dienen sollte. Die Information Algebra ging dabei von einem bestimmten Weltverständnis aus, durch das die algebraische Notation zu einem adäquaten Beschreibungsmittel von Wirklichkeit werden konnte: ”An Information system deals with objects and events in the real world that are of interest. These real objects and events, called ’entities’, are represented in the system by

[17] [Lynch 1969] 190f
[18] [CODASYL 1962] S.235
[19] [CODASYL 1962] S.235

data (…) Information about a particular entity is in the form of 'values' which describe quantitatively or qualitatively a set of attributes or 'properties' that have significance in the system (…). Each property has a set of values associated with it …). There is one and only one value associated with each property of each entity".[20]

Der Automatisierungsgedanke, der bei der Information Algebra schon angedacht worden war, sollte durch die Entwicklung der Problem Stating Language (PSL) in die Praxis umgesetzt werden. Ansätze von ADS und der Information Algebra sollten dabei integriert werden. Die Entwicklung von PSL war Teil des ehrgeizigen Projektes Information System Design and Optimizing System (ISDOS), das Ende der sechziger Jahre an der Universität Michigan initiiert wurde. PSL sollte als nicht prozedurale Eingabesprache für ein System dienen, daß aus der Anforderungsspezifikation in PSL automatisch ein entsprechendes Informationssystem generieren sollte. Bei PSL geht es zwar um Kommunikation, aber an die Stelle der Kommunikation zwischen am Entwicklungsprozeß beteiligten Personen wie bei ADS setzt PSL die Kommunikation zwischen Mensch und Rechner, es ist "a language for communicating requirements which are understandable by both management and computer."[21] PSL übernimmt das Entity-Relationship-Modell der Information Algebra und treibt es weiter. Definiert die Information Algebra eine formale Struktur der Wirklichkeit, indem sie die Welt nach diskreten Entitäten mit eindeutigen Eigenschaften strukturiert, so macht PSL weitere Aussagen über die inhaltliche Ausgestaltung der Welt, indem es genau 22 verschiedene Typen von Entitäten und genau 55 verschiedene Typen von Relationen zwischen ihnen festschreibt. Alle Probleme der Informationsverarbeitung sollen durch diese 22 Entitäten und 55 verschiedenen Möglichkeiten, sie zueinander ins Verhältnis zu setzen, darstellbar sein.

6.3 Zur Modellierungsperspektive

Im Folgenden möchte ich zunächst einige Kriterien der RDL auf ihre Relevanz für die Wahl einer bestimmten Modellierungsperspektive untersuchen. Die Frage ist also, ob sich aus dem Vorhandensein oder der Ausgestaltung von bestimmten Kriterien die Tendenz des jeweiligen Ansatzes zur einen oder anderen Modellierungsperspektive ableiten läßt.

Ein wichtiges Unterscheidungskriterium scheint zu sein, inwieweit durch das Repertoire der Sprache und ihre Struktur die Darstellung der Wirklichkeit schon in einer bestimmten Richtung festgelegt ist und dadurch auch nur bestimmte Perspektiven zur Geltung kommen können. Die Strukturierung der Wirklichkeit in

[20] [CODASYL 1962] S.235
[21] [Teichroew 1972] S. 1204

der Information Algebra nach unterscheidbaren Entitäten mit eindeutigen Eigenschaften ist ja beispielsweise nicht naturgegeben, sondern Folge einer bewußten Designentscheidung. Aus einer konstruktivien Perspektive würde z.B. die Eindeutigkeit der Eigenschaften nicht unbedingt Sinn machen, da die Art der Eigenschaft ja von der Sichtweise des ”Betrachters” abhängig sein kann, also durchaus die Möglichkeit mehrdeutiger Eigenschaften besteht. Eindeutigkeit wäre in diesem Fall nichts, was sich von vornherein postulieren ließe, sondern das Ergebnis eines erfolgreichen Verständigungsprozesses. Auch die Entscheidung bei PSL, daß sich die Welt sozusagen aus 22 verschiedenen Objekttypen zusammensetzt, schließt ja alle Sichtweisen der Welt, die sich nicht in diesen 22 Objekttypen fassen lassen, von vornherein aus.

Die verschiedenen Ansätze ließen sich auch danach unterscheiden, inwieweit sie formalisierte Beschreibungselemente benutzen. Mir scheint Formalisierung als Unterscheidungskriterium für eine bestimmte Modellierungsperspektive aber unzureichend. Die unterschiedlichen Sprachansätze gebrauchen Formalisierung aus ganz unterschiedlichen Gründen. Bei PSL werden beispielsweise formalisierte Elemente im Glauben benutzt, durch den Formalismus, also die 22 definierten Objekttypen mit den 55 verschiedenen Relationsarten, eine korrekte Repräsentation der Wirklichkeit zu ermöglichen. Das mit diesem Ansatz entwickelte formale Modell soll dann automatisch, also ohne Rückbezug auf die Bedeutung der einzelnen Elemente und ihrer Beziehungen zueinander, modifizierbar und analysierbar sein. Formalisierung geschieht hier also mit dem Ziel der Automatisierung unter der Voraussetzung, daß das formale Modell ein korrektes Abbild der Wirklichkeit ist. Die Perspektive, unter der hier formale Elemente eingeführt werden, ist also eher eine Abbildperspektive. Bei IPC verfolgt der Einsatz von formalisierten Darstellungselementen, den eindeutig definierten Verben, dagegen das Ziel, durch Eindeutigkeit Mißverständnisse in der Kommunikation zwischen den beteiligten NutzerInnen des Ansatzes zu verringern. Formalisierung zielt hier also nicht auf die Welt und ihre Abbildbarkeit, sondern auf die Bedingungen einer erfolgreichen Verständigung. Dieser Ansatz der Formalisierung ist also durchaus auch mit der Konstruktionsperspektive vereinbar. Formalisierung kann also aus beiden Perspektiven nützlich sein und ist deshalb kein sinnvolles Kriterium, um einen Ansatz mit der Konstruktions- oder der Abbildperspektive zu identifizieren.

Die Art der Formalisierung in den beschriebenen Ansätzen liefert aber Hinweise auf weitere mögliche Unterscheidungskriterien. Formalisierung wird beim eher abbildorientierten PSL-Ansatz zum Zwecke der Automatisierung eingeführt. Bei IPC, das in bezug auf Formalisierung eher konstruktionsorientiert ist, diente Formalisierung dagegen der Kommunikationsunterstützung. Da liegt es nahe, die Eignung von Automatisierung bzw. Kommunikationsunterstützung als Unterscheidungskriterium zu untersuchen.

ADS scheint mir ein Werkzeug zu sein, das zur Unterstützung der Kommunikation im Entwicklungsprozeß entwickelt wurde. Die Art, wie bei ADS die Notwen-

digkeit eines solchen Werkzeuges begründet wird, deutet meiner Meinung nach auf eine Tendenz zur Konstruktionsperspektive hin. Im Vordergrund stehen die unterschiedlichen Sichtweisen und Backgrounds der am Entwicklungsprozeß Beteiligten. Die Macht der Interpretation von Ergebnissen in Abhängigkeit von unterschiedlichen Wertmaßstäben wird deutlich benannt. ADS dient in diesem Zusammenhang nicht dazu, der ”richtigen” Perspektive zum Sieg zu verhelfen, sondern eine ”Brücke zwischen den unterschiedlichen Welten” zu bauen. Im abbildungsorientierten Ansatz von PSL dagegen kommt Kommunikation nur als Befehlsübermittlung zwischen NutzerIn und Computer vor. Dem kommunikativen Prozeß während der Modellierung wird keine Bedeutung zugemessen.

Läßt sich nun daraus schließen, daß die Tatsache, daß in einem Ansatz intersubjektive Kommunikation behandelt wird, ein Hinweis auf eine Orientierung an der Konstruktionsperspektive ist? Nicht ohne weiteres. Wie ich oben beschrieben habe, sollte die Kommunikation zwischen NutzerIn und EntwicklerIn teilweise auch deshalb verbessert werden um die NutzerInnen der Fehlerhaftigkeit ihres eigenen Standpunktes zu überführen und sie vom richtigen Standpunkt der SystementwicklerInnen zu überzeugen. Diese Unterscheidung in richtiger und falscher Standpunkt wäre aber aus der Konstruktionsperspektive nicht zu treffen. Kommunikationsunterstützung erfolgt hier also aus einer eher abbildorientierten Perspektive. Die Tatsache alleine, daß Kommunikationsprozesse eine Rolle spielen, deutet also nicht unbedingt auf die Konstruktionsperspektive hin. Andererseits kann die Konstruktionsperspektive nicht eingenommen werden, ohne auf Kommunikationsprozesse einzugehen.

Kommunikation kann also insofern ein nützliches Unterscheidungskriterium sein, als daß ihre Nichtbehandlung die Abbildperspektive impliziert. Die genauere Untersuchung, in welcher Form Sprache und Kommunikation behandelt werden, läßt Schlüsse auf die Orientierung des Ansatzes an Konstruktions- oder Abbildperspektive zu.

Das Beispiel ADS macht auch deutlich, daß Automatisierung als Kriterium einige Schwierigkeiten aufwirft. ADS als eher ”konstruktiv” orientierter Ansatz wurde nachträglich auch automatisiert und diente in dieser Version als Anregung für den klar abbildorientierten PSL-Ansatz.

Ein wichtiger Aspekt kann auch sein, welche organisatorische Struktur den EntwicklerInnen für den Einsatz ihrer Sprachen vorgeschwebt hat und in welcher organisatorischen Struktur sie selbst gearbeitet haben. Läßt beispielsweise eine hierarchische Kommunikationsstruktur, in der übergeordnete Personen ihre Sichtweise gegenüber untergeordneten durchsetzen können, eine konstruktive Modellierungsperspektive zu, oder impliziert eine solche Struktur zwangsläufig die Abbildperspektive?

Dieser letzte Punkt lenkt die Aufmerksamkeit auch auf eine andere Schwierigkeit, die unterschiedlichen RDL-Ansätze anhand bestimmter Kriterien der einen oder

anderen Perspektive zuzuordnen: Die Intentionen und Orientierungen, die von den EntwicklerInnen in den RDL angelegt waren, bestimmen nicht zwangsläufig die Orientierung, mit der die RDL dann tatsächlich eingesetzt werden. Eine Methode wie ADS, deren EntwicklerInnen eine eher konstruktionsorientierte Sichtweise von Wirklichkeit (zumindest von Kommunikation) vertraten, kann natürlich genauso gut Kommunikationsprozesse unterstützen, in denen eine "richtige" Sichtweise durchgesetzt werden soll. Auch bei der Automatisierung von ADS oder der Übernahme von bestimmten Features in PSL blieb von der Struktur des Ansatzes viel, von seiner Orientierung aber nichts übrig. Keine der Methoden sagt über die organisatorische Struktur, in der die Methoden angewendet werden, etwas aus. Und doch scheint mir der organisatorische Rahmen ein wichtiger Einflußfaktor auf die Wahl einer bestimmten Perspektive zu sein. Dieser Einflußfaktor wird also vollständig durch die Anwendungspraxis, und nicht durch die Intentionen der EntwicklerInnen bestimmt.

Andererseits gibt es auch Kriterien, die die Orientierung auf eine bestimmte Perspektive nahelegen, und die fest in die Struktur der Ansätze eingeschrieben sind. Die Strukturierung von PSL beispielsweise gibt meiner Meinung nach eine bestimmte Struktur der Welt derartig fest vor, daß ein konstruktiver Gebrauch dieses Ansatzes praktisch nicht möglich ist. Das Verhältnis von intendierter und praktisch realisierter Perspektive scheint eine Spezialisierung eines allgemeinen Phänomens der Technik zu sein, das beispielsweise Perrow oder Joerges beschreiben: Einerseits realisieren sich in Technik bestimmte Gebrauchszwecke, die von den EntwicklerInnen intendiert waren und ihren Entwicklungsentscheidungen zugrundelagen. Tatsächlich läßt sich aber mit den intendierten Zwecken nie der tatsächliche Gebrauch von Technik vollständig determinieren. Technik bleibt kontingent und deutbar und die NutzerInnen haben grundsätzlich die Möglichkeit, technische Geräte (oder auch Methoden) entsprechend ihren eigenen Zwecken zu gebrauchen. Auch die Wahl der Perspektive scheint zu einem Teil im Entscheidungsspielraum der NutzerInnen von RDL zu liegen. Zu ihrer Bestimmung muß deshalb auch der jeweilige Gebrauch und nicht nur die Struktur einer RDL untersucht werden.

Kapitel 7

Grundkonzepte der Softwaremodellierung AutorInnen: Melahat Elis, Michael Freitag

7.1 Softwarekrise und Softwaremodellierung

In diesem Kapitel beschäftigen wir uns mit den *Grundkonzepten der Softwaremodellierung*. Hierzu untersuchen wir Texte von drei im Bereich der Informatik sehr bekannten Autoren mit verschiedenen Ansätzen. Dies sind John Backus, David Lorge Parnas und Peter Naur, die wir auch in der angegebenen Reihenfolge betrachten und miteinander vergleichen werden. Naurs Vorstellungen werden wir sehr stark über die Abgrenzung zu Backus und Parnas herausstellen, deren Konzepte für die Softwareentwicklung sehr einflußreich waren, und die Einzug in den Informatik-Mainstream gefunden haben. Naur hingegen hat eine grundlegend andere Herangehensweise an das Thema, als dies im Informatik-Mainstream der Fall ist.

Im Kontext von Softwaremodellierung fällt häufig das Stichwort *Softwarekrise*. Auch John Backus, dessen Modellierungsansatz wir untersuchen wollen, macht die Softwarekrise zum Ausgangspunkt der Begründung für den neu von ihm entwickelten Modellierungsansatz. Deshalb scheint es uns notwendig, kurz darauf einzugehen, wie die Softwarekrise in der Fachliteratur wahrgenommen wird, bevor wir uns mit den Ansätzen zur Softwaremodellierung, die von den drei oben genannten Autoren vorgeschlagen werden, auseinandersetzen.

Bis heute scheint die Softwarekrise nicht überwunden zu sein. So führen beispielsweise [Suhr & Suhr 1993, S. 14] an:

> „Jüngeren Studien zufolge machte die Softwareindustrie in den USA Mitte der 80er Jahre bereits 8% des Bruttosozialproduktes aus. Die erzielten Produktivitätszuwächse reichen nicht aus, um einer jährlichen Steigerungsrate der Anforderungen von etwa 12% nachzukommen. Die Softwarekrise hat Bestand, ihre Symptome (funktionelle Mängel, Termin-, Kosten- und Qualitätsprobleme) sind weitgehend die selben geblieben, aber ihre Qualität hat sich gewandelt.
> Nachdem die Mikroelektronik und Softwareindustrie zu umfangreichen Rationalisierungswellen in den Volkswirtschaften genutzt wurden, muß man sich verstärkt darum bemühen, die Softwarebranche selbst zu rationalisieren. Rechnergestütztes Software Engineering, die Softwarefactory sowie Objektorientierte Methoden und Sprachen sind einige der Ansatzpunkte, mit denen man Herausforderungen der Praxis zu begegnen versucht."

Wir möchten den Begriff Rationalisierung aus dem Zitat aufgreifen und näher betrachten, weil dies unserer Ansicht nach typisch für den Informatik-Mainstream ist; eine detailliertere Begründung wird im Laufe dieser Arbeit geliefert. Rationalisierung des Arbeitsprozesses von EntwicklerInnen und AnwenderInnen hat Bedeutung in zweierlei Hinsicht:

1. Automatisierung dient zur *Unterstützung* des Menschen in seinem Arbeitsprozeß.

2. Durch Automatisierung wird die *Ersetzung* des Menschen im Arbeitsprozeß angestrebt.

Bereits Ende der 60er Jahre gab es kritische Meinungen[1], u.a. auf der ersten Konferenz für Software-Engineering 1968, aus den Reihen von Managern, die das Verhalten von EntwicklerInnen gegenüber AnwenderInnen kritisierten, da sich EntwicklerInnen über den möglichen Gebrauch von informatischen Artefakten keine Gedanken machten und es zu Kommunikationsproblemen zwischen diesen kam. Dies wird auch in [Pasch und Biskup 1995, S. 85] so dargestellt, daß

> „... schon auf der Geburtskonferenz ... des neuen Fachgebiets Software-Engineering 1968 in Garmisch ... ein beträchtlicher Teil der Teilnehmer die Kommunikation der Software-Entwickler untereinander als die eigentliche Ursache der sogenannten Software-Krise bezeichnet hat. Folgerichtig wurde die Software-Krise von diesen schon damals für eine *Krise des Managements* gehalten und nicht für eine Krise des Standes der Kunst der Technik. Allerdings haben seitdem die Technokraten an den Universitäten fast immer nur in Letzteres investiert und so die Kluft zwischen Wissenschaft und Praxis erheblich erweitert."

[1] Vgl. Florian Theissing in Teil 3 dieses Bandes.

Auf die Forderung, den Entwicklungsprozeß partizipativ zu gestalten, wurde nicht eingegangen, stattdessen wurde als Lösungsstrategie die Automatisierung angestrebt. Es ging um Automatisierung der Programmierung durch Steigerung der Effizienz der ProgrammiererInnen sowie um die vollständige Ersetzung der Arbeitskraft der ProgrammiererInnen durch automatische Systementwicklung. Gründe dieser Entwicklung sind in der Tatsache zu sehen, daß relativ schnell konkrete Lösungsansätze und Produkte für die Automatisierung zur Verfügung standen, während bis zum heutigen Tag partizipative Lösungsansätze wie z.B. das Modell PETS[2] in der Praxis wenig Beachtung finden. Solange sich diesbezüglich nichts bewegt, wird sich die Industrie und damit der Informatik-Mainstream ausschließlich auf die Weiterentwicklung der Automatisierung im Sinne der Ersetzung des Menschen konzentrieren.

Angestrebtes Ziel dieses Kapitels ist die mögliche Einordnung der Autoren in die Konstruktions- oder Abbildperspektive[3]. Darüber hinaus wollen wir versuchen die beiden folgenden von uns aufgestellten Thesen zu belegen bzw. näher zu beleuchten:

1. Der Informatik-Mainstream ist abbildorientiert.

2. Die Konstruktionsperspektive ist die angemessene Perspektive zur Modellierung in der Informatik.

Untersuchungsfragen

Bei der Untersuchung stellen sich uns folgende Fragen:

Welchen Problembereich betrachten die Autoren: z.B. den gesamten Software-Entwicklungsprozeß, Teile des Prozesses, alle beteiligten Personen oder nur EntwicklerIn?

Von welchem Ansatz (Voraussetzungen) gehen sie dabei aus? Was genau erfassen sie bzw. wollen sie erfassen: z.B. vollständige Spezifikation, Unmündigkeit der AnwenderInnen, Konstruktion einer gemeinsamen Realität?

Begründen die Autoren ihr Vorgehen? Wie begründen sie es? Welche Konzepte zur Lösung der Softwarekrise nennen sie: z.B. nicht ingenieursmäßig, nicht rational?

Welche Personen(gruppen) sind beteiligt: z.B. EntwicklerIn, AnwenderIn?

[2] S. Irina Leyde in Teil 3 dieses Bandes.
[3] Siehe Gernot Grube in Teil 1 dieses Bandes.

7.2 John Backus

Einleitung

Backus setzt in [Backus 1985] beim Stichwort ”Softwarekrise” ein, die seiner Ansicht nach aus den schlechten Konzepten des Programmierens resultiert. Diese machten Programmierung zu einer Kunst statt zu einer Ingenieursdisziplin. Solch eine Disziplin würde wenigstens fordern, daß es einen Bestand von nützlichen Programmen und eine Menge von Standardtheoremen gäbe, die wiederholt angewandt werden könnten.

Backus' Vorstellung von Programmierung, die wir im folgenden als sein Vorgehensmodell bezeichnen, sieht folgendermaßen aus:
Es gibt eine Menge von Standardtheoremen, mit Hilfe derer ein gegebenes Problem gelöst wird. Dieses Problem ist aufgrund der verwendeten Standardtheoreme korrekt gelöst, d.h. es ist somit Bestandteil der Menge von Standardkonstruktionen. Korrespondierend zu seiner mathematischen Vorgehensweise übernimmt er einen Grundstamm an Theoremen aus der Mathematik.

Sein Vorgehensmodell berücksichtigt ausschließlich die Programmierung.
Er nimmt eine bestimmte Sicht auf die Programmierung ein, die er als ”function level view” bezeichnet. Diese besagt, daß Programme als Funktionen darstellbar sind, wodurch die Programmierung zu einem mathematischen System wird.

Mathematische Systeme sind laut Backus definiert durch eine Menge von Operationen, die einfach zu verstehende Ergebnisse liefern und einer Menge von strengen algebraischen Regeln Folge leisten.

Modellierungsverständnis

In bezug auf die Herstellung informatischer Artefakte existiert in Backus' Vorgehensmodell *die Formulierungsphase*[4] nicht. Daraus schließen wir, daß er diese als gegeben voraussetzt. Sein Vorgehensmodell beginnt erst mit der *Formalisierungsphase*[5]. Es finden dann *die Transformation formaler Modelle*[6] und *die Ausführung formaler Modelle auf Computern*[7] statt. Wir sehen bei diesen

[4] Diese Phase beinhaltet das Herstellen eines ersten Modells, d.h. die Formulierung des Problems und/oder Gegenstandsbereichs. Die Formulierungsmittel sind hier nicht unbedingt formal, es werden z.B. die natürliche Sprache oder graphische Repräsentationsmittel verwendet.

[5] Es geht hier um die Erstellung von Modellen in einem formalen Repräsentationsformat aus nicht formalisierten Modellen. Die Mittel der Formalisierung sind formale Sprachen.

[6] Die Transformation von formalen Modellen in andere formale Modelle steht im Vordergrund. Ziel ist hier die Herstellung eines effizient ausführbaren operationalen Modells. Diese Transformationen können, müssen aber nicht, selbst nach formalen Regeln ablaufen.

[7] In dieser Phase werden operationale Modelle auf Computern ausgeführt und von Menschen genutzt.

Vorgängen nicht zwangsläufig eine Sequentialität, die es aber bei Backus' Vorgehensmodell gibt. Aufgrund der Sequentialität sind Wechselwirkungen zwischen den Vorgängen nicht vorgesehen.

Durch die Verwendung der korrekten Standardtheoreme in der funktionalen Programmierung entstehen wiederum informatische Artefakte. Backus scheint ähnlich wie Bacon [8] davon auszugehen, daß ein solches Vorgehen zu per se nützlichen Artefakten bzw. Programmen führt. Diese Art der Erstellung ist nach Backus' Ansicht ein ingenieursmäßiges Vorgehen. Daraus folgt für ihn, daß die Disziplin der Informatik eine Ingenieursdisziplin ist. Aus diesem Grund treten nur InformatikerInnen in seinem Vorgehensmodell auf, die als AnwenderInnen seines Vorgehensmodells die "function level view" einnehmen. D.h. die InformatikerInnen führen die oben aufgeführten Phasen durch.

Interessant ist unserer Ansicht nach die Frage, ob Backus die „Schwächen" des Menschen (sowohl EntwicklerIn als auch AnwenderIn) in bezug auf die Modellierung „erkennt", so wie dies bei Parnas der Fall ist (siehe 7.3). Backus zählt nicht die „Schwächen" der EntwicklerInnen und der AnwenderInnen auf, aber daß er dazu doch eine Ansicht vertritt, ist sichtbar. Er strebt unserer Ansicht nach die Automatisierung im Sinne der Ersetzung des Menschen im Arbeitsprozeß (s. 7.1) an. Die AnwenderInnen werden eliminiert, da erstens bei der Herstellung informatischer Artefakte die *Formulierungsphase*, in dem die AnwenderInen sehr stark partizipativ teilnehmen könnten, bei ihm nicht existiert bzw. nicht angesprochen wird und zweitens eine Spezifikation gleich auf abstrakter Ebene in der *Formalisierungsphase* „entsteht". Durch das Vorhandensein von Standardtheoremen wird eine Art Automatisierung angedacht, wodurch die Transformationsphase formaler Modelle quasi zu einem maschinellen Akt wird, dem sich die EntwicklerInnen unterzuordnen haben und wo nicht menschliche Fähigkeiten wie Kreativität oder Intuition, wie sie etwa Naur herausstellt (s. 7.4), erforderlich sind., Hierdurch wird im Grunde die Eliminierung der EntwicklerInnen im Arbeitsprozeß anvisiert.

Backus stellt in bezug auf Programme folgende Kriterien auf:

- *Korrektheit* bezüglich der Spezifikation.
  Dieses Kriterium besagt, daß ein Modell, das Programm, ein Orginal, die Spezifikation, korrekt abbildet. Dazu müssen die Kriterien Unabhängigkeit und Vergleichbarkeit[9] zwischen Modell und Orginal erfüllt sein. Das Kriterium Korrektheit impliziert ein weiteres Bewertungskriterium: *Vollständigkeit*.

- *Wiederverwendbarkeit* von Programmen und Programmteilen.
  Darunter ist ein mehrmaliges Benutzen von korrekten und vollständig er-

[8] Vgl. Martin Fischer in Teil 2 dieses Bandes.
[9] Siehe Gernot Grube in Teil 1 dieses Bandes.

stellten Modellen zu verstehen, um ein gleiches Original wieder abzubilden oder einen Teil eines Orginals abzubilden, wobei das wiederverwendbare Modell ebenfalls nur ein Teil eines Größeren ist.

- *Optimierung* der Programmerstellung.
  Es soll erreicht werden, daß die Programmerstellung schnell und fehlerfrei ablaufen kann mit Hilfe der Bewertungskriterien *Korrektheit* und *Wiederverwendbarkeit*. Dieses Kriterium weist eine Tendenz zur Automatisierung, d.h. eine maschinelle, menschenunabhängige Programmerstellung, auf, die wir Backus unterstellen.

Die funktionale Programmierung ist für ihn **die** Möglichkeit, um dies alles zu erreichen.

Aufgrund der von Backus an Programme gestellten Kriterien läßt sich sein Ansatz der Abbildperspektive zuordnen, was wir im folgenden genauer begründen:
Er konzentriert sich auf die korrekte Abbildung einer Spezifikation auf ein Programm. Diese Korrektheit der Abbildug wird durch Wiederverwendung bzw. Zusammenstellung bereits existierender korrekter Programme und Programmfragmente erreicht. Diese inhärente Korrektheit resultiert aus der Anwendung von als korrekt bewiesenen Standardtheoremen. Dadurch soll der Programmiervorgang rationalisiert werden, so daß diese Tätigkeit automatisch erfolgt. In seiner Vorstellung des Programmierens berücksichtigt er nicht die besonderen Fähigkeiten des Menschen wie bspw. Intuition (siehe 7.4).

7.3 David L. Parnas

Einleitung

Nach Parnas in [Parnas 1985] suchen InformatikerInnen nach dem idealen Softwareentwicklungsprozeß, d.h. einem Prozeß, in dem Programme aus Spezifikationen abgeleitet werden, wie Lemmata und Theoreme von Axiomen in veröffentlichten Beweisen.

Parnas beschreibt in seinem Papier solch einen idealen Softwareprozeß anhand einer Reihe von Dokumenten, welche im Laufe der Softwareentwicklung erstellt werden sollen, und den Zweck, dem die Dokumente dienen. So sollen sie als Basis für ein vorbereitendes Design-Review, als Referenz während der Kodierphase und als Anleitung der Pflege- oder WartungsprogrammiererInnen dienen. Er stellt weiter vor, wie diese Dokumente erstellt werden können, mit Hilfe derselben Prinzipien, welche das Softwaredesign leitet. Diese Dokumentation stellt weit mehr dar, als die Dokumente, die üblicherweise parallel zum Entwicklungsprozeß erstellt werden. Diese Dokumentation, wenn sie auf dem aktuellen Stand

gehalten wird, stellt einen rationalen Software-Entwicklungsprozeß dar. Daraus folgt für Parnas, daß die Arbeitsweise der InformatikerInnen identisch ist mit anderen Ingeniuersdisziplinen. Ingenieursmäßiges Arbeiten beinhaltet für ihn eine solide wissenschaftliche Basis, welche aus den Grundlagen der Informatik und Mathematik bestehen, eine hervorragende und geübte Problemerkennung und Lösungsfähigkeit und außerdem, sich der Verantwortung der Tätigkeit voll bewußt zu sein, um nützliche Artefakte korrekt zu erstellen. Diese Art zu arbeiten, setzt natürlich eine entsprechende Ausbildung voraus, welche dahingend anzupassen sei.[10]

Allerdings gibt Parnas in seinem Papier auch die Gründe an, weshalb dieser ideale Softwareentwicklungsprozeß nicht erreichbar ist. Deswegen stellt er einen tatsächlich realisierbaren Softwareentwicklungsprozeß dar, im folgenden kurz Realmodell genannt, in dem die Dokumentation den zentralen Stellenwert einnimmt.

Modellierungsverständnis

Mit Parnas' Idealmodell soll ein rationaler systematischer Weg zur Entwicklung von Software erfolgen. Die Methode dabei kann als „top down" eingeordnet werden. Als Teil des Idealmodells ist anzusehen, daß aus den vorher aufgestellten Anforderungen das Programm entsteht. Im Idealmodell sind Testen und Reviewen nicht erforderlich, da keine Fehler auftreten können, die behoben werden müßten.

Ein Idealmodell wird laut Parnas benötigt:

- um Softwareentwicklung auf rationale Weise perfekt zu erstellen
- da EntwicklerInnen eine Führung benötigen, die ihnen mit dem Idealmodell zur Verfügung steht.
- um Aussagen über den tatsächlich zu realisierenden Softwareprozeß und informatische Artefakte durch Heranziehen der Dokumentation des Idealmodells als Meßgrundlage machen zu können.
- um den zu realisierenden Softwareentwicklungsprozeß und informatische Artefakte so nah wie möglich an das Idealmodell heranzuführen.

Allerdings ist Parnas der Ansicht, daß das Idealmodell aus folgenden Gründen nicht realisierbar ist:

- Auftraggeber/AnwenderInnen wissen nicht, was sie wollen und brauchen. Deshalb sind sie unfähig, präzise Anforderungen zu stellen.

[10] Siehe [Parnas 1990]

- Selbst wenn die Anforderungen bekannt wären, gibt es aber andere Fakten, die zur Entwicklung von Software benötigt werden. Ein Backtrack zwischen den Phasen in seinem Idealmodell ist erforderlich, da z.B. durch Bekanntwerden neuer Details während der Implementierung das Design überarbeitet werden muß.

- Auch wenn alle relevanten Fakten vor dem Beginn der Softwareentwicklung bekannt sind, kann aufgrund der möglichen Nicht-Bewältigung der Fülle an Fakten, weshalb abstrahiert wird, keine korrekte Software erstellt werden.

- Selbst wenn alle benötigten Details zu bewältigen wären, sind Projekte noch von externen Einflüssen abhängig.

- Menschlich bedingte Fehler sind nur dann vermeidbar, wenn der Einsatz von Menschen vermieden wird.

Er hält zwar die Realisierung des Idealmodells nicht für durchführbar, aber durch die Dokumentation von dem, was aus dem Idealmodell entstanden wäre, ist eine Art Vergleichsmodell für EntwicklerInnen bei der Erstellung des Realmodells gegeben, wodurch weniger Fehler während der Entwicklung und im informatischen Artefakt entstehen.

Parnas spricht auch noch von einer weiteren Dokumentation, und zwar handelt es sich dabei um die für das Realmodell. Eine Dokumentation, die im nachhinein, d.h. nach der Erstellung der Software erstellt wird, ist immer eine schlechte Dokumentation. Er betrachtet die Dokumentation als ein Hauptprodukt eines Softwareprojekts.
Eine Dokumentation von Software im Sinne von Parnas dient dazu, neuen EntwicklerInnen, die neu in die Herstellung des informatischen Artefakts einsteigen, den aktuellen Stand der Dinge zu vermitteln und getroffene Designentscheidungen anhand gegebener Anforderungen nachvollziehbar zu machen.
Das zentrale Dokument ist das Anforderungsdokument, das vor Beginn der Analysephase im Softwareentwicklungsprozeß zu erstellen ist. Dessen Kernstück ist eine Menge von mathematischen Funktionen in tabellarischer Form. Jede Funktion spezifiziert den Wert eines einzelnen Outputs als eine Funktion von externen Zustandsvariablen, die für die Applikation relevant sind.

Parnas' Herstellung informatischer Artefakte ist streng sequentiell. Sie sieht folgendermaßen aus:

- Es erfolgt die *Formulierung* von Anforderungen an das zu erstellende informatische Artefakt.

- Dann findet die *Formalisierung* dieser Anforderungen als mathematische Funktionen statt.

- Zum Schluß werden die *formalen Modelle* (sprich: mathematische Funktionen) *ausgeführt*, d.h. eine Validierung des Designs findet statt.

An *Personengruppen* werden sowohl die InformatikerInnen/ProgrammiererInnen – sehr ausführlich – als auch die AnwenderInnen – nur knapp – erwähnt. Parnas nennt zwei Gründe für das nicht weitere Eingehen auf die AnwenderInnen:
Erstens sind die NutzerInnen unfähig zu sagen, was sie wollen.
Zweitens werden in der Disziplin Informatik per se nützliche Artefakte erstellt, d.h. die Einbeziehung der AnwenderInnen scheint ihm deshalb nicht notwendig zu sein.

Bezüglich der *Realisierbarkeit* seiner Lösungsvorschläge räumt Parnas ein, daß es sicherlich Entscheidungen gibt, die getroffen werden, „weil es funktioniert", jedoch sollte dies dann kenntlich gemacht werden, damit es nicht so aussieht, als gäbe es dafür einen tiefen und philosophischen Grund. Parnas sieht zwar das Problem, geht aber nicht weiter darauf ein. Mit dem Problem, das er anspricht, umschreibt er eigentlich etwas wie Intuition, ohne es als solches zu bezeichnen. Er beschäftigt sich aber nicht weiter damit, da es nicht in sein Konzept der Rationalität paßt.

Parnas stellt folgende *Kriterien*, die sich auf den Modellierungsgegenstand beziehen, auf:

- *Vollständigkeit* der aufgestellten Anforderungen.
  Darunter ist eine Abbildung eines Originals zu verstehen, die die wesentlichen Merkmale des Originals beinhaltet, um selbst als Original für den nachfolgenden Erstellungsprozeß eines informatischen Artefaktes zu dienen.

- *Konsistenz* in bezug auf die zu Beginn des Softwareentwicklungsprozesses aufgestellten Anforderungen innerhalb des weiteren Verlaufes des Entwicklungsprozesses und des zu erstellenden Artefaktes.
  Konsistenz bedeutet eine korrekte Abbildung einzelner Anforderungen innerhalb verschiedener erstellter Dokumente des Entwicklungsprozesses sowie eine korrekte und vollständige Abbildung der Anforderungen auf ein informatisches Artefakt.

- *Effizienz* bei der Erstellung von informatischen Artefakten.
  Der Modellierungsvorgang von der *Formulierung* bis zur *Ausführung formaler Modelle* sollte streng sequentiell sein. Dadurch ist die Möglichkeit der Automatisierung des Vorganges gegeben.

- *Modularität* und *Redundanzfreiheit* von Anforderungen.
  Anforderungen sind zu formalisieren, wodurch es auch möglich ist, die Anforderungen menschenunabhängig zu validieren und zu transformieren.

- *Nachvollziehbarkeit (Rationalität)* von getroffenen Designentscheidungen des Erstellungsprozesses von informatischen Artefakten.
  Damit soll eine Objektivität gewahrt werden, die es erleichtern soll, die Abbildung eines informatischen Artefaktes vom Orginal erklärbar zu machen.

Diese Kriterien zeigen, daß Parnas Vertreter der Abbildperspektive ist. Außerdem strebt Parnas wie bereits oben beschrieben ein Idealmodell an, wodurch eine Konzentration auf Formalisierung und Ausführung formaler Modelle auf dem Computer erzielt werden soll. Aus den bereits genannten Gründen, warum das Idealmodell nicht realisierbar ist, folgt, daß der Mensch zwar nicht ausgeschlossen werden kann, jedoch zumindest der Weg der Entscheidungsfindung rational sein soll. Deshalb versucht Parnas, menschliche Fähigkeiten, insbesondere Intuition, die er als die eigentliche Fehlerquelle ansieht, auszuschließen.

7.4 Peter Naur

Einleitung

Naur hat sich in[Naur 1985] mit dem Faktor Mensch, der bei den beiden anderen Autoren als Risikofaktor betrachtet wird und deshalb auch weitestgehend eleminiert werden sollte, beschäftigt. Eine Eleminierung des Menschen selbst ist zum jetzigen Stand der Technik im Softwareentwicklungsprozeß nicht möglich. Naur untersucht dabei im besonderen die menschliche Intuition und ihren Einfluß auf den Softwareentwicklungsprozeß im speziellen.

Definition von Intuition: Die Intuition eines Menschen umfaßt seine Erfahrungen, sein Wissen und seine Erinnerungen. Das permanente zur Verfügungstellen der geistigen Fähigkeiten wird als Intuition bezeichnet. Diese Intuition steht in einer Abhängigkeit zu der Sprache, Theorien und der Sicht auf die Welt von einem Menschen. Sie wird während des gesamten Lebens eines Menschen permanent erweitert und modifiziert.

Die menschliche Intuition ist permanent vorhanden und spielt in jeglichen menschlichen Handlungen eine Rolle. So z.B. beim Tragen einer Tasse Kaffee, beim Reden mit anderen Menschen, beim Arbeiten oder Lösen von Problemen. Die menschliche Intuition ist dabei nicht einfach abschaltbar. Deshalb wird auch ihre Bedeutung im Softwareentwicklungsprozeß von Naur untersucht.

Modellierungsverständnis

Ein Modell bezüglich der Softwareentwicklung, wie bei den Autoren Backus und Parnas dargestellt, existiert bei Naur nicht. Stattdessen stellt er Dinge in den Vordergrund, die in den vorhandenen Modellen bewußt oder unbewußt nicht berücksichtigt werden.

Sein Ansatz beginnt bei der Vorstellung, daß es möglich sei, eine eindeutige Abbildung zwischen Modell und der *Transformation des Modells* zu erhalten. Naur vertritt die Auffassung, daß diese nie zu erhalten ist, weil eine korrekte und vollständige Transformation von einer Spezifikation zu einem Programm bei menschlicher Beteiligung nicht realisierbar ist. Andere Autoren, wie Backus und Parnas, nehmen dieses zum Anlaß, den Menschen, mit seiner offenkundigen Inkompetenz, durch eine Automatisierung zu ersetzen. Naur hingegen hält den Menschen nicht für inkompetent, sondern stellt die abweichende Arbeitsweise des Menschen gegenüber einer Maschine als Stärke heraus. Die Intuition des Menschen findet ebenfalls bei der Transformation der Modelle Anwendung und stellt somit in diesem Punkt den Unterschied zu den anderen Autoren dar. Denn Backus und Parnas würden sicherlich den Einfluß von Intuition bei der *Formulierung* und *Formalisierung* nicht abstreiten, bei der Transformation dagegen sicher.

Die Annahme, wie sie Backus bei seinem Vorgehensmodell macht, daß eine adäquate Spezifikation per se vorhanden sei, bzw. laut Parnas, daß eine adäquate und vollständige Spezifikation zu erstellen sei, stellt Naur in Frage. Seine Behauptung geht dahin, daß es aufgrund der unterschiedlichen Sichten der *beteiligten Personengruppen*, sprich AnwenderIn und EntwicklerIn, keine solche Spezifikation gibt. Stattdessen muß ein Konsens zwischen den Personengruppen gefunden werden, wobei durch Kommunikation zwischen AnwenderIn und EntwicklerIn eine neue gemeinsame Sicht auf den zu modellierenden Ausschnitt der Realität entsteht. Es steht dabei außer Frage, daß jederzeit auf neue Anforderungen reagiert werden kann. Daraus ergibt sich, daß bei der Herstellung informatischer Artefakte keine sequentielle Abarbeitung der Phasen folgt, wie es bei Backus und Parnas der Fall ist.

Bezüglich der *Realisierbarkeit* seines Modells braucht sich Naur keine Gedanken zu machen, weil dies ohnehin in der Praxis funktioniert. Desweiteren ist dieser Ansatz auf etablierte oder oben dargestellte Modelle anwendbar.

Naur stellt keine Bewertungskriterien auf wie Backus und Parnas, da sein Fokus auf Methoden der Sofwareerstellung gerichtet ist, im Gegensatz zu den informatischen Artefakten, die Backus und Parnas betrachten.

Das Arbeiten von EntwicklerInnen nach *Methoden* und Regeln, um dadurch eine hohe Qualität bei der Softwareentwicklung zu erhalten, kann nicht erreicht werden, weil der Mensch selbst nicht wie eine Maschine arbeiten kann.

Naur teilt eine Methode in drei wesentliche unterstützende Bereiche ein:

- **Activity**
  Regeln, welche beschreiben, was getan und was hergestellt werden soll.

- **Forms of expression**
  Spezielle Begriffe und Sprachen, welche in der Softwareentwicklung benutzt werden sollen.

- **Ordering of activities**
  Angabe der Reihenfolge von Aktivitäten.

Die wesentliche Frage, die bei der Anwendung einer Methode gestellt werden muß, ist die Frage nach der Verbesserung des Entwicklungsprozesses.

Es gibt zwei wesentliche Arten von Mängeln, die bei der Anwendung einer Methode auftreten. Dies sind zum einen **Auslassungen** und **begangene Fehler**. Der Bereich der **Activity** einer Methode, der sich auf die auftretenden Mängel bezieht, wird von Naur als positiv erachtet, da dieser Bereich der **Activity** den Prozeß mit einer Aktivitätenliste und der Kontrolle der geleisteten Arbeiten unterstützt. Das Vorschreiben einer Reihenfolge von Aktivitäten durch eine Methode ist dagegen nicht sinnvoll. Der dritte Bereich **Forms of expression**, welcher durch eine Methode unterstützt werden soll, soll das Verursachen von Fehlern durch das Verwenden von Formalisierungen vermeiden helfen, wie bei Backus und auch Parnas erwähnt. Das Problem liegt nach Naurs Ansicht nicht in der Umsetzung einer Problemlösung in eine Sprache, sondern vielmehr in dem Verstehen des Problemes, das gelöst werden soll. Dazu ist intuitives Wissen um den Problembereich notwendig. Diese Sicht steht im ganz klaren Widerspruch zu Backus und Parnas.

Es zeigt sich, daß Naur ganz eindeutig kein Vertreter der Abbildperspektive ist, da er die dort im Vordergrund stehenden Bestrebungen nach Formalisierung und Automatisierung nicht als die Ansätze zur Lösung der Probleme bei der Softwareerstellung auffaßt, sondern den Menschen mit seiner Intuition in den Vordergrund stellt. Die Partizipation der AnwenderIn bei der Softwareerstellung ist für ihn eine Selbstverständlichkeit, was in der Abbildperspektive dagegen eher ausgeschlossen wird. Zudem gibt es in den Phasen zur Herstellung informatischer Artefakte statt der in der Abbildperspektive gewünschten Sequentialität (siehe Backus und Parnas) Wechselwirkungen zwischen den Phasen.

7.5 Einordnung der Autoren in die Perspektiven und Bewertung

Die Herangehensweise, um die Einordnung zu erreichen, sieht folgendermaßen aus:

Zuerst stellen wir die Schwächen der Abbildperspektive bzw. des Informatik-Mainstreams heraus, wodurch die Stärken der Konstruktionsperspektive sichtbar werden.
Dann erläutern wir die 2. These, daß die Konstruktionsperspektive die angemessene Perspektive zur Modellierung in der Informatik ist.

Schwächen der Abbildperspektive bzw. des Informatik-Mainstreams

Durch die Einordnung der Autoren Backus und Parnas in die Abbildperspektive schließen wir, daß, da beide Konzepte für die Softwareentwicklung sehr einflußreich waren und sie Einzug in den Informatik-Mainstream gefunden haben, der Informatik-Mainstream auch abbildorientiert ist.
In diesem Abschnitt stellen wir nun die Schwächen des Informatik-Mainstreams bzw. der Abbildperspektive vor.

Wie bei Backus beschrieben, soll eine Transformation aus der Spezifikation in einen programmiersprachlichen Code stattfinden. Der Mensch, dem unterstellt wird, daß er diesen Prozeß nicht fehlerfrei ausführen kann, soll durch eine Menge von Standardtheoremen (korrektem programmiersprachlichen Code) dazu angeleitet werden, die Transformation nach einem bestimmten vorgegebenen Schema durchzuführen. Die EntwicklerInnen werden dabei dem Transformationsschema untergeordnet und nehmen damit nur die Rolle einer ausführende Maschine ein. Parnas selbst hat in seinen Ausführungen festgestellt, daß der Mensch nicht wie eine Maschine arbeiten kann, und Naur liefert mit der Intuition des Menschen die Begründung hierfür. Desweiteren stellt Naur dar, daß die Transformation durch den Menschen nicht ohne seine Intuition ablaufen kann, und diese sogar die wesentliche Rolle dabei spielt. Dadurch stellt er selbst die Realisierung von Parnas' Forderung in Frage, wonach eine Formalisierung automatisch in eine weitere Formalisierung transformierbar sei. Dies ist ebenfalls Backus' Ziel, wofür er eine theoretische Grundlage vorstellt.

Backus selbst geht in seinem Text davon aus, daß das wesentliche Problem bei der Softwareentwicklung in der Transformation der Spezifikation in ein Programm liegt. Doch mittlerweile hat sich die Erkenntnis über die Ursache des Problems erweitert.

Die Sequentialität der Phasen bedingt, daß nicht schnell und flexibel auf neu gestellte Anforderungen reagiert werden kann, so ist die Konformität bezüglich der Änderungen der Anforderungen nicht gewährleistet. Die Änderbarkeit von informatischen Artefakten scheint nicht vorgesehen zu sein.

Die Nützlichkeit eines informatischen Artefakts wird vor allem aus Sicht der EntwicklerInnen bewertet und es geht dabei ausschließlich um die Realisierbarkeit einer Funktionalität.

Es findet keine bzw. kaum Kommunikation zwischen EntwicklerIn und AnwenderIn statt. Die Vorstellungen der EntwicklerInnen vom Original werden abgebildet, wodurch die EntwicklerInnen den AnwenderInnen eine neue Realität quasi aufoktroieren. Die EntwicklerInnen haben die absolute Kontrolle, was dazu verleitet, menschliche Fähigkeiten so zu ersetzen, daß die Tätigkeit Ausführenden „überflüssig" werden.

Die Konstruktionsperspektive als die geeignetere Perspektive

Aus den aufgezeigten Schwächen des Informatik-Mainstreams bzw. der Abbildperspektive stellt sich die Frage, ob die Konstruktionsperspektive geeigneter ist. Wir greifen deshalb auf die von uns in Kap. 7.1 aufgestellte 2. These wieder auf:

> *Die Konstruktionsperspektive ist die angemessene Perspektive zur Modellierung in der Informatik.*

Eines der wesentlichen Merkmale der Konstruktionsperspektive ist das nicht Vorhandensein von Korrektheit. Es wird hier durch den Begriff Aufgabenangemessenheit ersetzt. Dieses Merkmal führt zu Softwareprodukten, die den Anforderungen beider beteiligten Gruppen (AnwenderInnen und EntwicklerInnen) besser genügen. Hier zeigt sich der Unterschied zur Abbildperspektive, da die Anforderungen gemeinsam von AnwenderIn und EntwicklerIn in einem kommunikativen Prozeß aufgestellt werden. Dazu ist es innerhalb dieses Prozesses von Nöten ein gemeinsames Modell zu erstellen. Dieses Modell ist keine Abstraktion der Wirklichkeit, sondern eine Abstraktion einer neu zu erschaffenden Realität. Diese Realität beruht auf den unterschiedlichen Sichten der beteiligten Personen. Bei Naur werden die Sichten der Menschen mit Intuition bezeichnet. Es kommt innerhalb der Konstruktionsperspektive darauf an, den Kommunikationsprozeß wesentlich stärker zu betrachten und die dabei resultierenden Probleme nicht hinter technischen Lösungen zu verstecken. Dies ist unserer Meinung nach eines der Ursachen der Softwarekrise, daß die Maschine zum Hauptgegenstand des Prozesses der Softwareentwicklung geworden ist oder als dieser gesehen wird.

Durch das Abstrahieren bzw. Modellieren einer neu zu schaffenden Realität hat der Begriff der Wahrheit aus der Abbildperspektive hier seine Relevanz verloren. Daraus ergibt sich, daß eine Transformation eines Modells nicht hundertprozentig zu erfolgen hat und damit am besten zu automatisieren ist.

Weitere Probleme, die sich in der Abbildperspektive nicht stellen, sich aber durch die Modellierung einer Realität ergeben, sind einerseits die Vollständigkeit und zum anderen die Nützlichkeit der Anforderungen und damit auch des Modells sowie das daraus zu entwickelnde informatische Artefakt. Diese Fragen werden selten bzw. gar nicht in der Abbildperspektive gestellt, aufgrund des Modellierungsverständnisses, das ein zu abstrahierendes Original zum Gegenstand

hat. Deshalb wird davon ausgegangen, daß die EntwicklerInnen, und nur die EntwicklerInnen, in der Lage sind, Anforderungen vollständig aufzustellen, und diese dann auch meistens die Nützlichen sind. Eine Änderung ist hier nur zu Beginn möglich. Diese Fragen müssen jedoch immer wieder während des Entwicklungsprozesses gestellt und die daraus resultierenden Änderungen berücksichtigt werden. Denn es werden durch die Schaffung neuer Realitäten neue oder sich verändernde Anforderungen an informatische Artefakte entstehen. Die daraus resultierenden neuen oder sich veränderten Sichten gilt es innerhalb der zu modellierenden Realität zu berücksichtigen und das Modell anzupassen. Werden diese nicht berücksichtigt, geht die Nützlichkeit eines Artefaktes verloren und das schon vor Fertigstellung.

Änderbarkeit der Anforderung ist damit ein Merkmal, das von der Konstruktionsperspektive unterstützt wird. Es ist dynamisch in die Perspektive mit eingebunden aufgrund des Merkmals der Aufgabenangemessenheit.

Naur ist mit seiner Theorie der Intuition und der damit verbundenen Sichten auf die Welt, sowie der Herausstellung der Notwendigkeit von Kommunikation zwischen AnwenderIn und EntwicklerIn während des Entwicklungsprozesses der Kontruktionsperspektive zuzuordnen.

Ausblick und Kritik

Abschließend möchten wir noch eine eigene Kritik bezüglich unserer Untersuchung anführen. Wir stellen den Informatik-Mainstream anhand der Untersuchung von zwei Autoren als abbildorientiert dar. Einerseits läßt sich der Informatik-Mainstream nicht auf die Positionen zweier Autoren reduzieren, andererseits glauben wir, daß die Autoren Backus und Parnas repräsentativ sind, da sie führende Vertreter der Diszilpin Informatik sind.

Aus unseren und den anderen Kapiteln erhaltenen Ergebnissen sollten sich unserer Meinung nach Forderungen und Konsequenzen ergeben, vor allem im Bereich der Informatik-Ausbildung. Die Informatik-Ausbildung ist unserer Meinung nach ein Punkt, an dem angesetzt werden kann, die Modellierungs-Perspektive der Informatik von der bis heute überwiegenden Abbild- auf die unserer Ansicht nach angemessenere Konstruktionssicht umzusetzen.

Kapitel 8

Partizipative Entwicklung von Softwaresystemen - PETS
Autorin: Irina Leyde

Das erste Projektmodell zur Kontrolle von Softwareentwicklung war das Phasenmodell[1], das in dem Versuch entstand, Konzepte aus dem Bereich industrieller Fertigung auf die Entwicklung von Software zu übertragen. Erst lange nach der Entwicklung des „klassischen“ Phasenmodells wurden weitere Koordinationskonzepte für Softwareentwicklungsprojekte entwickelt, die einerseits dem Fortschritt der Technik Rechnung tragen sollten und andererseits auf Kritik an den klassischen produktorientierten Methoden basieren. Während das Phasenmodell den Anspruch erhebt, für jede Art von Softwareentwicklungsprojekt anwendbar zu sein, beschäftigt sich PETS[2] hauptsächlich mit Projekten, „mit denen Innovationsziele in Arbeitszusammenhängen gegebener Produktions– und Dienstleistungsorganisationen verfolgt werden“ [Reisin 1994].

Ich werde im Folgenden kurz auf das Phasenmodell eingehen und darauf, wie Softwareprodukte entstehen. Über den bei PETS verwendeten Qualitätsbegriff und Methodenrahmen partizipativer Softwareentwicklungsprojekte werde ich mich dem Verständnis von PETS nähern und abschließend meine These vertreten, daß die partizipative Softwareentwicklung der Konstruktionsperspektive zu-

[1] Ich beziehe mich hier auf das von Stahlknecht und Wedekind entwickelte erweiterte Modell.

[2] PETS steht laut [Reisin 1991] für den Organisationsansatz „Partizipation, Evolution und Transparenz bei der Softwareentwicklung“. Die Abkürzung geht zurück auf ein Projekt, das vom Minister für Arbeit, Gesundheit und Soziales des Landes Nordrhein Westfalen in der Zeit zwischen 1986 und 1989 im Rahmen des Landesprogrammes „Sozialverträgliche Technikgestaltung“ gefördert wurde. Dabei handelte es sich um ein Forschungs- und Entwicklungsprojekt, das in Kooperation mit dem zentralen Tarifarchiv beim Wirtschafts– und Sozialwissenschaftlichen Institut des DGB durchgeführt wurde. Das im Rahmen des Projekts entwickelte Softwareprodukt ist dort seither im Einsatz.

geordnet werden kann. Für meine Argumentation werde ich den Methodenrahmen von PETS, den verwendeten Qualitätsbegriff und ganz allgemein die Art der Durchführung von Softwareentwicklungsprojekten heranziehen.

8.1 Systemanalyse und Phasenmodell

Spätestens die Entwicklung größere Softwaresysteme erfordert eine Form von Projektmanagement zur Koordination der Arbeitsschritte und beteiligten Personen. Bei dem in der Systemanalyse verwirklichten Phasenmodell führt dies automatisch zu einer produktorientierten Sicht. Basierend auf der Annahme, daß die Produktion von Softwaresystemen sich letztendlich nicht von der Fertigung eines anderen technischen Produktes unterscheidet, übertrug man Konzepte zur Erstellung oder Verbesserung solcher Fertigungsprozesse direkt auf den Bereich der Softwareerstellung.

Eine Systemanalyse nach Stahlknecht und Wedekind ist ein Projekt zur Entwicklung oder Verbesserung eines Datenverarbeitungssystems in einem Betrieb, das genau einmal und in streng sequentialisierten Phasen durchgeführt wird (siehe auch Abbildung 8.1). Dabei werden die folgenden Phasen unterschieden:

1. Projektbegründung
2. Istanalyse
3. Sollkonzeption
4. Systementwurf
5. Systemimplementierung
6. Systemeinführung und -wartung (Systembetrieb)

Innerhalb der einzelnen Phasen sind Rückschritte zu vorhergehenden Teilphasen möglich. Ist eine Phase aber erst einmal abgeschlossen und das Abschlußdokument erstellt, sind die Ergebnisse unumstößlich. Das Abschlußdokument jeder Phase enthält die erarbeiteten Ergebnisse und bildet somit die Grundlage für die folgende Phase. Sind alle Phasen beendet und das Softwaresystem eingeführt, ist das Projekt beendet[3]. Der Projektabschluß beeinhaltet sowohl die Installation des Softwareproduktes, als auch die Erstellung von Arbeitsanweisungen und die Schulung der betroffenen MitarbeiterInnen. Für eine Revision des Softwareproduktes muß jedoch ein völlig neues Projekt gestartet werden.

[3]Das Phasenmodell wird gelegentlich auch in Stufenform dargestellt. Man spricht dann vom „Wasserfallmodell“.

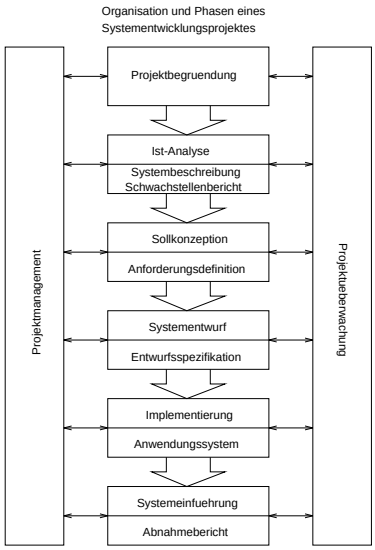


Abbildung 8.1: Das Phasenmodell in der Systemanalyse

Die Systemanalyse richtet sich nach Wedekind in der Hauptsache an die Entwickler, obwohl sie auch Auftraggebern die Möglichkeit bietet, den Projektfortschritt anhand von Meilensteinen (also Teilprodukten) zu überwachen. Die Entwickler führen die Systemanalyse durch und vergrößern die Benutzerschnittstelle, so daß der Computer über die universellen Funktionen hinaus um spezielle, auf ein bestimmtes Problem zugeschnittene Funktionen erweitert wird. Wedekind siedelt sein Projektszenario vor allem in der Industrie, der öffentlichen Verwaltung oder allgemein in betriebswirtschaftlichen Bereichen an, wo ein Team von SystemanalytikerInnen den Betrieb aufsucht, dort mit verschiedenen Techniken die Abläufe untersucht, um sie dann im Team zu optimieren und in ein Anwendungssystem umzusetzen. Eine Beteiligung der BenutzerInnen findet nur während der Ermittlung der zu modellierenden Arbeitsabläufe in Form von Interviews oder Beobachtungen statt. Die SystemanalytikerInnen müssen in der Folge auf dem zu modellierenden Gebiet ausreichend eigene Erfahrungen haben, um alle relevanten Teile zu berücksichtigen. Wedekind geht davon aus, daß durch die Systemabgrenzung und die abgeschlossene Darstellung der Abläufe in der Istanalyse fehlende oder nicht zum System gehörende Teile erkannt werden. Er spricht hier von der „Verantwortung des Systemanalytikers“.

Beim Phasenmodell steht das Qualitätskriterium der Korrektheit stark im Vordergrund, das vor allem durch das feste Vorgehen nach der sequentiellen Methode erfüllt werden soll. Kriterien wie Erweiterbarkeit oder Wiederverwendbarkeit spielen nur sekundär eine Rolle und zwar im Hinblick auf zukünftige Projekte.

8.2 Softwareentwicklung als kreativer Prozeß

Beim partizipativen Projektmodell PETS wird Softwareentwicklung als „Originalentwicklung“ gesehen. Es wird daher auch bewußt nicht von „Software–Fertigung“, sondern von „Software–Entwicklungs–Projekten“ gesprochen. Gemeint ist damit ein etablierter „Arbeitszusammenhang, in dem eine Zielvorstellung durch die Entwicklung eines Softwareprodukts realisiert werden soll“ [Reisin 1991]. Bei der Vereinbarung eines Projektes werden sowohl der zu modellierende Gegenstandsbereich als auch die mit der Durchführung des Projekts verfolgten Ziele und die „personellen und materiellen Rahmenbedingungen“ [Reisin 1992b] abgesteckt. Für die Durchführung eines Softwareentwicklungsprojektes verfolgte Ziele können laut [Reisin 1992b] sein:

- „Verbesserung der Qualität der Arbeit und der Arbeitsprodukte
- Steigerung der Produktivität und Effektivität
- Erhöhung der Transparenz von Organisationsstrukturen
- Durchlässigkeit von Entscheidungsstrukturen“

Diese Ziele können bei der Vereinbarung des Projektes quasi als Wunschziele genannt werden. Sie haben normativen Charakter und können bestenfalls durch „quantitative Kennziffern“ [Reisin 1991] näher bestimmt werden. Zu Beginn eines Softwareprojektes ist noch nicht festgelegt, welche Struktur und Qualität das Softwareprodukt haben wird, das der Erreichung dieser Ziele dienen soll. Hier liegt auch ein entscheidender Unterschied zwischen industrieller Fertigung eines technischen Produktes und der Entwicklung eines Softwareproduktes, denn Fertigungsprozesse (egal ob sie individuell oder kooperativ gestaltet sind) laufen nach einem vorgegebenen Muster und Fertigungsplan ab. Ausschlaggebend ist für sie in erster Linie die optimale Ausführung. Daher ist Routine, nicht Kreativität von besonderer Bedeutung[4].

Bei der Entwicklung von Softwareprodukten handelt es sich prinzipiell um ein neues Produkt, selbst wenn nur Teile einer bereits bestehenden Softwareanwendung völlig neu entwickelt werden, denn gerade diese Teile sind es, die das Projekt begründen. Im Gegensatz zur Fertigung industrieller Produkte ist hier nicht die der Erstentwicklung folgende Serienproduktion, sondern die Entwicklung des ersten Produkts, des **Originals** von besonderer Bedeutung. Im Vergleich zu dieser ersten Entwicklung ist die Massenfertigung trivial, zumindest aber von untergeordneter Bedeutung. Dies vor Augen, ist der „Prozeß der Softwareentwicklung, unabhängig davon, ob er mit oder ohne BenutzerInnen durchgeführt wird, als **kreativer Arbeitsprozeß** bestimmt“.

[4] Hierzu siehe auch: [Reisin 1991] und [Reisin 1992b].

Es ist ein Kennzeichen kreativer Arbeitsprozesse, daß ihre Resultate erst im Verlauf der Arbeit die spezifischen Qualitäts– und Strukturmerkmale gewinnen und nicht von vornherein determiniert sind[5]. Als Folge können auch die Mittel und der Verlauf kreativer Arbeitsprozesse selbst nicht vollständig vorherbestimmt werden. Der Verlauf ist diskontinuierlich, er umfaßt explorative Phasen und Lernprozesse, die im einzelnen weder plan– noch algorithmisierbar sind. Dies heißt für die Softwareentwicklung, daß „die Kompentenz zur zielgerichteten Gestaltung des Produkts und der Arbeitsprozesse seiner Realisierung im Zuge der Projektdurchführung“ erst aufgebaut werden muß. „Mit der wachsenden Kompetenz der Beteiligten erhält das Produkt seine Gestalt und die Aktivitäten und Mittel zu seiner Realisierung ihre konkrete Bestimmung“.

8.3 Partizipation und Kooperation bei der Softwareentwicklung

Im Kontext der Arbeitswelt werden dem Begriff „Partizipation“ je nach Blickwinkel des Betrachters unterschiedliche Bedeutungen zugeordnet[6]. In bezug auf technische sowie organisatorische Innovationen innerhalb eines Betriebes ist schlicht die Beteiligung der Arbeitnehmer an der Planung, Gestaltung und Realisierung der Innovationsmaßnahmen gemeint. Eine der Grundannahmen partizipativer Softwareentwicklungsprojekte ist, daß mit der Gestaltung und Einführung eines neuen Sofwareproduktes, das als Arbeits-, Kommunikations-, oder Organisationsmittel benutzt werden soll, nicht nur das Softwareprodukt selbst, sondern der gesamte Arbeitszusammenhang gestaltet wird. Mit der Partizipation der BenutzerInnen während des gesamten Softwareentwicklungsprojektes werden also nicht nur emanzipatorische Ziele, sondern ganz einfach die Qualitätssicherung des zu erstellenden Softwareproduktes verfolgt. „**Partizipation** steht daher im Kontext der Softwaretechnik für eine Organisationsform von Software–Projekten, die die *Beteiligung der Benutzer* des angestrebten Produkts vorsieht, und mehr noch, die kontinuierliche Kooperation zwischen den Entwicklern und Benutzern während des Entwicklungsprozesses systematisch unterstützt.“ [Reisin 1994]

Wenn nun aber die Entwicklung von Softwaresystemen ein kreativer Arbeitsprozeß ist, an dem BenutzerInnen, EntwicklerInnen und eventuell weitere „Statusgruppen“ gleichermaßen beteiligt sind, und die Gestaltungs– und Durchführungskompetenz im Verlauf des Projektes erst entwickelt werden muß, wird dadurch ein kontinuierlicher Lern– und Kommunikationsprozeß aller Beteiligten initiiert. Nur wenn dieser Prozeß erfolgreich verläuft, kann auch das Softwareprojekt selbst als erfolgreich angesehen werden und zu einem adäquaten

[5] Siehe hierzu [Reisin 1991] und [Reisin 1992b]

[6] Mehr dazu im Artikel [Reisin 1994], Einführung, Seite 299 ff.

Softwareprodukt führen. Partizipative Entwicklungsprojekte sind also „wesentlich durch die **kooperativen Arbeitsprozesse der BenutzerInnen und EntwicklerInnen** bestimmt.“ [Reisin 1992b]

Dabei ist sowohl das Expertenwissen der EntwicklerInnen, wie auch das der BenutzerInnen von Bedeutung, denn

- die EntwicklerInnen haben keine ausreichende Fachkompetenz im Arbeitsgebiet der BenutzerInnen und werden es realistischerweise auch nicht durch Interviews oder andere Techniken in kurzer Zeit erwerben können
- die BenutzerInnen haben meist keine Erfahrung in der Strukturierung, Darstellung, Begründung und Dokumentation ihrer Arbeitsbereiche sowie mit den Möglichkeiten technischer Gestaltung von Softwaresystemen
- die BenutzerInnen verfügen über ein implizites Erfahrungswissen, das nur sehr schwer vermittelt werden kann und auch erst im Verlauf mehrerer Gespräche klar wird
- wie bereits angedeutet, verändern und erweitern sich Anforderungen im Verlauf des Projekts

Kooperative Arbeitsprozesse sind „generell dadurch charakterisiert, daß die Beteiligten sich bei ihren jeweiligen Aktivitäten nicht nur auf sachlich–gegenständliche Gegebenheiten, sondern auch aufeinander beziehen“ [Reisin 1992b]. Die dadurch entstehende sachlich–gegenständliche Ebene kann nicht von der intersubjektiven getrennt oder eine auf die andere reduziert werden. Die so entstehende spezifische Projektsituation bildet keinen bloßen „sozialen Kontext“ des Projektes, sondern ist ein bestimmendes Moment der Softwareentwicklung und muß als solches theoretisch wie methodisch entsprechend berücksichtigt werden.

Qualitätsanforderungen bei partizipativen Softwareprojekten

Im Gegensatz zum Phasenmodell, bei dem die produktbezogene Korrektheit als vorherrschende Qualitätsanforderung angestrebt wird, gibt es bei PETS eine Reihe von Qualitätsmerkmalen, die in Anforderungen „an die innere Beschaffenheit“, „an das Produkt im Einsatz“, „an den Entwurf“ und in „benutzerInnenorientierte Qualitätsmerkmale“ unterteilt werden können und sich auf das Softwareprodukt, wie auf den Prozeß seiner Entwicklung und die gestalteten Arbeitszusammenhänge beziehen. Da PETS insbesondere für die Entwicklung von Softwaresystemen, die in Arbeitszusammenhängen benutzt werden, entwickelt wurde, ist

Zweckangemessenheit eine grundlegende Forderung, die sich in allen Bereichen niederschlägt. Ich nenne sie daher gleichsam als Eingangsanspruch[7].

Anforderungen an die innere Beschaffenheit des Produktes Diese ganz grundlegenden Anforderungen beschränken sich auf:

- Lesbarkeit
- sinnvolle Struktur
- Trennung von Außen– und Innenverhalten des Systems und seiner Komponenten

Anforderungen an das Produkt im Einsatz Unter die Anforderungen an das Produkt im Einsatz fallen:

- **Zuverlässigkeit:** zusammengesetzt aus den Forderungen nach
  - **Korrektheit:** das Programm soll für alle in seiner Aufgabenstellung definierten Eingaben die gewünschten Ausgaben liefern
  - **Robustheit:** das Programm soll fehlerhafte Eingaben zurückweisen ohne zuvor in gefährliche Teilverarbeitungen einzusteigen oder sinnlose Ergebnisse zu berechnen
  - **Ausfallsicherheit:** das Programm berücksichtigt während seines Einsatzes die Möglichkeit eines Versagens seiner technischen Umgebung angemessen
- **sinnvolle Ausnutzung der Betriebsmittel:** Speicherplatz, Laufzeit, Kosten der Programmentwicklung, indirekte Kosten des Programmeinsatzes
- **Angemessenheit:** bestehend aus den Forderungen nach
  - **Durchschaubarkeit:** jede durch das Programm direkt oder indirekt betroffene Person versteht das Programm in einer Weise, die für ihr Zusammenwirken mit dem System geeignet ist und ausreicht

[7] Ich entnehme diese Qualitätsanforderungen dem Endbericht des PETS–Projektes, das sich seinerseits auf das Projekt STEPS bezieht, das seinen Ursprung am selben Institut für Softwaretechnik der TU Berlin hat. Nachdem eine der Protagonistinnen, Frau Prof. Dr. Floyd einem Ruf nach Hamburg gefolgt ist, wurden die gemeinsam entwickelten Ideen in unterschiedlichen Projekten in Berlin und Hamburg weitergeführt: in Berlin unter dem Namen PETS, in Hamburg unter dem Namen STEPS. Für die subtilen Unterschiede zwischen den Zugängen zu den Projekten verweise ich auf [Reisin 1990], Kapitel 4.4, Seite 72 ff.

  - **Konvivialität:** ein Programm soll Resultate so erarbeiten, wie der Benutzer sie wirklich braucht und so arbeiten, daß der Benutzer nicht seine Arbeitsweise künstlich der des Programms anpassen muß
  - **Erreichen der Zielvorgaben durch die gewählte Lösung**

- **Änderbarkeit:** aufgeteilt in drei Kategorien
  - **strukturelle Änderbarkeit:** die Möglichkeit, Fehler oder ungünstige Teile eines Programms lokal zu verbessern, ohne den Rest des Programms zu gefährden
  - **technische Änderbarkeit:** zum Beispiel das Umsteigen auf ein anders Betriebssystem
  - **problemorientierte Änderbarkeit:** nötig durch Fehler in der Anforderungsermittlung oder Veränderungen im Umfeld des Systems (unter Umständen auch durch das System selbst)

BenutzerInnenorientierte Qualitätsmerkmale Übergeordnetes Gestaltungsanliegen sozialverträglicher Systementwicklung durch PETS sind die „Humanisierung und die Demokratisierung der Arbeitsverhältnisse bei der Gestaltung DV–gestützter Arbeitsplätze“ (PETS–Endbericht, S.18). Dies gilt sowohl für das zu entwickelnde System, als auch für den Prozeß seiner Entwicklung. Wesentliche benutzerInnenorientierte Qualitätsmerkmale sind:

- aufgabenbezogene Angemessenheit
- benutzerInnenbezogene Verständlichkeit
- benutzerInnenbezogene Handhabbarkeit
- aufgaben– und benutzerInnenbezogene Transparenz

Dabei ist ein Softwaresystem **angemessen**, „wenn es den BenutzerInnen die problemlose Bearbeitung ihrer Aufgaben gestattet und seine Benutzung leicht erlernbar und wunschgemäß veränderbar und effektivierbar ist“ [Floyd et al. 1989].

Die bei PETS vertretene Sichtweise kreativer, kooperativer Arbeitsprozesse und die benutzerInnenorientierten Qualitätsmerkmale stellen besondere Anforderungen an Organisation und Methodik, die ich im Folgenden näher betrachten werde.

Organisation und Methoden kooperativer Softwareentwicklung

Die kooperativen Arbeitsprozesse in partizipativen Projekten erfordern Methoden[8] und Arbeitsmittel, die „ihrerseits an die konkrete Projektsituation und Zweckbestimmung anpaßbar und projektspezifisch gestaltbar“[Reisin 1990] sind. Das heißt, die Methoden müssen die konkreten Anforderungen der jeweiligen Projektsituation und die Gestaltung der gemeinsamen Aktivitäten und daraus resultierenden Produkte unterstützen. Methoden und Arbeitsmittel müssen also projektspezifisch gestaltbar sein und dem Umstand Rechnung tragen, daß in den „wechselseitige(n) Kommunikations– und Lernprozesse(n) zwischen den EntwicklerInnen und BenutzerInnen (…) gewissermaßen ein neuer Realitätsbereich geschaffen wird“ [Reisin 1992b].

PETS wie STEPS arbeiten mit einem zyklischen, evolutionären, auf Systemversionen aufbauenden Projektmodell (siehe Abb. 8.2). Der Methodenrahmen umfaßt dieses Projektmodell, sowie eine Vielzahl aufeinander abgestimmter Methoden zur Unterstützung der kooperativen Arbeitsprozesse.

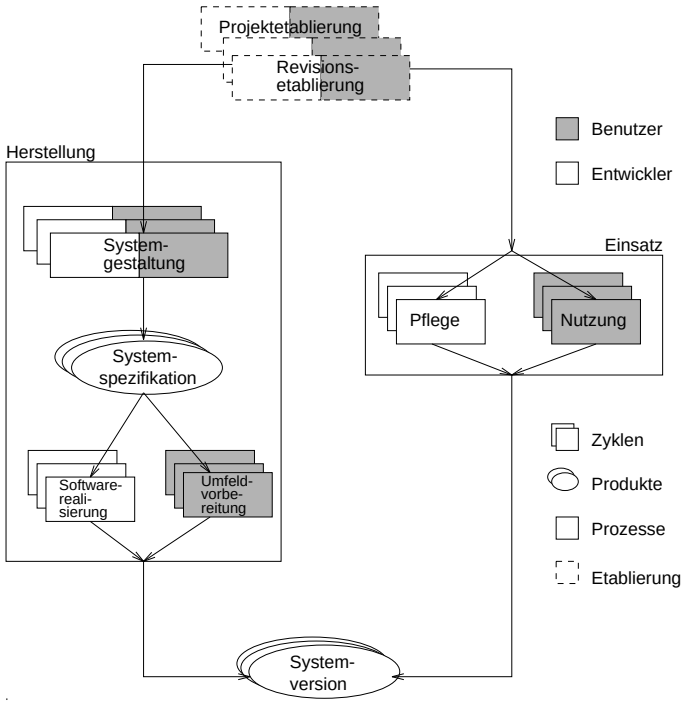


Abbildung 8.2: Das Projektmodell von STEPS

[8]Eine Definition der Begriffe „Methode“ und „Methodenrahmen“ findet sich in [Reisin 1990], Kapitel 4.2, Seite 62 und 63.

Projektinvariant gestaltbare Teile

Das Projektmodell zeigt die Entwicklung des Softwaresystems als eine Folge von **Entwicklungszyklen**, das System muß also an veränderte Anforderungen aus dem Einsatzbereich anpaßbar und somit revidierbar sein. Dabei wird der logische Zusammenhang zwischen Herstellung und Einsatz des Softwaresystems, die Notwendigkeit der Revidierbarkeit, deutlich. Jeder Entwicklungszyklus hat die Erstellung und den Einsatz einer **Systemversion**[9] zum Gegenstand. Am Anfang jedes Zyklus steht die **Zyklusetablierung**, da ein kooperativer Arbeitsprozeß nur auf der Basis ausdrücklich vereinbarter Ziele und Rahmenbedingungen möglich ist. Die Zyklusetablierung ist von besonderer Bedeutung, da sie das ganze Projekt umfaßt und an ihr nicht nur BenutzerInnen und EntwicklerInnen, sondern auch AuftraggeberInnen und andere InteressenvertreterInnen beteiligt sind. Sie ist quasi die Instanziierung des Projektes.

Der Erstzyklus wird durch die Etablierung des gesamten Projektes initiiert, nachfolgende Entwicklungszyklen starten durch eine Revisionsetablierung.

Das Projektmodell zeigt außerdem die für jedes partizipative Softwareentwicklungsprojekt invarianten „unterschiedlichen und gemeinsamen **Aktivitäten** der BenutzerInnen und EntwicklerInnen“[Reisin 1992b]. Im Wesentlichen handelt es sich um Aktivitäten der Gestaltung, Realisierung und Benutzung des Softwaresystems.

Das Projektmodell weist lediglich zwei **Produkte** aus: die von den EntwicklerInnen erstellte **Systemspezifikation** als Ergebnisprodukt der Systemgestaltung und die **Systemversion** als Ergebnisprodukt und als Realisierung der Systemspezifikation des jeweiligen Entwicklungszyklus (wobei hier die bei der Etablierung festgelegten Ziele erfüllt werden müssen).

Projektspezifisch gestaltbare Teile

Aussagen über die „zeitliche Abfolge von Aktivitäten, über die Art und Weise ihrer Ausführung und über die Anzahl, den Inhalt und die Form der hierbei entstehenden Zwischenprodukte“[Reisin 1992b] können nur projekt– und situationsspezifisch getroffen werden, da sie aus den konkreten Arbeitsprozessen des Projektes heraus gestaltet werden.

Die **Zyklizität** des Projektmodells zeigt die Notwendigkeit bzw. Anforderung zur Revision einer Version, die sich aus den Arbeitsprozessen der Benutzung ergibt. Diese müssen systematisch berücksichtigt werden und dürfen nicht als „Wartung des Produktes“ bagatellisiert und versteckt werden. Die Anzahl der Entwick-

[9] Der Versionsbezug des Projektmodells bietet allerdings durchaus die Möglichkeit, bereits zu Beginn eines Entwicklungszyklus Revisionen systematisch zu planan. Hierzu siehe auch [Reisin 1992b], Abschnitt 3.3.

lungszyklen ist nicht vorherbestimmbar.

Wichtigster Teil der **Zyklusetablierung** und Ergebnis der Projektetablierung ist der „Projektvertrag“, der zwischen den Beteiligten geschlossen wird und unter anderem ein grobes Systemkonzept und Projektablaufsmodell enthält. „Wie und ob ein solcher Vertrag ausgehandelt wird, ist durch das Projektmodell nicht festgelegt“ [Reisin 1990].

Das Projektmodell entkoppelt „grundlegende Diskursbereiche der Softwareentwicklung, wie Ermittlung und Bestimmung der Funktions– und Strukturanforderung, Entwurf der Architektur, Konstruktion und Implementierung der Softwarefunktionen“ [Reisin 1990] von vorgegebenen zeitlichen Phasen oder Meilensteinen. Die einzelnen **Aktivitäten** werden ebenso wie die beiden aus dem Entwicklungsprozeß heraus zusätzlich vereinbarten Zwischenprodukte im Verlauf des Projekts konkretisiert und zeitlich systematisiert.

Referenzlinien als Koordinationskonzept kooperativer Arbeitsprozesse

Als Koordinationskonzept zum prozeßbezogenen Projektmanagement von Softwareentwicklungsprojekten ist alternativ das Konzept der Referenzlinien entwickelt worden (siehe Abb. 8.3). Dieses Konzept wurde bei PETS und STEPS von dem dänischen Projekt MARS übernommen.

Das Projektmodell selbst fordert mit der Systemspezifikation und der Systemversion nur zwei Zwischenprodukte als Koordinationspunkte. Mit Hilfe von Referenzlinien kann der Entwicklungsprozeß genauer koordiniert und auf spezifische Gegebenheiten eines Projektes flexibel reagiert werden. Rererenzlinien werden aus dem „Entwicklungsprozeß heraus von den Beteiligten festgelegt“ [Reisin 1992b]. Mit ihnen wird ein jeweils zu erreichender Zwischenstand des Projektes durch das Vorliegen von Zwischenprodukten definiert. Beginnend mit dem initialen Schritt der Projektetablierung legen die Beteiligten im Verlauf des Entwicklungsprozesses jeweils den als nächstes zu erreichenden Projektzustand fest. Eine neue Referenzlinie muß spätestens mit Erreichen der gerade „durchlaufenen“ festgelegt sein[10].

Durch das Konzept der Referenzlinien kann das Projekt nie in einen undefinierten Zustand geraten. Entscheidend ist, „daß die Form und die Zeitpunkte für Zwischenprodukte durch das Projektmodell nicht vorgegeben, sondern prozeßbezogen definiert werden“. Eine Anforderungsbeschreibung oder ein Prototyp können ebenso eine Referenzlinie sein, wie ein für den Auftraggeber faßbares Zwischenprodukt.

[10] Zur Erläuterung der Referenzlinien siehe auch [Reisin 1992b], Abschnitt 3.3 und [Reisin und Schmidt 1988], Abschnitt 2.3.

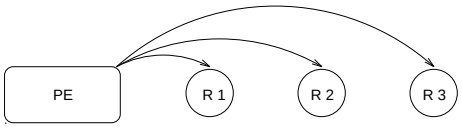


Abbildung 8.3: Koordinierung eines Prozesses durch Referenzlinien

Sollte der Auftraggeber zur Kontrolle des Projektforschritts Meilensteine fordern, die wie beim Phasenmodell produktbezogene Zwischenstufen darstellen, so können diese in das Konzept der Referenzlinien integriert werden. In diesem Fall stellt eine Referenzlinie einen Meilenstein dar. Der gravierende Unterschied zwischen Meilensteinen und Referenzlinien ist, daß Meilensteine rein produktorientiert festgelegt sind, während Referenzlinien während des Projektes dynamisch und an den Arbeitsprozessen orientiert definiert werden.

8.4 Das Modellierungsverständnis von PETS

Um das Modellierungsverständnis von PETS zu untersuchen, betrachte ich zuerst das **Verhältnis von Modell und Original**. Mit den angewendeten Methoden und in der dargestellten Organisationsform des Projektes wird in einem kreativen und kooperativen Prozeß und unter Verfolgung bestimmter Zwecke das Original mit dem Modell neu konstruiert. Damit wird ein neuer Realitätsbereich geschaffen und auch das Original ist ein Modell im Sinne der Modellierung von Arbeitsprozessen und -zusammenhängen. Dies wird auch durch die im Projektmodell bereits angelegte Zyklizität unterstrichen. Mit jeder neuen Revisionsetablierung wird ein Modell Teil des Originals, das somit ebenfalls als Modell erkennbar wird. Eine klare Abgrenzung findet durch das Versionskonzept statt.

Ein weiteres Indiz für die Konstruktionsperspektive bei PETS liegt im verwendeten Qualitätsbegriff. Hier ist die strikte **Zweckantizipation**, die sowohl explizit als Qualitätsanforderung genannt wird, als auch in den benutzerInnenorientierten Qualitätsmerkmalen implizit zum Tragen kommt, ein starker Anhaltspunkt. Im Sinne von PETS ist ein Softwaresystem **angemessen**, „wenn es den BenutzerInnen die problemlose Bearbeitung ihrer Aufgaben gestattet und seine Benutzung leicht erlernbar und wunschgemäß veränderbar und effektivierbar ist“ [Floyd et al. 1989]. Die **verfolgten Zwecke** gehen dabei ebenfalls aus den Qualitätsanforderungen (siehe Abschnitt 8.3) hervor, daher gehe ich an dieser Stelle nicht noch einmal gesondert darauf ein.

Untersucht man, in wieweit das Projekt seinen Zwecken angemessen ist, muß man sich die Art und Weise der Durchführung des Softwareprojektes, sowie die dabei zur Anwendung kommenden Methoden näher anschauen. Mit PETS sollen EntwicklerInnen und BenutzerInnen dabei unterstützt werden, „zweckan-

gemessene Systeme zu gestalten und dabei den Interessen der Benutzer(innen) an der Verbesserung der Arbeitsqualität und Demokratisierung der Arbeitsverhältnisse zu entsprechen“. Dabei muß „die Entwicklung solcher Softwaresysteme ... immer die Gestaltung von Technik und Arbeit zugleich im Auge haben“ [Reisin und Schmidt 1988]. Diese Ziele können nur „durch eine *versionsorientierte, zyklische Vorgehensweise* (, die) die *evolutionäre* Veränderung des Arbeitsgebietes und der Anforderungen an das Softwaresystem einbezieht“ [Wolf und Mehl 1991] und mit einem partizipativen Ansatz der Softwareentwicklung erreicht werden. Betrachtet man die projektinvarianten und projektspezifischen Anteile im Projektmodell von PETS, so werden genau diese Ansprüche verwirklicht.

In einem kooperativen Arbeitsprozeß tragen BenutzerInnen wie EntwicklerInnen gleichermaßen zum Fortschreiten des Projektes bei. Dabei kommt den „Kommunikations– und wechselseitigen Lernprozessen eine zentrale Bedeutung zu. Sie bilden die eigentlichen kreativen Arbeitsprozesse, in denen die gemeinsame Kompetenzbasis zur benutzerInnenorientierten Gestaltung des neuen Produkts ausgebildet wird“ [Reisin 1992b]. Das Konzept der Referenzlinien unterstützt die kooperativen Arbeitsprozesse, indem es mit der flexiblen Definition der jeweils nächsten Referenzlinie den nötigen Freiraum für die Kommunikations– und Lernprozesse läßt. Gleichzeitig kann durch die Projektetablierung (den Projektvertrag) das Projekt nicht „versacken“, da Ziele und Rahmenbedingungen von vornherein klar sind.

Mit ihrer gleichberechtigten Teilnahme am Projekt können die BenutzerInnen aktiv an der Verbesserung der Arbeitsqualität und Demokratisierung der Arbeitsverhältnisse arbeiten. Dabei bezieht sich die Demokratisierung sowohl auf die zukünftig gestalteten Arbeitszusammehänge, als auch auf die aktuelle Durchführung des Softwareentwicklungsprojektes.

Abschließende Bemerkung

Meine bereits eingangs bestehende Annahme, daß die partizipative Entwicklung von Softwaresystemen, die ich anhand von PETS untersucht habe, der Konstruktionsperspektive zuzuordnen ist, hat sich bestätigt. Gleichzeitig ist mir klargeworden, daß wo immer es sich bei einem Softwareentwicklungsprojekt nicht um eine standardisierte Anwendung handelt, ein für menschliche Arbeitszusammenhänge geeignetes Softwareprodukt nur mit den Methoden partizipativer Softwareentwicklung erreicht werden kann. Nach den im Studienprojekt „Modellierung“ gewonnenen Erkenntnissen und meinem Verständnis von PETS ist nur die Konstruktionsperspektive der Entwicklung von Software für Menschen angemessen.

Trotzdem haben sich für mich zwei Fragen ergeben, auf die ich keine einfache Antwort gefunden habe: die eine handelt von menschlichen Fähigkeiten, die an-

dere von menschlichen Schwächen. Die kooperativen Arbeitsprozesse eines partizipativen Softwareentwicklungsprojektes erfordern von allen Beteiligten in hohem Maß die Fähigkeit zu Kommunikation und eine gewisse Lernbereitschaft. Beides muß aber bei den Beteiligten nicht unbedingt im Übermaß vorhanden sein. Was dann? Nun kann man ja argumentieren, daß EntwicklerInnen, die nicht kommunizieren können und sich nicht auf neue Wissensgebiete einlassen wollen, sich lieber ein Arbeitsgebiet suchen sollen, auf dem sie möglichst wenig Kontakt mit Menschen haben. Und in einem Betrieb wird es ja irgendwo auch BenutzerInnen geben, die in der Lage sind, an kooperativen Arbeitsprozessen teilzunehmen (bei den BenutzerInnen mag man ja in der „Auswahl“ etwas flexibler sein …)[11]. Aber worauf ich hinaus will, ist, daß viele Menschen die Anforderungen zur Durchführung eines partizipativen Softwareentwicklungsprojektes im realen Leben nicht erfüllen können oder wollen. Hier ist entweder ein behutsames Umgehen aller Beteiligten erforderlich oder weitergehend eine Entwicklung der menschlichen Gesellschaft hin zu mehr Eigenverantwortung.

Die zweite Frage betrifft das immer wieder auftretende „Peer-problem“ in Gruppen. Vermutlich wird sich auch bei einem partizipativen Softwareentwicklungsprojekt die Frage nach der Leitung des Projektes ergeben. In der Praxis wird nicht etwa ein demokratisches Rotationssystem oder dergleichen gefunden werden, sondern einer oder eine Untergruppe wird den Prozeß dominieren, das vermute ich jedenfalls. Dies liegt zum einen vermutlich daran, daß die meisten Menschen in hierarchischen Systemen (und seien es die unmittelbaren Arbeitszusammenhänge) leben und diese oftmals auch nicht in Frage stellen, zum anderen sind viele auch einfach zu bequem zur Eigenverantwortung. Hier stellt sich die Frage, ob das Streben nach einer dominanten Position in einer Gruppe ganz einfach in der menschlichen Natur begründet ist, oder ob wir ganz einfach mehr Eigenverantwortung und Demokratieverständnis entwickeln müssen.

[11] Um Mißverständnissen vorzubeugen: Vorsicht, Ironie im lebenden Text!!

Kapitel 9

Der Wissensbegriff in der Künstlichen Intelligenz Autorin: Birgit Schelm

Bevor in den folgenden Kapiteln zwei Methoden zur Wissensmodellierung aus der Künstlichen Intelligenz dargestellt und analysiert werden, soll hier allgemeiner auf den Wissensbegriff der KI im Lichte von Abbild- und Konstruktionsperspektive eingegangen werden.[1]

9.1 Der Wissensbegriff

Das wesentliche Problem beim *Knowledge Engineering*, also der Disziplin der Künstlichen Intelligenz, die sich mit der Modellierung von Wissen für Expertensysteme befaßt, ist die Wissensakquisition. Wissensakquisition wird meistens als Transferprozeß betrachtet, wobei das Hauptproblem darin besteht, das Wissen „aus der Expertin[2] zu extrahieren“ und ein vollständiges und korrektes Modell dieses Wissens in das Expertensystem zu übertragen. Wissensakquisition stellt dabei einen Flaschenhals dar, wobei entweder das Expertensystem die Flasche darstellt und das Problem darin gesehen wird, das Wissen in das Expertensystem einzugeben, oder die Expertin die Flasche darstellt und das Problem darin gesehen wird, das Wissen aus der Expertin zu extrahieren[3].

Meines Erachtens stellt die Frage nach der den verschiedenen Methoden der Wis-

[1] Diese Modellierungsperspektiven werden ausführlich von Gernot Grube im 1. Teil dieses Bereichts erläutert.

[2] Da in den ganzen Texten, die ich mir im Zusammenhang mit dieser Ausarbeitung durchgelesen habe, ausschließlich Experten vorkommen, möchte ich an dieser Stelle den nie erwähnten Expertinnen den Vorzug geben.

[3] v. dazu [Morik 1989] S.107 – 112.

sensakquisition zugrundeliegenden Vorstellung von Wissen die Kernfrage für die Einordnung in die Modellierungsperspektiven dar. Daher ist es naheliegend, die verschiedenen Vorstellungen von Wissen, die in der Wissensakquisition anzutreffen sind, näher zu beleuchten.

In diesem Kapitel stelle ich das Wissenskonzept, das die meisten derzeit üblichen Wissensakquisitionssysteme implizit oder explizit voraussetzen, vor und erläutere die Anforderungen an ein Wissenskonzept, das aus wissenssoziologischer Sicht adäquat ist. Dabei beziehe ich mich auf mehrere Untersuchungen von Barbara Becker, Elke Steven und Sabine Strohbach, die das Wissenskonzept in der gängigen Wissensakquisitionspraxis untersucht und kritisiert haben. Sie begründen auf philosophischer und soziologischer Grundlage, welche Aspekte ein aus wissenssoziologischer Sicht adäquates Wissenskonzept berücksichtigen muß ([Becker and Steven 1991], [Becker 1991] und [Becker et al. 1991]). Dabei läßt sich das Wissenskonzept, das im Knowledge Engineering laut diesen Untersuchungen üblich ist, der Abbildperspektive zuordnen, während sich die Forderungen der Autorinnen an ein adäquates Wissenskonzept der Konstruktionsperspektive zuordnen lassen. Diese Einordnung ergibt sich daraus, daß die Einordnung für die Abbildperspektive fast schon direkt aus der Erläuterung der „computational theory of the mind“ folgt, während die Anforderungen an ein adäquates Wissenskonzept z.T. auf den gleichen philosophisch-soziologischen Quellen basieren, die auch die Grundlage für die Konstruktionsperspektive darstellen.

9.2 Das Wissenskonzept in der Abbildperspektive

In [Becker and Steven 1991] stellen Becker und Steven fest, daß die „computational theory of the mind“ in der Künstlichen Intelligenz und damit auch in der Wissensakquisition immer noch dominant ist. Diese Theorie basiert auf dem Symbolverarbeitungsansatz, dem wiederum mentale Repräsentationen als Konzept zugrunde liegen. Dabei wird von der Existenz atomarer mentaler Ausdrücke und mentaler Operationen ausgegangen. Mit deren Hilfe lassen sich dann beliebig komplexe mentale Ausdrücke zusammensetzen. Die Bedeutung dieser komplexen mentalen Ausdrücke wird durch die Bedeutung der atomaren mentalen Ausdrücke und die syntaktische Struktur des mentalen Ausdrucks bestimmt.

Dieser Ansatz läßt sich in folgenden Grundannahmen des Knowledge Engineering wiederfinden:

Wissen ist in Form symbolischer Repräsentationen mental verankert und damit potentiell explizierbar.

Wissen ist strukturiert und zergliederbar.

Wissen läßt sich durch mentale Kategorien ohne Berücksichtigung subjektiver, historischer und sozialer Kontexte beschreiben.

Wissen ist objektivierbar.

Wissen läßt sich unabhängig von der Basis seiner materiellen Realisierung betrachten. (aus [Becker et al. 1991][4])

Daraus lassen sich dann Aspekte für ein Wissenskonzept, das der Abbildperspektive zugeordnet werden kann, finden. Zu diesen Aspekten gehört, daß die Bedeutung mentaler Ausdrücke von ihrer syntaktischen Struktur bestimmt ist, das wissende Subjekt allerdings keine Rolle spielt, genausowenig wie die physikalische Realisierung der mentalen Ausdrücke. Die Bedeutung der mentalen Ausdrücke ist also vom jeweiligen Kontext unabhängig, der Einfluß einer Sprach- und Kulturgemeinschaft, die Geschichtlichkeit und soziale Einbettung von Wissen werden genauso ausgeblendet, wie kommunikative Aspekte und Mehrdeutigkeiten. Damit wird Wissen rein mit mentalen Kategorien beschreibbar, vor allem präzise beschreibbar, und ebenso wie die mentalen Ausdrücke vom wissenden Subjekt unabhängig. Wissen wird daher explizierbar, kann also außerhalb des wissenden Subjekts zugänglich gemacht werden. Ferner kann Wissen zergliedert und Teile von Wissen (meist also das Spezialgebiet einer Expertin) isoliert beschrieben werden.

Aber auch andere Aspekte für die Abbildperspektive, die an anderer Stelle[5] bereits erläutert wurden, lassen sich in diesem Zusammenhang wiederfinden:

Die Welt und das Wissen von der Welt ist feststellbar und damit beschreibbar (mit atomare Strukturen). ([Becker and Steven 1991] S. 8)

Die Beschreibung von Wissen muß konsistent und korrekt sein. „[…] consistence in the arrangement of known facts, and the validity of formal derivations are the basis of the search for knowledge.“ ([Becker and Steven 1991] S. 10)

Mathematik hat einen hohen Stellenwert, im allgemeinen werden formale Sprachen zur Repräsentation von Wissen verwendet.

Für den Wissensakquisitionsprozeß haben Becker et. al. damit u.a. folgende Konsequenzen festgestellt:

Wissen muß neutralisiert werden und von subjektiven Einflüssen, Beurteilungen und Wertungen befreit werden.

Überflüssiges muß eliminiert werden, man muß sich auf das Relevante beschränken (meist wird nur präzise beschreibbares Wissen als relevant angenommen).

Es werden nur Erklärungen kausaler Natur zugelassen.

Es wird angenommen, daß Weltausschnitte durch eine endliche, begrenzte Auflistung von Fakten beschreibbar wären.

---

[4] Ausführlicher sind diese Grundannahmen in [Becker and Steven 1991] und [Becker 1991] beschrieben.

[5] Siehe dazu den Beitrag von Gernot Grube im 1. Teil dieses Berichts, sowie die Beiträge von Martin Fischer im 2. Teil.

Subjektspezifisches Wissen wird als objektiv dargestellt.

9.3 Das Wissenskonzept aus wissenssoziologischer Sicht

Bereits in ihrer Kritik an dem im Knowledge Engineering weit verbreiteten Wissenskonzept machen Becker und Steven ihre Forderungen an ein adäquateres Wissenskonzept deutlich. Sie entwickeln diese Anforderungen aber noch ausführlicher in [Becker et al. 1991] aus den Erkenntnissen der Wissenssoziologie und einer ganzheitlichen Vorstellung von Wissen. Im Zusammenhang mit Knowledge Engineering erscheinen ihnen folgende Aspekte wichtig (siehe [Becker and Steven 1991], S. 16 - 19):

- Die Perspektive von Wissen.
- Die Berücksichtigung qualitativer Aspekte.
- Wissen im sozio-historischen Kontext.
- Die soziale Konstruktion von Wissen.
- Die Möglichkeit, Entwicklungen und Veränderungen zuzulassen.
- Das Zulassen von Vielfalt statt Standardisierung.

Dazu gehört für sie auch, daß individuelles Wissen nicht nur stark vom wissenden Subjekt beeinflußt ist, sondern Intentionen ebenso wie fundierte Kenntnis des Hintergrunds, der Herkunft von Wissen und des Kontextes beinhaltet[6]. Intention ist also genauso ein Bestandteil von Wissen, wie die Kenntnis entsprechender Hintergründe.

Ferner wird in [Becker and Steven 1991] herausgestellt, daß sich Wissen nicht nur auf präzise beschreibbare Fakten beschränkt, sondern auch das umfaßt, was jemand glaubt und denkt. Diese Faktoren können bei der Wissensakquisition nicht einfach ausgeklammert werden. Dazu schreibt Becker: „[…] the individual beliefs of an expert shape the articulated knowledge and gave it an individualistic colour. “([Becker 1991] S. 8)

Becker macht einen Unterschied zwischen theoretischem Wissen und menschlichem Handeln. Diesen Unterschied macht sie daran fest, daß Handlung keiner

[6] „[…] individual knowledge is not only strongly influenced by the knowing subject, but always implies intentions as well as fundamental insight into the background, the connections to the origin of knowledge, the contexts.“ ([Becker and Steven 1991] S. 9)

vorausgegangenen Reflexion bedarf. „Action [...] does not require previous reflection [...]“ ([Becker 1991] S. 9).

Ein weiterer Aspekt, der vor allem bei [Berger und Luckmann 1969] im Mittelpunkt steht, ist der interaktive Prozeß, durch den Wissen entsteht. Dabei beeinflußt die Umgebung das wissende Subjekt, genauso wie das dann mehr wissende Subjekt sich wieder auf seine Umgebung auswirkt.

Gerade im Zusammenhang mit Expertensystemen erscheint mir auch wichtig, wie auch Becker und Steven ausführlich darlegen, daß auch wissenschaftliches Wissen („scientific knowledge“) unter den genannten wissenssoziologischen Gesichtspunkten betrachtet werden muß und keine Ausnahme darstellt. Denn gerade bei Expertensystemen ist dieses wissenschaftliche Wissen häufig Gegenstand der Wissensdomäne.

Da Wissen nach Becker und Steven nicht von vorneherein schon explizierbar ist, sind die Möglichkeiten zur Beschreibung von Wissen und die daraus erwachsenden Konsequenzen ebenfalls von großer Bedeutung, wenn es darum geht, ein aus wissenssoziologischer Sicht adäquates Wissenskonzept zu entwickeln. Dazu gehören:

Wissen ist nicht eindeutig beschreibbar. „However, in colloquial, everyday communication, they [concepts][7] carry meanings which are beyond the concrete functions that they also take on here.“ ([Becker and Steven 1991] S. 8)

Die Beschreibung von Wissen enthält Widersprüche, die durch die subjektive und soziale Konstruktion von Wirklichkeit entstehen. ([Becker and Steven 1991] S. 8)

Wissen kann nur durch konkrete Aktionen und Handlungen identifiziert werden, ist also nur anhand von Fällen und Beispielen artikulierbar[8].

Der vortheoretische Hintergrund von Wissen bzw. von der Entstehung von Wissen ist nicht artikulierbar. Wird er dennoch artikuliert, so wurde bereits eine Theorie über diesen ehemals vortheoretischen Hintergrund gebildet, er ist also nicht mehr vortheoretisch. (siehe auch [Becker 1991] S. 7/8).

Die Artikulation von Wissen führt dazu, daß Wissen reorganisiert wird, was auch als konstruktiver Prozeß aufgefaßt werden kann. Dabei werden zu vortheoretischem Wissen Theorien aufgestellt, vages Wissen wird präzisiert, unstrukturiertes Wissen wird strukturiert etc. Auch die Wissende lernt bei der Artikulation von Wissen dazu, bildet sich eine Theorie bzw. ein Modell von ihrem Wissen. (siehe dazu auch [Becker 1991] S. 12) Durch die gängige Praxis in der Wissensakquisition kann die Expertin allerdings dazu verleitet werden, vages Wissen gar nicht zu artikulieren, da es ja nicht präzise beschreibbar und nicht exakt verifizierbar ist.

[7] Konzepte zur Beschreibung von Wissen.

[8] „Our knowledge of the world can only be identified in concretely definable actions;“ ([Becker and Steven 1991] S. 7) und „[...] knowledge [...] can only ever be described in terms or »with reference to something«, and not »in itself« with regard to its validity.“ ([Becker 1991] S. 8).

Wird Wissen unabhängig vom wissenden Subjekt dargestellt, erhält es einen objektiven Status. Becker und Steven weisen in diesem Zusammenhang auf die Gefahr hin, daß das Model über die Realität siegt („that model triumphs over reality“ ([Becker and Steven 1991] S. 7)).

Als abschließende Bemerkung wird in ([Becker 1991] S. 13) die Forderung gestellt, daß im Knowledge Engineering eine Sicht, die durch die Begriffe „knowledge modeling“ oder „knowledge construction“ beschrieben werden kann, eher adäquat ist, als die klassische „knowledge extraction“. Bei dem Versuch, Wissen explizit zu machen, kann also allenfalls ein Modell des Wissens einer Expertin explizit gemacht werden. Wenn das Modell adäquat sein soll, müssen die hier genannten Faktoren bei der Modellbildung, also der Wissensakquisition berücksichtigt werden. Ein Beispiel für einen Ansatz, der diesen Anforderungen gerecht wird, ist das Wissensakquisitionskonzept *sloppy modeling*, das in Kapitel 11 eingehender erläutert wird.

Kapitel 10

Wissensmodellierung mit KADS
Autor: Jürgen Schönig

Einleitung

Im folgenden wird die Modellierung mit der Methodologie KADS (Knowledge Analysis and Design Support[1]) nach Hinweisen auf die Modellierungsperspektiven[2] untersucht.

In Kapitel 10.1 werden die für die Beurteilung der Modellierungsperspektiven relevanten Aspekte von KADS vorgestellt. Aufbauend auf die Vorstellung der einzelnen Modelle und des gesamten Entwicklungsprozesses erfolgt in Kapitel 10.2 eine Bewertung der Modellierungsperspektive von KADS auf drei verschiedenen Ebenen der Methodologie. Zunächst wird die Modellierungsperspektive untersucht, die in der Theorie der Methodologie KADS eingenommen wird. Anschließend wird die Modellierungsperspektive des für die Modellierung mit KADS charakteristischen konzeptuellen Modells und abschließend werden die für diesen Modellierungsprozeß vorgesehenen Werkzeuge ansatzweise untersucht. Eine Zusammenfassung der Ergebnisse ist in 10.2 zu finden.

10.1 Modellierung mit KADS

Die Modellierung von Wissen zur Entwicklung eines wissensbasierten Systems (WBS) ist ein sehr komplexer Prozeß, der als Wissensakquisition bezeichnet wird. Er besteht aus der Analyse des Problembereichs, der Erhebung von Wissen aus verschiedenen Wissensquellen ('knowledge elicitation'), der Strukturierung,

[1] [Wielinga et al. 1993b]
[2] siehe Gernot Grube in Teil 2 dieses Bandes

der Interpretation der erhobenen Daten bis hin zur Umsetzung in eine operationale Wissensbasis und deren inhaltlicher Wartung [Wachsmuth 1993]. Die Komplexität dieses Prozesses erfordert ein methodisches Vorgehen bei der Wissensmodellierung, für das verschiedene, zum Teil entgegengesetzte Ansätze existieren. Zum einen gibt es für bestimmte Probleme spezifische Werkzeuge[3] in Form von Tools und Shells, zum anderen gibt es die modellbasierte Methodologie KADS.

Die Methodologie KADS soll eine strukturierte Vorgehensweise bei der Entwicklung und Wartung von wissensbasierten Systemen unterstützen. Dabei besteht eine Schwierigkeit darin, daß es sich bei einer Methodologie nicht nur um eine Methode, sondern vielmehr um eine Sammlung von Methoden handelt, die sich im Falle von KADS ständig weiterentwickeln und die von an der Entwicklung beteiligten Gruppen unterschiedlich interpretiert werden. Der Methodologiebegriff bei KADS umfaßt neben den Methoden auch Theorien, auf denen aber nur sehr vage beschrieben die Methoden basieren sollen, die werden [Wielinga et al. 1993b]. Die informelle Darstellung und Definition der zugrundeliegenden Konzepte erschwert den Umgang mit KADS erheblich.

KADS basiert auf der Grundlage eines modellbasierten Entwicklungsprozesses, bei dem ein von Details der Implementierung unabhängiges konzeptuelles Modell der Problemlösung über den Zwischenschritt eines Designmodells in die Implementierung übertragen wird. In diesem Ansatz wird Wissensakquisition nicht mehr einfach als Tansfer von Wissen eines Experten in ein System verstanden, wie es beispielsweise beim Rapid-Prototyping der Fall ist [Karbach und Linster 1990], sondern als Modellierung von Wissen auf verschiedenen Abstraktionsebenen (siehe Abb.10.1). Diese Vorgehensweise wird durch die Aufteilung der Modellierung in verschiedene Modelle strukturiert, wodurch es dem Knowledge Engineer möglich ist, seine Aufmerksamkeit auf einzelne, ausgewählte Aspekte der Problemlösung zu fokussieren.

KADS fordert eine Aufteilung in sechs verschiedene Modelle (siehe Abb.10.1), die allerdings nicht isoliert sondern stark miteinander verwoben betrachtet werden sollen. Die ersten vier Modelle, die sogenannten Umgebungsmodelle, sind das Organisations-, das Aufgaben-, das Akteur- und das Kommunikationsmodell. Das Organisationsmodell dient der Analyse des Einsatzbereiches und der Umgebung des zu realisierenden WBS. Hier wird ein Problemverständnis für die Domäne und die Rolle bzw. die Wirkung des WBS entwickelt. Im Aufgabenmodell wird die Struktur der globalen Aufgaben des WBS untersucht und der Umfang des zukünftigen Systems festgelegt. Die globalen Aufgaben werden in Teilaufgaben zerlegt, denen Akteure zugeordnet werden. Im Akteurmodell werden die Fähigkeiten und Rollen beschrieben, die beteiligte Akteure bei der Lösung von Teilproblemen benötigen und einnehmen. Akteure sind der Benutzer und

[3] Exemplarisch seien hier BABYLON und NEXPERT OBJECT genannt [Karbach und Linster 1990].

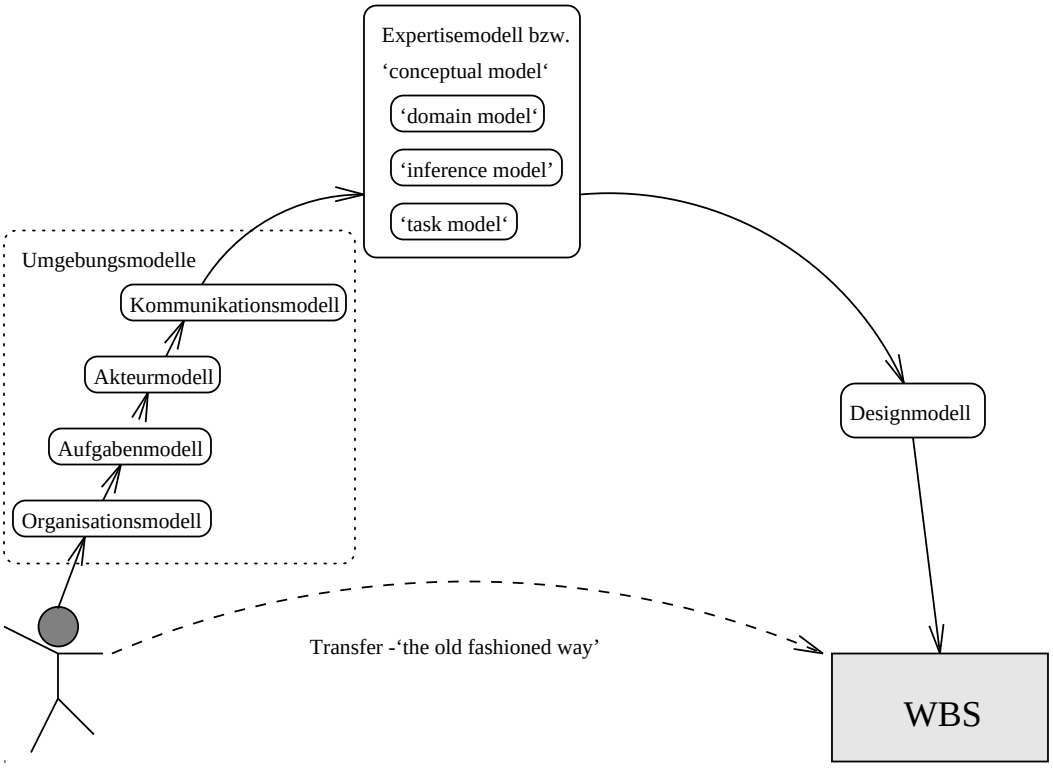


Abbildung 10.1: Modellierung mit KADS

einzelne Problemlösungskomponenten, Systemkomponenten, wie beispielsweise die Benutzerschnittstelle, oder externe Komponenten, wie Datenbanksysteme oder Simulatoren. Im Kommunikationsmodell wird die Interaktion zwischen den einzelnen Akteuren genauer festgelegt und dokumentiert.

Das Hauptgewicht bei der Modellierung mit KADS liegt bei der Entwicklung des Expertisemodells (’conceptual model’). Es dient der Beschreibung des zur Problemlösung benötigten Wissens des Experten und bildet dadurch den Kern der Wissensmodellierung. Dieses Modell wird auf einer abstrakten, implementierungsunabhängigen Ebene, dem von Newell [Newell 1982] eingeführten ’knowledge-level’ beschrieben. Hier wird das für den Problemlösungsprozeß benötigte Wissen strukturiert und durch die Beschreibung explizit gemacht. Der ’knowledge-level’ wurde von Newell in seiner ’knowledge level hypothesis’ [Newell 1982] als Lösungsansatz zur Beendigung der Stagnation der KI-Forschung formuliert. Die Stagnation führte er auf eine zu starke Konzentration auf konkrete Repräsentationsfragen zurück. Newell fehlte eine Auseinandersetzung mit den Fragen, die hinter den Techniken der Wissensrepräsentation stehen. Seiner Meinung nach sollten die Fragen nach dem ’Wie’, durch Fragen nach dem ’Warum’ abgelöst werden. Dazu führte er eine Unterscheidung von verschiedenen Ebenen der Wissensbeschreibung ein, durch die der ’knowledge-level’ von

dem ’symbol-level’ getrennt wird. Die Repräsentation des Wissens wird als eine Struktur auf der symbolischen Ebene betrachtet, die Wissen auf einer höheren Ebene beschreiben soll:

> *“There exists a distinct computer systems level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the law of behavior.“* ([Newell 1982] S.99)

KADS fordert bei der Modellierung zunächst die konzeptuelle Darstellung des zu modellierenden Sachverhalts auf dem ’knowledge-level’. Es soll also kein ausfühbares Modell auf dem ’symbol-level’, in Form einer Implementierung in einer Programmiersprache, sondern ein Modell auf einem abstrakteren Niveau entwickelt werden. Die Modellierung auf diesem ’knowledge-level’ beschreibt die Organisation des Wissens, wobei nicht nur das Wissen selbst, sondern auch die Rolle des Wissens im Problemlösungsprozeß implementierungsunabhängig beschrieben wird. Diese Beschreibung soll es den an der Modellierung beteiligten Personen ermöglichen, sowohl bei der Entwicklung, als auch bei der späteren Wartung, über das Modell zu kommunizieren. Charakteristisch für diese Modellierung auf dem ’knowledge level’ ist die Kategorisierung des Wissens in Domain- (’domain knowledge’), Inferenz- (’inference knowledge’) und Aufgabenwissen (’task knowledge’), wodurch drei weitere Modelle entsehen:

1. Das ’domain model’ ist eine koherente Sammlung einzelner Aussagen, die einen spezielle Sicht auf die Domäne ermöglichen und diese beschreiben. Typische ’domain models’ sind kausale oder hierarchische Strukturen. Hier werden die für die Problemlösung relevanten Konzepte mit ihren Eigenschaften und die Relationen zwischen den Konzepten beschrieben.

2. Im ’inference model’ werden elementare Inferenzen definiert. Es wird also festgelegt, welche Konzepte und Relationen aus dem ’domain model’ im Problemlösungsprozeß benötigt werden.

3. Im ’task model’ werden die elementaren bzw. atomaren Inferenzen aus dem ’inference model’ zu komplexeren Inferenzen kombiniert, wodurch eine Hierarchie von Teilaufgaben entwickelt wird. Hier wird beschrieben, welche Schlußfolgerungsschritte auf dem ’domain knowledge’ durchgeführt werden können und wie dies geschehen soll. Dadurch verbindet dieses Modell Wissen aus dem ’domain model’ mit Wissen aus dem ’inference model’.

Eine Umsetzung des ’conceptual model’ in ein operationales Modell soll durch die Entwicklung eines Designmodells unterstützt werden. Hier werden Entscheidungen getroffen, die zur technischen Realisierung des Systems notwendig sind.

Dabei werden die Architektur des Systems und die Spezifikation für die Implementierung festgelegt.

Diese Modelle dienen neben der Strukturierung des Entwicklungsprozesses auch als Checkliste für die Analyse des Problembereichs und zur Strukturierung der Dokumentation, die parallel erstellt werden soll.

Um den Entwicklungsprozeß so flexibel wie möglich zu gestalten, schlägt KADS ein Projektmanagement in Form eines spiralförmigen Prozeßmodells vor, das aus aufeinanderfolgenden Zyklen ('life-cycle-model') besteht [Bauer und Löckenhoff 1994]. Ein Zyklus beginnt mit der Festlegung der Zyklusziele in Abhängigkeit von den bereits erreichten Ergebnissen vorheriger Zyklen und der Vorgehensweisen zur Erfüllung der Ziele. Anschließend werden in einer Risikoanalyse allgemeine Projektrisiken identifiziert und bewertet, um die Vorgehensweise entsprechend anpassen zu können. Beendet wird ein Zyklus nach der eigentlichen Modellierung durch eine Qualitätskontrolle, bei der überprüft wird, ob die gesteckten Ziele tatsächlich erreicht wurden.

10.2 Modellierungsperspektiven bei KADS

Im folgenden wird auf die Rolle der Modellierungstheorie, der Modellierung des konzeptuellen Modells und der Werkzeuge im Hinblick auf die Modellierungsperspektiven eingegangen.

Theorie der Wissensmodellierung

Das Ziel der Wissensmodellierung und des Software Engineering, ist die strukturierte Entwicklung von Programmen[4]. Es scheint jedoch so zu sein, daß die Entwicklung von WBS einem anderen Paradigma als dem des klassischen Software Engineering unterliegt [Puppe 1991]. Das Paradigma des klassischen Software Engineering[5] basiert auf dem Vorhandensein einer präzisen Spezifikation des zu lösenden Problems, mit der die Implementierung validiert und mit der ihre Korrektheit (zumindest im Prinzip) bewiesen werden kann. Der Anwendungsbereich der Wissensmodellierung zeichnet sich jedoch gerade durch das Fehlen dieser Spezifikation aus. Ein typisches Beispiel für die Wissensmodellierung ist die Entwicklung von Diagnosesystemen. Wie soll hier der Zusammenhang zwischen Symptomen und Diagnosen spezifiziert werden, wenn nicht durch den Aufbau einer geeigneten Wissensbasis[6]? Die Spezifikation eines Diagnosesystems wäre al-

[4]WBS sind Softwaresysteme.

[5]F. Puppe verweist (in [Puppe 1991] S.179) auf [Dijkstra 1976] als einen Vertreter des klassischen Software Engineering.

[6]Unter einer Wissensbasis ist die Ansammlung von Fakten und Regeln zu verstehen.

so identisch mit seiner Wissensbasis.

Bei der Wissensmodellierung muß auf Grund der fehlenden Spezifikation und dem Wunsch nach Kommunizierbarkeit des Modells, der Anspruch auf Korrektheit und Vollständigkeit durch mehr oder weniger vage Minimalanforderungen ersetzt werden. Zu diesen Minimalanforderungen gehört beispielsweise, daß das System keine groben Fehler machen darf. Die Überprüfung dieser Anforderung erfolgt durch die Evaluierung des Modells. Es kann also nur eine Aussage bezüglich des bereits getesteten Verhaltens gemacht werden. Weitere Anforderungen sind Plausibilität, Nachvollziehbarkeit, Erweiterbarkeit und Änderbarkeit der Modelle. Die Änderbarkeit ist ein weiterer Grund für eine fehlende Spezifikation bei der Wissensmodellierung, da sich die Anforderungen an ein WBS häufig während dessen Entwicklung und späteren Wartung ändern. Für die Abbildperspetive ist eine Spezifikation erforderlich. Insofern kann das Fehlen einer Spezifikation bei Wissensmodellierung als ein Hinweis auf die Konstruktionsperspektive gewertet werden.

Modellierung von Wissen erfordert zunächst die Erhebung des zu modellierenden Sachverhalts im Rahmen der 'knowledge elicitation'. Typische Erhebungstechniken sind unstrukturierte, strukturierte und fokussierte Interviews, Beobachtungen in Form von Protokollanalysen, Introspektion und indirekte Techniken wie beispielsweise hierachisches Clustering, multidimensionale Skalierung oder gewichtete Netze [Karbach und Linster 1990]. Oft sind Experten aber unfähig ihr Wissen zu beschreiben, da sie Probleme nicht durch die Benutzung von symbolischen Wissen, sondern durch holistische Modelle lösen [Dreyfus und Dreyfus 1986]. Wird ein Experte zu seinem Wissen befragt, so denkt dieser über seine eigenen Problemlöseprozesse nach und bildet sich so ein eigenes Modell seines Vorgehens. Bei der Artikulation seines Modells wird dieses wieder reorganisiert, da er es im Rahmen der Kommunikation mit dem Knowledge Engineer symbolisch repräsentieren muß. Eine mentale Repräsentation reicht nicht aus. Der zu modellierende Sachverhalt ist also nichts Gegenständliches und kann deshalb weder vom Experten noch vom Knowledge Engineer direkt 'erfaßt' werden. Da dieses Modell nicht operational ist und nicht auf Korrektheit überprüft werden kann, muß man von einer 'naiven' Theorie des Experten über sein Wissen sprechen. Grundlage für die weitere Modellierung ist also eine 'naive' und nicht eine explizit beschriebene Theorie. Aus dieser Theorie müssen die für das Modell relevanten Daten extrahiert und anschließend formalisiert werden. Wichtig ist die Feststellung, daß diese Theorie nicht Bestandteil des Wissens ist, das modelliert werden soll, sondern aus dem Wissen bereits modelliert wird. Bei der Modellierung kann also kein Abbild der Wirklichkeit, sondern maximal nur ein Abbild einer Theorie von kognitiven Prozessen gemacht werden. Das heißt, der Sachverhalt, der modelliert werden soll, existiert nicht ohne das zu erstellende Modell. Diese Abhängigkeit zwischen Sachverhalt und Modell spricht gegen eine Abbildperspektive bei der Modellierung.

Ursprünglich wurde dieser Prozeß nur als Transformation von Wissen aus einem menschlichen Experten in ein WBS betrachtet. In der Praxis hat sich gezeigt, daß die Überbrückung der Kluft zwischen dem menschlichen Wissen und dem lauffähigen WBS eine besonders schwierige und nur approximativ lösbare Aufgabe ist, da es sich um schlecht strukturierbare bzw. algorithmisierbare, diffuse Problembereiche handelt. Hier zeigen sich in der Praxis, bei der Modellierung mit den klassischen Modellierungsmethoden des Software-Engineering, nicht zu vernachlässigende Probleme. Das Wasserfallmodell ist in der Regel unangemessen, da es eine exakte Spezifikation vorraussetzt. Bei der Modellierung von Wissen ist eine derartige Spezifikation aber nicht möglich, zumal wechselnde Anforderungen, wie sie im Knowledge Engineering üblich sind, im starren Entwicklungsprozess des Wasserfallmodells nicht unterstützt werden. Beim Rapid-Prototyping treten Probleme auf, da zwischen dem Experten und dem Knowledge Engineer erst dann über das Modell kommuniziert werden kann, wenn der erste Prototyp lauffähig ist [7]. Die Kommunikation beschränkt sich auch nur auf die Funktionalität des Modells, da dessen Beschreibung in Form einer Implementierung zu weit von der Terminologie des Experten entfernt ist. Auf Grund dieser Probleme wird die Wissensmodellierung bei KADS durch die Einführung eines konzeptuellen Modells strukturiert. Das konzeptuelle Modell dient als Spezifikation für die weitere Modellierung und kann als neu konstruiertes Original aufgefaßt werden. Da für KADS die Entwicklung des konzeptuellen Modells charakteristisch ist und die Umsetzung des konzeptuellen Modells in ein operationales Modell über das Designmodell bislang noch nicht ausgearbeitet ist, wird im folgenden nur auf die Modellierung des konzepuellen Modells eingegangen.

Modellierung des konzeptuellen Modells

Im modellbasierten Ansatz KADS wird versucht, auf eine strenge Trennung von Analyse auf der einen Seite und der Repräsentation auf der anderen Seite zu achten. Aus den analysierten Daten wird zunächst ein Modell auf einem höheren Abstraktionsniveau formuliert, das nicht auf der Ebene der üblichen Implementierungsformalismen angesiedelt ist, sondern welches eine Beschreibung des zu modellierenden Wissens ermöglicht. Bei dieser Modellierung werden unwesentliche Aspekte der Realität weglassen und damit die Aufmerksamkeit auf die wesentlichen Gesichtspunkte konzentriert. Dieser Abstraktionsprozeß muß nachvollziehbar sein, so daß eine inhaltliche Beurteilung der Modellierung möglich ist und die an der Modellierung beteiligten Personen, also auch der oder die Experten, über das Modell kommunizieren können. Diese Einbeziehung des Menschen in den Entwicklungsprozeß ist charakteristisch für KADS und kann als Hinweis auf die Konstruktionsperspektive gewertet werden.

[7] Dies gilt zumindest für das Rapid-Prototyping in der extremen, rein theoretischen Form.

Kommunizierbarkeit des konzeptuellen Modells ist die Vorraussetzung für die Entwicklung einer gemeinsamen Sicht auf das zu modellierende Wissen und soll die Entwicklung eines gemeinsamen Modells unterstützen. Das konzeptuelle Modell steht zwischen dem Experten auf der einen und dem Knowledge Engineer auf der anderen Seite. Dabei unterscheidet sich die Interpretation des konzeptuellen Modells durch den Experten mit Sicherheit von der des Knowledge Engineer. Auf beiden Seiten werden unterschiedliche Kontexte gegeben sein, die eine gemeinsame Interpretation des Modells ausschließen. Gegen die Abbildperspektive und für die Konstruktionsperspektive spricht, daß die Struktur des Expertenwissens nicht erfaßt werden kann. Die Beziehung zwischen dem Modell und dem Expertenwissen kann nicht strukturerhaltend sein, da eine angenommene Struktur des Expertenwissens sich erst durch das Modell ergibt. Es gibt also keine feste Beziehung zwischen dem zu modellierenden Wissen und dem konzeptuellen Modell. Ziel der Modellierung muß es sein, beide Seiten auf etwas Ähnliches referenzieren zu lassen, d.h. einen Konsens zu finden.

Werkzeuge zur Modellierung des konzeptuellen Modells

Werkzeuge zur Unterstützung der Modellierung des konzeptuellen Modells können nicht als eigenständig betrachtet werden. Sie sollen nur den Modellierungsprozeß im Rahmen einer gewählten Problemlösungsmethode unterstützen. Folglich läßt sich der Einfluß der Werkzeuge auf die Modellierungsperspektive nicht von den Methoden trennen. Unterschiedliche Methoden können auch unterschiedliche Werkzeuge erfordern. Bei der Betrachtung der Wekzeuge lassen sich allenfalls stärkende oder schwächende Argumente finden, die bei der Betrachtung der Theorie der Modellierung bzw. bei der Modellierung der Modelle nicht sichtbar sind. Sicher ist es aber möglich, daß sich die Modellierungsperspektive durch die Anwendung von Werkzeugen und die damit verbundene Automatisierung des Modellierungsprozesses verändert.

Bei KADS werden als Werkzeuge sogenannte KADS-orientierte Sprachen verwendet, die unterschiedliche Formalisierungen bezüglich der Form und des Inhalts der Modellierung ermöglichen [Wielinga et al. 1993b]. Es gibt also keinen einheitlichen Formalismus für die Modellierung mit KADS, sondern diverse formale Sprachen, die sich mehr oder weniger für eine konkrete Modellierung eignen. Eine diese formalen Sprachen ist die besonders ausdrucksmächtige und deklarative Sprache ML^2, bei der eine Programmverifikation über einen Beweiser unterstützt wird. Weitere KADS-orientierte Sprachen sind KARL, MODEL-K, OMOS und MoMo. MODEL-K und OMOS besitzen die Möglichkeit der Übersetzung von Formalismen nach BABYLON, einem hybriden Werkzeug zur Wissensrepräsentation. Bei MoMo wird auf eine starke Formalisierung, wie sie beispielsweise bei ML^2 gefordert wird, zugunsten einer graphischen Repräsentation verzichtet.

Gemeinsam ist diesen Sprachen eine relativ deklarative Darstellung, die Möglichkeit der Trennung zwischen von der Domäne abhängigem und von ihr unabhängigem Wissen, sowie die Möglichkeit der Trennung zwischen Inferenz- und Aufgabenwissen. Sie unterscheiden sich allerdings im Maß ihrer Ausführbarkeit, ihrer Verifizierbarkeit und ihrer Ausdrucksstärke. Je deklarativer und ausdrucksmächtiger eine Sprache ist, desto größer sind die Probleme ein ausführbares Modell zu entwickeln. Dieses Problem steht zunächst aber bei der Modellierung auf dem 'knowledge level' im Hintergrund, da hier ja implementierungsunabhängig modelliert werden soll. Bei der Modellierung eines konkreten Sachverhalts ist davon auszugehen, daß keine dieser Sprachen alle an sie gestellten Anforderungen ausreichend erfüllt. In diesem Fall wird bei der Modellierung zwischen verschiedenen Formalismen gewechselt.

Alle diese Sprachen haben letztendlich das Ziel, eine konzeptuelle Darstellung des zu modellierenden Sachverhalts zu ermöglichen. Ziel dieser Sprachen sollte sein, daß sie möglichst leicht zu verstehen und anzuwenden sind[8]. Die Kommunizierbarkeit des konzeptuellen Modells soll neben der Verwendung von Werkzeugen durch eine parallele Dokumentation der Modellierung unterstützt werden. Über Form und Umfang wird nichts gesagt.

Bei der Modellierung des konzeptuellen Modells wird von der Annahme ausgegangen, daß sich Wissen auf dem 'knowledge level' formalisieren läßt. Die Formalisierung des Wissens beinhaltet Aspekte der Abbildperspektive. Jede Formalisierung des konzeptuellen Modells verändert die Perspektive auf den zu modellierenden Sachverhalt durch die Einschränkung auf einen Formalismus. Jeder Formalismus ist notwendigerweise auf dem 'symbol-level' angesiedelt. Die bei KADS geforderte Trennung zwischen 'knowledge-level' und 'symbol-level' ist also nur theoretischer Natur. Wenn beide Ebenen vonneinander klar zu trennen wären, dann wäre es gar nicht möglich Sachverhalte auf dem 'knowledge-level' formal zu beschreiben. Wie oben beschrieben unterscheiden sich die Werkzeuge allerdings in der Stärke der Formalisierung. Je stärker diese Formalisierung ist, desto näher ist das konzeptuelle Modell am 'symbol-level'. Und umgekehrt, je weniger formal das konzeptuelle Modell ist, desto näher ist es am 'knowledge-level'. Ziel des Modellierungsprozesses ist die Entwicklung eines Modells auf dem 'symbol-level' in Form einer Implementierung. Insgesamt kann der Modellierungsprozeß als Entwicklung von verschiedenen, unterschiedlich formalen Darstellungen betrachtet werden. Dabei wird umsomehr die Abbildperspektive eingenommen, je näher eine Formalisierung am 'symbol-level' liegt. Stärkere Formalisierungswerkzeuge, wie ML^2, besitzen die Möglichkeit der Verifikation des Modells. Korrektheit ist ein wichtiges Kriterium der Abbildperspektive. Die Zuordnung von KADS zur Konstruktionsperspektive muß also eingeschränkt werden.

[8] Dies scheint aber nicht der Fall zu sein.

Durch die Strukturierung des Modellierungsprozesses und die geforderte Standardisierung von Problemlösungsprozessen, sowie durch die Verwendung von Werkzeugen bei der Modellierung wird eine Automatisierung ermöglicht. Mit Hilfe von sogenannten KADS-Tools sollen in Zukunft Bibliotheken zur Verfügung gestellt werden, in denen beispielsweise Rahmen für diverse Such-, Klassifikations-, Diagnose-, Design- und Planungsmethoden zu finden sind. Rahmen sind Grundbausteine in Form von Standardmodellen, die für die jeweilige Anwendung nur noch ausgefüllt werden müssen. Dem Knowledge Engineer würde sich aber nach wie vor die Frage stellen, welches Wissen benötigt wird. Er müßte entscheiden, welche Instanziierung der Rahmen geeignet ist und wie die durch die Rahmen entstandenen Teilmodelle kombiniert werden könnten. Die Automatisierung hätte insofern einen Einfluß auf die Modellierungsperspektive, als dadurch Formalismen vorgegeben wären, die vielleicht stärker oder schwächer sind als Formalismen, die ohne Automatisierung verwendet worden wären.

Zusammenfassung

Als Ergebnis der Untersuchung läßt sich festhalten, daß sich die Einordnung von KADS in Abbild- und Konstruktionsperspektive nicht eindeutig durchführen läßt. Der theoretische Anspruch von KADS fordert die Konstruktionsperspektive. Die Realisierung der Modellierung verführt aber zur Abbildperspektive.

Die Modellierungsperspektive ist abhängig von diversen Faktoren, die zum einen auf der Seite der zu modellierenden Probleme und Aufgaben, also der jeweiligen Anwendungsbereiche, und zum anderen auf der Seite der zur Verfügung stehenden Methoden bzw. der dazugehörigen Werkzeuge und der damit verbundenen Abstraktionsebenen der Modellierung liegen. Unter der Berücksichtigung dieser Einschränkung läßt sich der Modellierungsansatz von KADS insgesamt eher der Konstruktionsperspektive zuordnen, was für den Anwendungsbereich der Wissensmodellierung angemessen erscheint. Im Rahmen der Formalisierung des konzeptuellen Modells wird aber auch die Abbildperspektive eingenommen (siehe Abb.10.2).

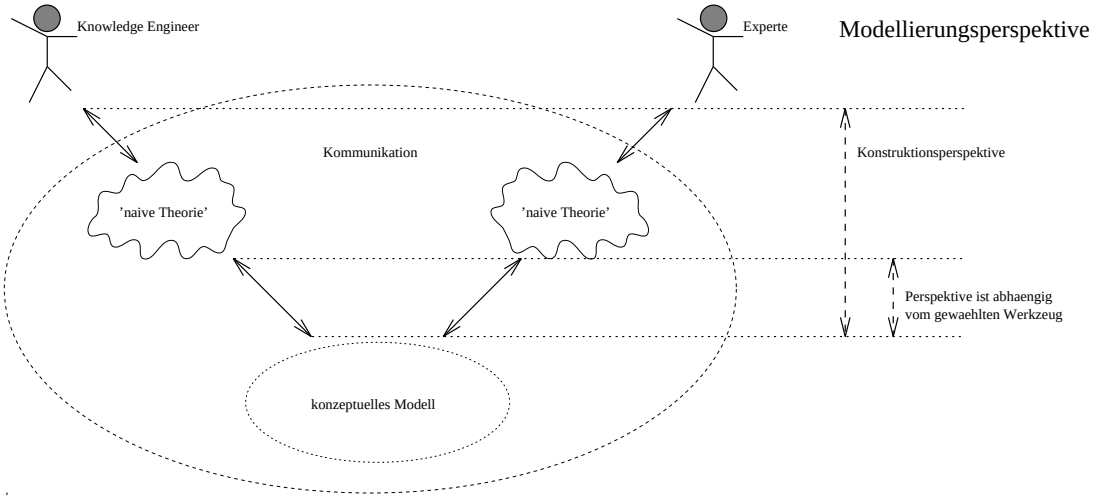


Abbildung 10.2: Modellierungsperspektiven

Kapitel 11

Sloppy Modeling
Autorin: Birgit Schelm

Im folgenden stelle ich das Wissensakquisitionskonzept „*sloppy modeling*“ vor, einen Ansatz für die Entwicklung von Systemen, die die Wissensmodellierung nicht nur unterstützen, sondern durch den Einsatz von maschinellem Lernen das entstehende Modell aktiv mitgestalten. Die Einordnung des *sloppy modeling* Konzepts in die Modellierungsperspektiven möchte ich im wesentlichen auf die explizit und implizit in diesem Konzept enthaltenen Aspekte von Wissen, im Vergleich zu den in Kapitel 9 erläuterten Vorstellungen von Wissen, stützen. Meine abschließende These, daß der *sloppy modeling* Ansatz der Konstruktionsperspektive zugeordnet werden kann, werde ich allerdings auch auf der Grundlage der Verwendung des Modellierungsbegriffs und der berücksichtigten Personengruppen, einiger weiterer Kriterien, die aus dem *sloppy modeling* Ansatz hervorgehen, sowie der Kriterien für die Konstruktionsperspektive[1] begründen.

11.1 Sloppy Modeling

Sloppy modeling ist ein alternatives Konzept für Wissensakquisitionssysteme, das Katharina Morik in [Morik 1989] aus der Kritik an bestehenden Wissensakquisitionssystemen entwickelt. *Sloppy modeling* basiert auf der Annahme, daß Wissensakquisition kein Transferprozeß, sondern eine Modellierungsaufgabe ist, die von einem Wissensakquisitionssystem entsprechend unterstützt werden muß, und zu der das System seinen Teil beiträgt. Wie Morik ihren Ansatz entwickelt und begründet und was in diesem Zusammenhang unter Wissen, Wissensakquisition und Modellierung verstanden wird, erläutere ich im folgenden.

[1] Diese Kriterien werden von Gernot Grube im 1. Teil dieses Berichts erläutert.

Unterscheidung in Wissen und Können

Morik betrachtet in ihrem Ansatz nicht nur Wissen, sondern unterscheidet Wissen und Können. In diesem Zusammenhang stellt Wissen etwas Explizierbares und mehr oder weniger Bewußtes dar, während Können nicht explizierbar und unbewußt ist. Auf dieser Grundlage werden auch unterschiedliche Arten von Expertisen[2] unterschieden: Expertisen, die eher auf Können basieren, und Expertisen, die eher auf Wissen basieren.

Um in einem wissensbasierten System (WBS) beide Arten von Expertise zu berücksichtigen, ist es also nötig, auch das Können von Expertinnen zu beschreiben. Diese Beschreibung von Kompetenzen, stellt für Morik eine wissenschaftliche Aufgabe („scientific task“) dar, die sie als Modellierung bezeichnet. Das entstehende Modell ist eine explizite, erklärbare und operationale Theorie der Domäne[3]. An den Anforderungen, die an dieses Modell gestellt werden, lassen sich dann auch die unterschiedlichen Richtungen der KI erkennen: Die eine Richtung behauptet, das Modell sei eine isomorphe Repräsentation der menschlichen Kompetenz und Handlung[4], während der anderen Richtung ausreicht, daß das Modell ein gutes Verhalten zeigt[5].

Wissenschaftliche und „naive“ Theorien

Wie bereits erwähnt, bezeichnet Morik mit Modellierung eine wissenschaftliche Aufgabe bzw. einen wissenschaftlichen Prozeß. Dabei wird aber herausgestellt, daß beim Erstellen einer wissenschaftlichen Theorie keine besonderen kognitiven Prozesse zum Einsatz kommen. Im Gegenteil: Beim Erstellen einer wissenschaftlichen und einer alltäglichen Theorie kommen die gleichen kognitiven Prozesse zum Tragen[6]. Der einzige Unterschied zwischen alltäglichen und wissenschaftlichen Theorien ist der, daß wissenschaftliche Theorien durch die verschiedenen Gepflogenheiten der Gemeinschaft der Wissenschaftler (“scientific community“) systematisch Zweifeln und Gegenbeispielen ausgesetzt werden. Für die Modellierung folgt daraus, daß eine Expertin, die Expertin im Ausführen einer Tätigkeit ist und nicht im Erklären der Ausführung dieser Tätigkeit, zunächst eine „naive“ Theorie über die von ihr ausgeführte Tätigkeit entwickeln wird, wenn sie danach gefragt wird. Diese naive Theorie wird als erste Annäherung im Modellierungs-

[2] Unter Expertise wird in [Morik 1989] das Wissen und/oder Können verstanden, das eine Expertin ausmacht, allerdings auf einer vortheoretischen Ebene. Expertise wird hier nicht als erste oder „fertige“ Theorie des Expertinnenwissens bzw. -könnens verstanden.

[3] „explicit, explainable, and operational theory of a domain“([Morik 1989] S. 113)

[4] „an isomorphic representation of the human competence and performance“([Morik 1989] S. 114)

[5] „that the model gives a good performance“([Morik 1989], S.112 - 114)

[6] „The same cognitive processes are involved in building scientific and every-day life theories.“([Morik 1989] S. 114)

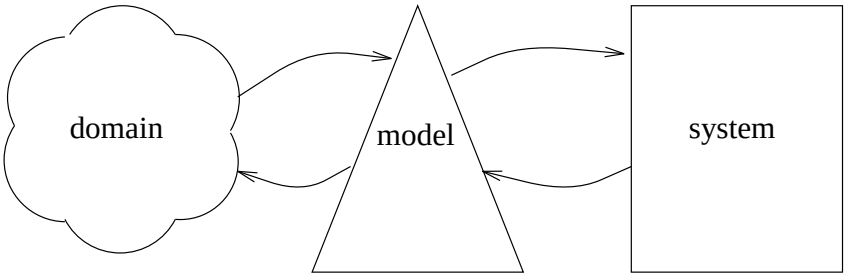
prozeß betrachtet. Um aber dem Ansatz gerecht zu werden, daß Modellierung ein wissenschaftlicher Prozeß sei, muß diese naive Theorie einen wissenschaftlichen Prozeß durchlaufen, d.h. systematisch Zweifeln und Gegenbeispielen ausgesetzt werden, um aus der naiven Theorie eine wissenschaftliche Theorie zu entwickeln.

Der interaktive Charakter der Modellierung

Morik geht von der Annahme aus, daß das benötigte Modell nicht schon vor Beginn des Wissensakquisitionsprozesses im Kopf der Expertin ist, sondern erst während der Wissensakquisition entsteht. Daß das Entstehen dieses Modells ein interaktiver Prozeß ist, wird daran fest gemacht, daß die Qualität des entstehenden Modells z.B. dadurch verbessert werden kann, indem beim Interviewen der Expertin das entstehende Modell durch Gegenbeispiele getestet und abgesichert wird. Selbst allgemeingültige Grundannahmen, auf denen die Modellbildung basiert, oder allgemeine Bedenken gegen eine Theorie, die Wissensingenieurin und Expertin unausgesprochen miteinander teilen, sind irgendwann durch interaktive Prozesse entstanden. Daraus wird gefolgert, daß der Wissensakquisitionsprozeß als interaktiver Prozeß betrachtet werden muß und daß das Modell im Verlaufe dieses Prozesses interaktiv von Expertin und Wissensingenieurin konstruiert wird.

Modellierung als zyklischer Prozeß

Die bis dahin beschriebenen Eigenschaften des Modellierungsprozesses verdeutlicht Morik in folgendem Bild:



In dieser Darstellung wird deutlich, daß Modellierung, ebenso wie Wissenschaft, ein zyklischer und kein linearer Prozeß ist, wie er in der „traditionellen“ Transfersichtweise der Wissensakquisition dargestellt wird.

Bei der Untersuchung des Modellierungszyklus unterscheidet Morik drei Phasen und drei Arten der Revision:

1. Phase Hier wird der Modellierungsrahmen festgelegt. Dazu gehört das Festlegen relevanter Aspekte der Domäne, des Vokabulars, in dem Phänomene beschrieben werden können etc. Üblicherweise wird diese Phase nicht vom System unterstützt, sondern meist mit Papier und Bleistift vollzogen.

2. Phase In der zweiten Phase wird der Rahmen, der in der ersten Phase festgelegt wurde, mit Fakten und Regeln ausgefüllt.

3. Phase Die dritte Phase dient dazu, das bis zu diesem Zeitpunkt erstellte Modell zu testen und zu evaluieren. Dabei sollen sowohl Konflikte aufgezeigt werden, die mit den gegebenen Informationen nicht gelöst werden können, als auch Lücken im Modell entdeckt werden.

Um diese Unterscheidung in Phasen sinnvoll vornehmen zu können, müssen auch Änderungen des Modells in den jeweils anderen Phasen vorgenommen werden können, wenn der zyklische Charakter des Modellierungsprozesses erhalten bleiben soll. Morik unterscheidet dabei drei Arten von Änderungen:

1. Falls während der zweiten Phase Beobachtungen in der Domäne gemacht werden, die innerhalb des in der ersten Phase festgelegten Modellierungsrahmens nicht beschrieben werden können, so muß es möglich sein, diesen Modellierungsrahmen auch rückwirkend zu ändern.

2. Eine Evaluation des Modells in der dritten Phase kann nur dann sinnvoll sein, wenn Fakten und Regeln revidiert werden können.

3. In der dritten Phase kann aber auch deutlich werden, daß Grundannahmen, die im Modellierungsrahmen festgelegt wurden, revidiert werden müssen.

Aus diesen drei Arten von Revisionsmöglichkeiten folgert Morik, daß ein Wissensakquisitionssystem, das den Modellierungszyklus unterstützt, sowohl die Revision von Fakten und Regeln, als auch die Revision der Terminologie und der Grundannahmen, unterstützen muß.

Modellierung im Gegensatz zur schrittweisen Verfeinerung

Morik stellt heraus, daß das Prinzip der schrittweisen Verfeinerung, das in vielen Wissensakquisitionssystemen verwendet wird, nicht ausreichend ist, um die von ihr geforderten Revisionsmöglichkeiten zu gewährleisten. Den Grund dafür sieht sie darin, daß bei der schrittweisen Verfeinerung die Möglichkeiten für weitere Definitionen, Fakten und Regeln mit jeder neuen Information immer weiter eingeschränkt werden. Demgegenüber muß es möglich sein, Definitionen Beobachtungen anzupassen. Daraus folgt für Morik, daß ein System, das den Modellierungszyklus unterstützt, zunächst alle möglichen Konsequenzen einer Änderung berechnen und dann die beste Änderung auswählen können muß.

Maschinelles Lernen

Aus der Überlegung, daß maschinelles Lernen als automatische Modellierung betrachtet werden kann und daß Wissensakquisition eine Modellierungsaufgabe ist, zieht Morik den Schluß, daß es naheliegend sei, maschinelles Lernen in der Wissensakquisition einzusetzen. Verschiedene bereits existierende Prototypen für maschinelle Lernsysteme lassen sich den bereits erläuterten Modellierungsphasen zuordnen. So können für die erste Phase Systeme zum Einsatz kommen, die Konzepte lernen, während in der zweiten Phase vor allem das Erkennen von Gesetzmäßigkeiten, die Umformung von Regeln, um Konflikte und Fehler zu beseitigen, und das Finden von Erklärungen wichtig sind. Für die dritte Phase ist vor allem die Generierung von Experimenten interessant, mit denen das Modell getestet werden kann.

Damit wird die Expertin nicht gezwungen, eine Theorie direkt explizit zu machen, die sie unter Umständen gar nicht hat, sondern kann Beispiele angeben. Das System versucht, entsprechende Gesetzmäßigkeiten und Strukturen, die in den Beispielen stecken, zu entdecken und explizit zu machen. Auf diese Art und Weise trägt das System dazu bei, überhaupt erst eine Theorie der Expertin zu entwickeln. Auch Experimente zur Überprüfung des Modells werden mit diesem Ansatz möglich.

Balanced Cooperative Modeling

Um die beschriebenen Möglichkeiten maschinellen Lernens in ein Wissensakquisitionssystem zu integrieren, schlägt Morik das Konzept des *balanced cooperative modeling* (siehe dazu [Morik 1989] S. 122) vor. Das Wissensakquisitionssystem unterstützt bei diesem Konzept die Benutzerin, erweitert aber auch das Modell durch Lernen. Dabei muß die Benutzerin keine Expertin im Problemlösen sein, sondern nur Phämonene beschreiben können. Das Verhältnis zwischen Benutzerin und System vergleicht Morik mit dem Verhältnis zwischen einer Wissenschaftlerin und ihrer Assistentin: Die Rolle der Benutzerin ist die einer Wissenschaftlerin, die Beobachtungen aufschreibt, strukturiert und Regeln findet, die die Phänomene erklären. Die Rolle des Systems ist die einer Assistentin, die über die Schulter der Benutzerin schaut, Informationen zusammenträgt, sich um die Buchführung kümmert und Konsequenzen aus Änderungen der Benutzerin bereinigt, auf verborgene Konflikte hinweist, sowie Erweiterungen des Modells empfiehlt[7].

[7] „The role of the user is that of a scientist who writes down obeservations, structures them, and finds rules which cover the phenomena. The role of the system is that of an assistant looking over the user's shoulder, compiling information, taking care of the book-keeping, and cleaning-up consequences of the user's changes, pointing to hidden conflicts, and recommending enhancements for the model." ([Morik 1989] S. 122)

Als Konsequenz aus diesem Konzept verlangt Morik, daß ein System mit einem unvollständigen und möglicherweise falschen Modell arbeiten kann, da die Enstehung des Modells und das Lernen aus dem Modell zur gleichen Zeit stattfindet. Ferner muß das Modell in allen Teilen änderbar sein, da Benutzerin oder System durchaus Fehler machen können. Diesen Ansatz bezeichnet Morik schließlich als *sloppy modeling*, also „schlampiges Modellieren“, da die Benutzerin zunächst mit einem schlampigen Modell beginnen kann, das im Verlauf des Modellierungsprozesses korrigiert und erweitert werden kann.

11.2 Das Modellierungsverständnis von Sloppy Modeling

In diesem Abschnitt möchte ich meine These, daß *sloppy modeling* der Konstruktionsperspektive zugeordnet werden kann, zunächst anhand des in diesem Ansatz enthaltenen Wissenskonzepts erläutern. Anschließend werde ich begründen, daß auch die Vorstellung von Wissensakquisition, die in [Morik 1989] deutlich wird, sowie die Verwendung des Modellierungsbegriffs diese These belegen. In der Erläuterung des *sloppy modeling* Ansatzes selbst sind einige weitere Aspekte enthalten, die als Kriterien für die Zuordnung zur Konstruktionsperspektive herangezogen werden können. Abschließend werde ich noch auf die Kriterien für die Modellierungsperspektiven eingehen.

Das Wissenskonzept

In bezug auf das verwendete Wissenskonzept ist *sloppy modeling* eindeutig der Konstruktionsperspektive zuzuordnen. Der gesamte Ansatz macht deutlich, daß Wissen in diesem Fall nicht als per se explizierbar betrachtet wird, sondern davon ausgegangen wird, daß immer nur ein Modell bzw. eine Theorie der Expertise während des Modellierungsprozesses hergestellt werden kann. Dies zeigt sich schon allein durch die Tatsache, daß es Morik bei der gesamten Erläuterung des Ansatzes im wesentlichen darum geht, wie sich das Modell der Expertise entwickelt und wie diese Entwicklung adäquat unterstützt werden kann. Auch in der Unterscheidung von Expertisen, die auf Wissen und Expertisen, die auf Können basieren, macht Morik diesen Punkt deutlich. Sie sieht Können – im Gegensatz zu Wissen – als etwas nicht Explizierbares an, das mehr oder weniger unbewußt vorhanden ist[8]. Dadurch entsteht für sie die Notwendigkeit, die Kompetenz einer Expertin in irgendeiner Form zu beschreiben. Diese Beschreibung muß allerdings erst entwickelt werden, ist also nicht per se schon vorhanden oder direkt aus der Kompetenz ableitbar. Die Entwicklung einer solchen Beschreibung sieht Morik

[8] „Skill is not explicable or conscious. “ ([Morik 1989] S. 112)

als eine wissenschaftliche Aufgabe an[9] und den Prozeß, eine Beschreibung für die Kompetenz einer Expertin zu entwickeln, als Modellierung[10].

Der nächste Punkt, der für meine These spricht, ist, daß Morik davon ausgeht, daß Expertinnen erst im Verlauf des Wissensakquisitionsprozesses eine Theorie ihrer Expertise entwickeln[11].

Des weiteren berücksichtigt sie, daß sich sowohl Wissen, als auch die Beschreibung von Wissen ändert. *Sloppy modeling* unterstützt die Anforderung an die Änderbarkeit der Beschreibung von Wissen auf allen Ebenen. Durch die Integration von maschinellen Lernkomponenten ermöglicht es Morik außerdem, das Expertinnenwissen anhand von Beispielen explizit zu machen, was ebenfalls Bestandteil der in Kapitel 9.3 genannten Kriterien für ein aus Sicht der Konstruktionsperspektive adäquates Wissenskonzept ist.

In der Umsetzung von *sloppy modeling* in MOBAL, auf die ich hier nicht eingegangen bin, wird der möglichen Widersprüchlichkeit und Unvollständigkeit der Beschreibung von Wissen Rechnung getragen, indem zusätzliche „Wahrheitswerte“ `contradictory` und `unknown` ([Morik 1991] S. 69/70) zu den üblichen Werten `true` und `false` hinzugenommen werden.

Der Wissensakquisitionsprozeß

Auch die Vorstellung, die Morik von dem Wissensakquisitionsprozeß selber hat, spricht für eine Einordnung in die Konstruktionsperspektive. Morik bezeichnet den Wissensakquisitionsprozeß explizit als Modellierungsprozeß[12], dessen interaktive Natur[13] sie ebenfalls herausstellt. Welche Vorstellung Morik dabei von Modellierung hat, werde ich im nächsten Abschnitt vorstellen. An dieser Stelle wäre noch anzumerken, daß Morik ihren Ansatz klar gegen die Transfersichtweise [14] abgrenzt. In [Morik 1989] entwickelt sie ihr Konzept aus der Kritik an „herkömmlichen“ Wissensakquisitionssystemen, die aus dieser Transfersichtweise heraus konzipiert wurden. Dabei stellt sie aus unterschiedlichen Gründen fest, daß diese Systeme die Modellierung der Wissensdomäne nicht unterstützen. Die Revision einer Entscheidung sollte vom System unterstützt werden. Nur dann

[9] „Describing competence is a scientific task.“ ([Morik 1989] S. 113)

[10] „We call the process of describing competence **modeling**.“ ([Morik 1989] S. 113)

[11] „[…] the acquired model was not in the head of the expert before the acquisition process started, but was built up during the knowledge acquisition.“ ([Morik 1989] S. 115)

[12] „knowledge acquisition as Modeling“ ([Morik 1989] S. 112)

[13] „[…] we also regard the process of knowledge acquisition as an interactive one.“([Morik 1989] S. 115)

[14] Diese „Transfer View“ entspricht eher einem Bacon’schen Weltbild, das davon ausgeht, daß alle Phänomene der Welt erkennbar und damit auch beschreibbar sind.

kann es als System bezeichnet werden, das die Modellierung unterstützt[15].

Die Verwendung des Modellierungsbegriffs

Morik sieht Modellierung im Zusammenhang mit Wissensakquisition als Konstruktionsprozeß, der interaktiv[16] und zyklisch ist. Der interessanteste Aspekt ihres Modellierungsbegriffs ist jedoch, daß sie Modellierung als wissenschaftlichen Prozeß sieht. Die Rolle, die diese Sichtweise spielt, und die Ansätze, die sich für *sloppy modeling* daraus ergeben, möchte ich gesondert behandeln.

Ebenfalls als Modellieren, wenn auch als automatisches Modellieren, bezeichnet Morik in diesem Zusammenhang maschinelles Lernen. Dies leitet sie daraus ab, daß durch maschinelles Lernen ein Modell einer Domäne erweitert und verändert werden kann, maschinelle Lernkomponenten also das Modell mitkonstruieren.

Als Modell bezeichnet Morik eine explizite, erklärbare und operationale Theorie einer Domäne, bezeichnet also mit Theorie und Modell das gleiche. Als Auslöser für Modellierung gibt Morik den Bedarf von Erklärungen an. „Modeling is driven by a need for explanation.“ ([Morik 1989] S. 115).

Das Ziel der Modellierung ist in diesem Zusammenhang ein Modell der Wissensdomäne. Allerdings steht dieses Ziel bei Morik weniger im Vordergrund, viel wichtiger ist ihr die adäquate Unterstützung des Modellierungsprozesses durch das System, wodurch ihre Anforderungen an die Darstellung und die Entwicklung einer Darstellung von Wissen erfüllt werden.

Die Rolle wissenschaftlicher Prozesse in Sloppy Modeling

Wie bereits erwähnt betrachtet Morik Modellierung als wissenschaftlichen Prozeß, das Modell einer Wissensdomäne als eine wissenschaftliche Theorie dieser Wissensdomäne. Ein wesentlicher Aspekt dieses wissenschaftlichen Prozesses ist dabei, die Absicherung des Modells bzw. der wissenschaftlichen Theorie durch die Konfrontation mit Gegenbeispielen und systematischen Zweifeln[17]. Morik vergleicht dies mit den Gepflogenheiten einer Gemeinschaft von Wissenschaftlerinnen, vor allem mit den Prüfprozeduren, in denen über die Annahme von Beiträgen zu Konferenzen entschieden wird, sowie Unterhaltungen und Diskussionen im Rahmen von Konferenzen[18]. Als Teil dieses wissenschaftlichen Prozes-

[15] „Revising […] a decision should be supported by the system. Only then can it be called a system supporting modeling.“ ([Morik 1989] S. 111)

[16] „The model of the domain is not already in the head of the expert but is interactively constructed by the expert and the knowledge engineer.“ ([Morik 1989] S. 115)

[17] „[…] scientific theories are systematically exposed to doubt and counter-examples.“ ([Morik 1989] S. 114)

[18] „Institutions such as conferences with their refereeing procedure for acceptance and the discussions after talks guarantee that counter-examples or arguments against a theory are brought up

ses kann man aber auch das Testen des Modells mit Hilfe von Experimenten, die durch maschinelle Lernkomponenten erzeugt werden sollen, sehen.

Morik sieht als Aufgabe der Wissenschaften das Finden von Erklärungen, was in ihrem Ansatz auch vom System, nämlich den maschinellen Lernkomponenten geleistet werden soll[19].

Ein weiterer Punkt, der Aufschluß über das Verständnis von Wissenschaft bei Morik bietet, ist die bereits erwähnte Rollenverteilung von System und Wissensingenieurin.

Die genannten Punkte sprechen für eine Vorstellung von Wissenschaft, die der von T.S. Kuhn[20] entspricht, wie er sie in [Kuhn 1973] beschreibt. Bei seiner Darstellung hebt Kuhn vor allem kommunikative und soziale Aspekte hervor, die sich in der Gemeinschaft der Wissenschaftlerinnen im Zusammenhang mit ihrer Arbeit abspielen. Ein weiterer und wichtiger Punkt ist bei Kuhn, daß die Sichtweise der Wissenschaftlerinnen von dem jeweiligen Paradigma oder den Paradigmata geprägt ist, nach denen sich die jeweilige Gruppe richtet. „In einem Sinne [...] üben die Befürworter konkurrierender Paradigmata ihre Tätigkeit in verschiedenen Welten aus. [...] Da sie in verschiedenen Welten arbeiten, sehen die beiden Gruppen von Wissenschaftlern verschiedene Dinge, wenn sie vom gleichen Punkt aus in die gleiche Richtung schauen.“ ([Kuhn 1973] S. 161). Damit wird das Finden in der Natur angeblich eindeutig vorliegender Sachverhalte, wovon z.B. Bacon ausgeht, unmöglich. Ein weiterer Punkt, in dem sich Kuhn und Bacon klar widersprechen, ist das Ziel wissenschaftlicher Forschung. Bacon sieht das Ziel darin, der Natur ihre Geheimnisse zu entreißen, während Kuhn kein eigentliches Ziel der Wissenschaften festmacht, sondern die Aufgabe der Wissenschaftler im Lösen der Rätsel sieht, die das gerade aktuelle Paradigma aufwirft.

Ausgehend von dem Kuhnschen Wissenschaftsbegriff, von dem Morik ausgeht, läßt sich also ein weiteres Argument für die Zuordnung von *sloppy modeling* zur Konstruktionsperspektive finden.

Die von Sloppy Modeling berücksichtigten Personengruppen

Ein Kriterium, daß für die Zuordnung von Modellierungsansätzen zur Konstruktionsperspektive spricht, ist die Berücksichtigung des Menschen, vor allem der Benutzerinnen. Dieser Aspekt von *sloppy modeling* ist ein weiteres Argument für die Zuordnung zur Konstruktionsperspektive. Da sich der von Morik vorgestellte Ansatz im wesentlichen auf die adäquate Unterstützung des Modellie-

---

and lead to a revision of the theory built up so far. This makes scientific theories more robust with respect to unforeseen events.“ ([Morik 1989] S. 114)

[19] „Thus, just as science is always creating explanations, so do learning programms.“ ([Morik 1989] S. 121)

[20] v. [Kuhn 1973].

rungsprozesses bezieht, und nicht auf die spätere Anwendung des wissensbasierten Systems, spielen auch nur die Personen, die an diesem Prozeß beteiligt sind, eine Rolle. Das sind im wesentlichen Wissensingenieurin und Expertin, also die potentiellen Nutzerinnen von Wissensakquisitionssystemen, die nach dem *sloppy modeling* Ansatz konzipiert wurden. Diese Personen werden durch den Ansatz des *balanced cooperative modeling* allerdings bei der Modellierung sehr wenig eingeschränkt: „Between the extremes of modeling by the user alone and some automatic contribution to the modeling by the system, all variations of work share are possible." ([Morik 1991] S. 68). Benutzerinnen haben also die Möglichkeit, einen eigenen Arbeitsstil zu entwickeln, ohne durch das System bereits auf feste Bahnen gelenkt zu werden. Allerdings macht Morik auch auf die Probleme aufmerksam, die aus dieser relativ großen Freiheit der Benutzerinnen erwachsen: „This flexibility also has a disadvantage which should not be hidden: new users of the system miss the strict guidance which is given by interactive systems." ([Morik 1991] S. 68).

Leider geht Morik in ihrer Beschreibung von *sloppy modeling* nicht auf die Anwenderin der „fertigen" Wissensbasis ein, denn hier wäre es sicherlich sehr interessant, ob und, wenn ja, welche Unterschiede sich für die Anwenderin einer Wissensbasis, die mit einem auf *sloppy modeling* basierenden Wissensakquisitionssystem modelliert wurde, im Vergleich zu einer „herkömmlichen" Wissensbasis ergeben.

Abschließend möchte ich noch hinzufügen, daß Morik mit ihrem Ansatz die typisch menschliche, „schlampige" Arbeitsweise unterstützen will, ohne der Anwenderin zu viele Vorschriften machen zu wollen.

Die Qualität der Wissensbasis

Auch in diesem Punkt lassen sich Aspekte finden, die für die Zuordnung zur Konstruktionsperspektive sprechen. Einen großen Vorteil für die spätere Wartung der Wissensbasis bietet sicher die von vornherein vorgesehene Änderbarkeit des Modells in allen Punkten. Alle Komponenten des Systems, einschließlich der Benutzerin, können auf alle vorhandenen Daten zugreifen. In dem auf *sloppy modeling* basierenden System MOBAL können alle Schritte des Systems nachvollzogen werden, es kann also nachvollzogen werden, wie die Daten zustande gekommen sind. Durch das Anbieten von Alternativen durch das System, und durch die Möglichkeit der Benutzerin, Bewertungen vorzunehmen, erscheinen die Daten der Anwenderin nicht in dem Maß als objektive Wahrheit, wie das bei „herkömmlichen" Expertensystemen der Fall ist. Durch die Implementierung von Wissen im Computer präsentiert sich das dargestellte Wissen der späteren Benutzerin der Wissensbasis häufig als objektivierte Fakten, da der Bezug der Benutzerin zur wissenden Person verloren geht.

Die Qualität des Modells selber wird schon durch die vorgesehenen Mechanismen der wissenschaftlichen Prozesse gesichert, die ich bereits beschrieben habe.

Sloppy Modeling im Hinblick auf Modellierungskriterien

Abschließend möchte ich *sloppy modeling* noch im Hinblick auf die Kriterien für die Modellierungsperspektive untersuchen. Die meisten dieser Kriterien sind bereits implizit oder explizit in den schon untersuchten Fragestellungen vorgekommen, aus Gründen der Vollständigkeit und Übersichtlichkeit möchte ich sie an dieser Stelle dennoch noch einmal zusammenfassen.

Korrektheit und Vollständigkeit spielen bei Morik auf keinen Fall die Rolle, die diese Kriterien in der Abbildperspektive spielen. Für Morik ist die formal beweisbare Korrektheit und Vollständigkeit ihres Modells nicht wichtig, die Qualität des Modells wird vielmehr durch den Prozeß seiner Entstehung gesichert.

Morik geht in ihrem Ansatz davon aus, daß kein Original im Sinne der Abbildperspektive existiert, welches dann abgebildet werden kann bzw. als Vorbild für das spätere Modell dient. Wenn man dennoch ein Original in *sloppy modeling* festmachen möchte, so ist es das Modell, welches sich die Expertin von ihrer Expertise während des Modellierungsprozesses macht, was nicht dem Modell entspricht, das konstruiert wird und die Wissensbasis des späteren Expertensystems darstellt. Allerdings hat dieses „Originalmodell“ nicht den Stellenwert, den ein Original in der Abbildperspektive hat. Da also kein Original im Sinne der Abbildperspektive existiert, ist weder das Unabhängigkeits- noch das Vergleichbarkeitskriterium für die Untersuchung von *sloppy modeling* anwendbar.

Das Angemessenheitskriterium dagegen, das für die Konstruktionsperspektive eine wichtige Rolle spielt, ist erfüllt. Wichtig ist bei *sloppy modeling*, daß das Modell seine Aufgabe erfüllt, eine Überprüfung findet anhand von Ergebnissen, die Konstruktion u.a. anhand von Beispielen statt. Ferner ist für *sloppy modeling* entscheidend, daß ein entsprechendes System den Modellierungsprozeß mit allen von Morik formulierten Anforderungen adäquat unterstützt, und nicht die formal korrekte Abbildung eines Originals erzwingt.

Das Kriterium der Verständlichkeit (oder Explizierbarkeit), das die Transparenz der Modellkonstruktion für die Modellbenutzerinnen ermöglichen soll, spielt bei Morik eine sehr große Rolle. Der ganze Ansatz des *balanced cooperative modeling* wäre ohne die Erfüllung dieses Kriterium überhaupt nicht realisierbar, da dieser Ansatz vorsieht, daß alle am Modellierungsprozeß Beteiligten (also sowohl Benutzerin als auch System) zu jedem Zeitpunkt auf alle Daten, die das Modell ausmachen, zugreifen und diese manipulieren können.

Abschließende Bemerkungen

Als Fazit läßt sich feststellen, daß auch die Untersuchung im Hinblick auf die aufgestellten Kriterien die These, daß *sloppy modeling* der Konstruktionsperspektive zugeordnet werden kann, weiter stützt. Die Kriterien, die typischerweise in der Abbildperspektive anzutreffen sind, spielen bei Morik keine Rolle, bzw. sind nicht anwendbar, wogegen die Kriterien, die für die Konstruktionsperspektive sprechen, in diesem Ansatz eine wichtige Rolle spielen.

Eine Frage, die mir interessant erscheint, ist allerdings noch offen geblieben: Ob ein *sloppy modeling* vergleichbarer Ansatz, der die Schlampigkeit der menschlichen Arbeit unterstützt, auch für andere Disziplinen in der Informatik, z.B. für die Softwareentwicklung, möglich, denkbar und/oder sinnvoll wäre.

Teil IV

Zusammenfassung

Kapitel 12

Resümee und Ausblick
Autor: Martin Fischer

Ausgangspunkt unserer Überlegungen war die Frage nach Modellierung oder Modellbildung. Wenn Modell ein zentraler Begriff in der Informatik ist[1], dann gehört die Frage Modellierung zu den zentralen Fragen der Disziplin. Herstellung, Transformation und Nutzung von (formalen) Modellen und deren Realisierungen mit Computern sind wesentliche Aspekte der Tätigkeiten von Informatiker/innen.

Wir haben Modellierung als Oberbegriff für die Begriffe Repräsentation, Design, Konstruktion betrachtet, die wie Modellierung eine Doppelbedeutung als Tätigkeit, Vorgang, Prozeß und als Produkt, Ergebnis, Gegenstand haben. Unser Interesse galt dabei stärker dem Modellieren, Programmieren, etc. als etwa den Programmen als Gegenständen oder Produkten.

Traditionelle informatische Zugänge zu Modellierung gehen meist bereits von Modellen (Spezifikationen, Pflichtenheften, etc.) aus und sind stark auf deren (formale) Transformationen fokussiert. Wir waren dagegen stärker an der Frage interessiert, wie erste Modelle (von Realität) zustande kommen. Dazu gehört die Formulierung der Problemstellung ebenso wie die Formulierung erster Lösungen. Modellieren, so verstanden, soll im folgenden Modellierung im engeren Sinne genannt werden. Aspekte von Modellierung im engeren Sinne sind aber auch immer dann im Spiel, wenn Modelle verändert werden.

Modellierung im weiteren Sinne soll den gesamten Prozeß der Entwicklung informationstechnischer Artefakte einschließlich ihrer Nutzung umfassen, denn der informatische Modellierungsprozeß beschränkt sich nicht auf das Herstellen von z.B. Software, vielmehr gibt es mannigfaltige Wechselwirkungen zwischen Herstellung und Nutzung von Computersystemen. Die Nutzung selbst setzt z.B. wiederum eine Modellbildung bei den NutzerInnen voraus.

[1] Vgl. Gernot Grube in Teil 1 dieses Bandes.

Modellieren meint das Herstellen von Modellen. Ein Modell ist immer Modell von etwas, Vorbild oder Nachbild eines Originals. Abbild- und Konstruktionsperspektive bezeichenen zwei grundlegend verschiedenene Sichten auf informatische Modellierung. Ihre wichtigsten Charakteristika in bezug auf Modell, Original und die Beziehung zwischen beiden möchte ich hier noch einmal kurz rekapitulieren.[2]

Die korrespondenztheoretische oder Abbildperspektive geht von folgenden Annahmen aus:

1) Die Realität ist gegeben. Sie ist strukturiert (z.B. in Objekte, Eigenschaften, Relationen) und im Prinzip, wenn auch nicht unbedingt vollständig, erkennbar. Mit Realität muß hier nicht unbedingt eine physikalische, materielle gemeint sein. Es kann sich auch um eine ideale, z.B. ein Reich abstrakter Informationen, oder um eine geistige Realität, z.B. mentale Modelle, handeln. Die Realität hat den Status des Originals.

2) Ein Modell der Realität ist ein Abbild, es richtet sich nach dem Original.

3) Die Beziehung zwischen Original und Modell ist als Abbildung analog zu einem mathematischen Homomorphismus zu verstehen. Sie ist also strukturerhaltend und im allgemeinen partiell. Diese Verkürzungen werden mit dem Terminus Abstraktion bezeichnet.

4) Ein Modell ist gut, wenn es die Realität korrekt abbildet.

5) Die Konstruktion etwa von technischen Artefakten nach Modellen ist im Rahmen der Abbildperspektive als eine neue Kombination von vorher erkannten objektiven Gegebenheiten zu verstehen.

Der Abbildperspektive haben wir eine an konstruktivistische Positionen angelehnte, Konstruktionsperspektive genannte Sicht entgegengesetzt:

1) Hier gibt es keine, zumindest keine zugängliche objektive Realität. Realität ist immer erfahrene Realität und damit nicht unabhängig vom erfahrenden Subjekt.

2) Modelle (Theorien, Begriffe, Handlungsmuster, etc.) konstruieren, konstituieren, strukturieren Realität.

3) Die Beziehung von Modell und Realität ist wechselseitig. Realität kann als Modell des sie konstruierenden Modells aufgefaßt werden. Unterschiedliche Modelle erzeugen unterschiedliche Realitäten. Das heißt aber nicht, daß mit Hilfe von Modellen beliebige Realitäten erzeugt werden könnten. Realität ist widerständig, das gilt sowohl für physische als auch für soziale Realitäten.

4) Ein gutes Modell ermöglicht neue Erfahrungen und bestätigt sich durch Erfahrung der Realität, oder, anders ausgedrückt, dadurch daß erfolgreiches Handeln möglich ist.

5) Die Herstellung von Modellen und Artefakten erzeugt und verändert hier Sich-

[2] Ausführlicher wurde diese Unterscheidung von Genot Grube in Teil 1 dieses Bandes dargelegt.

ten und Handlungsmuster und konstruiert damit Realität neu.

Im Rahmen der Abbildperspektive wird die Frage nach Modellierung im engeren Sinne, also die Frage, wie man überhaupt zu Modellen (z.B. Spezifikationen) kommt, die man dann entlang von Abbildungsbeziehungen transformieren kann, üblicherweise gar nicht gestellt. Ebensowenig kommen Aspekte der Nutzung, die für Modellierung im weiteren Sinne relevant sind, systematisch in den Blick. Im Rahmen der Konstruktionperspektive dagegen kann Modellierung im engeren Sinne gerade als wesentlich gelten für die konstruktiven Aspekte des Modellierens. Ebenso sind Modelle hier nicht zu isolieren von den sozialen Kontexten ihrer Herstellung und den sozialen Kontexten ihrer Nutzung.

Abbild- und Konstruktionsperspektive können auf verschiedene Weise miteinander in Beziehung gesetzt werden. Je nach dem ergeben sich daraus unterschiedliche Fragestellungen in bezug auf informatische Modellierung.

12.1 Abbild versus Konstruktion

Korrespondenztheoretische und konstruktivistische Theorien können als gegensätzliche, unvereinbare Positionen zur Welterklärung angesehen werden. Die Annahme einer gegebenen, im prinzip erkennbaren oder zumindest immer besser wissenschaftlich erschließbaren Welt und die These, daß Welten immer individuell oder sozial konstruiert sind, sind nicht vereinbar. Versteht man Abbild- und Konstruktionsperspektive in diesem Sinne als Gegensatz, so kann der Konflikt innerhalb der Informatik um ein ingenieurwissenschaftliches[3] Verständnis der Disziplin gerade an der Bruchlinie zwischen diesen beiden Perspektiven rekonstruiert werden. Abbild- und Konstruktionsperspektive beschreiben dabei weniger explizit eingenommene Positionen, sie können eher als Pole aufgefaßt werden, zwischen denen sich der Raum der Auseinandersetzung aufspannt.

Florian Theißing[4] hat am Beispiel der Requirement Definition Languages gezeigt, daß dieser Konflikt, dort zwischen Formalisierung des Entwicklungsprozesses und kommunikationsorientiertem Vorgehen, keineswegs neu ist.

Abbild- und Konstruktionsperspektive können dazu dienen, grundlegende Positionen in der Informatik zu analysieren und einzuordnen.[5] Es können aber auch konkrete Modellierungswerkzeuge und Methoden anhand ihrer grundlegenden Annahmen, anhand ihrer Interpretation der Problemlage informatischer Modellierung und anhand der Kriterien für die Qualität von Modellen, die sie in

[3] Zum hier gemeinten Verständnis von ingenieurwissenschaftlich siehe auch den Text zu Bacon in Teil 2 dieses Bandes.

[4] Vgl. Florian Theißing in Teil 3 dieses Bandes.

[5] Vgl. die Untersuchung von Melahat Elis und Michael Freitag zu den Konzepten von Baccus, Parnas und Naur im Teil 3 dieses Bandes

den Vordergrund stellen, klassifiziert und analysiert werden. Phasenmodell, objektorientierte und partizipative Softwareentwicklung[6], um nur die drei im Projekt behandelten Ansätze zur Softwareentwicklung zu nennen, können so in ihren Unterschieden analysiert und verstanden werden. Gleiches gilt für grundlegenden Ansätze, Methoden und Werkzeuge zur Wissensmodellierung in der KI-Forschung.[7]

Ein weiteres Feld, das hier einer genaueren Untersuchung Wert wäre, sind die langjährigen Diskussionen und Konflikte um die akademische Ausbildung im Fach Informatik bzw. Computer Science.

12.2 Abbild und Konstruktion

Abbild- und Konstruktionsperspektive können auch als sich gegenseitig ergänzende Zugänge zu informatischer Modellierung betrachtet werden.

Dann ergeben sich zumindest zwei Fragenkomplexe. Einerseits kann der Modellierungsprozeß selbst daraufhin untersucht werden, für welche Aspekte oder Phasen welche der beiden Perspektiven angemessen erscheint. Zum zweiten gibt es möglicherweise unterschiedliche Anwendungssituationen oder unterschiedliche Typen informatischer Artefakte, bei denen jeweils eine der beiden Perspektiven bei der Modellierung angemessen erscheint.

Wir haben beide Fragen im Studienprojekt nicht im Detail untersucht. Trotzdem will ich hier einige Thesen vorstellen, die andeuten sollen, inwiefern die Unterscheidung von Abbild- und Konstruktionsperspektive hier produktiv sein kann. Eine genauere Untersuchung der aufgeworfenen Fragen ist eine noch zu leistende Forschungsaufgabe.

Aspekte der Modellierung

Für die Transformation von Modellen, etwa von einer Spezifikation in ein Programm, scheint die Abbildperspektive angemessen. Das Kriterium Korrektheit, im Sinne von Isomophie der Abbildung zwischen Modellen, ist hier wesentlich. Auch wenn Korrektheit nicht wirklich sichergestellt werden kann, ist sie zumindest als Zielvorstellung angemessen. Eine korrekte Abbildung der Welt kann oder muß hier implizit unterstellt werden, um die Adäquatheit des Ausgangsmodells zu sichern. Allerdings betrifft diese Unterstellung die praktische Tätigkeit des Transformierens von Modellen wenig bis gar nicht[8].

[6] Irina Leyde analysiert in Teil 3 dieses Bandes einen partizipativen Ansatz im Gegensatz zum Phasenmodell.

[7] Vgl. Birgit Schelm und Jürgen Schöning in Teil 3 dieses Bandes.

[8] Vgl. auch die Anmerkungen zu Tarski in Teil 2 dieses Bandes.

Für die Modellierung im engeren Sinne, d.h. die Erstellung erster Modelle dürfte dagegen die Konstruktionsperspektive adäquat sein. Modellierung im engeren Sinne hat immer kreative und gestaltende Aspekte. Aus einer Aufgabenstellung folgt nie eindeutig das Modell zu ihrer Lösung, und umgekehrt spiegelt jedes Modell auch eine bestimmte Sicht der Aufgabenstellung wider.[9] Verschiedene Personen und Gruppen haben je unterschiedliche Interessen, Ziele und Vorstellungen. Hier eine Abbildperspektive einzunehmen beinhaltet die Annahme, daß diese subjektiven Faktoren zur Erstellung eines objektiven Modells ausgeglichen werden könnten. Gegen diese Annahme gibt es starke philosophische Einwände. Aber auch die praktischen Schwierigkeiten bei der Entwicklung von Software sprechen gegen eine solche Annahme.

Bezieht man die Nutzung von informatischen Artefakten mit ein, so gilt zumindest für solche, mit denen Menschen direkt interagieren, daß sich dadurch für die NutzerInnen neue Handlungsmöglichkeiten ergeben und damit ihre Realität verändert wird. Auch hier scheint eine konstruktive Perspektive angemessen zu sein.[10]

Typen von Artefakten

In Hinblick auf informatische Artefakte kann man unterscheiden, ob sie von Menschen genutzt werden, oder ob sie lediglich Bestandteil von technischen Systemen sind. Im Rahmen technischer Systeme, etwa bei Steuerungsprogrammen, erscheint die Abbildperspektive angemessen, sofern eine mathematische Beschreibung der zu lösenden Aufgabe vorliegt. Hier ist wiederum die korrekte Transformation dieser Beschreibung in ein Programm oder einen Schaltkreis die wesentliche Aufgabe informatischen Modellierens. Allerdings kommt auch hier im Rahmen des Umgangs mit Fehlern des technischen Systems Modellierung im engeren Sinne ins Spiel, insofern die Sicherheit des Systems davon abhängt, welche Situationen bei der Modellierung überhaupt berücksichtigt wurden.

Für Systeme, die von Menschen, etwa in Arbeitsprozessen, benutzt werden, scheint dagegen eine konstruktive Perspektive unverzichtbar zu sein. Zum einen gilt es verschiedene Sichten oder Realitäten der Personen und Gruppen, die an der Entwicklung beteiligt sind, zu berücksichtigen und auszugleichen. Zum anderen scheint in bezug auf die Nutzung solcher Systeme eine Abbildperspektive, die nur die (formale) Abbildung von bestehenden Vorgängen und deren Automatisierung in den Blick nehmen kann, zu eng zu sein.

---

[9] Die Erfahrungen bei den praktischen Arbeiten im Studienprojekt, stützen ebenfalls die These, daß es sich bei Modellierung im engeren Sinne nicht um eine Abbildung von (vor)gegebener Realität handelt.

[10] Jürgen Schöning deutet in Teil 3 dieses Bandes die Rolle der beiden Perpektiven bezüglich der Modellierung als Ganzes (Konstruktion) und bestimmte Aspekte der Formalisierung (Abbild) an, dort in bezug auf die Modellierungsmethodologie KADS.

Ob Abbild und Konstruktion als Modellierungsperspektiven relevant sind für die Unterscheidung zwischen der Entwicklung von Standardsoftware und Softwareprojekten, die spezielle Lösungen, z.B. für einzelne Betriebe, entwickeln, scheint mir dagegen offen zu sein.

Falls die angestellten Überlegungen einer genaueren Prüfung standhalten, wäre daraus jeweils zu folgern, welche Methoden für die Entwicklung bestimmter Arten von informatischen Artefakten und für bestimmte Aspekte des Modellierungsprozesses angemessen sind.

12.3 Konstruktion als adäquate Perspektive

Konstruktions- und Abbildperspektive müssen nicht als gegensätzlich oder zumindest unvereinbar angesehen werden, sondern die Abbildperspektive kann in die Konstruktionsperspektive integriert werden. Wenn die Konstruktionsperspektive einerseits umfassender ist und andererseits für bestimmte Aspekte informatischer Modellierung unverzichtbar, so ist sie als adäquat für informatische Modellierung im allgemeinen anzusehen.

Die folgenden Überlegungen sollen wiederum als Hinweise und Thesen zur Plausibilität dieser Sichtweise und den sich daraus ergebenden Folgerungen gelten, deren Überprüfung weiterer empirischer und theoretischer Studien bedarf.

Betrachtet man die Konstruktionsperspektive als adäquat für informatische Modellierung im allgemeinen und die Abbildperspektive als eine mögliche Sicht, die im Rahmen der Konstruktionsperspektive eingenommen werden kann, dann können die in den vorherigen Abschnitten entwickelten Fragestellungen mit leicht verändertem Tenor weiterhin verfolgt werden. Auch wenn Abbild- und Konstruktionsperspektive nicht als Gegensatz interpretiert werden, kann man Modellierungskonzepte, -methoden und -werkzeuge daraufhin untersuchen, ob sie auf die Abbildperspektive beschränkt bleiben. Ebenso kann man, auch ohne beide Perspektiven als gleichrangig anzusehen, fragen, in welchen Zusammenhängen eine Beschränkung auf die Abbildperspektive möglich, sinnvoll oder adäquat ist.

Abbild als eine Sicht im Rahmen der Konstruktionsperspektive

Für die Konstruktionsperspektive ist die Wechselwirkung zwischen Modell und Realität charakteristisch. Damit wird zugleich die Annahme einer gegebenen Realität als auch die Annahme wahrer Modelle und damit einer wahren Sicht auf Realität verabschiedet. Es ergeben sich für unterschiedliche Personen, Personengruppen (EntwicklerInnen, NutzerInnen, Management etc.) mit unterschiedlichen Interessen und Zielen auch verschiedene Realitäten oder Sichten. Daraus ergeben sich unterschiedliche Gesichtspunkte, die bei der Modellierung im Vor-

dergrund stehen, die sich etwa an den Kriterien für die Güte von Modellen festmachen.

Die Abbildperspektive kann dann als eine Sicht im Rahmen der Konstruktionsperspektive identifiziert werden, die einen bestimmten Gesichtspunkt, die Korrektheit der Abbildung, in den Vordergrund rückt und andere Gesichtspunkte wie Nutzer- und Nutzungsangemessenheit vernachlässigt oder zumindest nicht systematisch in den Blick nehmen kann.[11]

Aus unserer Sicht ergibt sich für informatische Modellierung aber die Forderung, verschiedene Gesichtspunkte systematisch zu berücksichtigen und zu integrieren. Dazu gehört, daß verschiedene Gesichtspunkte überhaupt thematisiert werden können und damit offen gelegt werden kann, welche für die jeweilige Aufgabe für relevant gehalten werden. Dazu bedarf es einer konstruktiven Perspektive.

Ich will hier einige Bemerkungen anschließen, die die Vorzüge einer konstruktiven Perspektive einerseits in Hinblick auf die Analyse von Problemen bei der Softwareentwicklung und andererseits in Hinblick auf neuere Entwicklungen in der Computertechnik zeigen.

Softwarenutzung, Softwareentwicklung, Softwareeigenschaften

Da die Nutzung der Artefakte in der Abbildperspektive nur insofern eine Rolle spielt als durch sie die ‘richtige’ Abstraktion bestimmt wird, spielen die Erfahrungen der NutzerInnen hier kaum eine Rolle. Den Problemen bei der Entwicklung zu immer komplexeren technischen Systemen versucht man mit ausgefeilteren, im wesentlichen formalen Methoden Herr zu werden.

Von den EntwicklerInnen wird in der Abbildperspektive Erfahrung im Umgang mit Methoden und Mitteln gefordert, individuelle Sichten und Erfahrungen sind gerade systematisch aus dem Modellierungsprozeß auszuschließen, um ein objektives Modell zu erhalten.

Softwareeigenschaften, wie Komplexität oder Zuverlässigkeit, beziehen sich ausschließlich auf Modelle oder Systeme als Gegenstände bzw. Produkte. Software ist z.B. zuverlässig, wenn sie korrekt und auf zuverlässiger Hardware installiert ist.

Das heißt nicht, daß etwa Fragen der Ergonomie nicht als relevant und wichtig angesehen werden, sie können nur nicht systematisch in ein abbildorientiertes Modell von Modellierung intergriert werden. Die Erstellung korrekter Software und die Gestaltung von Benutzeroberflächen sind hier getrennte, unvermittelte Bereiche. Allgemeiner gesprochen gilt: Softwareentwicklung, Softwareeigenschaften und Softwarenutzung sind im Rahmen der Abbildperspektive klar voneinander getrennte und auch systematisch trennbare Sichtweisen auf Software. Soziale Zu-

[11] vgl. auch Elis und Freitag, sowie Leyde in Teil 3 dieses Bandes.

sammenhänge sind damit von der Modellierungsproblematik, sofern sie sich für die Informatik stellt, abgekoppelt.

Im Rahmen einer konstruktiven Perspektive kann diese Trennung nicht aufrechterhalten werden. Hier kommen eine ganze Reihe von Wechselwirkungen zwischen ganz verschiedenen Ebenen ins Spiel. Damit ergeben sich Probleme, die über die im Rahmen der Abbildperspektive relevanten Gesichtspunkte hinausgehen.

Die Sichten und Erfahrungen der EntwicklerInnen beeinflussen das Modell und auch die damit konstruierte Realität. Gleiches gilt für die verwendeten Methoden und Darstellungsmittel, in denen sich wiederum bestimmte Sichten ihrer ErfinderInnnen und der Gruppen, die sie nutzen, letztlich disziplinäre Sichten von InformatikerInnen spiegeln.

Im Rahmen des Modellierungsprozesses sind neben den EntwicklerInnen eine Reihe anderer Personen in anderen Rollen (Management, verschiedene Nutzergruppen, etc.) betroffen. Diese Personen haben ihre je eigene, nur zum Teil explizite und explizierbare Realitätssicht. Werden diese Personen oder Gruppen nicht an der Modellierung beteiligt, so bleiben bestimmte Sichten ausgeblendet, was als eine Ursache der mangelnden Einsatzfähigkeit von Softwaresystemen gelten kann. Werden verschiedene Gruppen beteiligt, steigt damit die Komplexität der Modellierungsaufgabe enorm, da die verschiedenen Sichten, resp. verschiedenen Realitäten, in Kommunikationsprozessen aus- und angeglichen werden müssen.

Selbst wenn man die Möglichkeit einer erfolgreichen Konsensbildung unterstellt, heißt das nicht, daß solch ein explizites Modell nicht mit dem Handlungswissen der NutzerInnen, mit nicht explizierten Erfahrungen kollidiert. Die Angemessenheit von Modellen kann daher nur im konkreten Handlungszusammenhang, d.h bei ihrer Nutzung, beurteilt werden. Die Forderung nach Entwicklungszyklen korrespondiert zu dieser Einsicht.

Ein weiteres Problem bei der Entwicklung von Software, ist die Tatsache, daß sich die Anforderungen an Computersysteme mit ihrer Nutzung selbst dauernd verändern. Aus der Abbildperspektive kann dies nur als der Modellbildung äußerliches, unerklärbares Faktum, als Fluch, gesehen werden. Wenn das relevante Kriterium für die Güte von Modellen die Korrektheit ist, so müßte bei der technischen Realisierung von korrekten Modellen ein stabiler Zustand erreicht werden. Daß Programme nie wirklich korrekt sind, erklärt auch nicht warum sich Anforderungen massiv verändern.

Aus konstruktiver Sicht bietet sich für das genannte Phänomen folgende Interpretation an. Ausgangspunkt für einen Computereinsatz ist eine bestimmte Arbeits- resp. Handlungssituation. Die Situation, oder genauer die Sichten auf diese Situation, bilden den Ausgangspunkt für die informatische Modellierung. Wenn sich mit dem Einsatz von Computersystemen auch die Arbeitssituation selbst verändert, d.h. eine neue Realität konstruiert wird, ist es wenig überraschend, daß

sich mit dieser neuen Situation auch neue Sichten ergeben, auf Grund derer sich wiederum die Anforderungen an das Computersystem selbst verändern.

Damit besteht auch im Rahmen eines zyklischen Entwicklungsmodells kaum Aussicht, sich einem stabilen Zustand anzunähern. Modellierung erweist sich dann als ein komplexer, unendlicher Prozeß unter Beteiligung von Nutzer- und EntwicklerInnen.

Für den Softwareentwicklungsprozeß bedeutet das, daß im Rahmen einer konstruktiven Sicht Entwicklung und Nutzung über die Akteure, die an der Modellierung beteiligt sind (Nutzer und Entwicklerinnen) verwoben sind. Darüberhinaus ist eine wechselseitige Beeinflussung von Nutzung und Entwicklung zu konstatieren, die nur in einem zyklischen im Prinzip unendlichen Entwicklungs- und Nutzungsprozeß adäqut behandelt werden kann.

In der Praxis stößt die Organisation unendlicher Entwicklungsprozesse natürlich an Grenzen. In gewisser Weise stellt die hier vertretene Sicht deshalb ein idealisiertes Modell für die Softwareentwicklung dar, das aber konträr ist zu Idealmodellen, die im Rahmen der Abbildperspektive konzipiert worden sind.[12]

Neben einer veränderten Sicht auf den Softwareentwicklungsprozeß legt eine konstruktive Perspektive auch eine andere Sicht auf Softwareeigenschaften nahe, was ich am Beispiel: Zuverlässigkeit erläutern möchte. Während Zuverlässigkeit in der Abbildperspektive als äquivalent zu Korrektheit angesehen wird, kommt im Rahmen der Konstruktionsperspektive auch hier die Nutzung der Systeme mit in den Blick.

Technische Systeme, die ohne Rücksicht auf den Handlungszusammenhang, in dem sie eingesetzt werden sollen, und die Sichten und Gewohnheiten ihrer NutzerInnen entwickelt werden, wirken komplex oder kompliziert. Diese Komplexität ergibt sich aus der Diskrepanz zwischen den sich in den Systemen spiegelnden Modellen und den verfügbaren Handlungsmöglichkeiten ihren NutzerInnen.[13] Die Geschwindigkeit von Veränderungen ist ein wesentlicher Faktor, um solche Diskrepanzen zwischen der Realitätssicht, die sich in einem informatischen Modell spiegelt, und den Handlungsmustern und Gewohnheiten, in deren Rahmen die NutzerInnen agieren, entstehen zu lassen.

Solche Diskrepanzen betreffen die Zuverlässigkeit von Systemen, bei denen Menschen mit solchen modellerzeugten Realitäten interagieren. Sie hängt dabei wesentlich davon ab, ob beide, technische Systeme und menschliche Erfahrungen,

[12] Vgl. Melahat Elis und Michael Freitag zu Parnas' Idealmodell in Teil 3 dieses Bandes.

[13] Metaphern haben in der Informatik möglicherweise deshalb einen so hohen Stellenwert, weil sie als (im wesentlichen sprachliches) Mittel angesehen werden können, um aus Bekanntem neue Realitäten zu konstruieren. Indem Strukturen aus einem bekannten Bereich in einen anderen übertragen werden, können dort neue Zusammenhänge gesehen, konstruiert werden und vor allem auch verfügbar gemacht, d.h. in gewohnte Interpretations- und Handlungsmuster intergriert werden.

in einem verträglichen Zusammenhang stehen. Der Umgang mit nicht verstandenen Systemen[14] kann keine zuverlässigen Ergebnisse liefern. Es entstehen dann unbeherrschte und unbeherrschbare Systeme.

Die Eigenschaften von Software sind hier ebenfalls nicht unabhängig von ihrer Nutzung zu verstehen. Daraus ergibt sich wiederum die Anforderung, dies bereits im Entwicklungsprozeß zu berücksichtigen und den Kontext der Nutzung von informatischen Systemen, d.h. den Handlungsrahmen, in dem sie verwendet werden, mit zu berücksichtigen.

Neue Artefakte

Informatische Modellierung kann in zweifacher Hinsicht als Realitätskonstruktion verstanden werden.

Im Rahmen konstruktivistischer Positionen in z.B. Erkenntnistheorie, Sprachphilosophie oder Wissensoziologie ist Realität nie unabhängig von Beschreibungs- und Handlungsmustern gegeben. Mit Hilfe von Modellen wird Realität erst geordnet, erzeugt, konstituiert, als Realität konstruiert. Im wesentlichen hatten wir die Konstruktionsperspektive bis jetzt vor diesem Hintergrund verstanden. Mit Hilfe von Computer werden aber auch direkt Handlungsräume für Menschen erzeugt, technisch konstruiert. Beide Aspekte sind eng miteinander verbunden.

Falls es zutrifft, daß mit Computern formale Modelle in Realität transformiert werden können, dann werden durch informatische Modellierung neue Realitäten konstruiert, die Menschen erleben, in denen Menschen handeln. Für Virtuelle–Realitäts–Systeme scheint evident zu sein, daß Computerprogramme soziale Handlungsmuster definieren. Gleiches gilt meines Erachtens für alle informationstechnischen Systeme, sofern Menschen mit ihnen umgehen. In gewisser Weise stellt der Computer eine Radikalisierung der technischen Möglichkeiten dar, soziale Handlungsräume zu verändern und neu zu schaffen.

Wenn die These richtig ist, daß Computer in sehr unmittelbarer Weise die Konstruktion neuer Realitäten erlauben, ist völlig unklar, wie solche Systeme im Rahmen einer abbildorientierten Perspektive verstanden werden sollen. Für diese Sicht auf informatische Systeme, die auch mit dem Schlagwort ‘Computer als Medium’ verknüpft wird, erweist sich eine ausschließlich abbildorientierte Herangehensweise als unzureichend und inadäquat.

Für offene, verteilte Anwendungssysteme gilt ebenfalls, daß sowohl bei der Modellierung als auch bei der Nutzung Perspektiven unterschiedlicher Entwickler- und Nutzergruppen berücksichtigt werden müssen. Ein homogener Blick auf ‘das’ System ist sowohl aufgrund der Größe und Heterogenität solcher Systeme

[14] In Heideggerscher Terminologie könnte man auch, vielleicht treffender, sagen: ‘nicht zuhandenen Systemen’.

als auch aufgrund ihrer schnellen Veränderbarkeit nicht mehr zu gewinnen.

12.4 Subjektive Perspektiven beim Modellieren

Wir hatten gesehen, daß Konzepte, Methoden und Werkzeuge in der Informatik im Hinblick auf die Perspektive, die sie widerspiegeln, analysiert werden können. Dabei werden bei der Abbildperspektive die ‘richtige’ Abbildung der Realität und der Aspekt Korrektheit betont, während bei der Konstruktionsperspektive Interessenausgleich durch Kommunikation bei der Modellierung und Nutzer- und Nutzungsangemessenheit im Vordergrund stehen.

Es zeigt sich jedoch, daß bestimmte Mittel zwar bestimmte Perspektiven nahelegen, daß andererseits dieselben Mittel auch in unterschiedlichen Perspektiven verwendet werden können.[15] Darüber hinaus lassen sich Methoden und Werkzeuge nicht unbedingt eindeutig der Abbild- oder Konstruktionsperspektive zuordnen. Manche Modellierungskonzepte, wie z.B. der objektorientierte Ansatz, sind sowohl im Rahmen der Abbild- als auch der Konstruktionsperspektive interpretierbar und anwendbar.

Neben den Mitteln und Methoden zur Modellierung spielt die subjektive Sicht der EntwicklerInnen (und der AuftraggeberInnen) eine entscheidende Rolle dafür, in welcher Perspektive die Entwicklung informatischer Artefakte erfolgt.

Wenn es zutrifft, daß eine Konstruktionsperspektive auf informatische Modellierung die angemessene ist, ergibt sich daraus, daß es einer stärkeren Berücksichtigung dieser Perspektive nicht nur bei der (Weiter-)Entwicklung und verstärkten Verwendung von Methoden und Werkzeugen bedarf, die diese Sicht spiegeln, sondern daß eine konstruktive Perspektive insbesondere in der Informatikausbildung vermittelt werden muß.[16]

Betrachtet man das Studium an der TU Berlin, insbesondere die Grundausbildung, so steht hier eine abbildorientierte Perspektive nach wie vor, oder sogar zunehmend, im Vordergrund. Das zeigt sich unter anderem daran, daß im Rahmen praktischer Übungen im wesentlichen mathematisch formulierte oder leicht formulierbare kleine Probleme programmiert werden, um anschließend die Korrektheit zwischen Programm und mathematischer Spezifikation zu beweisen. Weder Modellierung im engeren Sinne noch Modellierung im weiteren Sinne geraten hier mehr als peripher in den Blick.[17]

[15] Vgl. z.B. Florian Theißing in Teil 3 dieses Bandes.
[16] Vgl. Melahat Elis und Michael Freitag in Teil 3 dieses Bandes.
[17] Das genauer und nicht nur für die TU Berlin zu untersuchen, bedürfte einer eigenen Studie.

Zusammenfassung

Konstruktions- und Abbildperspektive sind produktiv für die Analyse informatischer Konzeptionen, Methoden und Werkzeuge zur Modellierung. Die Abbildperspektive erweist sich allerdings als zu eng, um den Prozeß der Modellierung umfassend in den Blick zu bekommen; eine Konstruktionsperspektive ist erforderlich, um diesem äußerst komplexen Prozeß gerecht zu werden. Das subjektive Modellierungsverständnis der EntwicklerInnen ist ein wesentlicher Faktor bei informatischer Modellierung.

Diese drei Thesen zusammen mit der Annahme, daß der Informatik–Mainstream nach wie vor weitgehend in der Abbildperspektive denkt, arbeitet und lehrt[18], bilden den Kern dessen, was wir im Rahmen des Studienprojektes Modellierung und im Rahmen dieses Bandes angedacht haben.

Die hier aufgeworfenen Fragen bedürfen der weiteren Untersuchung, die formulierten Thesen weiterer Begründung. Auch die Beziehungen zu aktueller Literatur, wo, wie uns scheint, zum Teil ähnliche Fragen aufgeworfen werden, haben wir nicht ausgeleuchtet.[19] Letztlich haben wir auch für uns selbst mehr Fragen aufgeworfen als beantwortet.

Der spannende Punkt bleibt aber, sofern wir auf einer richtigen Spur sind, welche konkreten praktischen Konsequenzen aus unseren Überlegungen für informatische Modellierung, für Methoden und Werkzeuge zur Softwareentwicklung und nicht zuletzt für die Informatikausbildung gezogen werden können oder müssen. Es scheint allerdings zweifelhaft, ob angesichts der massenhaften Verbreitung informatischer Artefakte und deren Auswirkungen auf gesellschaftliche Prozesse und Strukturen auf eine Veränderung der Perspektive verzichtet werden kann.

[18] Vgl. insbesondere Melahat Elis und Michael Freitag in Teil 3 dieses Bandes, sowie die Texte zu Bacon und Tarski in Teil 2.

[19] z.B. [Floyd et al. 1992], [Luft und Kötter 1994] und [Hirschheim et al. 1995]

Literaturverzeichnis

[Backus 1985] John Backus. From Function Level Semantics to Program Transformation and Optimization. In Ehrig et al. (Eds.), *Formal Methods and Software Development, LNCS*, No. 186, Vol. 1, pages 60–91, Springer Verlag, Berlin, 1985.

[Bacon 1620] Francis Bacon. *Neues Organon (Novum Organum)*. Wolfgang Krohn (Hrsg.). Felix Meiner Verlag, Hamburg, 1990. 2 Bände.

[Bauer und Goos 1982] Friedrich L. Bauer und Gerhard Goos. *Informatik. Eine einführende Übersicht 1. Teil*. Springer Verlag, Berlin, 3. Aufl., 1982. 1. Aufl. 1971.

[Bauer und Löckenhoff 1994] C. Bauer und C. Löckenhoff. Das ESPRIT-Projekt KADS-II: nicht alles für die Katz! *Künstliche Intelligenz*, (3):51–54, 1994.

[Becker and Steven 1991] Elke Steven and Barbara Becker. Computational theory of the mind. Verbundprojekt Veränderungen der Wissensproduktion und -verteilung durch Expertensysteme. Technikfolgenabschätzung zur Künstlichen Intelligenz (TAIKI), Fachbericht 3, July 1991.

[Becker et al. 1991] Sabine Stohbach, Barbara Becker und Elke Steven. *Epistemologische und wissenssoziologische Aspekte maschineller Wissensverarbeitung*. Arbeitspapiere der GMD, Nr. 501. Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1991.

[Becker 1991] Barbara Becker. The Concept of Knowledge in Expert System Technology. Verbundprojekt Veränderungen der Wissensproduktion und -verteilung durch Expertensysteme. Technikfolgenabschätzung zur Künstlichen Intelligenz (TAIKI), Fachbericht 3, July 1991.

[Berger und Luckmann 1969] Peter L. Berger und Thomas Luckmann. *Die gesellschaftliche Konstruktion der Wirklichkeit.* Fischer Taschenbuch Verlag, Frankfurt a. M., 1969.

[Berger und Luckmann 1969] Thomas Luckmann und Peter L. Berger. *Die gesellschaftliche Konstruktion der Wirklichkeit.* Fischer Taschenbuch Verlag, Frankfurt a. M., 1969.

[Berka und Kreiser 1986] Karel Berka und Lothar Kreiser. *Logik-Texte. Kommentierte Auswahl zur Geschichte der modernen Logik.* Akademie-Verlag, Berlin, 1986.

[CODASYL 1962] CODASYL Development Committee. An Information Algebra. In *[Couger und Knapp 1974].* Original in Communications of the ACM 1962.

[Couger und Knapp 1974] D. Couger und R. Knapp (Hrsg.). *Systems Analysis Techniques.* New York, 1974.

[Dijkstra 1976] E. Dijkstra. *A Discipline of Programming*. Prentice-Hall, New Jersey, 1976.

[Dreyfus und Dreyfus 1986] H. Dreyfus und S. Dreyfus. *Mind over Mashine.* Free Press, New York, 1986.

[Floyd et al. 1989] Christiane Floyd, Wolf-Michael Mehl, Fanny-Michaela Reisin und Gregor Wolf. Projekt PEtS – Endbericht. Technische Universität Berlin, Fachbereich Informatik, Forschungsgruppe Softwaretechnik, 1989.

[Floyd et al. 1992] Ch. Floyd, H. Züllinghoven, R. Budde, R. Keil-Slawik. *Software Development and Reality Construction.* Springer, 1992.

[Friedman und Cornford 1989] A. Friedman und D. Cornford. *Computer Systems Development: History, Organization and Implementation.* Chichester, 1989.

[Greenberger 1962] M. Greenberger (Hrsg.). *Management and the Computer of the Future*. Cambridge Mass, 1962.

[Goodman 1973] N. Goodman. *Sprachen der Kunst. Ein Ansatz zu einer Symboltheorie*. Suhrkamp, Frankfurt a.M., 1973.

[Hirschheim et al. 1995] Rudy Hirschheim, Heinz K. Klein und Kalle Lyytinen *Information Systems Development and Data Modeling. Conceptual and Philosophical Foundations*. Cambridge University Press. 1995.

[James 1977] William James. *Der Pragmatismus: Ein neuer Name für alte Denkmethoden.* Felix Meiner Verlag, Hamburg, 1977.

[Karbach und Linster 1990] Werner Karbach und Marc Linster. *Wissensakquisition für Expertensysteme: Techniken, Modelle und Softwarewerkzeuge.* Hanser, München, 1990.

[Kraft 1977] P. Kraft. *Programmers and Managers.* New York. 1977.

[Kuhn 1973] Thomas S. Kuhn. *Die Struktur wissenschaftlicher Revolutionen.* Suhrkamp Taschenbuch Wissenschaft, Frankfurt a. M., 1973.

[Löckenhoff 1992] Christiane Löckenhoff. 2nd KADS User Meeting: Inferenz- und Aufgabenstrukturen. *Künstliche Intelligenz*, (4):35–36, 1992.

[Luft und Kötter 1994] Alfred L. Luft und Rudolf Kötter *Informatik – eine moderne Wissenstechnik* BI Wissenschaftsverlag, Mannhein. 1994.

[Lynch 1969] H. J. Lynch. ADS - A Technique in Systems Documentation In *[Couger und Knapp 1974]*. Original in Database 1969.

[McCarn 1970] D. McCarn. Getting Ready. *Datamation*, pages 22–26, August, 1970.

[Morik 1989] Katharina Morik. Sloppy Modelling. In Katharina Morik (Eds.), *Knowledge Representation and Organization in Mashine Learning*, pages 107–134. Springer Verlag, Berlin, 1989.

[Morik 1991] Katharina Morik. Balanced Cooperative Modeling. In G. Tecule and R.S. Michalski (Eds.), *1st Workshop on Multistrategy Learning*, pages 65–80. George Mason University, Fairfax, Virginia, 1991.

[Mumford 1972] E. Mumford. *Job Satisfaction: A Study of Computer Specialists*. London, 1972.

[Naur et al. 1968] P. Naur, B. Randell und J. Buxton. *Software Engineering - Concepts and Techniques.* New York, 1976.

[Naur 1985] Peter Naur. Intuition in Software Development. In Ehrig et al. (Eds.), *Formal Methods and Software Development*, *LNCS*, No. 186, Vol. 2, pages 60–79, Springer Verlag, Berlin, 1985.

[Newell 1982] Allen Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[Pagel und Six 1994] B.U. Pagel und H.W. Six. *Software Engineering, Band 1: Die Phasen der Softwareentwicklung.* Addison-Wesley, Bonn, 1994.

[Parnas 1985] David Lorge Parnas. A Rational Design Process: How and Why to Fake It. In Ehrig et al. (Eds.), *Formal Methods and Software Development*, *LNCS*, No. 186, Vol. 1, S. 80–100, Springer Verlag, Berlin, 1985.

[Parnas 1990] David Lorge Parnas. Education for Computing Professionals *IEEE Computer*, 23(1), S. 17–22, 1990.

[Pasch und Biskup 1995] Jürgen Pasch und Hubert Biskup. Software-Engineering - Ausbildung für die Praxis? *Informatik-Spektrum 4/1995*, 18(2), S. 84–94, 1995.

[Projektbericht] *Projekt: Modellierung.* Abschlußbericht des Studienprojektes Modellierung im SoSe 94 und WS 94/95 am Fachbereich Informatik der Technischen Universität Berlin.

[Puppe 1991] Frank Puppe. *Einführung in Expertensysteme.* Springer Verlag, Berlin, 1991.

[Reisin und Schmidt 1988] Fanny-Michaela Reisin und Gerhard Schmidt. Steps – Ein Ansatz zur evolutionären Systementwicklung. *Computer Magazin* (7/8), 1988.

[Reisin 1990] Fanny-Michaela Reisin. *Kooperative Gestaltung in partizipatven Softwareprojekten.* Europäische Hochschulschriften, Band 7. Verlag Peter Lang, Berlin, 1990.

[Reisin 1991] Fanny-Michaela Reisin. Kooperativer Aufbau einer gemeinsamen Referenztheorie. In *Arbeitsgestaltung und partizipative Systementwicklung.* Leske + Budrich, 1991.

[Reisin 1992a] Fanny-Michaela Reisin. Kooperation in Softwareprojekten - Von der Betroffenenpartizipation zur partizipativen Betroffenheit. In *Kontrast-Programm Mensch-Maschine-Arbeiten in der HighTex-Welt,* S. 233 - 249. Bund Verlag, 1992.

[Reisin 1992b] Fanny-Michaela Reisin. Gestaltbarkeit und Gestaltung von Methoden - zwei notwendige Bedingungen kooperativer Softwareentwicklung. *Mitteilungen der Fachgruppe Software-Engineering der GI,* 9(2), S. 15 - 26. September 1992.

[Reisin 1994] Fanny-Michaela Reisin. Software-Ergonomie braucht Partizipation. In *Einführung in die Software-Ergonomie,* Mensch Computer Kommunikation. deGruyter, Berlin, 2. Aufl., S. 301 - 333. 1994.

[Rorty 1981] Richard Rorty. *Der Spiegel der Natur. Eine Kritik der Philosophie.* Suhrkamp, Frankfurt a. M., 1981.

[Skirbekk 1977] Gunnar Skirbekk (Hrsg.). *Wahrheitstheorien.* Suhrkamp, Frankfurt a. M., 1977.

[Stachowiak 1973] Herbert Stachowiak. *Allgemeine Modelltheorie* Springer Verlag, Wien, 1973.

[Suhr & Suhr 1993] R. Suhr und R. Suhr. *Software Engineering. Technik und Methodik.* Oldenbourg-Verlag, München, 1993.

[Tarski 1935] Alfred Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. In *[Berka und Kreiser 1986],* S. 445–546. Orginal in Studia Philosophica Commentarrii Societatis Polanorum, Vol. 1, Leopoli, 1935.

[Tarski 1944] Alfred Tarski. Die semantische Konzeption der Wahrheit und die Grundlagen der Semantik. In *[Skirbekk 1977],* S. 140–188. Orginal in The Semantic Conception of Truth and the Foundations of Semantics, Philosophy and Phenomenological Research, Vol. IV, 1994.

[Teichroew 1972] D. Teichroew. *A Survey of Languages for Stating Requirements for Computer-Based Information Systems.* AFIPS Conference Proceedings of the AFIPS fall Joint Computer Conference 1972, pages 1203–1224, 1972.

[Voß und Studer 1994] H. Voß und R. Studer (Hrsg.). *Proceedings of the 4th KADS Meeting*. Arbeitspapiere der GMD, Nr. 832. Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1994.

[Wachsmuth 1993] I. Wachsmuth. In Günther Görz (Hrsg.) *Einführung in die künstliche Intelligenz.* Addison-Wesley, Bonn, 1993.

[Watzlawik 1976] Paul Watzlawik. *Wie wirklich ist die Wirklichkeit? Wahn - Täuschung - Verstehen.* Piper, München, 1976.

[Wedekind 1976] H. Wedekind (Hrsg.). *Systemanalyse.* Carl Hanser Verlag, München, 1979.

[Wielinga et al. 1993a] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. KADS: A modelling approach to knowledge engineering. In B.G. Buchanan and D.C. Wilkins (Eds.), *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems,* pages 92–116. Kaufmann, San Mateo, CA, 1993.

[Wielinga et al. 1993b] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. *KADS: A Principled Appoach to Knowledge-Based System Development.* Academic Press, San Diego, 1993.

[Wolf und Mehl 1991] Gregor Wolf und Wolf-Michael Mehl. Koordinationskonzepte in partizipativen Softwareprojekten. In *Arbeitsgestaltung und partizipative Systementwicklung*. Leske + Budrich, S. 81 - 94. 1991.