Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-93-16

Design & KI

**Bernd Bachmann, Ansgar Bernardi,
Christoph Klauck, Gabriele Schmidt**

Dezember 1993

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Friedrich J. Wendl
Director

Design & KI

Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt

DFKI-D-93-16

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-9304/3 C4).

Design & KI

B. Bachmann, A. Bernardi, Ch. Klauck, G. Schmidt (Hrsg.)

Zusammenfassung

In diesem Bericht werden die wichtigsten Begriffe aus dem Bereich des (Produkt)Designs aus dem Blickwinkel der künstlichen Intelligenz untersucht. Der Schwerpunkt liegt dabei nicht so sehr auf den technischen Details von existierenden Designsystemen als vielmehr in der Untersuchung der wesentlichen Konzepte wie Designmodelle, Modelle des wissensbasierten Designs, innovatives Design, etc. auf einem informellen Level, die der grundsätzlichen Beschreibung des Vorgehens eines Experten beim Design genügen.

Inhaltsverzeichnis

1

Design und Designmodelle

Elisabeth Daub

ZUSAMMENFASSUNG In diesem Kapitel über Design und Designmodelle werden nach einer Einleitung im Abschnitt 1.2 Merkmale von Design zusammengestellt und im Abschnitt 1.3 verschiedene Designmodelle beschrieben. Im Abschnitt 1.4 werden dann Beziehungen zwischen den Designmerkmalen und Designmodellen in Betracht gezogen. Daran schließt sich eine Betrachtung der Beziehungen zwischen Designmerkmalen und Merkmalen des Designprozesses am speziellen Beispiel des Problem–Lösens in Abschnitt 1.5 an. Als Kontrast dazu enthält Abschnitt 1.6 Kritik an der Betrachtung von Design als ein Prozeß des Problem–Lösens. Sich auf die kreative Seite von Design beziehend, wird dieses Kapitel mit dem Abschnitt 1.7 über innovatives Design abgeschlossen.

1.1 Einleitung

Design und Designmodelle beeinflussen - wenn auch indirekt - unser Leben und unsere Lebensqualität. Design ist in einer Vielzahl von Bereichen zu finden: in der Architektur, der Grafik, dem Maschinenbau, der Elektrotechnik, dem Industriedesign etc. Noch vielfältiger sind die Produktarten, die in diesen Bereichen auftreten; so kann ein Produkt, wie z.B. ein Gebäude riesengroß und von rein statischer Natur sein oder winzig-klein sein und auf physikalische Weise funktionieren, wie z.B. eine James-Bond-Kamera. Angesichts dieser Vielzahl von Bereichen, in denen Design vorkommt, und den vielfältigen Produktarten, die Design hervorbringt, stellt sich die Frage: Was ist Design?

Diese Frage ist wohl kaum mit einem Satz zu beantworten, denn Design als ein intelligentes Verhalten ist nach Smithers Ansicht nicht leicht zu verstehen – in mancher Hinsicht sogar überhaupt nicht [SCD+89]. Dennoch werden im folgenden, in Anlehnung an einige Versuche verschiedener Designforscher Design zu definieren, Merkmale herausgestellt, um zumindest einen ersten Eindruck von Design zu vermitteln.

1.2 Designmerkmale

Ein wesentliches Merkmal von Design ist, daß es *zielgerichtet* ist. Beim Entwerfen hat man also ein Ziel vor Augen, d.h. man hat eine Vorstellung von dem gewünschten Zustand. Dieser gewünschte Zustand kann nicht konkret beschrieben werden. Könnte er konkret beschrieben werden, wäre Design überflüssig. Aber dieser gewünschte Zustand soll ganz bestimmte Eigenschaften haben, oder anders ausgedrückt, er soll die Zielvorstellungen erfüllen. Deshalb besteht Design nicht nur daraus Artefakte zu finden, sondern auch daraus, vorherzusagen, wie diese Artefakte Zielvorstellungen erfüllen werden. Zielgerichtet bedeutet aber auch, daß Design nicht [illegible]

Betrachtung des Designprozesses bringt einige Probleme mit sich:

Ziele selbst sind Produkte von Prozessen, die wenig bekannt sind. Ziele, die der Designer setzt und verfolgt, werden beeinflußt durch das Wertesystem und die Kultur des Designers. Zudem sind die Ziele abhängig von der individuellen Interpretation der Designer, d.h. gleiche Ziele werden von einzelnen Menschen verschieden ausgelegt und auf verschiedene Weise zu erreichen versucht. Außerdem sind Ziele keine autonomen Objekte, die unabhängig voneinander realisiert werden können, sondern sie wirken auf vielfältige Weise aufeinander ein.

Ebenfalls Probleme bereitet ein weiteres Merkmal von Design: Design ist *schlecht-strukturiert*. Thomas und Caroll verstehen darunter, daß die Anfangsbedingungen, die Zielvorstellungen und die erlaubten Transformationen nicht eindeutig festgelegt sind [TC79]. Dadurch gibt es auch keinen eindeutig festgelegten Ausgangspunkt oder klar vorgegebene Vorgehensweisen in Design, die man befolgen könnte. Um den Designprozeß voranzutreiben, scheint es aber erforderlich, sich in Entscheidungssituationen auf eine Richtung festzulegen. Deshalb wird die Beschreibung von Design erweitert zu einer zielgerichteten Tätigkeit, die das Fällen von Entscheidungen umfaßt. Diese Definition kann mengentheoretisch übertragen werden in das Aufstellen einer Menge von Beschreibungen eines Zustandes bzw. Gegenstandes, der eine Menge von Einschränkungen und Bedingungen an seine Gestalt erfüllt; wobei diese Einschränkungen und Bedingungen durch die getroffenen Entscheidungen bestimmt sind. Auf ähnliche Weise beschreibt Darke Design als ein Prozeß, der eine große Anzahl von möglichen Lösungen durch externe und interne Einschränkungen reduziert. Dabei werden die externen Einschränkungen durch die spezielle Designaufgabe auferlegt und die internen Einschränkungen durch den Denkprozeß des Designers selbst [Dar79]. Diese Ansichten zeigen, daß bereits in einer frühen Phase des Designs eine große Anzahl möglicher Designlösungen durch eine Konvention, den Stil des Designers, eine persönliche Vorliebe oder durch eine willkürliche Entscheidung beträchtlich eingeschränkt werden kann.

Aber es wird auch bereits angedeutet, daß Design *dynamisch* ist, z.B. können die Unterprobleme, in die ein Designproblem zu zerfallen scheint, nicht zu Beginn des Designprozesses festgelegt werden, sondern werden vielmehr dynamisch, im Verlauf des Designprozesses produziert. Insbesondere kann Design in diesem Sinne als ein Prozeß gekennzeichnet werden, in dem einzelne Teilantworten zu einer Neudefinition der Ziele führen. Dieser Aspekt von Design wird auch von Carroll und Rosson erfaßt, indem sie Design als ein Transformationsprozeß beschreiben. D.h. erstens, daß Design statisch nicht adäquat dargestellt werden kann, und zweitens, daß Design die Entwicklung von Teil- und Übergangslösungen mit sich bringt, die im Enddesign keine Rolle mehr spielen [CR85]. Durch den Vergleich von Design und Wissenschaft lassen sich weitere typische Merkmale von Design aufstellen.

Archer stellt Design als ein Prozeß, der darauf abzielt, ein spezielles Bedürfnis zu erfüllen und etwas zu erfinden, zu konstruieren oder anzuordnen , das es vorher noch nicht gegeben hat, der Wissenschaft gegenüber. Wobei Wissenschaft ein Prozeß ist, der danach strebt, ein Phänomen von der Differenziertheit einer Situation zu isolieren und generelle Prinzipien aus Beobachtung und Versuch abzuleiten [Arc91]. Durch diese Gegenüberstellung erfaßt Archer folgende weitere Merkmale von Design: Design ist *nützlich*, d.h. Designprodukte erfüllen einen Zweck; Design ist *produktiv*, d.h. es erzeugt Produkte bzw. Produktbeschreibungen, und Design ist *innovativ*, d.h. es bringt die Entdeckung von etwas neuem mit sich. Im Gegensatz dazu ist Wissenschaft *deskriptiv*. Dies entspricht in etwa der Aussage von Simon, der danach unterscheidet, daß Wissenschaft von natürlichen Dingen handelt, davon wie sie beschaffen sind und wie sie funktionieren, während Design davon handelt, wie Dinge sein sollten [Sim69]. Zudem erkennt Archer, daß die Absichten von Design und Wissenschaft entgegengesetzt gerichtet sind: Design *spezialisiert* und Wissenschaft *generalisiert*, oder anders formuliert, Design schaut sich die Ergebnisse an, die benötigt werden, und versucht, die Zustände der Dinge vorherzusagen, die nötig sind, um zu den Ergebnissen zu gelangen; Wissenschaft dagegen schaut sich die Zustände der Dinge an und versucht Hypothesen aufzustellen, die diese Zustände erklären. In engem Zusammenhang damit steht, daß Design „Wissen verbraucht“ und Wissenschaft „Wissen aufstellt“. Während Design mit den Zielen startet und verfügbares Wissen benutzt, um zu einem Objekt zu gelangen, das die ursprünglichen Ziele erfüllt, versucht Wissenschaft durch das Ab-

damit Designtätigkeit rechtfertigen. Den Vergleich von Design und Wissenschaft abschließend wird festgehalten, daß Design im Gegensatz zur Wissenschaft *pragmatisch* ist, d.h. selbst wenn Design nicht korrekt oder vollständig ist, so stellt es dennoch einen Kompromiß dar, der die gegebenen Ziele zumindest zu einem gewissen Grade erfüllt, während es in der Wissenschaft so etwas wie „halbe" Wahrheiten nicht gibt. Insgesamt wird Design durch folgende Merkmale charakterisert:

- zielgerichtet
- schlecht–strukturiert
- dynamisch
- nützlich
- produktiv
- innovativ
- spezialisierend
- integrativ
- pragmatisch

Im Anschluß an diese Betrachtungen bzgl. Design werden verschiedene Modelle

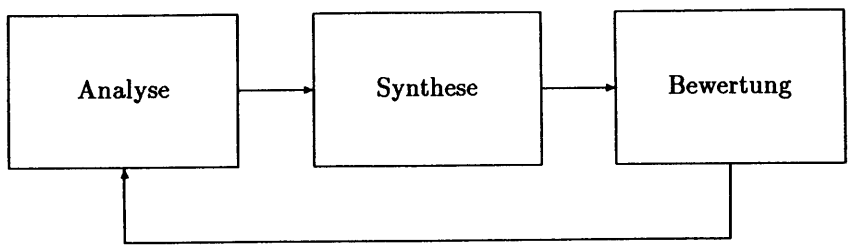


ABBILDUNG 1.1. 3–Phasen–Modell

Aber auch das 3–Phasen–Modell läßt Fragen offen: Was ist, wenn keine Lösung gefunden wird, die den Zielvorstellungen entspricht und die Rekursion nicht endet? Gibt es dann keine Synthese?

Das ständige Wiederholen von Analyse, Synthese und Bewertung legt außerdem die Betrachtung des 3–Phasen–Modells als Suchprozeß nahe. In dieser Hinsicht überlappt sich das 3–Phasen–Modell mit dem nächsten Modell.

1.3.2 Das Problem–Lösungs–Modell

Das *Problem–Lösungs–Modell* ist ursprünglich ein Modell zur Entwicklung von Theorien in der Wissenschaft. In diesem Modell geht man von einem Phänomen aus und stellt eine Hypothese oder Vermutung auf, die dieses Phänomen erklärt. Die Hypothese ist solange gültig, bis sie widerlegt wird. Ist dies der Fall, wird eine neue Hypothese aufgestellt und das Verfahren beginnt von neuem. D.h. eine Hypothese kann durch Deduktion nur widerlegt, aber nie als tatsächlich gültig bewiesen werden. Diese Ansicht von Problemlösen als ein Erzeugungs- und Testzyklus ist von Popper auf Design übertragen worden:

Zunächst wird ein Design erzeugt und wenn es sich als nicht zufriedenstellend erweist, wird ein neues Design erzeugt und so weiter [RK72]. Das Verfahren kann also als ein Suchprozeß beschrieben werden. Simon vergleicht diesen Suchprozeß mit dem Erkunden eines Labyrinthes [Sim69]. Das Labyrinth selbst kann als ein Zustandsraum aller möglichen Lösungen angesehen werden. Die Kreuzungen innerhalb des Labyrinthes entsprechen dann einzelnen Zuständen bzw. möglichen Lösungen. Ausgehend von einem ersten Design, dem Ausgangsort im Labyrinth, werden systematisch alle Gänge des Labyrinthes abgegangen, d.h. es werden systematisch Zustände bzw. mögliche Lösungen erzeugt, bis der Bestimmungsort, der Endzustand, der die Zielvorstellungen erfüllt, erreicht ist. Ähnlich einem Labyrinth kann dieser Zustandsraum groß und komplex sein, d.h. er kann viele Zustände enthalten, und es kann viele Beziehungen zwischen den Zuständen geben. Die Suche wird also das Fällen von Entscheidungen und das Aufstellen von Einschränkungen erfordern, die sich auf die Ziele beziehen und auf diese Weise bestimmte Lösungen ausschließen. Dies steht in Zusammemhang mit der Ansicht Simons, daß Probleme nicht vorgezeichnet sind als schlecht–strukturiert oder gut–strukturiert, sondern, daß es Aufgabe des Designers ist, ein unter Umständen schlecht–strukturiertes Problem in ein gut–strukturiertes umzuwandeln. Indem der Designer bestehende Informationen ignoriert oder fehlende Informationen aus seinem Langzeitgedächtnis oder anderen externen Hilfsmitteln hinzufügt, erreicht er, daß er sich zu jedem Zeitpunkt einem gut–strukturierten Problem gegenübersieht. Das impliziert, daß das Problem und damit auch die Zielvorstellungen während des Designprozesses einer ständigen Neuformulierung unterzogen werden. Die Neuformulierung der Ziele bringt aber auch neue Bewertungskriterien und damit auch neue Lösungen mit sich; in diesem Sinne ändern sich Problemaufgabe, Ziele, Bewertungskriterien und Lösungen dynamisch mit dem Fortschreiten des Designprozesses.

Basierend auf der Betrachtung von Design als ein Suchprozeß haben Simon und Newell ein automatisiertes Problem–Lösungssystem, den General Problem Solver (GPS) entworfen [NS72]. Nach Simon und Newell muß ein solches System folgende Anforderungen erfüllen: Es muß den aktuellen

Zustand, den gewünschten Zustand und Unterschiede zwischen diesen beiden Zuständen darstellen können, sowie Mechanismen enthalten, die Zustände verändern können. Außerdem muß es Aktionen auswählen können, die wahrscheinlich die Unterschiede zwischen aktuellem und gewünschtem Zustand verringern oder beseitigen, d.h. es muß Mechanismen besitzen, die das Näherrücken an die Ziele bewerten. Solche Kontrollmechanismen sind erforderlich, um die Suchmenge zu beschränken und die Designtätigkeit in die richtige Richtung zu lenken. Im folgenden werden zwei bekannte Mechanismen, um Suche zu kontrollieren, die Simulation und die Optimierung, vorgestellt.

Dazu werden zunächst die Beziehungen zwischen Zielen, Zielvorstellungen, Gestaltungsvariablen und Entscheidungsvariablen erklärt: Ein Designziel wird beschrieben durch eine Menge von Zielvorstellungen. Diese Zielvorstellungen können aber nur erfüllt werden, wenn die Gestaltungsvariablen Werte innerhalb bestimmter Wertebereiche annehmen. Die Wertebereiche werden in speziellen Aussagen als Gestaltungs–Einschränkungen und Kriterien festgelegt. Weiter wird der Designzustand durch eine Menge von Entscheidungsvariablen beschrieben. Entscheidungsvariablen und Gestaltungsvariablen stehen in kausalen Beziehungen zueinander, die eine Art Black-Box bilden, wie in Abb. 1.2 dargestellt.

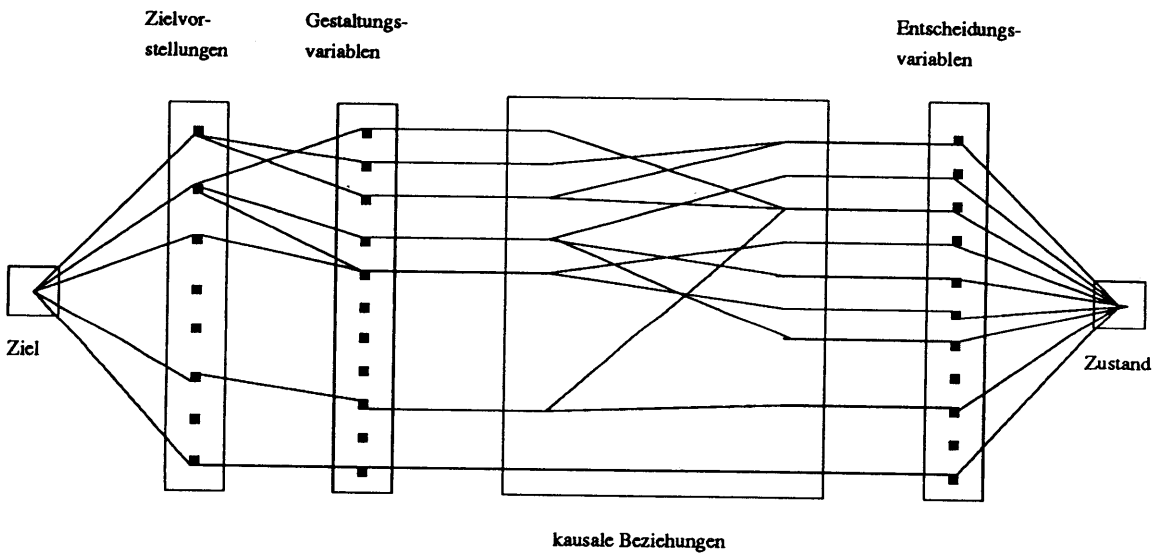


ABBILDUNG 1.2. Beziehungen zwischen Zielen, Zielvorstellungen, Gestaltungsvariablen und Entscheidungsvariablen

Die Simulation

Bei der *Simulation* wird eine mögliche Designlösung vorausgesetzt, d.h. die Entscheidungsvariablen und ihre Werte sind bereits festgelegt. Damit können auch die Werte der Gestaltungsvariablen bestimmmt werden. Liegen diese Gestaltungsvariablen in den Wertebereichen, die durch die Gestaltungskriterien angegeben sind, erfüllen sie die Zielvorstellungen. Ist dies nicht der Fall werden einige Werte der Entscheidungsvariablen geändert und die Simulation wiederholt. D.h. der Prozeß endet, sobald eine zufriedenstellende Gestalt–Ebene erreicht ist. Er läßt jedoch keine Aussage darüber zu, wie gut die gefundene Lösung im Vergleich zu anderen möglichen Designlösungen ist.

Die Optimierung

Optimierung dagegen sucht als Lösung nicht irgendein Design, sondern das best–mögliche Design. Das best–mögliche Design bedeutet das Design, das die Zielvorstellungen unter den festgesetzten Einschränkungen und Kriterien der Problemformulierung am besten erfüllt. Bei der Optimierung werden die Gestaltungs–Einschränkungen und Kriterien festgesetzt. Damit sind die Entscheidungsvariablen eingeschränkt auf gegebene Wertebereiche oder diskrete Werte. Die Gestaltungs– Einschränkungen und Kriterien werden benutzt, um die Suche durch den Zustandsraum zu kontrollieren und damit die Lösungen zu finden, die die Kriterien am besten erfüllen. Außerdem können durch

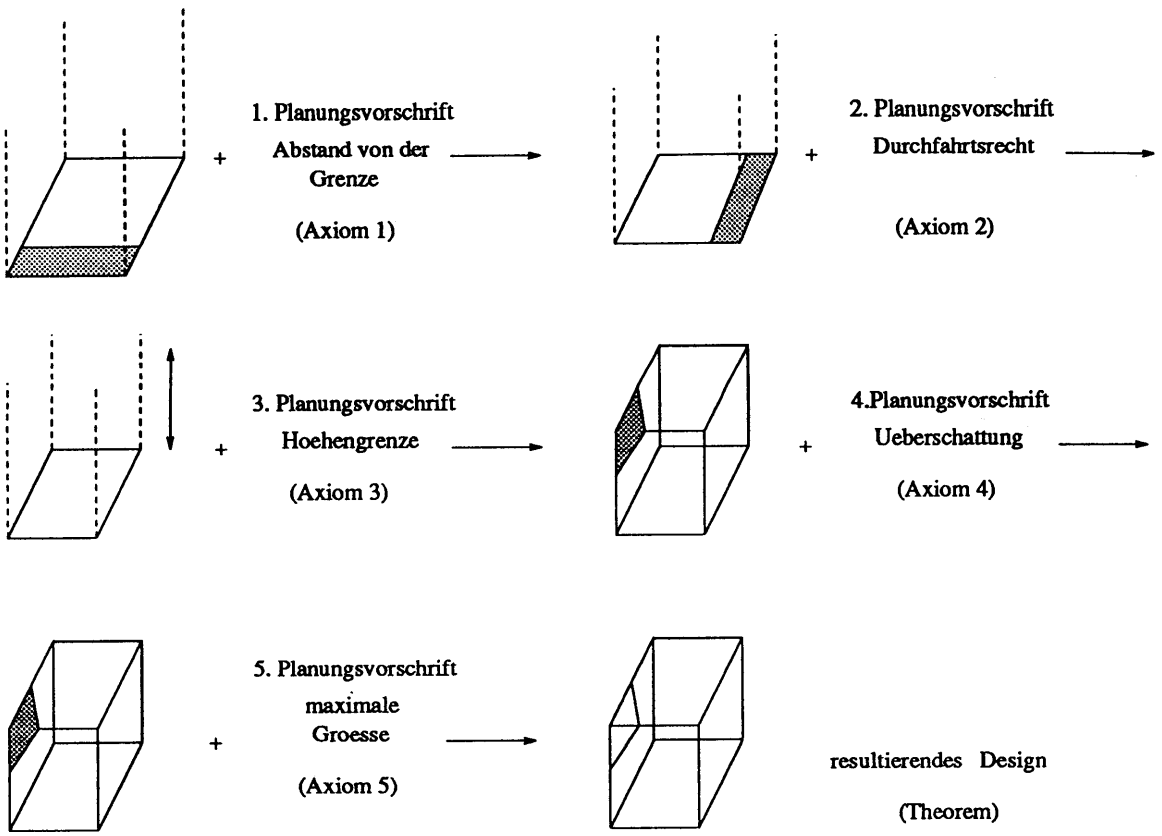


ABBILDUNG 1.3. Beispiel: Gegeben sind als Ausgangspunkt(Prämisse) ein Baugelände und als Designbedingungen(Axiome) eine Reihe von Planungsvorschriften. Daraus läßt sich durch eine Folge von Deduktionen das resultierende Design(Theorem) herleiten.

das Festlegen von Gestaltungs–Kriterien Lösungen, die nicht explizit im Zustandsraum enthalten sind, erzeugt werden. D.h. die Optimierung verkörpert im Gegensatz zur Simulation nicht nur einen Bewertungsprozeß, sondern auch einen Sytheseprozeß.

Obwohl die Optimierung eine mächtige Suchstrategie darstellt und ein Problem in Aussagen von Kriterien, Einschränkungen, Gestaltungs– und Entscheidungs–Variablen formuliert, hat sie das Gebiet des Designs nicht stark beeinflußt, mitunter deshalb, weil die Optimierung nicht der Frage nachgeht, wie man zu solchen Problemformulierungen gelangt.

Ein weiterer Versuch Design zu formalisieren, besteht darin, Design in Aussagen der Logik zu betrachten.

1.3.3 Design und Logik

Vergleicht man den Designprozeß mit der Herleitung eines mathematischen Theorems, so stellt der Designprozeß eine *Folge von Deduktionen* dar. Dabei entsprechen die Designbedingungen den Axiomen und das resultierende Design dem hergeleiteten Theorem. Wie Abb. 1.3 an einem Beispiel zeigt, schreiben die Design–Bedingungen bzw. Einschränkungen die Gestalt des Designs vor.

Mengentheoretisch kann dieser Prozeß als ein „Bedingungs–Erfüllungs"–Modell beschrieben werden. Ausgehend von der Menge aller möglichen Lösungen werden durch sukzessives Anwenden der Bedingungen bzw. Einschränkungen Teilmengen von Lösungen nacheinander ausgeschlossen. Das resultierende Design, das alle Einschränkungen erfüllt, ist dann das Ergebnis einer sukzessiven Schnittmengenbildung.

Probleme mit diesem Ansatz treten dann auf, wenn die Designaufgabe unter– oder überbestimmt ist. Ist die Designaufgabe unterbestimmt, d.h. sind nicht genügend Einschränkungen bekannt, führt dieser Prozeß nicht zu einer einzigen Lösung, sondern zu einer Menge möglicher Lösungen. Ist die

Designaufgabe überbestimmt, können unter Umständen Einschränkungen in Konkurrenz zueinander stehen und auf komplexe Weise aufeinander einwirken. Aus diesem Grunde ist dieser Ansatz nur für Designaufgaben geeignet, die nicht überbestimmt sind und deren Einschränkungen unabhängig voneinander sind.

Eine zweite Art, eine Analogie zwischen Design und Logik herzustellen, liegt darin, Design nicht als *Produkt* eines deduktiven Prozesses anzusehen, sondern als *Ausgangspunkt* von logischen Deduktionen. Aus einem Design könnten demnach mit geeigneten Theorien das Verhalten und Eigenschaften des Designs vorausgesagt werden. Aber das Verhalten und die Eigenschaften des Designs sind in Form von Zielvorstellungen bereits bekannt und was fehlt, sind die Theorien, wie man die Zielvorstellungen erfüllen kann. Deshalb ist man daran interessiert, diesen Prozeß umzukehren, so daß man aus den Zielvorstellungen die Designbedingungen herleiten könnte. Ein entsprechendes Modell (abduktives Schließen) wird im zweiten Kapitel vorgestellt.

Hier wird der Versuch, Design zu formalisieren und damit einen effektiven Einsatz von Computersystemen zu ermöglichen, fortgesetzt, indem Analogien zwischen Design und Sprache in Betracht gezogen werden.

1.3.4 Design und Sprache

Design scheint eine große Ähnlichkeit mit Sprache aufzuweisen, denn Ausdrücke wie Komposition, Stil, Kontext und Bedeutung benutzten wir im Bezug auf Design genauso wie im Bezug auf Sprache. Außerdem lassen sich direkte Parallelen ziehen zwischen dem Erzeugen eines Satzes in der natürlichen Sprache und dem Erzeugen eines Designs: Wie es Regeln, in Form einer Grammatik, gibt, die vorschreiben, wie man Wörter zu einem Satz zusammensetzt, so scheint es auch, Kompositionsregeln zu geben, die vorschreiben, wie man einzelne Komponenten beim Design zusammensetzt. Die Bedeutung von Sprache für Design liegt jedoch vorwiegend in der Repräsentation von Designbeschreibungen. Designbeschreibungen sind in Computersystemen als Zeichenketten dargestellt. Durch das Aufstellen und Manipulieren von Zeichenketten durch die Sprache können damit auch Designbeschreibungen aufgestellt und manipuliert werden.

Aber das Aufstellen und Manipulieren von Zeichenketten sind nicht die einzigen Anliegen, die sich Sprache und Design teilen; sie teilen sich auch das Anliegen, Objekte zu kennzeichnen. Auf dieser Grundlage basiert der folgende Ansatz.

1.3.5 Design und Typisierung

Indem man in Sprache und Design einem Objekt einen Namen gibt, weist man es einer bestimmten *Klasse* zu. Die Einteilung der Objekte in Klassen kann je nach Zweckmäßigkeit wechseln und ist als eine menschliche Tätigkeit abhängig von der Lebensauffassung des einzelnen. Die Abstraktion einer Klasse nennt man einen *Typ*. Genauer unterscheidet man zwei Arten von Typen, Urtypen und Prototypen. Als *Urtyp* bezeichnet man ein spezielles Design, das die Merkmale einer Klasse verkörpert. Z.B. stellt ein roter Ferrari einen urtypischen Sportwagen dar. Indem man ein Objekt als Urtyp betrachtet, wendet man Wissen an, um zu erkennen, was wesentlich und was nebensächlich für die Klasse ist und was nicht dargestellt ist. Auf diese Weise repräsentiert ein spezielles Beispiel, eine Instanz der Klasse, die Klasse selbst.

Unter einem *Prototyp* kann man ein spezielles Design verstehen, das eine Klasse veranschaulicht, z.B. in Form von Diagrammen oder einer Liste von Klassenmerkmalen. Aber ein Prototyp kann auch eine Wissensbasis zur Definition eines Designraumes darstellen. Im Unterschied zu einem Urtyp ist ein Prototyp also ein Design, aus dem andere Designs entstehen. In diesem Sinne kann Design als ein *Prozeß der Instantiation* betrachtet werden: Beginnend mit einem Prototyp, der den Designraum definiert, setzt man Komponenten zusammen, die ein Objekt erzeugen, das den Typ veranschaulicht, indem es die Merkmale der Klasse erfüllt. D.h. der Designprozeß führt zur Beschreibung eines speziellen Designs, einer Instanz der Klasse [Mon78].

Es gibt drei Kategorien von Designtätigkeit, um auf diese Weise neue Designs zu erzeugen: Prototypverfeinerung, Prototypanpassung und Prototyperzeugung. Bei der *Prototypverfeinerung* arbeitet man innerhalb der Einschränkungen des Designraumes, der durch den Prototyp festgelegt ist. Ein Prototyp, der das konventionelle Wissen über Flugzeugentwurf verkörpert, z.B. indem er die Form der Tragflächen und die Plazierung der Motoren vorschreibt, kann durch das Abändern einzelner Parameter, wie z.B. der Form der Nase oder der Länge der Tragflächen, zu einem neuen Design

Design-merk-male	Designmodelle				
	3–Phasen Modell	Problem-Lösungs-Modell	Logik-Modell	Typologie	
				Prototyp-verfeinerung	Prototyp-erzeugung
zielgerichtet	×	×	–	+	+
schlecht-sruktutriert	×	×	–	+	+
dynamisch	×	×	–	+	+
nützlich	–	–	–	–	–
produktiv	–	–	–	–	–
innovativ	+	+	+	+	×
integrativ	+	+	+	×	×
spezialisierend	+	+	+	×	+
pragmatisch	+	+	–	+	+

Zeichenerklärung: – :Modell berücksichtigt Merkmal nicht
+ :Modell berücksichtigt Merkmal
× :Modell berücksichtigt Merkmal stark

Tabelle 1.1. Beziehungen zwischen Designmerkmalen und Designmodellen

verfeinert werden. *Prototypanpassung* dagegen setzt sich über Einschränkungen des Designraumes hinweg. Im Flugzeugentwurf hat man z.B. konventionelles Wissen überschritten, indem man die Motoren über den Tragflächen plaziert hat, so daß die Auspuffgase über die Tragflächen gehen und den Auftrieb erhöhen. Diese Anpassung hat zu einem neuen, besseren Design geführt. Die dritte Kategorie, *Prototyperzeugung*, liegt vor, wenn ein völlig neuer Prototyp entsteht, z.B. das Design des ersten Flugzeuges. Da neue Prototypen oft durch eine Reihe von Anpassungen zustande kommen, kann Prototyperzeugung als Prototypanpassung hinreichenden Ausmaßes angesehen werden.

Diese Betrachtungen deuten bereits an, daß Prototypverfeinerung für Routine–Designs eingesetzt wird und daß innovative Designs gerade dann entstehen, wenn sich der Designer außerhalb des Designraumes bewegt und sich auf andere Wissensquellen beruft. Innovative Designs und Methoden, die zu innovativen Designs führen, werden jedoch erst an einer späteren Stelle in diesem Kapitel diskutiert.

1.4 Beziehungen zwischen Designmerkmalen und Designmodellen

An dieser Stelle werden Designmodelle aus dem Abschnitt 1.3 in Bezug gesetzt zu den Designmerkmalen, wie sie im Abschnitt 1.2 aufgestellt worden sind: Auf diese Art soll hervorgehoben werden, daß bestimmte Designmodelle bestimmte Designmerkmale berücksichtigen, nicht berücksichtigen oder stark berücksichtigen. Dazu dient die Tabelle 1.1.

Es fällt auf, daß die beiden Merkmale nützlich und produktiv von keinem der Modelle berücksichtigt werden. Dies liegt daran, daß diese beiden Merkmale sich eher auf die Produkte des Designprozesses beziehen, als auf den Designprozeß selbst. Von daher sind sie im wesentlichen unabhängig von der angewandten Designmethode. Dagegen spielen die drei Merkmale zielgerichtet, schlecht–strukturiert und dynamisch sowohl in dem 3–Phasen–Modell als auch in dem Problem–Lösungs–Modell eine große Rolle. Beide Modelle können als ein Suchprozeß betrachtet werden; eine solche Betrachtung setzt jedoch die Existenz von Zielen voraus, so daß die Suche eine Richtung bekommt. Um diese Richtung festzulegen, müssen i.a. Entscheidungen getroffen und Zielvorstellungen revidiert werden. Dies erklärt, warum die drei Merkmale zielgerichtet, schlecht–strukturiert und dynamisch in diesen zwei Modellen stark ausgeprägt sind. Im Gegensatz dazu berücksichtigt das Logik-Modell diese drei Merkmale überhaupt nicht. Die Merkmale schlecht–strukturiert und integrativ treten in

zesses nicht und die Zielvorstellungen werden nicht geändert, d.h. auch die Merkmale zielgerichtet und dynamisch werden hier nicht berücksichtigt. Schließlich fließt auch das Merkmal pragmatisch nicht in dieses Modell ein, da das Ergebnis des Designprozesses alle Einschränkungen erfüllt. Ansonsten kann man noch festhalten, daß der Ansatz mit Prototypen stark integrativ ist, in dem Sinne, daß man bei diesem Ansatz bereits von einem bestimmten Prototyp ausgeht, der eine bestimmte Wissensbasis definiert. Bei der Prototypverfeinerung bewegt sich der Designer innerhalb des Designraumes, der durch den Prototyp festgelegt ist, so daß das Merkmal spezialisierend stärker betont wird, während bei der Prototypanpassung bzw. Prototyperzeugung die innovative Komponente von Design stärker hervortritt, da der Designer die Grenzen des Designraumes überschreitet.

Die obige Zuordnung ist nicht als starre Abgrenzung aufzufassen, insbesondere da eine solche Abgrenzung nahezu unmöglich ist, denn schon einzelne Designmerkmale können nicht vollkommen getrennt voneinander betrachtet werden. So scheinen die zwei Merkmale schlecht–strukturiert und dynamisch miteinander verknüpft zu sein und jedes Design, das diese Merkmale aufweist, wird irgendwo auch pragmatisch sein, ebenso wie jedes innovative Design auch integrativ sein wird. Mit dem Hinweis, daß die Zuordnung lediglich unternommen worden ist, um Schwerpunkte und Besonderheiten zu erkennen, wird diese Betrachtung abgeschlossen.

Anschließend wird ein Ansatz von Goel und Pirolli vorgestellt, der für unsere Betrachtungen interessant ist, weil er zwischen Designmerkmalen und Merkmalen des Designprozesses trennt und insbesondere auf Beziehungen zwischen den beiden eingeht, indem er beschreibt, wie sich einzelne Merkmale des Designprozesses aus den Designmerkmalen ergeben [GP89].

1.5 Beziehungen zwischen Designmerkmalen und Merkmalen des Designprozesses am Beispiel des Problem–Lösens

Der Ansatz von Goel und Pirolli betrachtet Design als ein Problem–Lösungsprozeß. Er ist entstanden als Versuch, Design–Problem–Lösen von Nicht–Design–Problem–Lösen zu unterscheiden, indem man den Design–Problem–Raum durch bestimmte Merkmale kennzeichnet [GP89]. Dazu haben Goel und Pirolli acht Merkmale zusammengestellt, die in der Aufgabenumgebung typischer Designaufgaben enthalten sind, und haben deren Auswirkung auf den Design–Problem–Raum untersucht. Auf diese Weise haben sie im wesentlichen sieben Merkmale gefunden, die den Design–Problem–Raum aufbauen und damit den Designprozeß charakterisieren. Abb. 1.4 zeigt die Merkmale des Designprozesses und wie sich diese aus den Designmerkmalen ergeben. Da die Betrachtung von den acht Designmerkmalen ausgeht, werden diese zunächst zusammengestellt:

1. Es existieren viele Freiheitsgrade in der Problemformulierung. Dieses Merkmal hängt zusammen mit der früheren Aussage, daß Design integrativ ist.

2. Die Rückmeldung aus der Welt ist während dem Problem–Lösen begrenzt und verspätet.

3. Die Eingabe in den Designprozeß besteht aus Zielvorstellungen und die Ausgabe ist eine Gegenstandsspezifikation. Dieses Merkmal bezieht sich auf die Tatsache, daß Design schlecht–strukturiert ist.

4. Der Gegenstand muß unabhängig von dem Designer arbeiten, d.h. der Designer kann später in der Welt nicht neben dem Gegenstand stehen und erklären, wie dieser funktioniert oder angewandt wird.

5. Spezifikation und Fertigung des Gegenstandes sind zeitlich getrennt.

6. Mit jeder Aktion in der Welt sind Kosten verbunden. Dies impliziert, daß Fehler beim Design bestraft werden.

7. Antworten sind weder richtig noch falsch, sondern nur besser oder schlechter. Durch diese Abstufung erfaßt das Merkmal die pragmatische Komponente von Design.

8. Das Problem tendiert dazu groß und komplex zu sein.

Darauf aufbauend werden im folgenden die sieben Merkmale des Designprozesses als Auswirkungen einzelner Designmerkmale beschrieben:

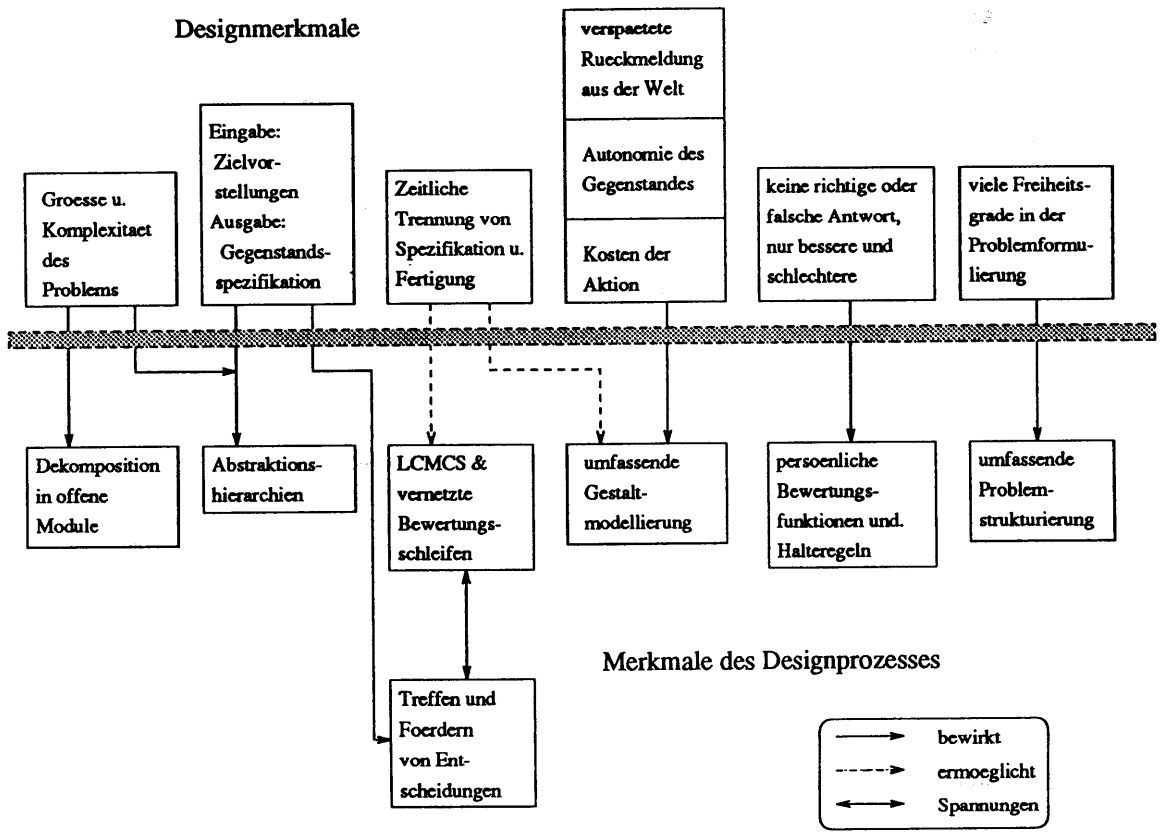


ABBILDUNG 1.4. Designmerkmale und Merkmale des Designprozeßes und ihre Beziehungen aus (Goel and Pirolli, 1989)

1. Die vielen Freiheitsgrade in der Problemformulierung bringen eine *umfassende Problemstrukturierung* mit sich. Unter Problemstrukturierung versteht man den Prozeß, fehlende Informationen zu finden und zu benutzen, um den Problemraum zu definieren.

2. Die verspätete oder begrenzte Rückmeldung aus der Welt, die Autonomie des Gegenstandes und die Bestrafung für Fehler zwingen den Designer zu einer *umfassenden Gestaltmodellierung*. D.h. er benötigt Mechanismen, die die aktuelle Problem–Lösung simulieren, so daß er während des Designprozesses kontinuierlich Rückmeldungen erhält und Bewertungen anstellen kann, um auf diese Weise Fehler zu vermeiden oder zumindest zu verringern. Ermöglicht wird eine umfassende Gestaltsmodellierung durch die zeitliche Trennung von Spezifikation und Fertigung des Gegenstandes.

3. Die Tatsache, daß es keine richtigen oder falschen Antworten zu einem Designproblem gibt, bringt den Gebrauch von *persönlichen Bewertungsfunktionen und Haltefunktionen* mit sich.

4. Die *Begrenzte–Entscheidungs–Methode–Kontroll–Strategie und vernetzte Bewertungsschleifen* (Limited Commitment Mode Controll Stategy with Nested Evaluation Cycles) ist eine Strategie, die den Designer nicht zwingt an bereits getroffenen Entscheidungen festzuhalten, indem sie ihm ermöglicht, Entscheidungen zurückzunehmen. D.h. eine Entscheidung, die zu einem früheren Zeitpunkt sehr attraktiv erschienen ist und sich in einem späteren Zusammenhang als ungeeignet erweist, kann rückgängig gemacht werden. Dies erfordert jedoch neue Bewertungen unter Berücksichtigung der geänderten Entscheidung. Erneut ermöglicht erst die zeitliche Trennung von Spezifikation und Fertigung diese Strategie.

5. Um aus Zielvorstellungen einen Gegenstand komplett zu spezifizieren, muß der Designer *Entscheidungen treffen und fördern*, damit der Designprozeß vorangetrieben wird. Dadurch besteht zwischen diesem Merkmal und dem vorherigen eine ständige Spannung: zum einen wird das Treffen von Entscheidungen gefordert, zum anderen wird versucht, Entscheidungsmöglichkeiten so lange wie möglich offenzuhalten.

6. Die Größe und Komplexität von Designproblemen führt zu einer *Dekomposition in offene Module*. Darunter versteht man die Zerlegung des Problems in Teilprobleme oder Module. Dabei wirkt sich eine Entscheidung innerhalb eines solchen Moduls auch auf alle anderen Module aus. Dies erfordert einen laufenden Überwachungsprozeß der Beziehungen der Module untereinander.

7. Die Tatsache, daß die Zielvorstellungen die Spezifikation des Gegenstandes nicht eindeutig festlegen, zusammen mit der Größe und Komplexität des Problems bringen das Verwenden von *Abstraktions–Hierarchien* mit sich. D.h. man bewegt sich auf verschiedenen Detailebenen abwärts, bis man die Ebene erreicht hat, die den Gegenstand spezifiziert. So kann z.B. eine Zwischenebene eingeführt werden, auf der eine Funktion gesucht wird, die die Zielvorstellungen erfüllt. Auf dieser Ebene interessiert nur die Funktion, der Gegenstand selbst wird als Black–Box betrachtet und die Bewertung bezieht sich nur darauf, wie gut diese Funktion den Zielvorstellungen nahekommt. Hat man eine geeignete Funktion gefunden, geht man eine Ebene tiefer und sucht dort nach Formen des Gegenstandes, die eben diese Funktion erfüllen. D.h. die Bewertung bezieht sich jetzt nur darauf, wie gut die Formen die Funktion erfüllen. Das Verwenden von Abstraktions–Hierarchien stellt also eine weitere Möglichkeit dar, das Problem zu zerlegen.

Diese Betrachtung abschließend, wird festgehalten, daß die vorgestellten Merkmale des Designprozesses bereits bekannte Konzepte, wie z.B. das Einbinden von Wissen, das Anwenden von Erzeugungs– und Bewertungszyklen, das Treffen von Entscheidungen und Aufgabenzerlegung, enthalten. Damit jedoch nicht der Eindruck entsteht, daß Design im wesentlichen Problem–Lösen ist, werden im nächsten Abschnitt Gründe angeführt, Design nicht nur als Problem–Lösen zu betrachten. Dabei bezieht man sich im wesentlichen auf Tunnicliffe und Scrivener [TS92].

1.6 Design ist nicht Problem–Lösen

[redacted]

verdeutlicht werden. Die Zutaten entsprechen der Technologie und die Anweisungsfolge für die Zubereitung entspricht der Technik. D.h. unter Technologien versteht man physikalische Teile oder Materialien und unter Techniken versteht man standardmäßige Verfahren oder Vorgehensweisen.

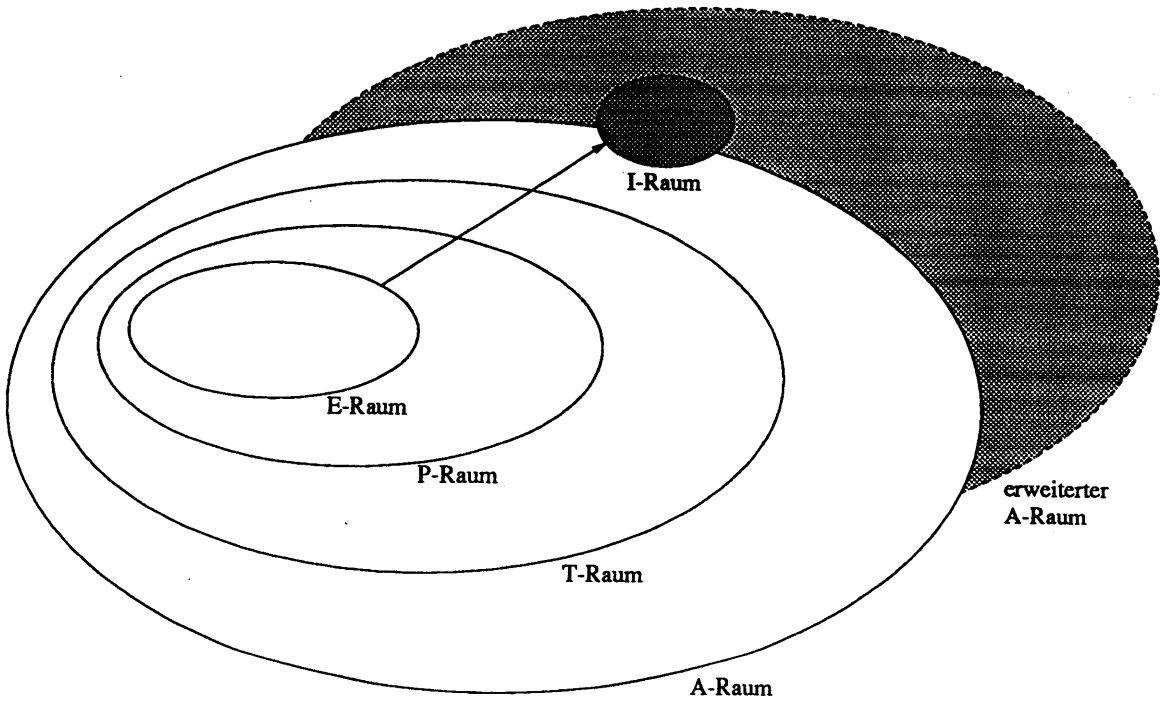


ABBILDUNG 1.5. Lösungsraum eines Designbereiches aus (Navin Chandra, 1992)

Teilt man den Lösungsraum eines Designbereiches auf in Unterräume, wie in Abb. 1.5 gezeigt, dann entspricht der P–Raum der Designkultur. Der P–Raum besteht aus den sogn. Puzzle–Lösungen, die allein durch das Anwenden bekannter Techniken und Technologien entstehen. Innerhalb diese P–Raumes liegt nur noch der E–Raum; er enthält alle bereits bestehenden Designs. Umgeben wird der P–Raum von dem T–Raum, der die Lösungen umfaßt, die bekannte Techniken auf neue Arten kombinieren und bekannte Technologien benutzen. Der äußerste, alle anderen Räume einschließende Raum, ist der A–Raum, in dem alle legalen Kombinationen einer Technologie, die in der Designkul

Der zweite Ansatz zu innovativem Design ist die Methode der *Assoziation*. Assoziation versucht frühere Designerfahrungen und Wissen außerhalb des Problembereichs heranzuziehen. Dies erfordert die Fähigkeit nützliche Analogien zwischen der aktuellen Designaufgabe und Designs aus der Vergangenheit oder aus anderen Designkulturen zu erkennen. Dabei erscheinen die Designs um so innovativer, je weniger verwandt die Kulturen sind, zwischen denen Analogien übertragen werden. Die Übertragung von Ideen zwischen nichtverwandten Kulturen umfaßt sowohl Technologien als auch Techniken: Das Importieren von Technologien in eine Kultur führt, wie in Abb. 1.5 angedeutet, zu einem erweiterten A–Raum, da sich dadurch die Anzahl der physikalischen Primitiven, die kombiniert werden können, erhöht. So bringt z.B. das Impotieren von Mikro–Elektronik in den Designbereich von Haushaltsgeräten eine beträchtliche Ausdehnung des A–Raumes mit sich. Das Importieren von Techniken kann einen Sprung von dem aktuellen P–Raum auf eine Insel von Lösungen bewirken. Dieser sog. I–Raum ermöglicht es dem Designer, bekannte Technologien in einem ande-

Schaltkreis–Designtechniken auf die Designkultur der Hydraulik.

Aber durch Assoziation sollen nicht nur Ähnlichkeiten zwischen der aktuellen Aufgabe und außerkulturellen Designaufgaben erkannt werden, sondern die aktuelle Aufgabe soll auch in Analogie zu der außerkulturellen Designaufgabe gelöst werden können. Beides erfordert eine Betrachtung von Objekten oder Sachverhalten aus verschiedenen Perspektiven und eine Wahrnehmung dieser Objekte bzw. Sachverhalte, die durch deren Kontext nicht beeinflußt wird. Damit enthält Assoziation Aspekte holistischer, feldunabhängiger Denkweisen. Außerdem bezieht sich Assoziation auf das Verfahren des Brainstorming aus der Psychologie, das versucht, durch Sammeln von spontanen Einfällen die beste Lösung zu einem Problem zu finden. Insgesamt kann die Methode der Assoziation durch folgende Merkmale gekennzeichnet werden:

- Der Idee, daß Kreativität erreicht wird durch das Transformieren von Fragen.
- Der Anwendung von Analogie
- Der Vorstellung, daß Innovation nicht aus einem bewußten Versuch entsteht innovativ zu sein, sondern durch das Erzeugen einer breiten Vielfalt von Alternativen und dem Wegwerfen der schlechten.

1.9 Literaturverzeichnis

[Ale69] C. Alexander. *Notes on the Synthesis of Form.* Harvard University Press, Cambridge, Massachusetts, 1969.

[Arc91] L.B. Archer. The nature of research into design and design education. In *Proc. 4th National Conference on Design and Technology Education Reseach and Curriculum Development*, pages 1–11, DATER 91, Loughborough University of Technology, September 1991.

[Asi62] W. Asimow. *Introduction to Design.* Prentice–Hall, Englewood Cliffs, New Jersey, 1962.

[Bij85] A. Bijl. An approach to design theory. In H. Yoshikawa and E.A. Warman, editors, *Design theory for CAD*, pages 3–25. North-Holland, 1985.

[CN80] A. Cross and M. Nathenson. Design methods and learning method. In R. Jacqques and J. Powell, editors, *Design : Science : Method*, pages 281–294. Westbury House., 1980.

[CR85] J.M. Carroll and M.B. Rosson. Usability specifications as a tool in iterative development. In H.R. Hartson, editor, *Advances in Human-Computer Interaction*, pages 1–28. Ablex, 1985.

[Cro84] A. Cross. Towards an understanding of the intrinsic values of design education. *Design Studies*, 5(1):31–39, January 1984.

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge–Based Design Systems.* Addison–Wesley Publishing Company, 1990.

[Dar79] J. Darke. The primary generator and the design process. *Design Studies*, 1(1):36–44, 1979. Butterworth.

[Eas81] M.C. Eastman. Recent developments in representation in the science of design. In *Proc. Design Automation*, pages 13–21, Washington, D.C.: Institute for Electronics and Electronical Engineers., 1981.

[GP89] V. Goel and P. Pirolli. Design within information–processing–theory: The design problem space. *AI MAGAZINE*, pages 19–36, 1989.

[Law72] B.R. Lawson. *Problem Solving in Architectural Design.* PhD thesis, University of Aston in Birmingham, 1972.

[Law79] B.R. Lawson. Cognitive strategies in architectural design. *Ergonomics*, 22(1):59–68, 1979.

[LR87] J. Landsdown and C. Roast. The possibilities and problems of knowledge–based systems for design. *Environment and Planning B*, pages 255–266, 1987.

[Mon78] R. Moneo. On typology. *Oppositions*, 13:23–45, 1978.

[NS72] A. Newell and H.A. Simon. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[RK72] Routledge and P. Kegan, editors. *Conjunctions and Refutations, the Growth of Scientific Knowledge*, London, 1972.

[SCD+89] T. Smithers, A. Conkie, J. Dohency, B. Logan, and K. Millington. Design as intelligent behaviour: an AI in design research programme. In J.S. Gero, editor, *Artificial Intelligence in design. Proc. 4th Int. Conf. on the application of Aritifical Intelligence in Design*, pages 293–334, Cambridge, UK, July 1989. Springer-Verlag.

[Sim69] H. A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, Massachusetts, 1969.

[Sim81] H.A. Simon. *The Sciences of the Artificial.* Mass.: MIT Press, Cambridge, 2d edition, 1981.

[TC79] J.C. Thomas and J.M. Carroll. The psychological study of design. *Design Studies*, 1(1):5–11, 1979.

[TS92] A. Tunnicliffe and S. Scrivener. Design issues for design knowledge elicitation. In R. Brian and A. Mark, editors, *Proc. 7th Banff Knowledge Acquisition for knowledge-based systems Workshop*, volume 2, pages 1–8, 1992.

2

Modelle des wissensbasierten Designs

Holger Huf

ZUSAMMENFASSUNG Dieser Abschnitt behandelt Grundlagen des wissensbasierten Designs. Dabei wird sowohl auf die Repräsentation, als auch auf den Designprozeß selbst eingegangen. Es werden verschiedene Modelle vorgestellt; ein Ansatz nach Coyne, Rosenmann, Radfort, Balachandran und Gero ([CRR+90]), ein taskorientierter Ansatz nach Chandrasekaran ([Cha90]) und ein KADS-Ansatz nach Kruger und Wielinga ([KW93]).

2.1 Repräsentation

In diesem Kapitel werden Möglichkeiten der Informationsrepräsentation vorgestellt. Dabei wird eine Form verwendet, die sich stark an der Prädikatenlogik orientiert.

2.1.1 Tatsachen (Facts)

Tatsachen kann man in zwei Bereiche aufteilen:

Objekte (Objects) dienen als grundlegende Informationseinheiten.

Beziehungen (Relations) beschreiben die Zusammenhänge zwischen den einzelnen Objekten oder bestimmen Eigenschaften von Objekten.

Beispiel:

In einer Klötzchenwelt könnten z. B. folgende Tatsachen gelten:

StehtAuf(A, B)	d. h. Klötzchen A steht auf Klötzchen B
StehtAuf(B, C)	d. h. Klötzchen B steht auf Klötzchen C
Farbe(A, Grün)	d. h. Klötzchen A hat eine grüne Farbe.

Diese Darstellung ist natürlich nur eine von vielen möglichen; die Aussage *Farbe (A, Grün)* könnte ebenso durch *Grün (A)* repräsentiert werden. Die erste Variante hat jedoch den Vorteil, daß sich in diesem Falle nicht nur *A*, *B* oder *C*, sondern auch *Grün* durch eine Variable ersetzen läßt. Diese können sowohl mit Tatsachen als auch mit Objekten instantiiert werden. Im folgenden werden Variablen durch einen kleinen Anfangsbuchstaben gekennzeichnet.

Die hier vorgestellte Form der Repräsentation von Tatsachen stellt die Grundlage einer relationalen Datenbank dar. Diese Grundlage wird im folgenden weiter ausgebaut.

2.1.2 Wissen (Knowledge)

Wissen kann als Zusammenhang zwischen Tatsachen betrachtet werden. Eine häufig verwendete Art solcher Zusammenhänge sind *logische* Beziehungen mit Hilfe von **and**, **or** und **if** (Wissensklauseln).

Beispiel:

In einer Klötzchenwelt gilt z. B.:

If(Über(a, b), StehtAuf(a, b))
If(Über(a, b), **And**(StehtAuf(a, c), Über(c, b))).

Beziehungen dieser Art werden oft auch in Infix-Notation geschrieben:

Über(a, b) **If** StehtAuf(a, b)
Über(a, b) **If** StehtAuf(a, c) **And** Über(c, b).

2.1.3 Kontrollwissen (Control Knowledge)

So wie Wissen Zusammenhänge zwischen Tatsachen beschreibt, so beschreibt Kontrollwissen Zusammenhänge zwischen Wissensklauseln. Dieses Meta–Wissen stellt Informationen zur Verfügung, wie das Wissen behandelt werden muß, z. B. in welcher Reihenfolge die einzelnen Wissensklauseln ausgewertet werden müssen.

2.1.4 Zusammenfassung

In diesem Kapitel wurde eine Möglichkeit beschrieben, Wissen im Computer zu repräsentieren. Dies geschieht in Form von Tatsachen über Design, Beziehungen zwischen Tatsachen (Wissen) und Beziehungen zwischen den verschiedenen Wissensklauseln (Meta–Wissen).

2.2 Schlußfolgern

Die Vorraussetzungen des Designprozesses liegen in logischer Form vor. Daher ist es naheliegend, den Designprozeß selbst durch automatisches Schlußfolgern (automatic reasoning) zu lösen.

Dabei kommen drei verschiedene Verfahren zum Zuge:

- Deduktion
- Induktion
- Abduktion

Deduktion und Induktion liefern dabei eindeutige (von der Spezifikation abgeleitete) Ergebnisse. Bei der Abduktion (Umkehrung der Deduktion) ist wegen der fehlenden Eindeutigkeit ein Backtracking–Mechanismus notwendig.

Während des Designprozesses versucht man nun, die im Rechner vorliegenden Beschreibungen — z. B. Aussagen über Größe, Form, Farbe des Produktes — zu vervollständigen, d. h. Attribute, die nicht in der Wissensbank vorhanden sind (z. B. Preis, Leistungsfähigkeit), aus den vorhandenen abzuleiten, in anderen Worten, die Wissensbank zu interpretieren (Deduktion). Das ist jedoch nicht die übliche Vorgehensweise eines Designers.

Normalerweise stehen allgemeinere Überlegungen — z. B. über Funktionalität — am Anfang. Mit diesen wird dann versucht, eine Designbeschreibung zu finden, die diese Bedingungen erfüllt. Das ist ein Prozeß analog der Abduktion und bringt auch die gleichen Probleme mit sich, d. h. die so gewonnene Designbeschreibung muß nicht notwendigerweise korrekt sein; eventuell muß ein erneuter Versuch unternommen werden.

Das aus Lehrbüchern zu extrahierende Wissen ist meist interpretativer Art. Unter gewissen Voraussetzungen kann dieses Wissen jedoch zur Erzeugung von Designbeschreibungen durch Abduktion herangezogen werden. Dies führt allerdings nicht zu einer einzigen Designbeschreibung, sondern zu einer Menge möglicher Designs, die alle die Designbedingungen erfüllen, einem *Designraum*, der in Kapitel 2.4 beschrieben wird.

Der Designprozeß besteht nun aus dem Parsen des Startzustandes, um zu versuchen, ihn in eine Form zu überführen, die die Beschreibung des Designs erfüllt.

Diese Art von Wissen wird als generatives (erzeugendes) oder syntaktisches Wissen bezeichnet.

2.4 Designräume

In diesem Kapitel wird beschrieben, wie das Konzept des interpretativen (siehe Kapitel 2.2) und des generativen Wissens (siehe Kapitel 2.3) dazu verwendet werden kann, ein Designsystem zu bilden.

2.4.1 Interpretation und Erzeugung (Generation)

Mit Hilfe von interpretativem Wissen kann man ein interpretatives System definieren, welches Zusammenhänge zwischen der Designbeschreibung und bestimmten, sich daraus ergebenden, Eigenschaften (performance) beschreibt.

Ein generatives System, basierend auf generatives Wissen, kann nun dazu verwendet werden, einen Designraum der möglichen Designsbeschreibungen zu bilden. Wenn z. B. als Designbeschreibung ein Auto vorgegeben ist, so sagt dies nichts über die Art des Autos aus. Das Ergebnis des Designs kann sowohl ein Kleinwagen als auch ein Tieflader sein. Die Materialien und Werkzeuge die zur Verfügung stehen, schränken den Designraum jedoch stark ein, so daß z.B. nur Wagen unter einer Tonne Gesamtgewicht in Frage kommen (generatives System, siehe Abbildung 2.1, *Designraum*). Nachdem eine konkrete Designbeschreibung ausgewählt (Abb. 2.1, *Design*) und der Designprozeß beendet ist — und somit die Designbeschreibung in die Eigenschaften überführt worden sind (interpretatives System) — muß überprüft werden, ob die Eigenschaften mit den gewünschten übereinstimmen, z. B. geringer Verbrauch, vier Sitzplätze, etc. Falls dies nicht der Fall ist, wird ein anderes spezielles Design ausgewäht und der Versuch wiederholt.

Der generative Prozeß liefert also die mit dem vorhandenen Vokabular möglichen Designs (abhängig von den Erzeugungsregeln). Die Realisierung eines speziellen dieser Designs ist dann Sache des interpretativen Systems. Diese *generate-and-test*-Methode ist jedoch, insbesondere bei umfangreicheren Designs, nicht praktikabel.

Eine andere Vorgehensweise ist es, durch Abduktion den sich durch die Eigenschaften (Abb. 2.1, *performances*) ableitbaren Designraum zu erzeugen. Aus der Schnittmenge dieses Designraums mit dem durch das generative Wissen erzeugten Designraum liefert dann die Menge aller möglichen Designs (Abbildung 2.1, schraffierte Fläche).

2.4.2 Repräsentation von Designwissen

Designwissen ist Wissen, mit dessen Hilfe Designräume definiert werden können. Designwissen kann man sich aber auch in Form generischer Beschreibungen vorstellen (Prototypen). Daher kann man erwarten, daß sich syntaktisches Wissen in Form von Typbeschreibungen darstellen läßt (z. B. akzeptable Wertebereiche für Attribute), so daß sich die Klasse von Designs, die man durch das generische Design erhält, mit Hilfe dieser Beschreibungen in eine konkrete Designbeschreibung instanziieren läßt. Die Übertragung der prototypischen Beschreibung in eine Designinstanz ist jedoch nicht immer einfach.

2.4.3 Designprozesse

In diesem Unterkapitel wird näher auf das Wissen über den Designprozeß selbst eingegangen.

Man kann die dem Designprozeß zugrundeliegenden Operationenen als Objekte (objects, siehe Kapitel 2.1.1) ansehen. Dann beinhalten das Wissen (knowledge) Informationen über das Schließen über diese Objekte. Kontrollwissen kann als Beziehungen zwischen Regeln angesehen werden (siehe Kapitel 2.1.3). Generative oder syntaktische Regeln können als Aktionen gesehen werden, die Tatsachen (facts) ändern. Im folgenden wird nun betrachtet, ob Tatsachen über Aktionen das „Vokabular", d. h. die Beschreibung des Designs (siehe Kapitel 2.3) innerhalb eines Designsystems verändern können.

Dieses Designkontrollsystem, welches ein Designsystem kontolliert, hat als Objekte Aktionen und

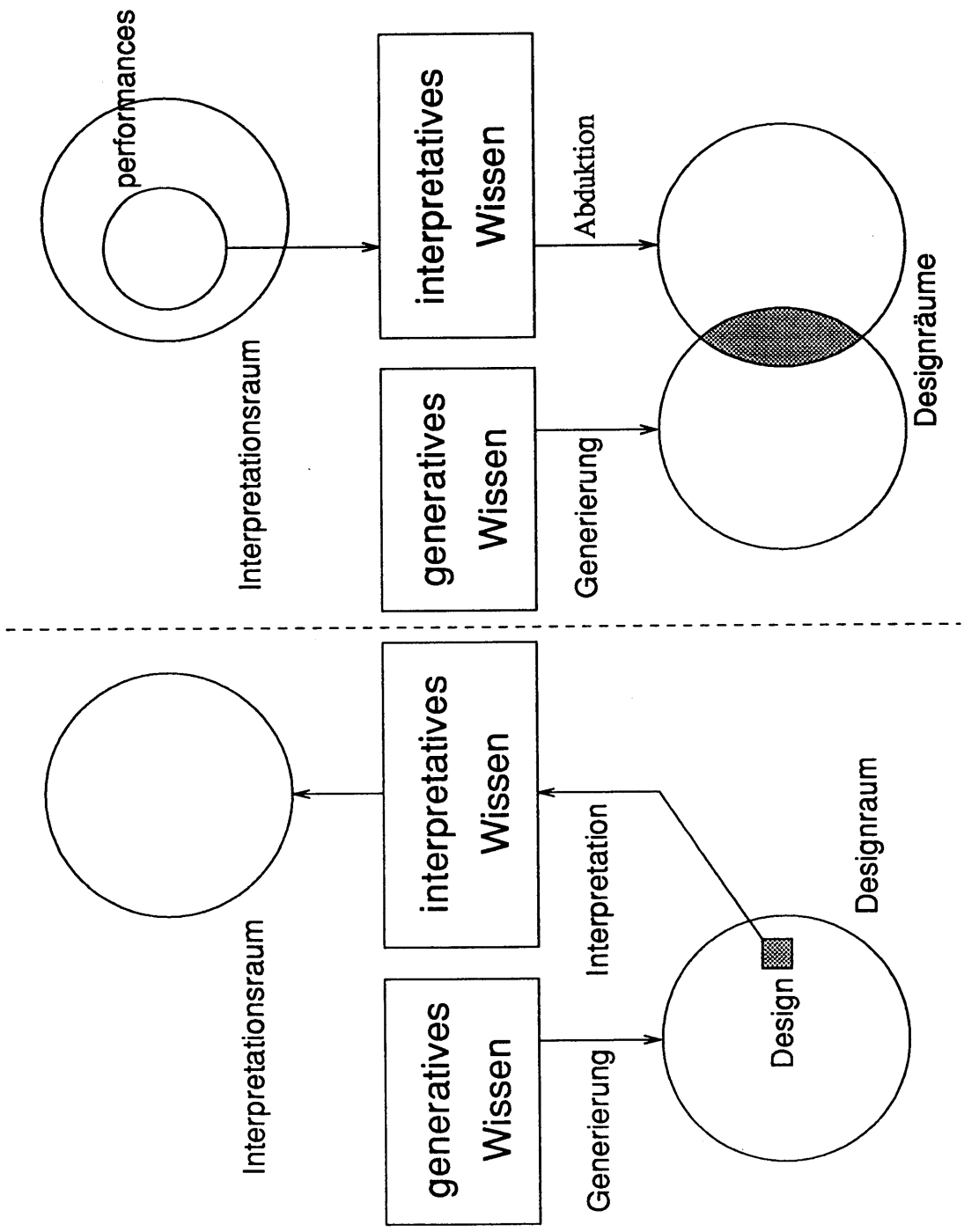


ABBILDUNG 2.1. Zwei Designverfahren

als Wissen (Knowledge, siehe Kapitel 2.1.2) Aussagen über Beziehungen zwischen Aktionen. Es existiert daher Wissen zur Interpretation der Aktionen und Wissen, welches die Syntax solcher Aktionen beschreibt.

Wenn man bedenkt, daß man Design auch als Auswahl geeigneter Aktionen in der richtigen Reihenfolge ansehen kann, wird deutlich, daß die Auswahl und Generierung von Aktionen einen hohen Stellenwert einnehmen kann. Dies beinhaltet auch die Vermeidung von eventuell auftretenden Konflikten.

2.4.4 Wissenserwerb

Die Repräsentation des Designwissens wurde in den letzten Kapiteln in Form von logischen Regeln dargestellt. Diese lassen sich relativ einfach in einem Computersystem verwenden. Es gibt jedoch auch andere Möglichkeiten, Wissen zu repräsentieren. Eine solche Möglichkeit ist Wissen in Form von real existierenden Fällen (design episodes). Diese können unterschiedlich verwendet werden:

- Durch direkte Übernahme ohne Veränderung (evtl. auch nur in Teilen),
- als Prototypen (siehe Kapitel 2.4.2),
- als Wissensquelle für allgemeine Regeln, die auch auf andere Designs angewendet werden können oder
- als Beispiele für Analogien.

Um einen solchen Fall zum Wissenserwerb nutzen zu können, ist es notwendig, die in ihm verborgenen Regeln zu erkennen und zu extrahieren. Dabei ist es keineswegs sicher, daß die so gewonnenen Informationen korrekt oder vollständig sind. Die so gewonnenen Generalisierungen sind wesentlich effizienter zu handhaben als eine große Menge von Designbeispielen. Dennoch ist zu beachten, daß

Bei genügend komplexen Designs stellt sich dabei das Problem der Größe des Suchraumes. Nur ein verschwindend geringer Anteil von Objekten in diesem Raum erfüllen jedoch die an sie gestellten Anforderungen. Was man braucht, sind also Designstrategien, die den Suchraum drastisch einschränken.

Für jede Designaufgabe kann eine Menge von primitiven Komponenten und Beziehungen zwischen Komponenten vorausgesetzt werden. Außerdem soll das Produkt verschiedene Funktionen zur Verfügung stellen und gewisse Bedingungen erfüllen. Funktionen sind Zustände, die das Produkt unter bestimmten Voraussetzungen erreichen oder vermeiden soll. Dabei können Funktionen sowohl zu der Designspezifikation gehören, als auch implizit in dem Domänenwissen enthalten sein. Außerdem müssen oft eine Anzahl Bedingungen (Constraints) erfüllt werden. Dabei lassen sich Funktionen und Constraints schwierig trennen; Funktionen sind Bedingungen über das Verhalten, Constraints sind Bedingungen über Eigenschaften des Produkts.

Ein möglicher Designprozeß ist es z. B., ein Design basierend auf den Funktionen zu generieren und dann solange zu verändern, bis die Constraints erfüllt werden. Beim Design eines Autos könnte man die Möglichkeit, vier Personen transportieren zu können als Funktion und die zulässige maximale Größe als Constraint ansehen. Man müßte dann solange versuchen, Autos für vier Personen zu generieren, bis man unterhalb der maximalen Größe bleibt.

Eine Designaufgabe (Design Task) wird definiert durch

1. eine Menge von Funktionen, die vom Produkt zu Verfügung gestellt werden müssen, sowie eine Menge von Constraints, die erfüllt werden müssen und
2. einer Technologie, d. h. einer Menge von verfügbaren Komponenten und einem Vokabular von Beziehungen zwischen Komponenten.

Die Lösung einer Designaufgabe besteht aus einer vollständigen Spezifikation einer Menge von Komponenten und deren Beziehungen untereinander, welche insgesamt ein Produkt beschreiben, das die Funktionen zu Verfügung stellt und die Constraints erfüllt. Dazu gehören auch implizite Annahmen, wie z. B., daß das Produkt nicht komplexer oder teurer ist als mögliche Alternativen.

Kann eine Designaufgabe nicht direkt gelöst werden, weil einzelne Komponenten nicht vorhanden sind, so kann dies als ein Subtask zum Design dieser Komponenten betrachtet werden. Design ist also häufig ein rekursiver Prozeß.

Eine Methode zur Lösung der Designaufgabe ist eine Suche im Problemraum[New80].

Eine andere Methode besteht darin, die Lösung direkt zu berechnen. Dies funktioniert jedoch nur bei sogenannten wohlstrukturierten Problemen (z. B. Statikberechnungen). Die meisten tatsächlichen Designaufgaben fallen nicht in diesen Bereich.

PROPOSE-CRITIQUE-MODIFY

Die bekanntesten Methoden zur Lösung von Designaufgaben lassen sich als Propose–Critique–Modify–Methoden (PCM) zusammenfassen. Diese Methoden zerfallen in vier Subtasks:

1. Generiere eine teilweise oder komplette Lösung (propose).
2. Überprüfe die Lösung auf Übereinstimmung mit der Spezifikation. Falls die Spezifikation erfüllt wird, ist die Designaufgabe (oder -unteraufgabe) gelöst.
3. Falls die Spezifikation nicht erfüllt wird, muß die Ursache für den Fehler gefunden werden (critique).
4. Modifiziere die Lösung entsprechend den Fehlerursachen.

Der Ablauf der einzelnen Phasen geschieht dabei nach Abbildung 2.2.

Dieses Verfahren kann unter Umständen auch nur auf Teile der Designaufgabe angewand werden. Die Lösung diese Teilaufgaben liefert dann weitere Constraints, die dann die Gesamtlösung ermöglichen.

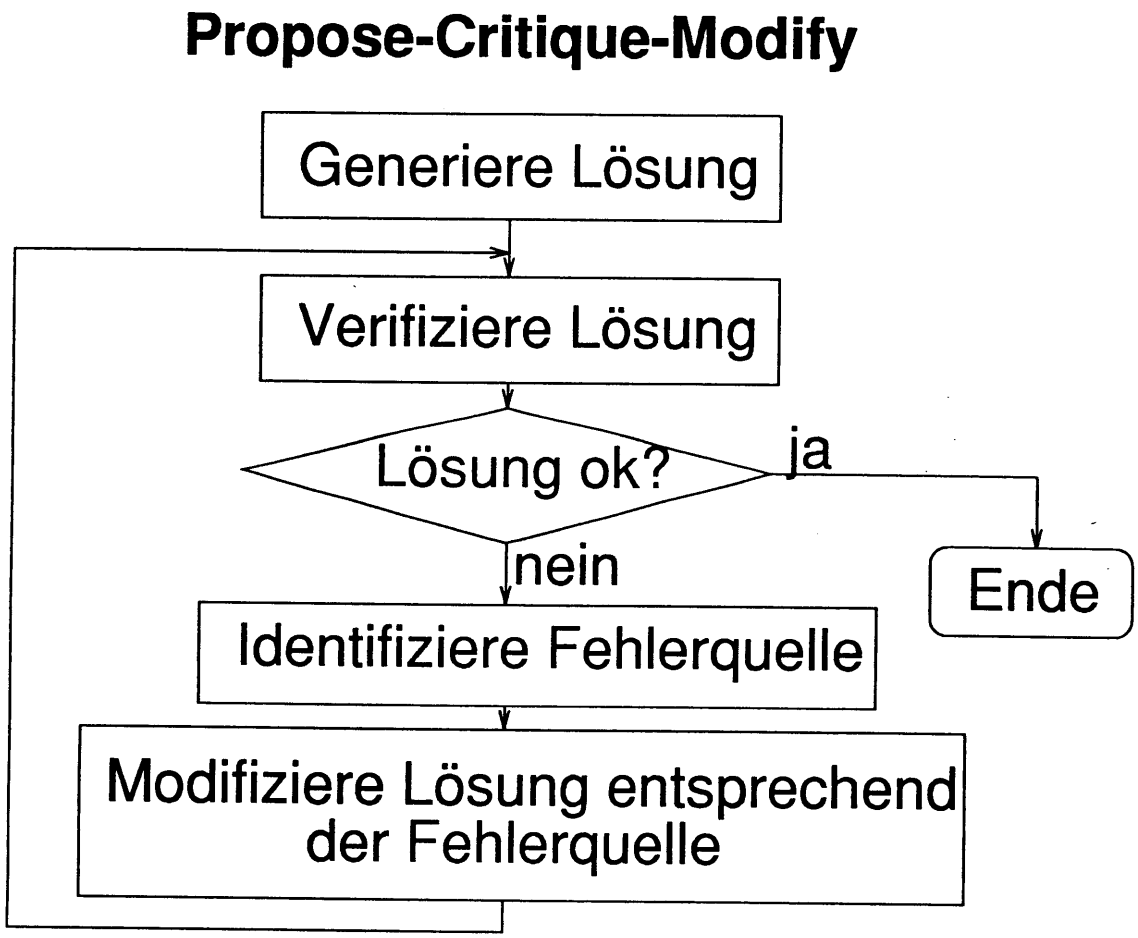


ABBILDUNG 2.2. Ablauf beim PCM-Modell

Zu 1.: Methoden zur Lösungsgenerierung

Im wesentlichen kann man die Methoden in drei Gruppen einteilen:

- problem-decomposition solution-composition.

  Die Designaufgabe muß sich in kleinere Unteraufgaben (Subtasks) zerlegen lassen. Falls es mehrere Zerlegungsmöglichkeiten gibt, muß eine Auswahl getroffen werden und die Möglichkeit des Backtrackings zur Verfügung stehen. In vertrauten Domänen sind die Zerlegungen jedoch schon bekannt, so daß nur wenig Zeit auf die Suche nach der besten Zerlegung verwendet werden muß. So hat sich z. B. im Automobilbau die Art der Zerlegung in den letzten 70 Jahren kaum geändert.

  Die zwei wichtigen Ziele dieses Verfahrens sind, die Spezifikation für die Unteraufgaben zu generieren und die erhaltenen Teillösungen zu einer Lösung der Gesamtaufgabe zusammenzusetzen. In vielen Domänen ändern sich Constraints in Abhängigkeit von Teillösungen. In einem solchen Fall ist es nicht immer möglich, das Gesamtproblem sofort in Einzelprobleme zu zerlegen. Schlimmstenfalls kann die Suche nach Parametern, die alle Teilprobleme lösbar machen, den Hauptteil der Zeit beanspruchen.

- Suche nach ähnlichen Fällen

  Bereits gelöste Designaufgaben (design cases) stellen eine Hauptwissensquelle für die Lösung von Designproblemen zur Verfügung. Solche Fälle können episodisch (siehe Kapitel 2.4.4) oder aber bereits abstrahierte Lösungen enthalten.

  Der Hauptaspekt des fallbasierten Designs ist die Vergleichsfunktion (matching). Einige Constraints sind offensichtlich wichtiger als andere. Daher wird man eher einen Fall auswählen, dessen wichtigen Constraints denen der Designaufgabe ähneln und wird bei weniger wichtigen Constraints größere Abweichungen zulassen. Es ergibt sich also die Notwendigkeit, Prioritäten festzulegen.

- Direktes Erfüllen der Constraints

  Bestimmte Klassen von Designaufgaben lassen sich durch Optimierung oder durch das Lösen von Gleichungen lösen. Diese haben gemeinsam, daß der Lösungsraum durch entsprechende Algorithmen direkt bestimmt werden kann.

Zu 2.: Verifikation

Es gibt im wesentlichen zwei Verifikationsverfahren. Das erste berechnet mit Hilfe domänenspezifischer Algorithmen die in Frage kommenden Attribute (z. B. Kosten, Gewicht, ...) und vergleicht diese mit der Spezifikation. Das zweite Verfahren ist die Simulation des Produkts. Diese Simulation erhält als Eingabe die Beschreibung des Produktes (die Lösung des Designproblems) und liefert als Ausgabe die interessierenden Attributwerte.

Zu 3.: Beurteilen der Fehlerquellen

Diese Beurteilung (Critiquing) ist eine allgemeine Variante des *diagnostic problem*. Dies ist ein

- Es wird eine Änderung vorgenommen, der Erfolg bewertet und dann die Änderung so korrigiert, daß ein maximaler Anstieg der Bewertung erreicht wird (hill climbing).
- Liegen auch Informationen über die Abhängigkeit von Bedingung und Ergebnis im Designvorschlag vor, kann durch Backtracking einfach eine andere Möglichkeit ausgewählt werden.

2.5.2 Ein KADS-Modell

Kruger und Wielinger verwenden in ihrem Artikel [KW93] ein KADS-Modell für das industrielle Design. Der Vorteil bei der Verwendung eines solchen Modells liegt darin, daß die Spezifikation systematisch und formal ist und daher das Modell mit anderen Problemlöseverfahren einfacher verglichen werden kann.

Ähnlich wie bei [Cha90] wird ein Designproblem mit Hilfe von funktionalen Anforderungen, nichtfunktionalen Anforderungen und Constraints beschrieben. Eine Lösung ist eine Menge von Komponenten und ihrer Beziehungen, die die Anforderungen und Constraints erfüllen. Die funktionalen Anforderungen beschreiben dabei die Funktionen, die das Produkt zur Verfügung stellen soll; die nichtfunktionalen Anforderungen stellen Anforderungen dar, die nicht direkt etwas mit der Funktion zu tun haben (z. B. Einfachheit oder Ästhetik) und die Constraints beziehen sich auf die Attribute des Produkts, wie z. B. Gewicht, Qualität, etc.

Der Designprozeß teilt sich in zwei Schritte auf:

- Analyse der Anforderungen; dabei kann es sich sowohl um eine Gesamtanalyse als auch um eine Anzahl von Teilanalysen handeln.
- Die Synthese von Lösungen: bei Teillösungen müssen diese dann [illegible]

mengesetzt werden.

Der von Chandrasekaran beschriebene PCM-Prozeß ist in diesem Fall nicht praktikabel. Auf der obersten Ebene des Designprozesses kommt es selten vor, daß ein komplettes Design gebildet, dieses bewertet und dann entsprechend geändert wird. Auf tieferen Ebenen mag diese Verfahren besser anwendbar sein. Im Gegensatz zum PCM–Ansatz von Chandrasekaran gehen Designer jedoch im allgemeinen so vor, daß sie Teillösungen, die nicht den Spezifikationen entsprechen, oft komplett verwerfen und eine völlig neue Teillösung versuchen.

Kruger/Wielinger kommen in ihrem Artikel aufgrund von Experimenten (siehe [KW93]) auf folgende Schlußfolgerungen:

Bei iterativem Design liegen noch nicht alle Constraints und Anforderungen vor Beginn des Designprozesses vor. Auch sucht der Designer nach Teillösungen, die am Ende zur Gesamtlösung zusammengesetzt werden. Die Experimente zeigen aber auch, daß ein menschlicher Designprozeß nicht so wohlstrukturiert ist, wie man das erwartet hatte. Ein Designer wendet für verschiedene Probleme unter Umständen völlig verschiedene Designmethoden an.

2.6 Zusammenfassung

Dieser Abschnitt diente dazu, einen konzeptuellen Rahmen für einige Arten von Design zur Verfügung

der Aktionen als Wissen (knowledge) interpretiert. Außerdem wurde auf die Möglichkeit der Wissenserhebung während des Designprozesses eingegangen.

Zum Abschluß wurden zwei weitere Designmodelle vorgestellt, ein Task-orientieres Modell nach Chandrasekaran und ein KADS-Modell nach Kruger und Wielinga.

Insgesamt läßt sich sagen, daß sich a priori kein Designverfahren vorziehen läßt. Die Vielfalt der Anwendungsgebiete für Design liefert notwendigerweise auch eine Vielzahl von Lösungsverfahren. Es hat sich gezeigt, daß die PCM–Methode nach Chandrasekaran sich nicht für das Designproblem von Kruger und Wielinga anwenden läßt. Insbesondere ist es schwierig ein formales und wohlstrukturiertes Modell fuer ein Problem zu erstellen, welches im allgemeinen — nämlich vom Designer — unstrukturiert und eher intuitiv gelöst wird. Welcher Ansatz für ein spezielles Problem geeignet ist, wird man wohl von Fall zu Fall unterscheiden müssen.

2.7 Literaturverzeichnis

[Cha90] B. Chandrasekaran. Design problem solving: A task analysis. *AI Magazine*, pages 59–71, Winter '90 1990.

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge-Based Design Systems*. Addison–Wesley Publishing Company, 1990.

[KW93] C. Kruger and B. Wielinga. A KADS model for the industrial design task. In Christiane Löckenhoff, Dieter Fensel, and Rudi Studer, editors, *CommonKADS, 3rd KADS Meeting*, pages 131–144, Siemens AG Munich, March 1993. GI special interest group 1.5.1 Knowledge Engeneering.

[New80] A. Newell. Reasoning, problem solving, and decision process. *Attention and Performance*, Vol. 8:693–718, 1980.

3

Die Interpretation von Design

Dirk Krechel

ZUSAMMENFASSUNG Dieser Beitrag beruht auf dem entsprechenden Kapitel aus dem Buch von Coyne und Rosenman *Knowledge-Based Design Systems*. Unter Interpretation versteht man den Prozeß einer Darstellung eines Designs seine Bedeutung zuzuweisen. Eine mächtige Methode zur Interpretation ist die logische Deduktion. Mit ihrer Hilfe können einfache Expertensysteme konstruiert werden. Bei der Interpretation leitet man Eigenschaften eines Designs ab, die nicht in der Designbeschreibung enthalten sind . Von besonderem Intresse sind dabei die Eigenschaften die eine Bewertung der Leistung des Designs ermöglichen. Auch werden Vergleiche der abgeleiteten Attribute mit erwarteten Attributen durchgeführt (machting). Durch Iteration solcher Vergleiche kann man sich an Ziele heranarbeiten. Interpretation spielt auch eine wichtige Rolle bei der Auswertung von Design. Bei der Auswertung werden Leistungen zweier Designs oder ein Design mit festen Vorgaben verglichen. Solche Auswertungen liefern Aussagen wie „das Gewicht ist zu groß" oder „ die Kosten sind zu hoch". Auch die Interpretation von geometrischen und topologischen Beschreibungen wird eine Rolle in diesem Seminar spielen. Ein weiteres Feld wird die Nutzung von Designkodes als Quelle von Wissen sein.

3.1 Die Rolle der Interpretation

Um zu verstehen was Interpretation bedeutet hilft es, ihre Rolle in der Sprache , der Logik und natürlich im Design zu betrachten. In der Sprache versuchen wir festzustellen was eine Menge von Sätzen aussagen will. In der Logik versucht man die Wahrheit von Theoremen und Aussagen festzulegen. Im Design werden Aussagen aus der Designbeschreibung abgeleitet.

3.1.1 Interpretation von Sprache

Mit der Interpretation von Sprache dürfte jeder wohl in seiner Schulzeit in Berührung gekommen sein. Man denke nur an die Suche nach der Bedeutung von modernen Gedichten oder an Erzählungen von Kafka. [CRR+90] bezieht sich auf eine Definition von Ogden und Richards in „Meaning of Meaning" die besonders nützlich im Zusammenhang mit automatischen Systemen und Design ist. Wörter und Wortgruppen bilden Symbole. Wenn sie entsprechend genutzt werden, sind sie dann Zeichen für eine Sache, z.B. eine Idee oder Objekt. Ein Zuhörer oder Leser interpretiert, wenn er neuen Zeichen eine Referenz zuweist. Wörter haben eine inhaltliche wie auch eine symbolische Funktion, aber es gibt keine feste Verbindung zwischen Wörtern und den Dingen, die sie bezeichnen. Faktoren die die Beziehung zwischen Sprecher und Hörer beeinflussen, bilden die sign situation. Das sind Erfahrungen die psychologische Reaktionen auf Zeichen festlegen, die wir in unserer Terminologie mit Wissen bezeichnen, das Mapping zwischen Zeichen, Ideen und Objekten erleichtert.

3.1.2 Interpretation von Logik

Interpretationen in der Logik dürfte jeder Informatikstudent nach dem Vordiplom zu genüge kennen. Es gibt zwei Wege wie man Interpretationen in der Logik betrachten kann. Erstens werden Symbolen der Prädikatenlogik, wie in der natürlichen Sprache, Ideen oder Objekte zugewiesen. In Bezug auf Kowalski ist die Bedeutung, die mit einer logischen Aussage oder Prädikat assoziert werden kann, relativ zu der Menge von Aussagen die einem axiomatisches System zugrunde liegen. Kowalski geht davon aus, daß jede Aussage sich mit Implikationen ausdrücken läßt. Die Interpretation einer Aussagemenge ist also nicht explizit festgelegt, sondern ist alles was aus dieser Menge abgeleitet werden kann. Deduktive Inferenz liefert also eine Methode für die Ableitung. Bei der Interpretation von Design werden also neue Aussagen (z.B. über die Leistung des Designs) aus einer gegebenen Menge von Aussagen (die in der Designbeschreibung enthalten sind) abgeleitet.

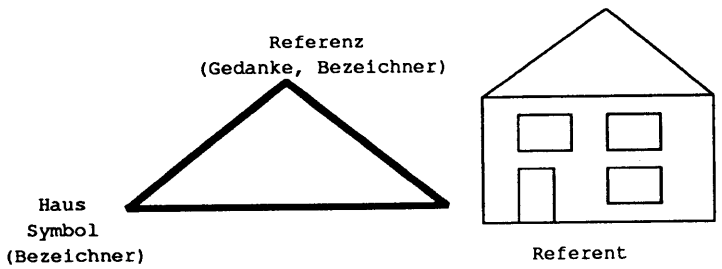


ABBILDUNG 3.1. Das „semiotic triangle“ aus Odgen and Richards (1923)

3.1.3 Interpretation von Design

Ein Designentwurf kann also durch eine Menge von logischen Aussagen beschrieben werden Um neue Aussagen über die Leistungsfähigkeit des Designs abzuleiten reicht die Beschreibung des Entwurfs nicht. Sie muß mit Wissen kombiniert werden. Die Interpretation eines Designs ist ähnlich schwierig wie die eines gesprochenen Satzes. Der Satz „Kaiserslautern ist eine Weltstadt“ kann nur im Zusammenhang mit der Betonung interpretiert werden. Obwohl es nur wenige Menschen gibt, die ihn ohne Ironie aussprechen können. Eine Formalisierung kann die folgende Form haben:

$I = \tau(Ki, D)$
τ: *Funktion*
Ki: Interpretatives Wissen
D: Designbeschreibung
I: Interpretation

3.2 Teilaspekte der Design Interpretation

In diesem Kapitel werden die Syntax und die Semantik von Design, die Rolle von Parsing, Auswirkungen von lückenhaftem Wissen und Mehrdeutigkeiten betrachtet.

3.2.1 Syntax und Sematik im Design

Eine Designbeschreibung ist normalerweise Syntax während die Interpretation einer Designbeschreibung semantischen Charakter hat. Syntax bezieht sich also auf die Gestalt, z.B. Punkte, Linien eines Elements oder auf explizite Aussagen über Eigenschaften des Elements. Semantik steht dagegen für abgeleitete Beschreibungen oder Leistungsangaben. In diesem Model sind Eigenschaften eines Designs nicht fest syntaktisch oder semantisch. Figure 3.2 zeigt, daß es verschiedene syntaktische und semantische Ebenen gibt. Aus Material und Abmessungen eines Messers kann man zum Beispiel das Gewicht ableiten. Dieses hat in diesem Augenblick semantischen Charakter. Für die Ableitung der Schärfe des Messers hat das Gewicht jedoch eine syntaktische Funktion.

Was auf der einen Ebene semantisch ist, kann auf der nächsten syntaktischen Charakter haben. Design kann charakterisiert werden als die Suche eines Mappings zwischen dem Raum der Interpreationen (Semantikraum) und dem Raum der Designbeschreibungen (Sytaxraum). In einem

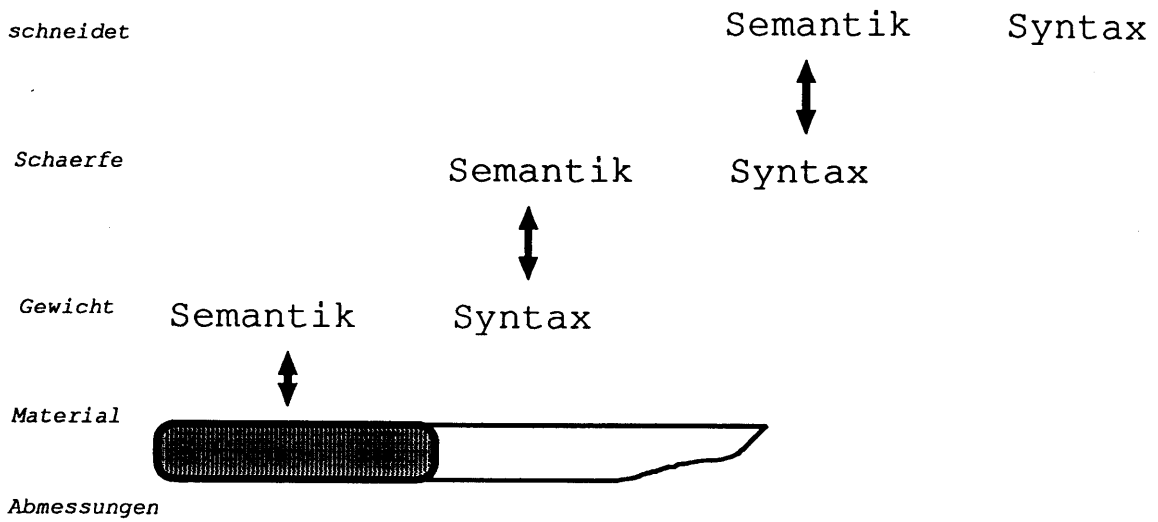


ABBILDUNG 3.2. Die Interpretationsebenen

vorausgesetzt wird, daß man Interpretation von dem Wissen über den Aufbau des Designs, z.B. die Grammatik oder die durchgeführten Designaktionen, trennen kann, ist es doch oft nützlich bei der Interpretation eines Designs zu wissen welche Schritte bei der Erzeugung des Designs durchgeführt wurden.

3.2.3 Der Einfluss von unvollständigem Wissen auf die Designinterpretation

Oft entstehen bei der Interpretation von Sprache Schwierigkeiten wenn die zugrundeliegende Aussa-

[redacted]

Interpretation benötigt, sondern kann oft nur im Kontext eines Textes verstanden werden. Auch bei der Interpretation von Design spielen solche Kontextinformationen eine wichtige Rolle. So sind die Temperaturverhältnisse in einem Getriebe, von dessen Plazierung abhängig. Solche Informationen

abgeleitet werden können. Dies legt den Gedanken nahe, daß es Interpretationshierarchien gibt. Zahlen können in einer Datenstruktur als Punkte oder Linien interpretiert werden, diese dann wieder zu einfachen Formen, die Formen zu Designelementen. Aus einer Menge von Designelelmenten können wiederum Objekte gefolgert werden. Bei wissensbasierten Systemen ist man an einer expliziten Ausarbeitung der Hierarchien bei der Organisation des Designs und des Wissens interesiert. Eine intuitive Methode Interpretationswissen darzustellen ist die von Regeln der Form:

if a1 and a2 and ... and an then b1 and b2 ... and bm

Solche if-regeln können leicht in Hornklauseln umgewandelt werden.

3.3 Design Auswertungen

Ein Design besitzt nicht nur die Eigenschaften, die explizit in seiner Beschreibung aufgeführt sind. So wird z.B. eine Brücke durch ihre Art , Hänge-, Bogen oder Auslegerbrücke, ihr Material, die Anzahl der Pfeiler und ihre Geometrie beschrieben. Sie besitzt aber auch andere Eigenschaften, wie Stabilität, Verkehrskapazität, Umweltverträglichkeit und ästhetische Qualitäten. Diese Eigenschaften sind impicit in der Designbeschreibung enthalten und können aus ihr abgeleitet werden. Aus einer gegebenen Designbeschreibung können solche Leistungsmerkmale abgeleitet werden. Eine Auswertung ist eine Menge von Leistungsvergleichen. Eine Möglichkeit ist zum Beispiel der Vergleich des Wertes eines Attributs bei verschieden Designs, z.B. die Kosten. Eine Auswertung wäre dann zum Beispiel der Satz:

Design 1 ist billiger als Design 2

Eine andere Art wäre der Vergleich mit einem Standard- oder Normwert. Eine Auswertung diesen Typs wäre:

Design 1 ist zu teuer.

Man kann Designsysteme erstellen bei denen es das Ziel ist eine Idealleistung zu erreichen. Es werden Minima oder Maxima angegeben und das System versucht sich diesen Werten anzunähern. Auswertungen liefern oft Aussagen wie die folgenden:

Die Leistung ist zufriedenstellend (oder ungenügend)
Dies ist die bestmöglichste Leistung für dieses Problem

In einem solche System wird zuerst ein Kandidatendesign erstellt und seine Leistungsmerkmale abgeleitet, indem man das Verhalten mit Hilfe von symbolischen und analogen Modellen simuliert. Ist der Designer mit den Ergebnissen der Auswertung gegen ein bekanntes Design oder in Bezug auf andere Kriterien zufrieden, dann wird das Kandidatendesign zum neuen Enddesign, ansonsten wird es verworfen. Durch mehrfache Iteration dieses Prozesses kann oft ein gutes Ergebnis erzielt werden, speziell wenn aus den einzelnen Auswertungen der Kandidatendesigns Vortschritte erkennbar sind. Es ist üblich Auswertungswissen als algorithmische Funktionen darzustellen. Oft ist es aber auch möglich es intuitiv in Form von Regeln anzugeben. Z.B.

if cost ist greater than 10.000 DM then evaluation of cost is „zu teuer“

Für iterative Designsystem sind auch Auswertungskriterien wie das folgende nützlich:

if performance of element under consideration is better than performance of previos element then prefer element under consideration

Die Interpretation von „better than“ hängt von spezifizierten Leistungen ab. Die algemeine Regel muß also durch spezielle Regeln ergänzt werden die festlegen wann die performance eines Elements „better than“ die eines andern ist. Z.B.

if performance is cost and cost of element1 is less than cost of element2 then performance of element1 is better than performance of element2

Auswertungen sind also hilfreich bei der Suche nach einem Design. Wiederholte Simulationen können dabei helfen, ein ideales oder wenigstens zufriedenstellendes Design zu finden.

3.4 Die Interpretation von räumlichen Designbeschreibungen

Designer sind oft an den räumlichen Qualitäten eines Designs interessiert. Bevorzugte Darstellungsart für solche räumlichen Eigenschaften sind Bilder. In CAD-Systemen werden solche Eigenschaften durch Zahlen und Texte beschrieben. Leider ist man noch nicht soweit, echtes automatisches „graphic reasoning", ohne Zwischenschritt über Zahlen und Texte, durchzuführen. Räumliche Eigenschaften die durch Zahlen und Text beschrieben werden, müssen zu Bildern interpretiert werden. In diesem Kapitel wird zuerst die Interpretation von Graphikbeschreibungen betrachtet. Danach die Ableitung von räumlichen Eigenschaften aus der geometrischen Beschreibung und wie dies in einem regelbasierten System gespeichert werden kann.

3.4.1 Die graphische Interpretation geometrischer Beschreibungen

Bilder können auch durch deduktives Schließen gemalt werden. Graphische Grundelemente können, ähnlich wie durch Graphikkommandes in prozeduralen Sprachen, auch durch Regeln dargestellt werden. Graphische Funktionen können durch Prädikate ersetzt werden die wahr liefern, wenn ihre Operationen ausgeführt werden können. Move und Draw Funktionen können in Regeln eingetragen werden, die Objekte zeichnen. Z.B. eine Regel zum Zeichnen eines Rechtecks.

$$DrawRectangle(x0,y0,w,h) \Leftarrow Move(x0,y0)\ and\ Draw(x0,y0+h)\ and\ Draw(x0+w,y0+h)\ and\ Draw(x0+w,y0)\ and\ Draw(x0,y0)$$

Der Beweis dieser Regel führt zum Zeichnen eines Rechtecks. Mit dem Formalismus der Logik wird durch die Einführung von Funktionsaufrufen ein Kompromis geschlossen, weil viel von der Resolutionsmethode des Beweisers abhängt. Pereira (1982) und Swinson (1983) haben Versuche unternommen, Graphik in PROLOG einzubinden.

3.4.2 Ableitung räumlicher Eigenschaften aus geometrischen Beschreibungen

Geometrische Informationen können im Prädikatenkalkül repräsentiert werden. Die Prädikatenlogik ist ein einfacher Weg, die Idee der relationalen Datenbanken darzustellen. In CAD-Systemen werden diese immer häufiger angewendet. Geometrische Informationen z.B. über einen Raum können wie folgt dargestellt werden:

$$Room(office(1),3.0,1.0,4.0,2.0)$$

Diese Aussage ist Fakt in einer CAD-Datenbank. Die fünf Argumente des room-Prädikates sind Name des Raums, und die Werte der Nord-,Süd-,Ost- und Westwand.

Komplexere Objekte können aus Mengen solcher Regeln gebildet werden. Wir sind daran interesiert Aussagen zu bilden, aus denen sich nicht nur graphische Interpretationen herleiten lassen. Z.B. läßt sich aus Aussagen wie der folgenden topologische Beziehungen schließen:

$$Direction(a,North,b) \Leftarrow Room(a,na,sa,ea,wa)\ and\ Room(b,nb,sb,eb,wb)\ and\ (sa>nb)$$

Wir setzen in dieser Regel fest, daß Raum a nördlich von Raum b ist, wenn die Südseite von a über der Nordseite von b ist. Ähnliche Regeln können für andere Richtungsbeziehungen definiert werden. Zum Beweisen des folgenden Theorems

$$\exists x[Direction(x,North,Office(1))]$$

muß das System in der Lage sein, alle Räume nördlich von Office(1) zu folgern, daß sind alle Räume mit denen x instanziert werden kann, damit das Theorem wahr wird. Ähnlich können auch noch weitere implicite Eigenschaften nachgewiesen werden. Es gibt aber noch höhere Ebenen von Interpretationen. Man kann auch komplexere logische Aussagen formulieren die es ermöglichen

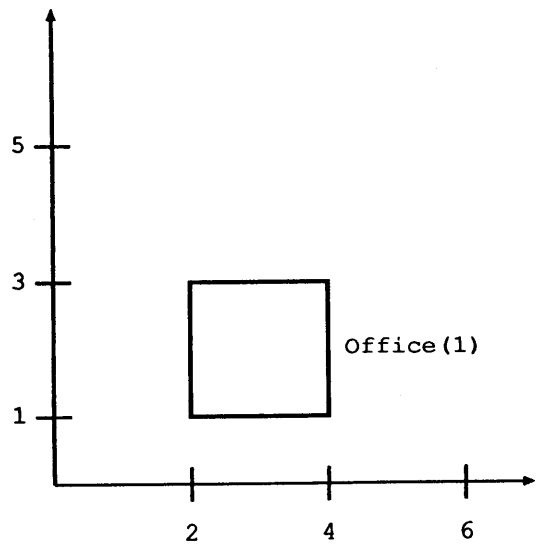


ABBILDUNG 3.3. Die Geometrie eines einfachen Raumes

b): Path(a,b,c,d) $\Longleftarrow$ *(Link(a,e) or Link(e,a)) and*
Not(Member(e,c)) and Path(e,b,e and c,d)

Das erste Argument ist der Startpunkt, das zweite das Ziel, das dritte die schon besuchten Räume und das vierte die Räume die den Pfad bilden. Der Definition liegt der Gedanke zugrunde, daß es einen Pfad von a nach b gibt, wenn es einen zu a benachbarten Raum gibt von dem aus ein Pfad nach b existiert. Diese Regel könnte Teil der Wissensbasis eines Expertensystems sein, welchem man den Auftrag erteilen könnte, einen Beweis zu führen, daß es einen Pfad zwischen zwei Räumen gibt. Das Ergebnis wäre der Pfad als Return-Wert des vierte Arguments.

$$\exists x[Path(a, b, Nil, x)]$$

Durch deduktive Inferenz können ähnlich noch weitere Eigenschaften von Designtopologien produziert werden.

3.5 Wissensbasierte Interpretationsysteme

In diesem Abschnitt werden wir uns die Realisation einiger vorher beschriebenen Ideen als Teil von Gesammtdesignsystemem anschauen.

3.5.1 Ein System zur Interpretation von Graphikbeschreibungen

Die Fakten in einer Graphikdatenbank erlauben uns zwar Aussagen über die physische Form eines Designs, unsere Systeme sollen aber in der Lage sein, Zeichnungen ähnlich wie ein Designer zu interpretieren. In dem nun folgenden Beispiel betrachten wir die Nutzung von Regeln zur Wissensdarstellung bei einfachen Punktverbindungen und formulieren ein System, welches Schlüße über die räumlichen Merkmale des repräsentierten Objekts zuläßt. Das Beipiel ist eine vereinfachte Version des in [CRR+90] angegebenen Beispiels Topology 1 (Akiner 1985;Gero,Akiner and Redford 1983). Der Zweck des Systems ist es den Designer bei der Interpretation einer Designkonfiguration zu unterstützen. Man kann die Prädikatenlogik 1.Stufe benutzen um Wissen über räumliche Zusammenhänge zu kodieren. Punkte bekommen ein Label und x,y,z als kartesische Koordinatenwerte.

$$\text{Coord}(V_1[\text{x,y,z}])$$

Scheitelpunkte werden mit Objekten verbunden durch Fakten. Z.B.:

$$\text{vertex}(\text{Objekt}, V_1)$$

Wissen wird als Aussagemenge über die Beziehungen einzelner Einheiten gespeichert. Dieses Wissen kann wie folgt klassifiziert werden:

1. Wissen über Einzelobjekt-Topologien, also Wissen über die Beziehungen zwischen z.B. Scheitelpunkten, Kanten und Flächen. Z.B.:
   If the two vertices belong to an Objekt and the two vertices are not identical and two of their respective coordinates match then an edge exists between two vertices of an Objekt.

2. Wissen über Einzelobjekt-Geometrie, also Wissen welches mit Attributen wie Länge, Breite, Fläche und Volumen zu tun hat. Z.B.:
   If an edge exists between two vertices of an objekt
   then the length of an edge is the distance between two vertices.

3. Wissen über Eigenschaften von Einzelobjekten, also Wissen das mit Eigenschaften wie geometrischer Typ, Farbe, Beschriftung, Preis oder Richtung verbunden ist. Z.B.:
   An objekt is a cube if ist length, with, and height are all equal.

4. Wissen über topologische Beziehungwen zwischen Objekten, z.B benachbart, über oder in. Diese Beziehungen werden abhängig von ihrer Herleitung hierarchich organisiert. Z.B.:
   If the common coordinate of the bottom face of a is greater than the common coordinate of the top face of b then a is above b.

5. Wissen über geometrische Beziehungen zwischen Objekten, z.B. größer als, länger als. Z.B.:
   If a is greater then b in terms of volume then a is bigger than b.

6. Wissen über Eigenschaften von Objektgruppen, das ist Wissen über Attributvergleiche wie teurer als, billiger als. Z.B.:
   If a is greater than b in terms of price then a is more expensive than b

7. Wissen über Verbindungsbeziehungen, also Wissen über Beziehungen wie verbunden mit, es gibt eine Pfad nach. Z.B.:
   If space a has a doorway to space b or space b has a dorway to space a then space a has access to space b.

Aus diesen Kategorion sieht man, daß man aus einer geometrischen Datenbank eine Menge Eigenschaften folgern kann. Diese Folgerungen bauen aufeinander auf. Das Volumen kann zum Beispiel aus Fläche und Höhe abgeleitet werden. Die Fläche wiederum direkt aus den Eckpunkten in dem Entwurf.
Ein Expertensystem mit einer so aufgebauten Regelbasis ist nun in der Lage bestimmte Fragen über die Topologie eines Designs, Verbindungen oder die Anzahl bestimmter Objekte zu beantworten. Das folgende Beispiel 3.4 ist eine Vereinfachung von Frank Lloyd Wright's Carsons House in [CRR+90].

Nun eine Beispielsitzung mit dem Expertensystem:

Was ist über Wohnzimmer ?
Schlafzimmer ist über Wohnzimmer im Haus.

Wieviele Räume hat das Hause?
Es sind 6 Räume im Hause.

Was sind die Pfade vom Wohnzimmer zum Bad ?
Pfade vom Wohnzimmer zum Bad sind
[Wohnzimmer, Flur(1), Flur(2), Bad]
[Wohnzimmer, Flur(1), Flur(2), Schlafzimmer, Bad]

Ist Schlafzimmer größer als Wohnzimmer?
Ja.

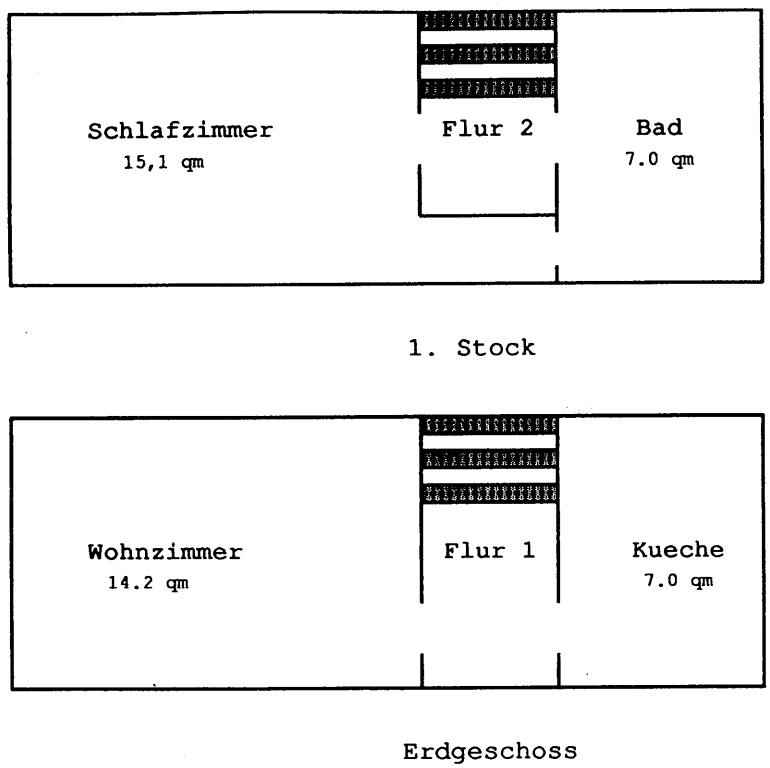


ABBILDUNG 3.4. Entwurf eines Hauses

3.5.2 Auf Erfahrungen basierendes Wissen

Bis jetzt haben wir uns nur mit eindeutig festgelegtem Wissen über räumliche Eigenschaften von Design befasst. Dieselben Repräsentationsformen und Schlußregeln können aber auch auf Heuristiken und ideosynkratische Wissensbasen angewendet werden. So etwas bezeichnet man mit Erfahrungswissen (experiential knowledge). Man benutzt also sogenannte Faustregeln. Nehmen wir als Beispiel den Entwurf von Küchen. Man kann ein System erstellen bei dem der Benutzer interactive am Bildschirm seine Küche entwirft und diese dann interpretiert und analysiert wird.

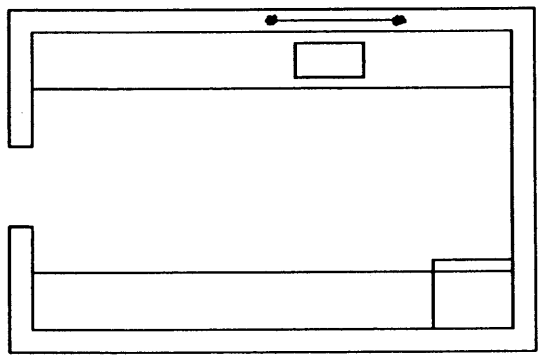


ABBILDUNG 3.5. Graphische Darstellung einer Küche

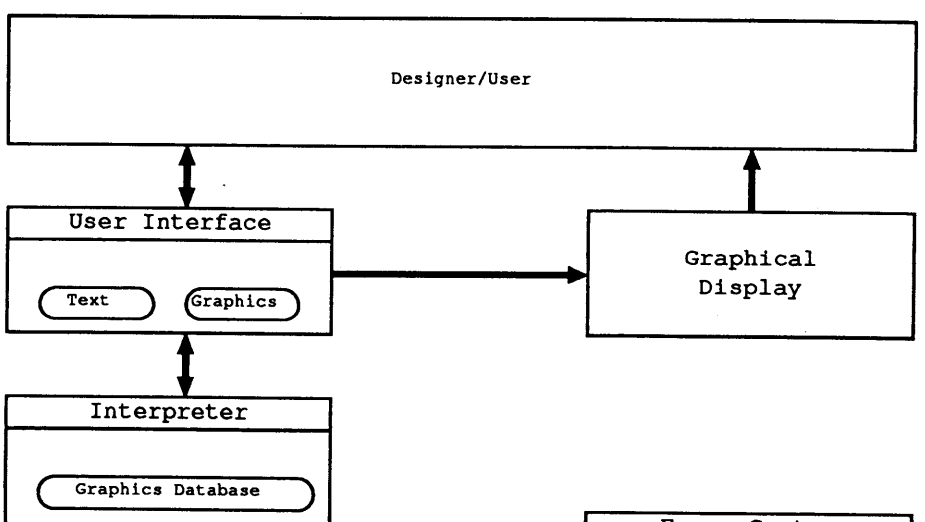
Dieser Entwurf wird dann wie in den vorhergehenden Kapiteln beschrieben, in Regeln einer Wissensbasis interpretiert. Zusammen mit Regeln die auf Erfahrungswerten, wie sie zum Beispiel aus Prototypen oder Designstandards enthalten sind, basieren, kann das System feststellen, ob der Entwurf vorteilhaft ist oder nicht. Beispiel für eine Erfahrungsregel in diesem Zusammenhang:

If es ein Fenster gibt and die Raumart Korridor ist and das Fenster an der langen Wand ist and und die Theke ist an der selben Wand wie das Fenster then Thekenlage wird akzeptiert.

Eine Beispielsitzung eines solche Systems (PREDIKT Oxman, Gero 1987) ist auf Abbildung 5.13 in [CRR+90] dargestellt.

3.5.3 Interpretation graphischer Beschreibungen von Ingenieurs-Objekten

Strukturen (Balachandran 1988). Es ist in der Lage bestimmte Merkmale von Objekten zu identifizieren und sie nach diesen in Klassen einzuteilen. Es kann Informationen die auf dem Bildschirm graphisch erstellt wurden, in ein deskriptives Model des Designs in Form einer Frame-Darstellung umwandeln. Figure 3.6 zeigt die Architektur eines solchen Systems:



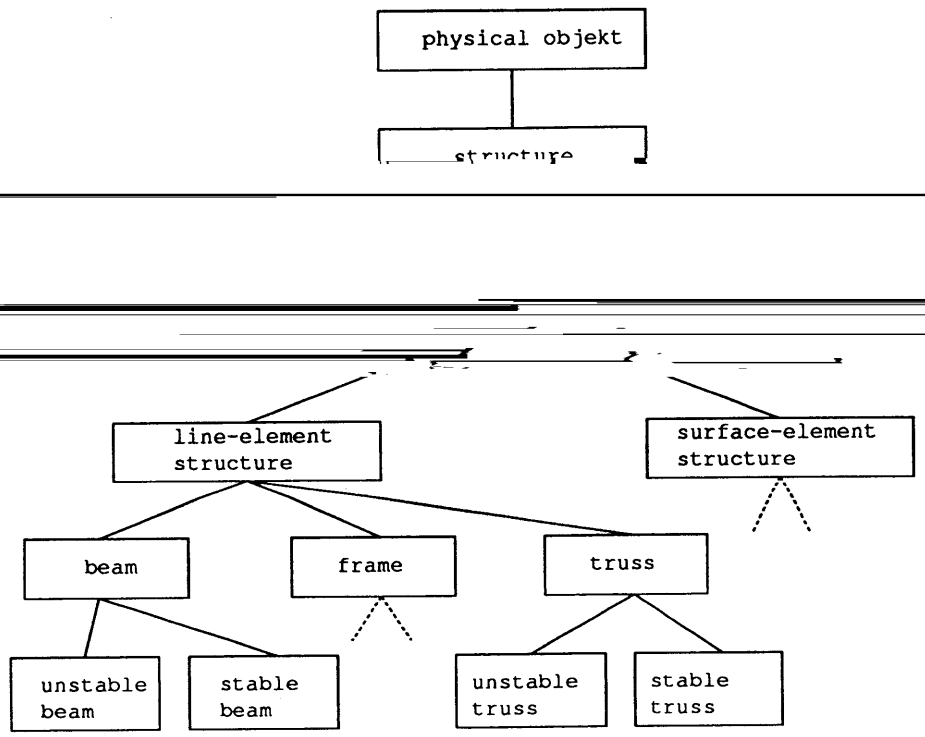


ABBILDUNG 3.7. Eine Objekthierarchie

name	:**Truss**		
slots	:**SuperClasses**	: value	line_element_structure
	:**SubClasses**	: value	: [stable_truss unstable_truss]
	:**MembershipCriteria**	: value	: [numberOfMembers numberOfJoints numberOfForces numberOfSupports locationOfSupports]
	:**NumberOfJoints**	: valueClass	: [intersection integers greater_than (2)]
	:**NumberOfMembers**	: valueClass	: [intersection integers greater_than(2)]
	:**NumberOfSupports**	: valueClass	: [intersection intergers greater_than (1)]
	:**LocationOfSupports**	: value	: joint_locations
	:**Span**	: if_needed	: truss_span
	:**Triangulation**	: if_needed	: truss_triangulation
	:**NumberOfRedundants**	: if_needed	: truss_redundants
	:**Stability**	: if_needed	: truss_stability

Eine Klassendefinition enthält bestimmte Teile. Einen Klassennamen, Metaklassen , Unterklassen, Attribute und Werte. Auch prozedurales Wissen wird benutzt, um zum Beispiel graphische Eigen-

nisse kann er einen Entwurf verwerfen, annehmen oder weiter verbessern. Die Wissensbasis enthält qualitative Regeln über den physikalichen Hintergrund der Wasserbewegungen. Die genaue Physik, wie sich das Wasser in den Zwischenräumen bewegt und wie turbulente Luftbewegungen diese beeinflussen, ist nicht sehr einfach, deshalb verwendet das System lieber einfachere Faustregeln. Beispiele für solche einfache Regeln mit denen festgelegt werden kann ob bestimmte Objekte naß sind.

> *If ein Teil naß ist then sind alle seine Ecken und Seiten naß.*
> *If ein Winkel an einem anderen Winkel am Rand anliegt and der höhere Punkt naß ist then ist der untere Winkel auch naß.*

Diese Regeln gehen von dem worst-case aus, daß keine Abdichtung zwischen zwei Fensterelementen dicht bleibt. Ausnahmen werden durch andere heuristische Regeln in der Wissensbasis abgefangen.

3.6 Designkodes und -standards als Interpretationswissen

Designkodes und standards bilden eine signifikante Grundlage von Designwissen. Das Wissen in diesen Kodes ist oft kausal oder experimentell erarbeitet. In vielen Bereichen der Industrie werden Standards und Kodes vorgegeben, um den Designentwurf zu kontrolieren. Ohwohl Kodes und Stan-

[redacted]

deduktives Expertensystem umwandeln ließe. Es ist also wichtig zu wissen, wie ein Designer sich durch Kodes findet und sich in das Wissen einarbeitet. Die Darstellung dieses Metawissens ist leider nicht trival.

Desingnkodes und -standards dienen oft dem Arbeits- und Verbraucherschutz. Normallerweise haben sie die Form von Bestimmungen und Definitionen. In diesem Kapitel wird nur die Nutzung des interpretativen Wissens, das in den Designstandards und -kodes beinhaltet ist und dessen Auswertung im Designprozess betrachtet.

Erstens können Designkodes benutzt werden, um vor dem Beginn des Designentwurfs bestimmte Leistungsanforderungen festzulegen. Dem Designer werden schon vor dem Entwurf gewisse Zwänge verdeutlicht die er bei seiner Arbeit berücksichtigen muß. Zweitens die Nutzung des [illegible]

Ein Teil kann in folgende Regeln transformiert werden:

R13:*if Art des Raumes ist Badezimmer and not Vergrößerungsanforderung der Fläche des Badezimmers*
then benötigte Minimallgröße des Raumes ist 2,2

R14:*if Art des Raumes ist Badezimmer and Vergrößerungsanforderung and zusätzlich benötigte Fläche ist INCR*
then benötigte Minimalgröße des Raumes ist 2,2+INCR

R15:*if Inhalt of Badezimmer include WC or Extradusche or Waschmaschine*
then Vergrößerungsanforderung für Badezimmer

Weitere Regeln sind in [CRR+90] ausgearbeitet.
Eine Sitzung an einem Expertensystem für unseren Kode könnte wie folgt aussehen:
enter command
?find Minimalfläche für ein Badezimmer!!!
Vergrößerungsanforderung für Badezimmer? j/n (how/why/explain)
?why
Art des Raumes ist Badezimmer und not Vergrößerungsanforderung
needed to prove
Minimalfläche für ein Badezimmer ist 2,2
Vergrößerungsanforderung für Badezimmer? j/n (how/why/explain)
?how
Der Inhalt eines Badezimmer ?
Die Optionen für den Badezimmerinhalt sind:
1 WC
2 Extradusche
3 Waschmaschine
4 Andere

? 1
Vergrößerungsanforderung für Badezimmer is true
Vergrößerungsfläche für WC is 0,7
Vergrößerungsfläche für Extradusche is 0,0
Vergrößerungsfläche für Waschmaschine is 0,0
Vergrößerungsanforderung für Badezimmer is 0,7
Minimalfläche für Badezimmer is 2,9

In diesem System ist das Wissen Teil einer Expertensystem-Shell mit Vorwärts- und Rückwärtsverkettung. In diesem Beispiel ist noch kein Design erzeugt, sondern ein Benutzer möchte sich über gewisse Vorschriften informieren. Das System fragt nach Informationen die es benötigt, um Benutzeranfragen zu bearbeiten. Auf Anfrage (how) erläutert es dem Benutzer auch, weshalb es gewisse Informationen braucht. Außerdem kann das System angewiesen werden seine Folgerungen und Schlüsse zu erklären. Bei unserem Beispiel handelt es sich um eine zielgerichtete Anfrage, dabei verkettet das System rückwärts von diesem Ziel bis es alle nötigen Informationen hat oder der Beweis fehlschlägt. Es sind aber auch Anfragen anderer Art möglich die den Vorwärtsverkettungsansatz nutzen.

3.6.2 Compilance Checking von Designkodes

Ein Design wird auf Einhaltung eines Kodes oder Standards geprüft. Dabei spielen gewisse Attribute des Designs eine Rolle. Bei Gebäuden sind das Attribute wie die Klasse des Gebäudes (Wohn-, öffentliches oder Industrie-Gebäude) oder die Feuerfestigkeit der Wände. Für Compilance Checking ist es nützlich, diese Attribute herauszuheben z.B. in Form einer Framestruktur, in die Attributwerte eingetragen werden können, wenn sie festgelegt worden sind. Es ist möglich ein solches Framesystem aus einer regelbasierten Wissensbasis zu extrahieren. Der Vorteil dieses Modells ist eine einfachere

Kommunikation zwischen einem Expertensystem und kommerzieller CAD-Systeme.
Man benötigt 3 Elemente für ein modelbasiertes Expertensystem

1. Eine Expertensystemshell.
2. Ein Interface das die Kommunikation zwischen der Regelbasis dem Modell (Frames) und der Datenbank des externen Systems unterstützt.
3. Einen Frame-Generator welcher das Modell aus einer regelbasierten Wissensbasis extahiert.

Behauptungen über Design können in den folgenden Formen repräsentiert werden:

<predicate>

Prädikate können die Werte True oder False annehmen.

<objekt><verb><value>

wobei *<verb>* ein vom Benutzer definierte Beziehung ist und *<objekt>* den Wert *<value>* annimmt.

<attribute>of<objekt><verb><value>

wobei das *<attribut>of<objekt>* den Wert *<value>* annimmt.
type of *<objekt>* ist *<class name>*
Regeln haben oft die if-then Form. Solche Regeln können in Frame-Beschreibungen übersetzt werden. Als Zwischenschritt müssen durch Parsing Optionslisten für Objekte abgeleitet werden. Die Vorgehensweise verdeutlichen wir uns an folgenden Regeln:

If Raum is a Schlafzimmer or Wohnzimmer
then type of Raum is bewohnbar
If Badezimmer contains Dusche or Waschmaschine
then Vergrößerungsanfrage ist nötig

Die folgende Optionsliste kann aus diesen Regeln abgeleitet werden:

Raum is a:*Schlafzimmer or Wohnzimmer*
type of Raum is:*bewohnbar*
Badezimmer contains:*Dusche or Waschmaschine*

Die Optionsliste muß natürlich um die erweitert werden die noch in anderen Regeln gefunden werden. Die Optionslisten können dann in Objekt(Frame-)darstellung überführt werden. Z.B. die Liste

Raum is a:*Schlafzimmer or Wohnzimmer*

Figure 3.8 zeigt eine solche Darstellung. Diese Art der Übersetzung liefert ein Model das Informatio-

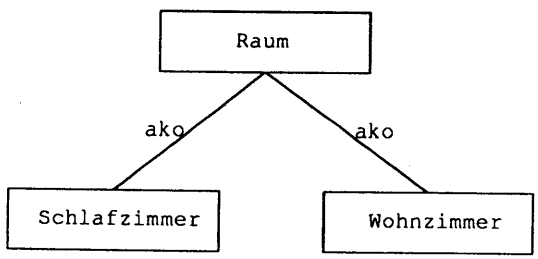


ABBILDUNG 3.8. Ein Raum-Frame

nen als Objekte, Attribute und Werte enthält. Solche Analysen der Designkode-Wissensbasis können vor Überprüfung des ersten Versuchsdesigns durchgeführt werden. Oft werden solche Modelle aber auch manuell erstellt und nur mit den Informationen aus den Regeln aufgefüllt.

Die Interface Komponente (KBI) des Systems beschäftigt sich mit der Interpretation der Datenbasis eines kommerziellen CAD-Systems und einfügen der dort enthaltenen Informationen in Frames.
Das KBI (3.9) enthält 3 Teile:

1. Das Frame-System
2. Ein Interpreter
3. Eine Definitionsmege

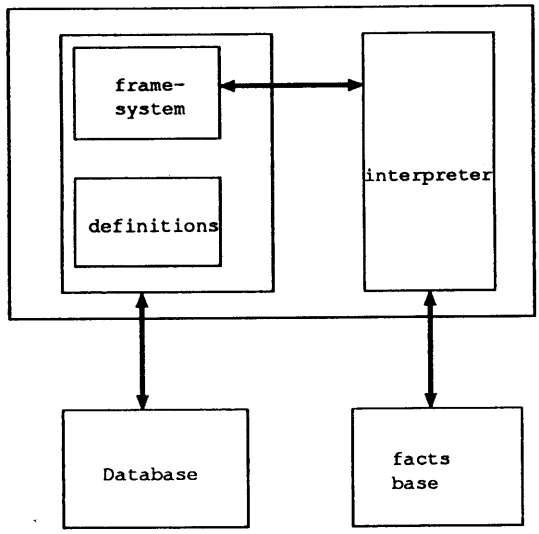


ABBILDUNG 3.9. Die Organisation des KBI

Das Frame-System beinhaltet das Model des Designs oder der Objekte mit denen das System sich befaßt. Im Model sind Informationen über Klassen, Superklassen und Eigenschaften der Objekte enthalten. Ein Frame-Generator analysiert die Regelbasis und konstruiert die in dieser implizit enthaltene Sicht des Modells.

Das KBI verbindet die Expertensystemshell und das Model, indem es die Anfragen des Expertensystems ordnet. Das System schaut immer zuerst in den Frames nach einer Information, bevor es beim Benutzer nachfragt. Wenn neue Fakten festgelegt oder bei der Beantwortung einer Frage abgeleitet wurden, dann werden sie an den passenden Stellen der Rahmendarstellung eingefügt.

3.7 Zusammenfassung

Den Interpretationsprozeß von Design haben wir definiert als die Extration impliziter Informationen aus einer Designbeschreibung. Diese Informationen dienen oft zur Leistungsbestimmung eines Designs. Als einfache Möglichkeit, dieses Interpretationswissen zu modelieren, wurde die Deduktion betrachtet. Wichtig ist auch die Feststellung, daß es Interpretationshierarchien gibt. Da nicht alle möglichen Ableitungen eines Designs interessant sind ist es wichtig, Strategien zur Auswahl der relevanten Ableitungen auszuarbeiten. Generel legt man ein Interpretationsziel fest und testet dessen Wahrheit gegen die Designbeschreibung. Beweisprozeduren spielen in diesem Zusammenhang eine große Rolle. Drei Faktoren sind bei der Designinterpretation zu beachten. Erstens, da auch Parsing eine wichtige Rolle bei der Interpretation spielt, kann man generatives und interpretatives Wissen nicht immer trennen. Zweitens ist die Wissensbasis nicht immer komplett. Drittens ist sie nicht immer konsistent, eindeutig und sicher.

Abschliesend kann man feststellen, daß Interpretationswissen viele Anwendungen in Designsystemen hat. Coyne und Rosenman in ihrem Buch jedoch nicht viel mehr als eine Begriffsdefinition und Beispiele liefern.

3.8 Literaturverzeichnis

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero.

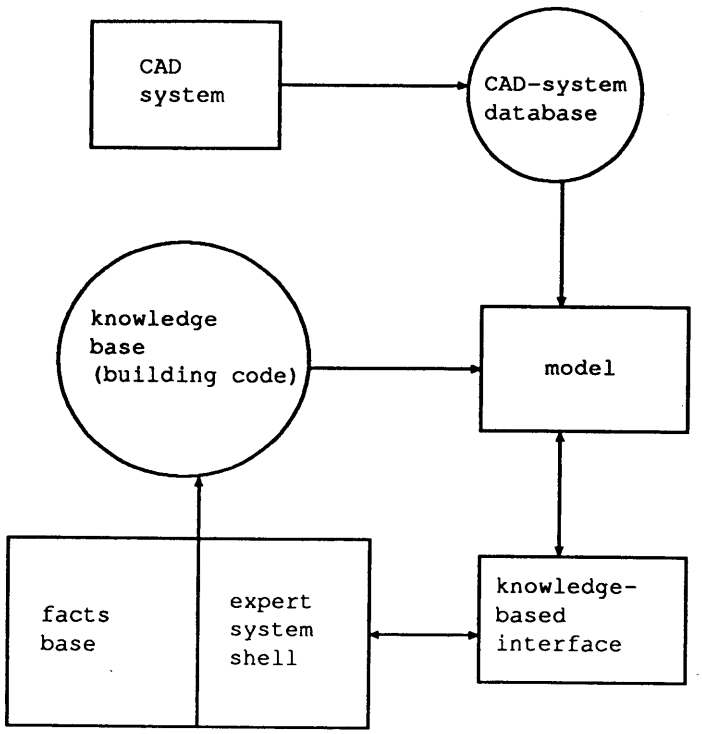


ABBILDUNG 3.10. Darstellung eines kompletten modelbasierten Expertensystems

Knowledge-Based Design Systems. Addison-Wesley Publishing Company, 1990

4

Designprozess

Martin Wagner

ZUSAMMENFASSUNG Um den Entwurfsprozeß modellieren zu können, muß dieser zuerst eingehender betrachtet werden. In diesem Kapitel werden wir den Entwurfsprozeß etwas genauer unter die Lupe nehmen und seinen Ablauf studieren, um uns eventuelle Methoden, Ansätze, Strategien und wichtige Werkzeuge für die Modellierung verschaffen zu können.
Wir betrachten zwei Ansätze des Entwurfsprozesses, Planung und Regulierung. Danach untersuchen wir die Rolle der Abstraktion und Zerlegung. Obwohl sie hier getrennt betrachtet werden, liegt es auf der Hand, daß ein intelligentes System Abstraktion und Zerlegung interaktiv benutzt. Auch, daß ein System beide Ansätze verwendet, indem es in verschiedenen Abstraktionsebenen den einen oder den anderen Ansatz benutzt, ist nicht undenkbar.
Im folgenden Abschnitt wird eine tiefergreifende Beschreibung der Modellierung eines durch Regulierung kontrollierten Entwurfsprozesses dargestellt Anschließend betrachten wir diverse Hilfsmittel für die Planung und schließen das Kapitel mit einem Beispiel der Prototypenverfeinerung.

4.1 Einleitung

Zunächst stellen wir uns die Frage, was der Entwurfsprozeß eigentlich ist. Um diese Frage beantworten zu können brauchen wir folgende Definitionen [CRR+90]:

- Entwurfselemente [*engl. Design elements*]: Ein Entwurf besteht aus verschiedenen Elementen.
- Entwurfshandlungen [*engl.* Design actions]: Operatoren, die Entwurfselemente konfigurieren, heißen Entwurfshandlungen.
- Entwurfsaufgaben [*engl.* Design tasks]: Operatoren, die Entwurfshandlungen konfigurieren, werden Entwurfstasks genannt.

Alle Zustände, die mein Entwurf nach beliebiger Anwendung von Entwurfshandlungen haben kann, spannen einen *Entwurfsraum* (*Design space*) auf. Irgendwo in diesem Suchraum befinden sich der Entwurf im Anfangszustand und vielleicht ein oder mehrere Zielzustände (siehe Abb. 4.1). Im Entwurfsprozeß geht es also einfach betrachtet darum, einen Weg zwischen dem Anfangszustand und einem Zielzustand in unserem Suchraum zu finden. Wichtig ist es also, einen effektiven Such- bzw. Optimierungsalgorithmus zu finden. Maschinelle Beweis- oder Schließtechniken sind erforderlich, um verschiedene Problem-Solving-Aktivitäten zu bewältigen [Smithers89]. Alternative Entwürfe müssen verglichen und berücksichtigt werden, während wir den Suchraum erforschen.

Der Generierungsprozeß wird mit Entscheidungen über Entwurfshandlungen kontrolliert. Dadurch erkennen wir, wie wichtig ein System ist, das die Fähigkeit besitzt, effekiv Entscheidungen über Entwurfshandlungen zu treffen. Der Ingeneur benutzt seine Erfahrung und Wissen, um diese Entscheidungen zu treffen. Es ist notwendig, dieses 'Entwurfswissen' strukturieren zu können, um es der Maschine verständlich zu machen.

Diese Punkte sind die Schwerpunkte dieses Kapitels. Ein kurzer Überblick auf die Themen:

- *Reasoning about design actions* : Schließen, urteilen, beweisen über/von/auf Entwurfshandlungen
- *Reasoning about design tasks*

Um uns den Prozeß etwas zu veranschaulichen, betrachten wir zuerst ein einfaches Beispiel: der Entwurf eines Hauses.

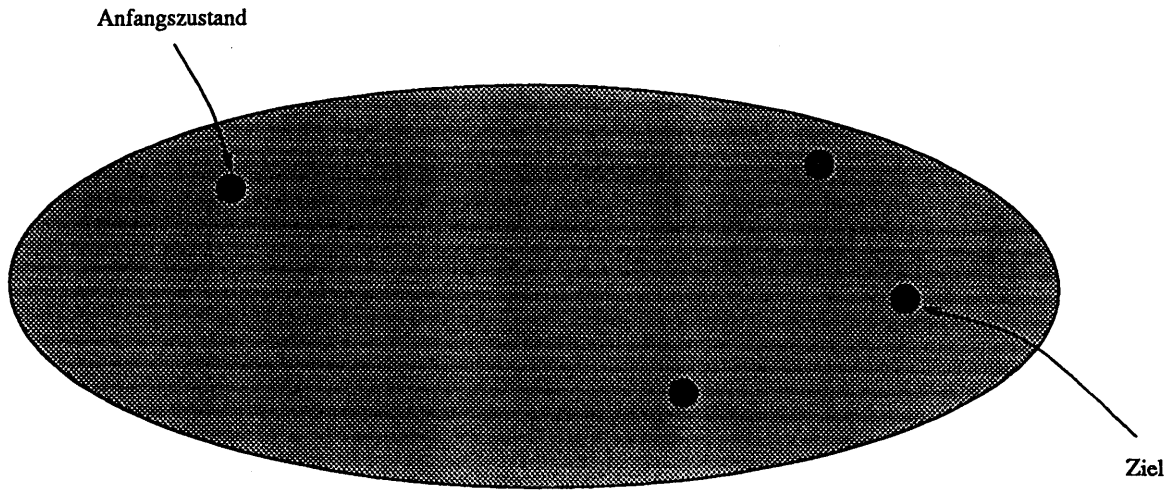


ABBILDUNG 4.1. Entwurfsraum

4.1.1 Ein einfacher Entwurf

Um ein Haus zu entwerfen, gibt es mehrere Ansätze: Ich kann, falls Material vorhanden, einfach damit anfangen, Stein auf Stein zu legen und Wände hochzuziehen, ohne die Form des Hauses vorher festgelegt zu haben. Ich kann mir aber auch vorher schon Gedanken über die Anzahl der Räume und Verbindungskomponenten machen. Daraufhin kann ich dann im Gespräch mit einem Architekten eine Skizze anfertigen (siehe Abb. 4.2). Dieser könnte dann den entgültigen Grundrißplan festlegen, um erst dann mit den Bauarbeiten zu beginnen.

Welche Eigenschaften des Entwurfsprozesses oder Hilfsmittel können wir uns aus dem obigen Beispiel verschaffen ? Zunächst können wir den Entwurf 'regeln', d.h. wir realisieren Entwurfshandlungen (indem wir Steine mauern), betrachten deren Wirkung auf den Entwurf, überlegen uns den nächsten Schritt, führen ihn aus, betrachten deren Wirkung auf den Entwurf, überlegen uns den nächsten Schritt, u.s.w., bis wir einen befriedigenden Endzustand erreicht haben. Ein zweiter Ansatz wäre, daß wir vor der Implementierung der Entwurfshandlungen vorher den Entwurf mit bestimmter Genauigkeit 'planen'. Wir überlegen uns eine bestimmte Reihenfolge für bestimmte Entwurfshandlungen.

Ein Hilfsmittel, das beim Planen intuitv benutzt, aber oft nicht richtig als Hilfsmittel wahrgenommen und 'übersehen' wird, ist die Abstraktion[*engl.* Abstraction]. Schon beim skizzieren bewege ich mich in einer höheren Abstaktionsebene, indem ich Wände durch Linien ersetze. Eng verbunden mit der Abstraktion ist die Zerlegung [*engl.* Decomposition]. Ich kann mein Ziel in mehrere Teilziele unterteilen, um diese leichter bearbeiten zu können. In unserem Beispiel kann ich das Dach getrennt von dem Keller planen, ohne daß sich die beiden verschiedenen Bereiche überschneiden.

Fassen wir also kurz die wichtigsten Schlagworte zusammen. Sie bilden den Leitfaden für die folgende Abschnitte:

- Planung und Regulierung.
- Abstraktionshierarchien und Ebenen
- Zerlegung

4.2 Reasoning about Design Actions

Der Titel ist etwas schwierig zu übersetzen. Möglichkeiten wären planen, konfigurieren, überlegen oder schließen über/von/auf Entwurfshandlungen. Ein besseres Verständnis soll in den folgenden Abschnitten entstehen. Zunächst wird anhand einfacher Beispiele der Unterschied zwischen den beiden Entwurfsansätzen Planung und Regulierung erläutert [*engl.* Planing and regulating]. Danach betrachten wir die Rolle der Abstraktion im Entwurfprozeß sowie verschiedene Darstellungsmöglich-

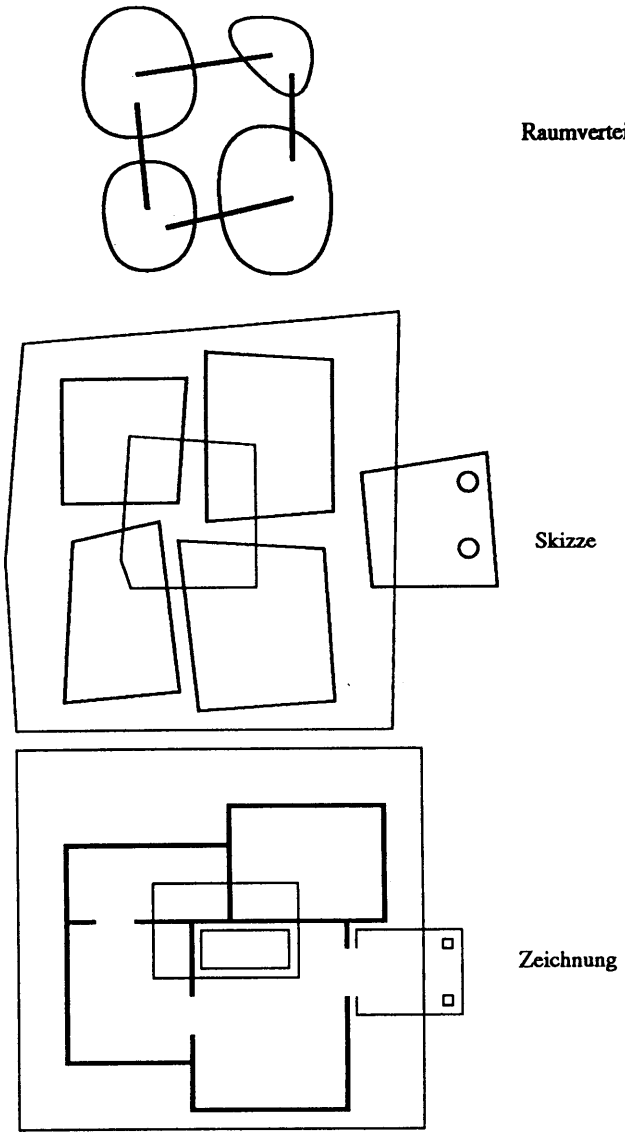


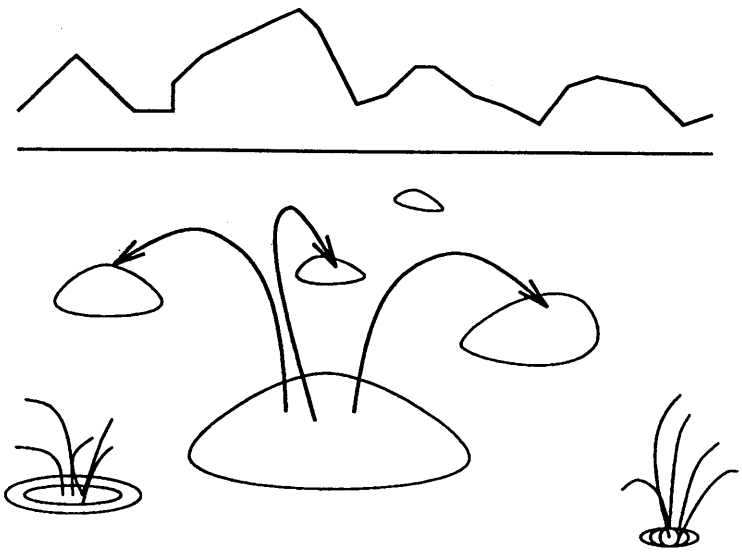
ABBILDUNG 4.2. Verschiedene Abstraktionsebenen des Hausentwurfes

keiten. Bevor wir uns dann mit konkreten Beispielen der Anwendungen beschäftigen, untersuchen wir kurz die Zerlegung [*engl.* Decomposition].

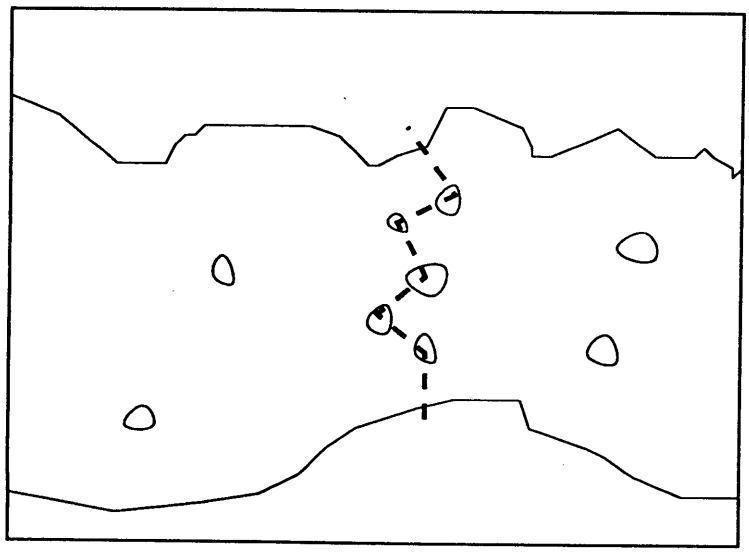
4.2.1 Planung und Regulierung

Bei der Regulierung betrachten wir unseren aktuellen Zustand und die möglichen Folgezustände, um den nächsten Schritt zu implementieren. Bei der Planung entscheide ich über eine Sequenz von Entwurfshandlungen bevor ich sie ausführe.

Betrachten wir uns die beiden Ansätze im Falle einer Autofahrt. Ziel ist es, vom Punkt A zu Punkt B zu kommen. Mögliche Entwurfshandlungen wären *Fahre Straße entlang*, *biege links ab*, *biege rechts ab*. Da ich bei der Regulierung nur die Folgezustände betrachten kann, ist nur die Handlung *Fahre Strasse entlang* möglich, falls keine Abzweigung erscheint. Kommen wir zu einer Kreuzung oder Abzweigung, müssen wir eine Entscheidung über den möglichen Folgezustand treffen (Anwendung von Heuristik). Ein Vorteil der Regulierung ist, daß der Suchraum des Entwurfes eingeschränkt wird, da ich nur die Folgezustände meines aktuellen Zustandes betrachten muß. Voraussetzung dafür ist aber eine gute Kontrolle, die effektiv über die nächste Handlung entscheidet. Nachteil des Ansatzes ist – im Falle der Autofahrt gut illustriert – die hohe Wahrscheinlichkeit von Fehlversuchen. In diesem Fall benötigt man Backtracking um einen erneuten Ansatz zu starten. Bei der Planung



Moegliche Handlungen betrachten



Planen der Ueberquerung

ABBILDUNG 4.3. Unterschiede beim Regeln und Planen

besitzen wir die Möglichkeit den gesamten Suchraum zu überblicken, z.B. mit einer Landkarte. Damit können wir die Fahrt planen und alle Entwurfshandlungen festlegen, ohne eine ausgeführt zu haben. Der Vorteil ist, daß wir in einer billigen Umgebung Entscheidungen über die Sequenz der Handlungen treffen können (deutlich im Falle des Hausentwurfs) und eventuelle Fehler billig korrigieren können.

Betrachten wir nun ein anderes Beispiel: den Entwurf einer Überquerung eines Flusses. Ziel ist es, von einer Seite zu der anderen Seite zu kommen. Die Entwurfshandlungen seien auf *waten* und *Springe auf Stein* beschränkt. Bei der Regulierung betrachte ich nur die Umgebung unmittelbar vor mir. Besitze ich die Möglichkeit, auf einen Stein zu springen, so tue ich das, falls mich diese Handlung an das gegenüberliegende Ufer näher heranbringt (siehe Abb. 4.3).

Im Planungsmodell gehe ich in eine Position, von der aus ich den gesamten Suchraum überblicke, z.B. auf die Spitze eines in der Nähe liegenden Berges. Wir überlegen uns alle Schritte um den Fluss zu überqueren, entwerfen vielleicht auch mehrere Wege und berücksichtigen dann den einfachsten (siehe Abb. 4.3).

4.2.2 ABSTRAKTION

Wieso sind Abstraktionsebenen und Hierarchien eine wichtige Hilfe im Entwurfsprozeß ? Intuitiv ist dies klar, da es z.B. billiger ist, auf Papier ein Haus zu entwerfen statt mit Stein und Mörtel. Der Vorteil der Abstraktion liegt jedoch nicht nur in einer billigen Entwurfsumgebung, sondern in der Erleichterung der Kontrolle des Prozesses. Der Generierungsprozeß wird, wie vorher schon angedeutet, durch Entscheidungen über Entwurfshandlungen kontrolliert. Reduziere ich die Anzahl der möglichen Handlungen so ist es leichter an ein Ziel zu kommen, da dies meinen Suchraum einschränkt. Genau dies geschieht mit der Abstraktion. Indem ich also Wände durch Linien repräsentiere, werden Handlungen wie *Lege Stein auf Stein* überflüssig.

Wie in Abb. 4.4 leicht zu erkennen ist, kann ich für den Hausentwurf mehrere Abstraktionsebenen haben, die meinen Entwurf jeweils mit unterschiedlichem Verfeinerungsgrad darstellen können. Zerlege ich meinen Enwurf in z.B. 'Entwerfe Dach' und 'Entwerfe Erdgeschoß', so kann ich verschiede Abstraktionshierarchien festlegen. Ich kann Abstraktionsebenen dadurch ausnutzen, indem ich in einer bestimmten Ebene mit kleinem Suchraum eine Lösung zu meinen Bedingungen finde, und mich dann mit Verfeinerung zu den tieferen Ebenen runterarbeite. Jetzt ist noch zu betrachten, *was* wir im Entwurfsprozeß abstrahieren können. Coyne erwähnt folgende drei Punkte, die sich als sinnvoll für Abstraktionen erweisen könnten:

- *Design description abstractions*
- *Design actions abstractions*
- *Control abstractions*

Abstraktion der Entwurfsbeschreibung

Der Hausentwurf wird in Abb. 4.2 in der Einleitung durch eine Zeichnung in Abhängigkeit der Abstraktionsebene entsprechend genau beschrieben. Bei der Modellierung eines Entwurfprozesses müssen wir darauf achten, daß das System anhand der Entwurfsbeschreibung den aktuellen Zustand *interpretieren* kann, um Entscheidungen über zukünftige Handlungen treffen zu können.

Abstraktion der Entwurfshandlungen

Ähnlich wie bei der Entwurfsbeschreibung ist es auch möglich, Abstraktionsebenen für Entwurfshandlungen festzulegen. Grundgedanke ist, von generellen Handlungen auf spezifischere Handlungen überzugehen. Betrachten wir als einfaches Beispiel die Handlung 'befestige A auf B'. Diese Handlung kann zu den Handlungen 'lege A auf B', 'Loch bohren in A', 'Loch bohren in B', und 'Schraube A mit B fest' *expandiert* werden (Abb. 4.5).

Finden wir einen Lösungsweg im Suchraum einer höheren Abstraktionsebene, so können wir die einzelnen Handlungen aus der erstellten Sequenz als Leitfaden für die tieferen Ebenen benutzen. Diese können wiederum in Handlungen expandiert werden, bis diese entweder unzerlegbar oder 'ausführbar' sind.

Abstraktion der Kontrolle

Nun behandeln wir Entwurfshandlungen wie Elemente. Operatoren, die diese konfigurieren, können eine Sequenz von Entwurfshandlungen bestimmen. Entsprechend des Entwurfraumes besteht die Möglichkeit, einen Suchraum zu bilden, der aus allen möglichen Sequenzen von Entwurfshandlungen aufgespannt wird.

Da Entwurfshandlungen mehrere Abstraktionsebenen besitzen können, können wir auch mehrere Abstraktionen für die Kontrolle vermuten. Eine Möglichkeit, diese zu definieren, wäre folgende: Operatoren, die Enwurfshandlungen konfigurieren, sind eine Art der Kontrolle, da sie eine Sequenz von Handlungen festlegen und damit den Ablauf 'kontrollieren'. Gehen wir eine Ebene höher, so sind auch Operatoren, die meine Entwurftasks konfigurieren eine Art der Kontrolle. Dies könnten wir auf beliebigen Ebenen weiterführen. Ziel oder Vorteil dieser Abstraktionen wäre z.B., daß wir zur einer Abstraktionsebene gelangen könnten, in der wir unser Problem als 'bereichunabhängig' betrachten und sie mit GPS (general problem solvers) bearbeiten könnten.

4.2.3 Zerlegung

Wie wir vorher schon gesehen haben, spielt die Zerlegung eine nicht unwichtige Rolle im Entwurfprozeß. Ich kann meine Ziele in Teilziele unterteilen, die einfacher zu lösen sind. Auch kann ich Entwurfshandlungen zerlegen, so wie wir es im letzten Abschnitt gesehen haben. Es kommt aber meistens nur selten vor, daß ich Entwurfshandlungen unabhängig von anderen implementieren kann. Generell werden Handlungen durch andere beinflußt und umgekehrt. Es können also Konflikte zwischen sogenannten 'rivalisierenden Handlungen' entstehen. Zusätzlich muß also jedes System in der Lage sein, diese Konflikte beheben zu können.

Betrachten wir unsere Überlegungen mit einem Problem aus der Blockwelt (siehe Abb. 4.6). Ich besitze drei Entwurfselemente, Block A, Block B und Block C. Alle drei befinden sich auf den Boden. Ziel ist es, einen Turm so zu bauen, das A auf B und B auf C ist. Dies sei ausgedrückt durch *Auf*(A,B) und *Auf*(B,C). Erlaubte Handlung sei *legeAuf(x,y)*. Es darf jeweils immer nur ein Block bewegt werden. Vorbedingungen für die Handlung sind, daß auf beiden Objekten kein anderes Objekt liegt, und daß x auf etwas liegt. Nachbedingungen sind, daß x auf y liegt, x immer noch frei ist, y aber nicht mehr. Wir können unser Ziel in zwei Teilziele zerlegen, nämlich *Auf*(A,B) und *Auf*(B,C). Diese Ziele werden durch die Handlungen *legeAuf*(A,B) und *legeAuf*(B,C) realisiert. Implementieren wir die

Handlungen, so erkennen wir, daß die Reihenfolge der Handlungen wichtig ist. Wie wir in Abb. 4.6 erkennen, liefert der Prozeß einen Fehler, falls wir die Handlung *legeAuf*(A,B) zuerst realisieren. Wir definieren einen Endzustand als den Zustand, in dem das System eine schon realisierte oder geplante Handlung rückgängig machen muß weil sie nicht zum Ziel führt. Eine mögliche Strategie, um diesen Konflikt zu lösen, ist die Umordnung oder Vertauschung der Teilzeile.

Diese Strategie führt jedoch nicht immer zum Ziel, wie Abb. 4.7 zeigt. Wir befriedigen eins der Teilziele, gelangen aber immer in einen Endzustand.

4.3 Regulierung der Entwurfshandlungen

In diesem Abschnitt betrachten wir die Kontrolle über den Entwurfprozeß mit dem Regulierungsansatz. Grundprinzip ist das einfache "Generiere und Teste". Die Umgebung liefert das blackboard model(BM).

4.3.1 The Blackboard: Eine kurze Zusammenfassung

Nach der Entwicklung der Expertensysteme zeigte sich bald ein kleines Problem [ReddyOHare91]. Um bereichsübergreifende Probleme lösen zu können, bedarf es mehrerer Experten, man war aber unfähig miteinander zu kommunizieren. Eine einfache Lösung fand man im Blackboard Model. Es funktioniert nach dem (hoffentlich bekannten) electronic news system. Habe ich ein Problem, so poste ich mein Problem. Verschiedene Experten geben mir verschiedene Lösungsvorschläge. Diese Lösungsvorschläge werden gewertet und schließlich führe ich die Lösung aus, die mir am besten erscheint. Nach diesem Konzept arbeitet das BM. Auf dem Blackboard sind verschiedene Anfragen(tasks) auf die jeder Experte zugreifen kann. Eine Kontrolleinheit weist dann den verschiedenen Experten die Möglichkeit zu, bestimmte Anfragen zu bearbeiten (triggern). Diese liefern dann verschieden Lösungen oder Handlungen . Nach Bewertung der Vorschläge wird einer von der Kontrolle ausgesucht und aktiviert. Der Zyklus wiederholt sich, bis alle Anfragen gelöst sind. Generell besteht ein BM aus drei Elementen:

1. Factsbase: Eine globale Datenbank die den momentanen Zustand darstellt.Alle Experten haben Zugriff auf diese.

2. knowledge sources: Jeder einzelne Experte bekommt dieses Attribut.

- trigger
- evaluate
- focus attention
- fire

Durch ein BM besitzen also verschiedene Experten die Möglichkeit, miteinander zu kommunizieren. Dies stellt also die Haupteigenschaft eines BM dar. Deutliche Unterschiede bezüglich der Haupteigenschaft gibt es bei Coyne. Er weist der Fähigkeit des Systems, auf bestimmte Fakten oder Handlungen Schwerpunkte zu legen, die größte Bedeutung zu.

4.3.2 Regulierung des Blackboard Models

Da wir im Regulierungsansatz nur die Folgezustände betrachten können, reduzieren sich die knowledge sources auf alle mögliche Entwurfshandlungen. Jeder 'Experte' schlägt also eine Entwurfshandlung vor.

Der Zyklus der Kontrolle wird nach dem Muster des vorherigen Abschnitts abgearbeitet. Die Kontrolleinheit bewertet den aktuellen Zustand des Entwurfes. In Abhängigkeit des Zustandes überprüft die Einheit, welche Handlungen möglich sind, und triggert diese. Mittels Heuristik wird dann entschieden, welche Handlung die effektiveste ist, und gefeuert. Dies wird solange wiederholt, bis die Bewertung des aktuellen Zustand den Anforderungen entspricht.

4.4 Planen der Entwurfshandlungen

Coyne hält Planung für den effizienteren Ansatz, um Entwurfshandlungen zu ordnen oder konfigurieren. Wichtiges Argument dafür ist, daß es bei der Planung deutlich mehr Hilfsmitteln gibt als bei der Regulierung. Einige werden in diesem Abschnitt angesprochen, so wie Vorwärtsverkettung, Rückwärtsverkettung, Abtraktionsebenen und hierarchisches Verfeinern, und Meta-Planung. Wie vorher schon angedeutet, werden meistens mehrere der Hilfsmittel in einem Entwurfssystem kombiniert. Zur Beschreibung werden wir sie aber einzeln betrachten.

4.4.1 Vorwärtsverkettung

Wie am Anfang schon angesprochen, besitzen wir beim Planen einen Überblick über denn gesamten Suchraum. Die Vorwärtsverkettung ist ein Suchalgorithmus, der kein Wissen anwendet (exhaustive search). Ein Anwendungsbeispiel liefert uns die besprochene Autofahrtplanung. Der gesamte Suchraum wird durch eine Landkarte repräsentiert. Wir beginnen in Stadt A und überprüfen alle mögliche Wege. Gelangen wir an einen Endzustand, das heißt wir müssen eine bereits geplante Handlung rückgängig machen, so müssen wir zu dem Punkt backtracken, an dem eine weitere mögliche Handlung existiert, und dort weitermachen.

Wenden wir jetzt dieses Prinzip auf unser bekanntes Problem in der Blockwelt an. Der Suchraum wird durch alle möglichen Zustände meiner drei Entwurfselemente dargestellt(siehe Abb. 4.10). Im Gegensatz zu unser früher angewandten Technik gelingt es uns nun, zu einem Zielzustand zu kommen.

Vorteil dieser Methode ist die relativ leichte Implementierung, und daß wir davon ausgehen können, die beste Möglichkeit zu finden. Nachteil ist der hohe Suchaufwand. In unserem Beispiel ist der Suchraum relativ klein, bedenken wir aber, daß bei etwas anspruchvolleren Aufgaben der Suchraum entsprechend groß ist.

4.4.2 Rückwärtsverkettung

Bei der Rückwärtsverkettung plane ich meine Autofahrt vom Ziel aus und versuche an den Anfang zu kommen. Allgemein betrachtet bedeutet dies, daß ich mir überlege welche Handlungen notwendig sind, um ein bestimmtes Ziel zu erreichen. Dann untersuche ich die Vorbedingungen meiner Handlungen und behandle diese als Teilziele. Dieser Prozeß wird so lange weitergeführt, bis eine Menge von Vorbedingungen gefunden wird, die mit dem Anfangszustand des Entwurfes übereinstimmen.

Im Beispiel der Blockwelt wird dies mit Abb. 4.12 dargestellt. Der Buchstabe T bedeutet, daß ein Teilziel dem momentanen Zustand entspricht. Das Symbol < in der Abbildung kennzeichnet die Stelle, an der das System 'backtrackt' um an ein korrektes Ergebnis zu kommen. Die endgültige Sequenz von Handlungen ist damit LegeAuf(C,Boden), LegeAuf(B,C), LegeAuf(A,B).

4.4.3 Abstraktionsebenen und hierarchische Verfeinerung von Pläne

In unserem Beispiel der Autofahrtplanung können wir verschiedene Abstraktionsebenen unseres Bereiches ausnützen. So ist es vielleicht vorteilhafter, als Ziel nicht die Fahrt zwischen A und B zu planen, sondern von einen Bereich A zu einen Bereich B. Falls die Bereiche geschickt gewählt werden, reduzieren sich die möglichen Pfade in unserem Suchraum.

Planen durch mehrere Abstraktionsebenen bedeutet im Design, daß ich mein Problem mit groben Skizzen löse, meinen Entwurf beim Übergang in eine tiefere Ebene verfeinere, bis ich eine Ebene erreiche, in denen die Handlungen 'erfüllbar' sind.

Dies, auf unsere Autofahrt übertragen, könnte folgenderweise aussehn: Wir verbinden die Stadt A mit Stadt B mit einer direkten Line (grobe Skizze der Lösung). Dann suchen wir Städte oder größere Dörfer die im Umfeld der gezogenen Linie liegen und verbinden wieder A mit B aber jetzt mit Einzug dieser. In der untersten Ebene unser Abstraktionshierarchie werden dann die einzelne Routen durch die gewählten Teilziele bestimmt.

Diese Methode wird bei Coyne als 'expansion' bezeichnet (Literaturhinweis NOAH planing system, Sacerdoti 1977). Ich kann *expansion rules* festlegen, um eine Sequenz von allgemeinen Entwurfshandlungen in spezifischere, ausführbare Handlungen zu transformieren. *Expansion rules* enthalten Wissen, wie individuelle Ziele erreicht oder realisiert werden. Jedoch sind *expansion rules* alleine nicht ausreichend. Im allgemeinen entstehen Konflikte bei der Expansion. Um diese zu lösen werden sogenannte *critics* eingeführt. *Critics* könnten folgende Aufgaben besitzen :

- Löse Konflikte
- Benutze existierende Objekte
- Entferne redundante Vorbedingungen

Es fehlt jetzt nur noch eine Struktur oder Umgebung, in der es möglich ist, eine partielle Ordnung zwischen Enwurfshandlungen herzustellen, um die effizientere Arbeit der *expansion rules* und *critics* zu gewährleisten. Sacerdoti bezeichnet das *prozedurelle Netzwerk* als adäquate Struktur, um Beziehungen zwischen Entwurfshandlungen zu beschreiben. Diese Beziehung kann seriell – d.h. die Reihenfolge, in der eine Gruppe von Handlungen auszuführen ist, ist bereits festgelegt – oder parallel sein. Der Anfangszustand ist meistens eine Konjunktion von Handlungen höherer Abstraktionsebe-

[redacted]

nen. Die Kontrolle ist relativ einfach, da diese hauptsächlich von den critics übernommen wird. In jedem Zyklus wird versucht, eine Expansion Rule anzuwenden. Falls dies geschieht, werden alle Critics durchgearbeitet. Ist keine Expansion Rule mehr anwendbar, so sind wir in einen Endzustand oder einen Zielzustand.

Betrachten wir diesen Ansatz mit den beiden Aufgaben "male Dach" und "male Leiter". Die Handlung "male Dach" läßt sich durch die drei Handlungen "hole Farbe","hole Leiter" und "streiche Dach" expandieren, "male Leiter" durch "hole Farbe" und "streiche Leiter". Expandieren wir das Netzwerk, so erscheint ein Konflikt. Es soll nicht möglich sein, etwas zu holen, das gerade bemalt worden ist. Die Reihenfolge der Handlungen muß festgelegt werden, hier kommen die Critics ins Spiel. In Abbildung 4.14 wird dann die endgültige Lösung angegeben.

Dieser Ansatz macht zwar Backtracking überflüssig, die Schwierigekeit liegt aber darin, Wissen für die Konfliktlösung den Critics zuzuführen.

Abb. 4.14 zeigt, wie die beiden Ziele Auf(C,B) und Auf(A,C) realisiert werden. Zustand 1 kann als Konjunktion zweier 'höherer Handlungen' verstanden werden. Expandieren wir diesen Zustand zu Zustand 2, so erkennen unsere Critics einen Konflikt. Die Handlung LegeAuf(A,C) hat als Nachbedingung die Handlung Frei(C) unmöglich gemacht. Um diesen Konflikt zu lösen, ändern die Critcs

darstellt. Zustand 4 wird nach Zustand 5 expandiert. Hier werden Variablen durch existiernde Objekte ersetzt. Wieder ist die Vorbedingung Frei(C) überflüssig, wir entfernen diese also und kommen dadurch zu Zustand 6. Die Knoten Frei(A) und Frei(B) existieren im initialen Zustand und können deshalb ignoriert werden. Der entgültige Plan besteht also aus der Sequenz LegeAuf(C,B), LegeAuf(A,C).

Betrachten wir den Ablauf des Generierungsprozesses, so erkennen wir, daß auch Zwischenergebnisse verständlich sind. Dies ist ein weiterer Vorteil dieses Systems gegenüber anderen.

4.4.4 META-PLANER

Einige entwickelte wissensbasierten Systeme besitzen eine Metaplaner genannte Komponente. Der Ausdruck Metaplaner wird in zwei verschiedenen Zusammenhängen gebraucht. Der erste bezeichnet den Metaplaner als eine bestimmte Abstraktionsebene der Kontrolle, meistens die höchste. Der Metaplaner besitzt also Wissen, um verschiedene Pläne für einen Bereich zu erstellen. Ein einfaches Beispiel wäre, um auf unsere Autofahrtplanung zurückzukommen, daß ich drei verschiedene Experten habe, die mir unter drei verschiedene Kriterien einen Plan erzeugen. Der erste versucht so viel wie möglich Autobahnen zu benutzen, der zweite achtet mehr auf das gastronomische Angebot und der dritte auf reizvolle Aussicht. Der Metaplaner unseres Systems müßte Wissen besitzen, die Alternativen abzuschätzen und zu vergleichen, um sich für eine entscheiden zu können.

Die zweite Möglichkeit konzentriert sich auf den Inhalt dieses Wissen. Metaplanen wird als bereichsunabhängig betrachtet. Es beschäftigt sich mit universellen Prinzipien angewandt auf generelles Entscheiden. Diese Prinzipien sollen den Planungsprozeß leiten. Einige Beispiele dieser Prinzipien wären:

- Schütze Ressourcen
- Versuche, soviel Ziele wie möglich zu erfüllen.
- Vermeide unmögliche Ziele
- Versuche, erwünschte Eigenschaften zu erfüllen.

Die Idee des Metaplanens spielt bei Coyne eine große Rolle. Seiner Meinung nach müßte viel in diesen Bereich gearbeitet werden, da dieser noch bedeutende Mängel aufweise. Ziel wäre es, ein Sytem zu entwickeln, in dem wir das Problem so abstrahieren können, daß wir dieses dann mit

4.5 Reasoning about Design Tasks

Ein Beispiel eines Decomposition Trees liefert Abbildung 4.15. Es beschreibt den Entwurf eines Photokopierertransportsystems. Das Entwurfsystem soll die Anzahl und Lage der Rollen bestimmen, die notwendig sind, um das Kopierpapier von dem Stapellager bis zum Ausgang zu befördern.

Auf oberster Ebene des Baumes finden wir allgemeine tasks wie *entwerfePapierPfad*. Der frame der task *bestimmeAnzahlUndLageDerRollstationen* könnte folgendermaßen aussehen:

```
frame
  Slot            Value
    name              Ziel 5
    bezeichner        bestimme Anzahl und Lage der Rollstationen
    status            initial
    Vaterziel         Entwerfe Papierweg
    Eingabeparameter  Papierweglaenge,...
    Ausgabe           Anzahl der Rollstationen, lage der Stationen,...
    Entwurfsmethoden  subtasks
                        Bestimme minimale Anzahl von Rollstationen
                        Bestimme grobe Lage der Stationen
                          ...
                          ...
    Bedingungen       Erste Station <= 100 millimeter
                      Minimaler Abstand zwischen den Stationen = 20 cm
                          ...
    Vorschlaege       Falls  zuWenigPlatz
                           Versuche Anzahl von Stationen zu verringern
                           Verkleinere Durchschnitt der Stationen
```

Sind Handlungen nicht mehr in subtasks zerlegbar, befinden wir uns auf Blättern. Blätter sehen im allgemeinen so aus:

```
frame
  Slot            Value
    name              Blatt 4
    bezeichner        bestimme Durchschnitt einer Rollstation
    status            abgearbeitet
    Vaterziel         Entwerfe Rollstationen
    Eingabeparameter  keiner
    Ausgabe           Durchschnitt einer Rollstation
    Entwurfsmethoden
                        Durchschnitt=Random(minDurch...maxDurch)
```

Wir erkennen, daß bei Blättern meistens nur der Wert eines Parameters festgelegt wird. Dies geschieht manchmal mit einer festen Wertzuweisung (Erfahrung) oder mit Hilfe eines Zufallsgenerators. Entsteht in den oberen Ebenen ein Konflikt, so wird versucht, ihn durch Werteanpassung der Parameter zu lösen.

Die Abarbeitung erfolgt mit der top-down Methode. Sind alle Ebenen abgearbeitet (status abgearbeitet), so sind wir fertig.

4.6 Schlußfolgerung

Wie der aufmerksame Leser vielleicht schon bemerkt hat, beschränkt sich nicht nur das letzte Modell, sondern alle vorgestellten Systeme auf die Prototypenverfeinerung und -anpassung. Dies folgt eigentlich schon aus unserer Definition des Suchraumes. Er ist statisch definiert, begrenzt.

Dadurch haben wir die möglichen Entwürfe eingeschränkt und produzieren nichts 'innovatives'. Die Kreativität bleibt also auf der Strecke.

Wie können wir dieses Problem lösen ? Eine Methode, den Entwurfsraum dynamischer zu gestalten, ist dadurch gegeben, wenn unser Entwurfssystem *lernfähig* ist. Durch neu erworbenes Wissen vergrößert sich unser Entwurfsraum. Altes Wissen kann nach einer bestimmten Zeit gelöscht werden. Dies kann zum Beispiel mit Neuronalen Netzen realisiert werden. Eine andere Möglichkeit ist vielleicht durch Implementierung 'zufälliger' Handlungen gegeben, d.h. es werden zufällige Handlungen mit zufälligen Parameter realisiert.

Heutzutage ist man mit der Modellierung der Kreativität noch nicht weit gekommen, weshalb wir uns die Frage stellen müssen : Ist Kreativität überhaupt modellierbar ?

4.7 Literaturverzeichnis

[Smithers89] T. Smithers. Why design cannot be supported by geometry alone. Computer Aided Design, 21(3), 1990.

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge-Based Design Systems.* Addison-Wesley Publishing Company, 1990.

[ReddyOHare91] M. Reddy and G.M.P. O'Hare. The blackboard model: a survey of its application. *AI Review*, 5, 1991.

[Watterson90] B. Watterson. The indispensable *Calvin and Hobbes*. Novexa, vol. 2, 53, 1990.

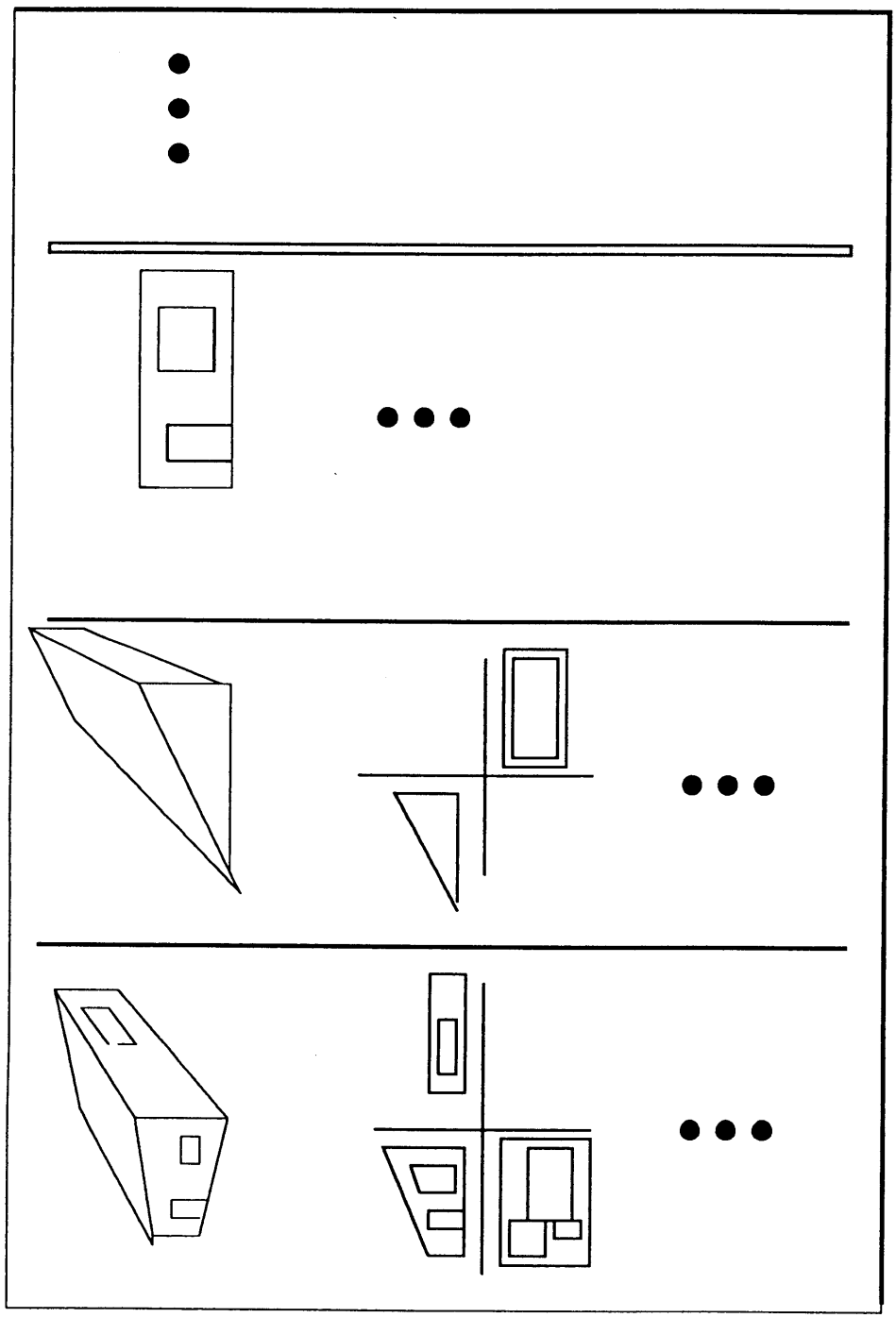


ABBILDUNG 4.4. Unterschiedliche Abstraktionsebenen und Hierarchien

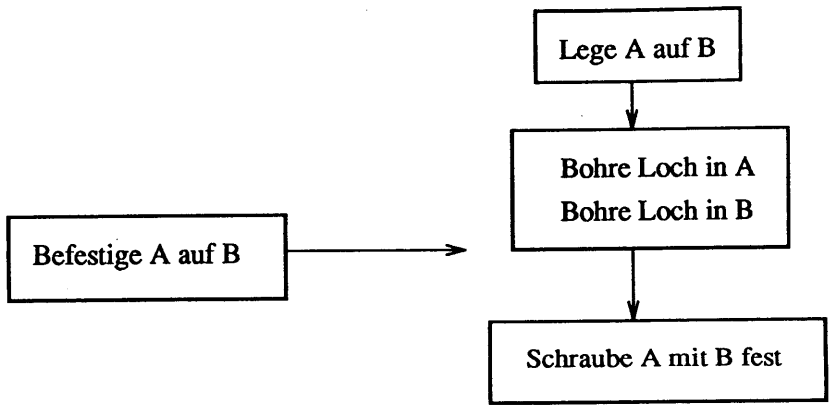


ABBILDUNG 4.5. Abstraktionshierarchien von Entwurfshandlungen

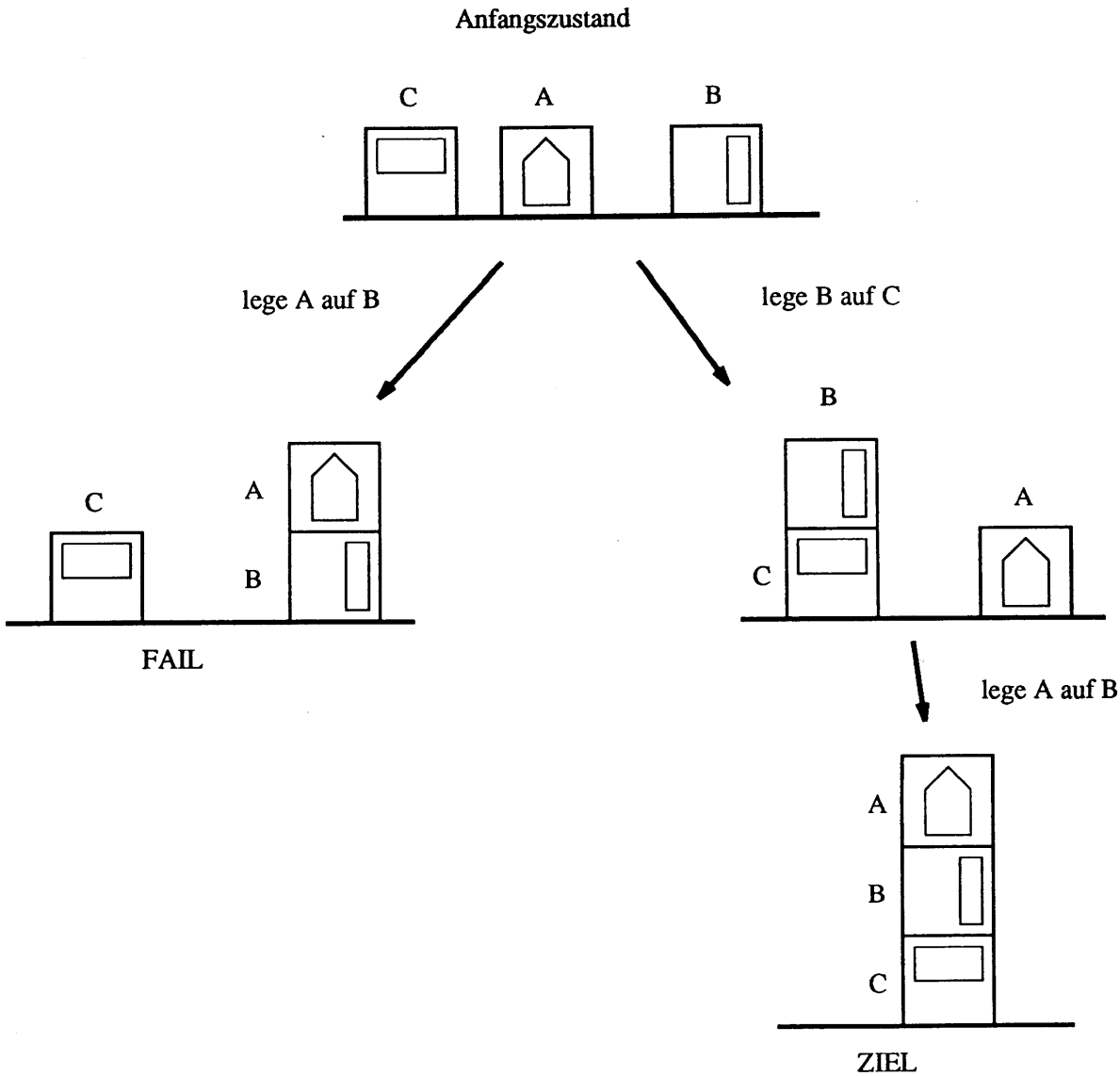


ABBILDUNG 4.6. Das Ziel wird in zwei Teilziele unterteilt. Die Reihenfolge der Handlungen erweist sich als wichtig. Mögliche Strategie für die Behebung ist die Umordnung der Teilziele.

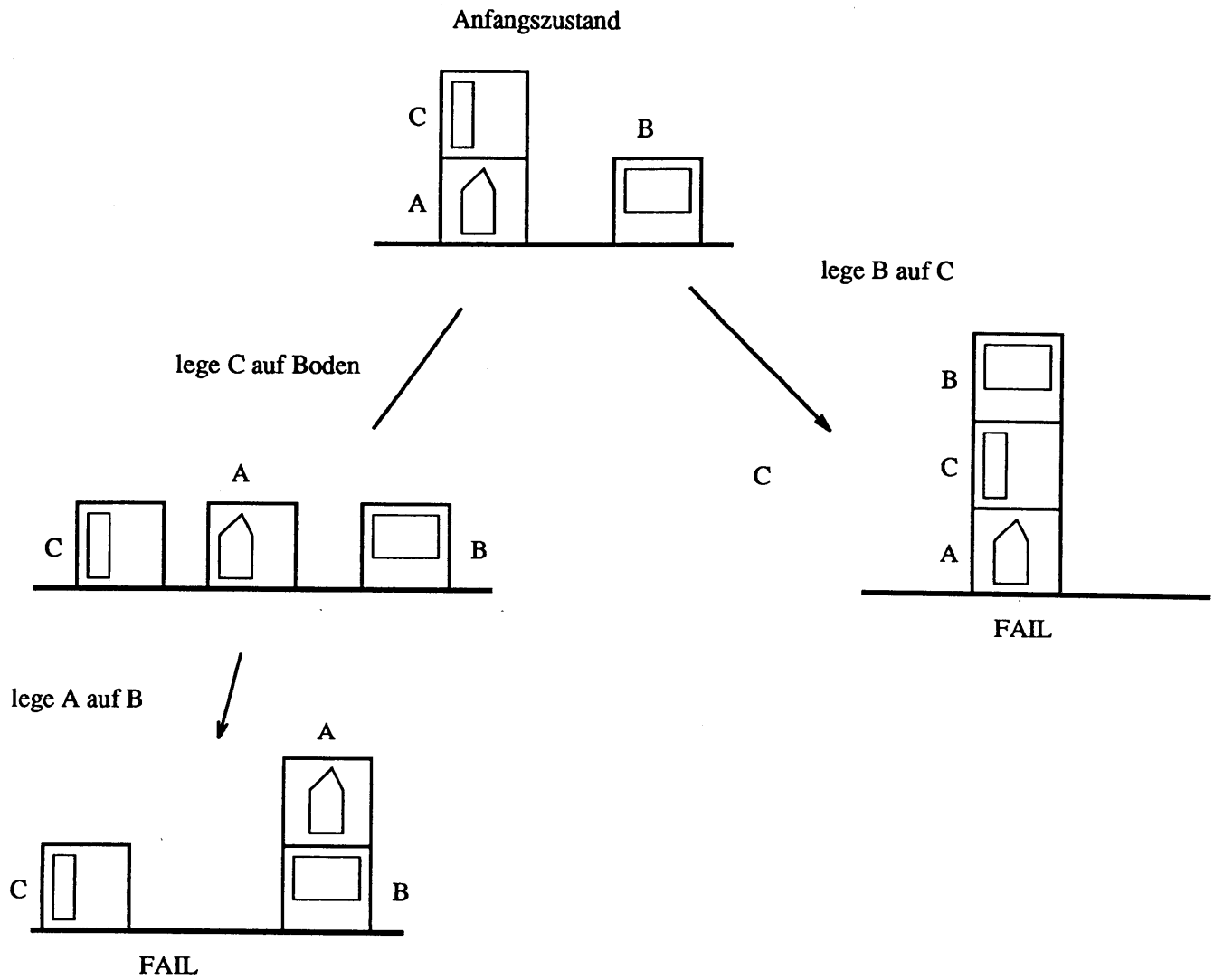


ABBILDUNG 4.7. Zwei Pfade, wo versucht wird, das Ziel zu erreichen. Beide versagen

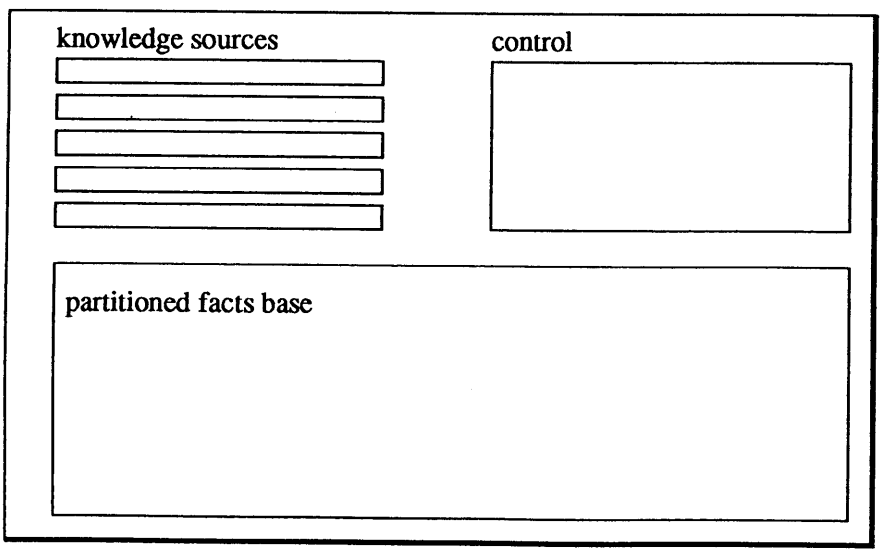


ABBILDUNG 4.8. Die wichtigsten Bestandteile eines blackboard systems

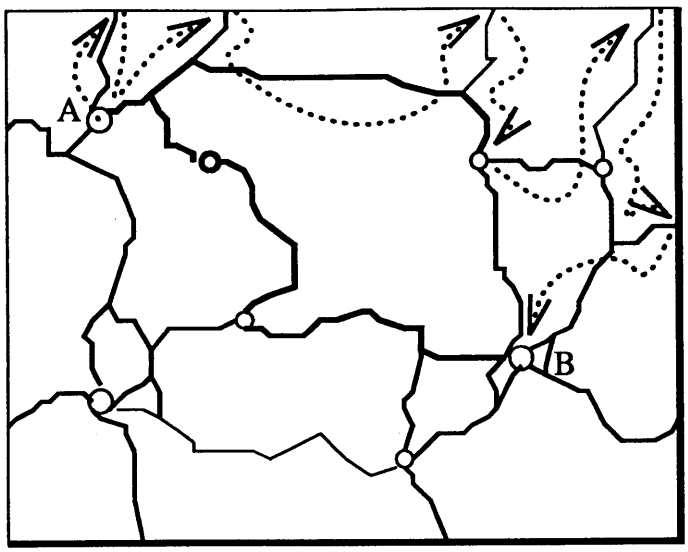


ABBILDUNG 4.9. Vorwärtsverketten der Autofahrt

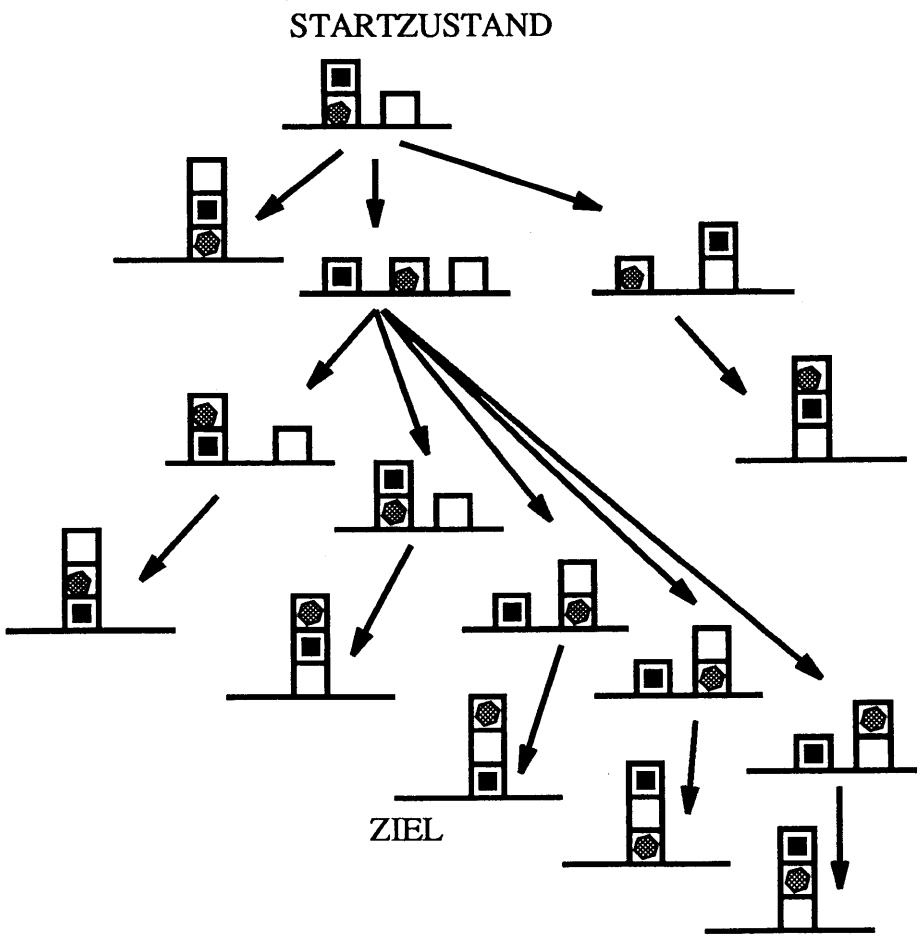


ABBILDUNG 4.10. Suchgraphen für das Blockproblem

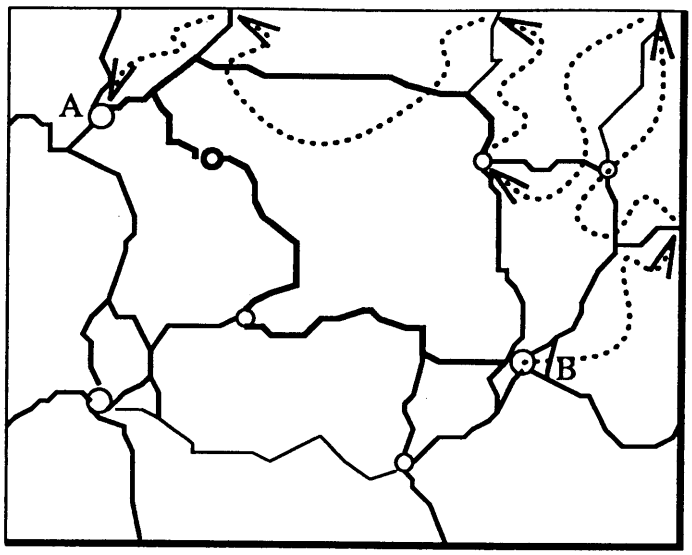


ABBILDUNG 4.11. Rückwärtsverketten der Autofahrt

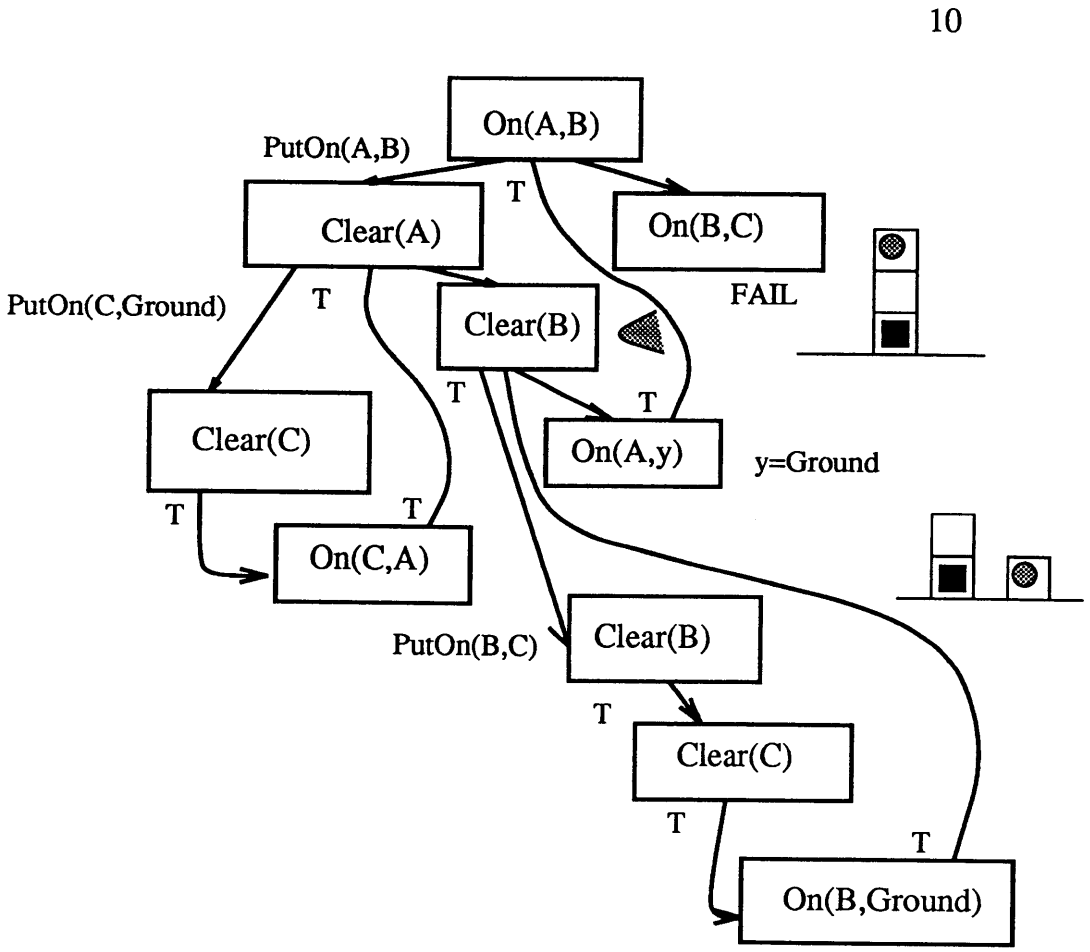


ABBILDUNG 4.12. Rückwärtsverkettung mit dem Blockproblem

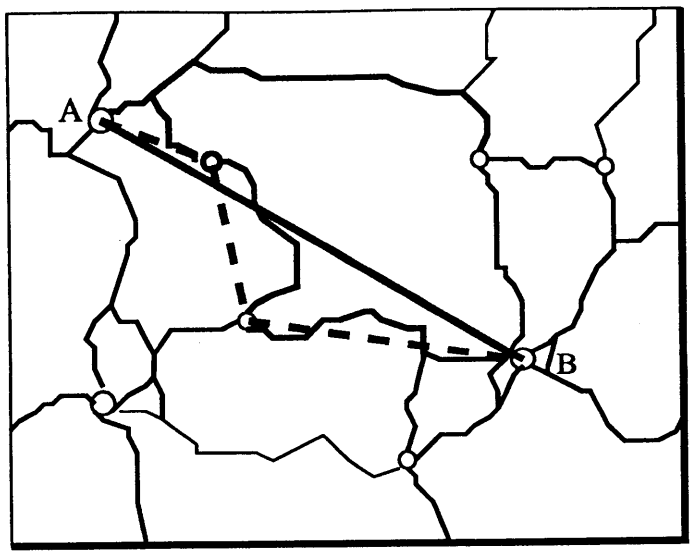


ABBILDUNG 4.13. Hierarchische Verfeinerung der Autofahrt

11

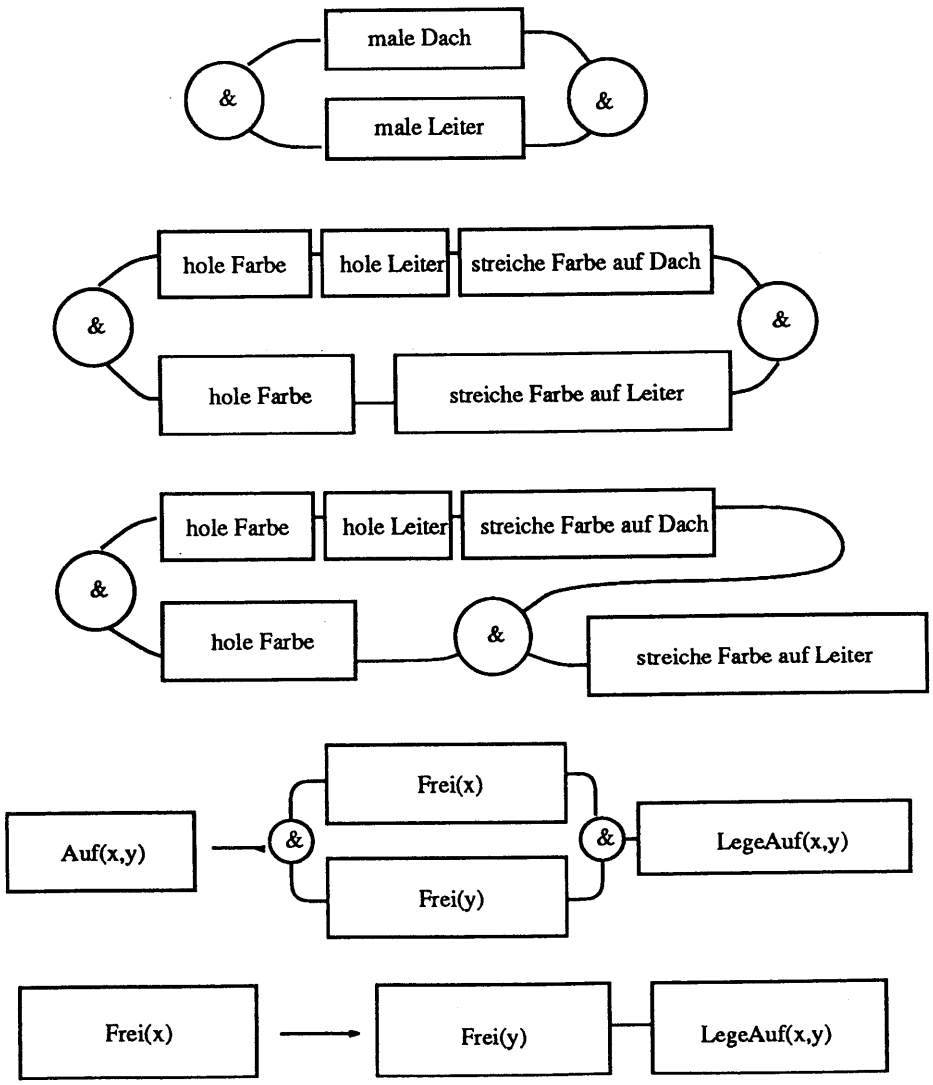


ABBILDUNG 4.14. Hierarchische Verfeinerung

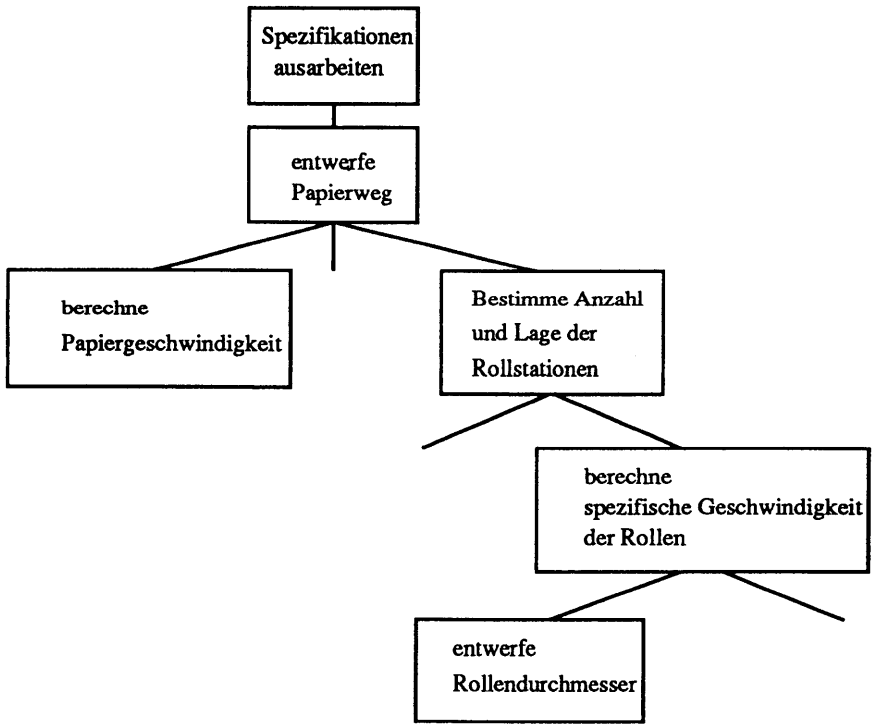


ABBILDUNG 4.15. Beispiel eines Decomposition Trees

5

Lernen und Kreativität in Designsystemen

Thomas Müller

ZUSAMMENFASSUNG Im folgenden Text sollen die Möglichkeiten der Einbringung von kreativen Gesichtspunkten und Lernfähigkeit in wissensbasierte Designsysteme betrachtet werden. Zunächst werden verschiedene Konzepte der Wissensaquisition für Interpretationswissen vorgestellt und eine Charakterisierung der einzelnen Einheiten eines solchen Lernsystems besprochen. Anschließend folgt eine Untersuchung von Generierungswissen mit grammatikalischem Aufbau. Alle weiteren Ansätze verfolgen prinzipiell beide Ziele, sowohl die Ableitung von Generierungs- wie auch von Interpretationswissen. Hier werden Schlußweisen über Ähnlichkeitsvergleiche und das Neuronale-Netz-Modell mit Anwendungen in wissensbasierten Designsystemen eingeführt. Danach wird versucht, den Begriff Kreativität für die Anwendung zu fassen. Dies geschieht jedoch eher auf kognitiver Ebene. Eine Umsetzung der gewonnen Erkentnisse auf die maschinelle Verarbeitung von Designaufgaben erweißt sich als schwierig, wodurch der Bezug zu wissenbasierten Designsystemen nicht so eng geknüpft werden kann.

5.1 Das Problem des Lernens im Designprozeß

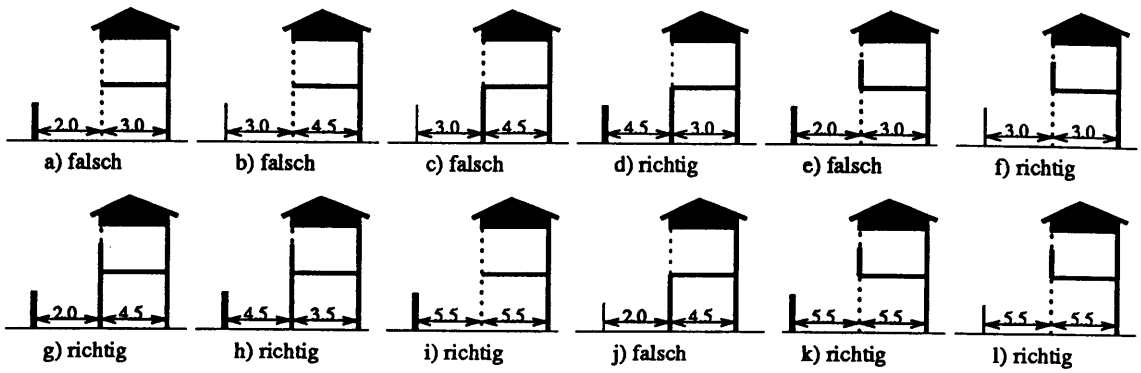
Da Lernen gewöhnlich in Verbindung mit intelligentem Handeln steht, kommt ihm eine Schlüsselrolle im kreativen Designprozeß zu. Mit fortschreitender Zeit verbessert ein Designer seine Fähigkeiten, und dies kann nicht vollständig mit dem bloßen Erwerb neuer Fakten beschrieben werden. Lernen verursacht eine Strukturveränderung im System: eine Veränderung sowohl im Kontrollfluß als auch in der Formulierung und Struktur der Ziele und Constraints.

Warum ist es überhaupt nötig ein Designsystem mit Lernkomponenten auszustatten?

Zunächst ist es sicherlich erwünscht, daß ein System seine Fähigkeiten mit fortschreitender Einsatzzeit steigert. Ein System ohne Lernmöglichkeiten wird nur in der Lage sein auf solche Situationen zu reagieren, für die es kreiert wurde bzw. Wissen besitzt. In ähnlichen aber ungleichen Situationen wird es scheitern. Gerade beim Designen von real-life Objekten ist es erwünscht explizites Wissen zur Konstruktion vieler verschiedener Objekte zur Verfügung zu haben. Es wird also gefordert aus speziellem Wissen generelle Prinzipien abzuleiten. Zudem ist die Verarbeitung und Speicherung von speziellem Wissen sehr zeit- und resourcenintensiv im Vergleich zu generalisiertem Wissen.da ein Designer während des Designprozesses hinzulernt sollte das System in der Lage sein sich anzupassen, um eine vernüftige Unterstützung gewährleisten zu können.

Was verseht man unter dem Begriff Lernen?

Wann kann überhaupt von einem intelligenten Lernverhalten gesprochen werden? Es wird z.B. als intelligent bezeichnet, wenn ein Mensch in dem Sinne aus einer schlechten Erfahrung gelernt hat, daß er eine Wiederholung dieser oder ähnlicher Handlungen unterläßt. Ein Lernprozeß scheint also Generalisierung und Abstraktion zu enthalten. Wir extrahieren nicht die Vorgänge oder Besonderheiten einer Situation, sondern die generalsierten, abstrakten Konzepte, um sie in einer artverwandten Begebenheit zur Anwendung zu bringen. In den meisten Fällen kann die Generalisierung aber nicht mit absoluter Sicherheit angenommen bzw. überprüft werden. In einem lernfähigen System stellt der Wissenskörper nur eine Hypothese dar, die mit fortschreitendem Prozeß weiterentwickelt und verfeinert wird. In der Theorie ist ein solches Vorgehen, das Schließen aus einigen Beispielen inakzeptabel. Der Mensch macht dies sehr oft. Er geht von der Annahme aus, das abgeleitete Wissen sei korrekt, solange ihn nicht eine neue Erfahrung vom Gegenteil überzeugt.



Lernmenge:

Design-Nr.:	a)	b)	c)	d)	e)	f)	g)	h)	i)	j)	k)	l)
Fenster oben	gross	gross	gross	gross	klein	klein	klein	klein	gross	gross	klein	klein
Fenster unten	gross	gross	nicht	nicht	gross	gross	nicht	nicht	gross	nicht	gross	gross
Abstand	2.0	3.0	3.0	4.5	2.0	3.0	2.0	4.5	5.5	2.0	5.5	5.5
Raumgroesse	3.0	4.5	4.5	3.0	3.0	3.0	4.5	3.5	5.5	4.5	5.5	5.5
Mauermaterial	Stein	Holz	Holz	Stein	Stein	Holz	Stein	Stein	Stein	Holz	Stein	Holz
Klasse	falsch	falsch	falsch	richtig	falsch	richtig	richtig	richtig	richtig	falsch	richtig	richtig

ABBILDUNG 5.1. Lernmenge für Grenzbebauungsprobleme

5.2 Wissensakquisition

Das folgende Verfahren zur Aquisition von Wissen kann allgemein eingesetzt werden und ist unabhängig von Lernvorgängen in Designsystemen. Man kann sich auf eine Designbeschreibung aus positiven und negativen Beispieldesigns als Ausgangsmenge beschränken. Die Aufgabe besteht nun darin, eine Menge von Interpretationswissen abzuleiten, die einen Design-Space definiert, der alle positiven Beispiele als Teilmenge enthält und alle Negativen ausschließt.

5.2.1 Lernen durch sukzessive Unterscheidung von Attributen

Ein einfacher Algorithmus zur Generalisierung ist unter dem Titel Concept Learning System (CLS) bekannt. Er soll an einem Beipiel erläutert werden: Bau- und Grenzbestimmungen beim Hausbau. Die Lernmenge umfaßt einige positive und negative Beispiele bestehender Baupläne. Die Aufgabe des Lernsystems ist es nun, die wesentlichen Gesichtspunkte d.h. Parameter des Designs zu bestimmen und abzugrenzen, welche darüber entscheiden, ob ein Plan akzepiert werden kann oder die Bestimmungen verletzt. Es sollen Regeln abgeleitet werden, mit denen sich die Richtigkeit eines Bauplans überprüfen läßt. Dem Lernsystem stehen als Eingabe die 5 Attribute mit ihren Werten für jedes Design, daß in die Klassen richtig oder falsch eingeordnet ist, zur Verfügung (Figur 5.1).

Der CLS-Algorithmus produziert nicht direkt abgeleitete Regeln, sondern einen Entscheidungsbaum, dessen Blätter die Aussage richtig oder falsch bezeichnen, dessen Knoten das für die Entscheidung relevante Attribut enthalten und dessen Kanten mit den zugehörigen Attributwerten markiert sind (Figur 5.2).

Die grundlegende Idee des Algorithmus ist sukzessive Unterteilungen der Eingabemenge gemäß einem bestimmten Attribut vorzunehmen, bis der Fall eintritt, daß alle Beispiele nur noch einer Klasse (hier die Klassen: richtige oder falsche Pläne) angehören. Figur 5.3 gibt einen Überblick wie der Entscheidungsbaum entsteht. Betrachtet man die Lernmenge unterteilt in Klassen, so stellt man fest, daß alle Designs mit einem Grenzabstand > 3.0m richtig sind. Dies ist die Basis für die erste Unterteilung. Man unterteilt die Lernmenge in eine Menge von Plänen, deren Grenzabstand > 3.0m und eine deren Abstand <= 3.0m beträgt. Die Pläne der ersten Menge sind bereits abgearbeitet, da hier der Entscheidungsbaum auf einem Blatt 'richtig' endet; die Unterteilung wurde gerade so

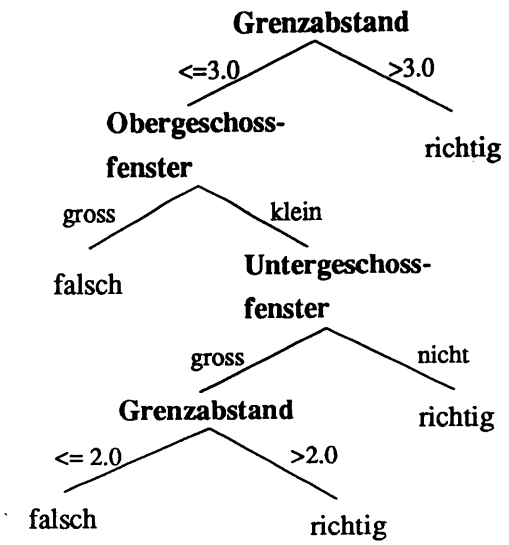


ABBILDUNG 5.2. Der fertige Entscheidungsbaum

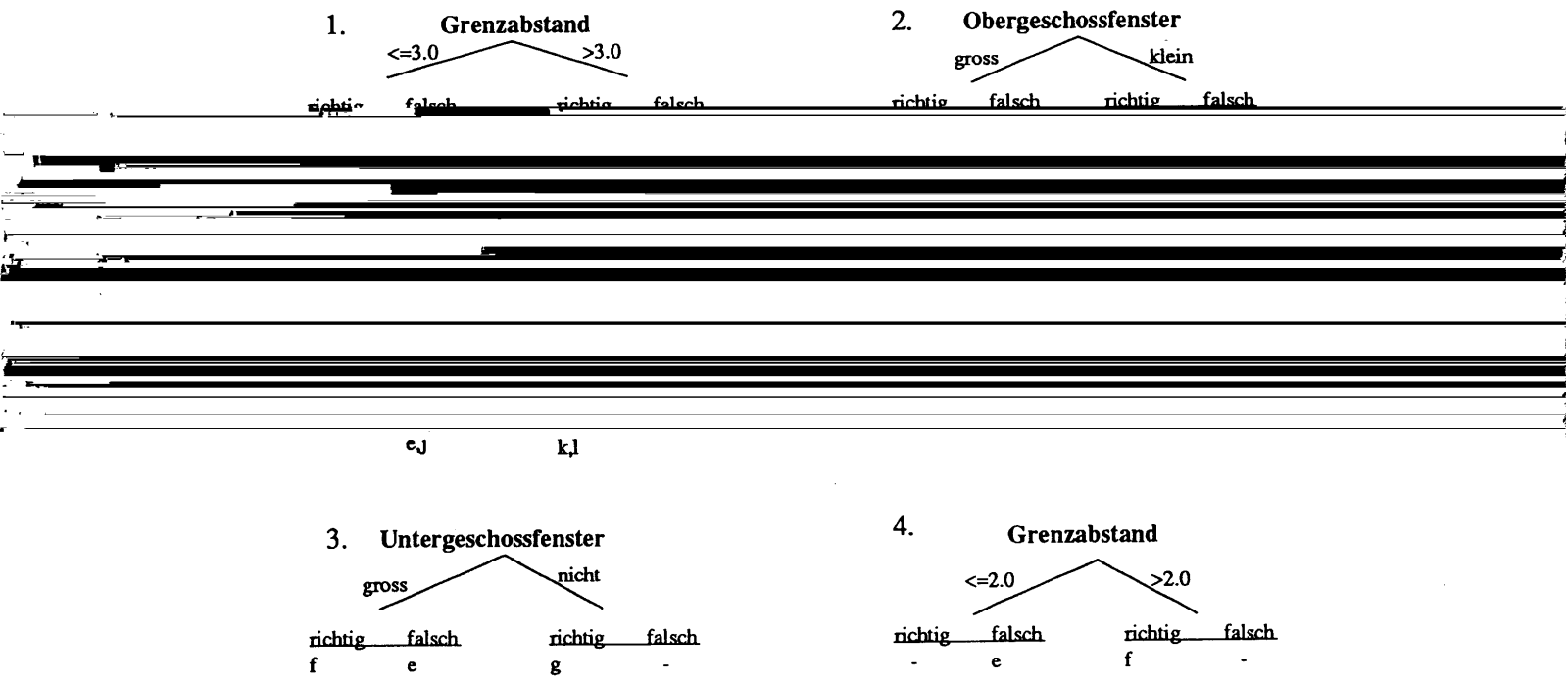


ABBILDUNG 5.3. Die Entwicklung des Entscheidungsbaumes

gewählt.

Der Algorithmus arbeitet nun an der Komplementärmenge weiter, da sie noch Pläne beider Klassen (richtige und falsche) enthält. Der Algorithmus sucht nach einem weiteren Attribut, anhand dessen die Menge sich abermals in eine mit eindeutiger Klassenzugehörigkeit und eine weiter abzuarbeitende Komplementärmenge teilen läßt. Ein solches Attribut ist z.B. die Größe des Obergeschoßfensters. Der Algorithmus endet, wenn alle verbleibenden Pläne nur noch einer Klasse angehören. Der Entscheidungsbaum ist somit vollständig und läßt sich sehr leicht in einfache Regeln umwandeln.

Schwierigkeiten des CLS-Algorithmus:

Die größte Schwierigkeit beim CLS besteht darin zu entscheiden, auf welcher Basis d.h. nach welchem Attribut die Unterteilungen getroffen werden sollen. Gefordert ist natürlich der kleinste Baum. Offenbar sind auch nicht immer alle Attribute relevant. In obigen Beispiel hat das Mauermaterial und die Raumgröße keinen Einfluß auf die Korrektheit eines Planes. Es kann vorkommen, daß der Entscheidungsbaum ebensoviele Knoten hat, wie die Lernmenge Beispiele enthält, dann hat sicherlich keine Generalisierung stattgefunden. Es zeigt sich also, daß die wesentliche Aufgabe von einer, dem Problem angepaßten, ausgefeilten Heuristik zu leisten ist. Ein weiterer Schwierigkeit liegt in der Menge der Attribute, die dem System vorgegeben werden. Ist im Extremfall das entscheidende Attribut nicht in der vorgegebenen Attributmenge, so liefert der CLS ein Ergebnis, das über den Rahmen der Beispielmenge hinaus nicht genutzt werden kann. Gleiches gilt natürlich für den Fall, daß sich ein falsch bewertetes Beispiel in der Lernmenge befindet. Der CLS-Algorithmus ist nicht in der Lage Beziehungen zwischen Attributen aufzedecken, z.B. die Raumlänge muß doppelt so groß sein wie der Grenzabstand. Außerdem erzeugt der Algorithmus kein Wissen, daß quasi zwischen den Attributen und Klassen steht. Es wäre etwa sinnvoll zu wissen, daß ein kleines Obergeschoßfenster Blickdiskretion, oder ein großer Grenzabstand, Abstandsdiskretion bedeutet. Diese Information würde das abgeleitete Wissen lesbarer machen. Um solches zu produzieren wären allerdings erweiterte Methoden und weiteres Hintergrundwissen notwendig.

5.2.2 Lernen mit Hilfe von Generalisierungswissen

Wenn wir mit Interpretationswissen arbeiten, kann die Lernaufgabe dadurch charakterisiert werden, daß Wissen produziert bzw. abgleitet wird, mit dem verschiedene Interpretationen erzeugt werden können. Es ist möglich ein einfaches Lernsystem für Interpretationswissen als ein System von Produktionen anzusehen, dessen Komponenten wie folgt charakterisiert werden können:

- Lernmenge: Nutzbar sind solche Beispiele, für die Interpretationen existieren. Im einfachesten Fall läßt sich die Beispielmenge in zwei Klassen aufteilen (siehe vorheriges Beispiel).
- Konzepte: Beim Ableiten von Interpretationswissen ist es nötig sich auf bestimmte Interpretationen festzulegen. Diese Interpretationen werden auch als Konzepte bezeichnet, äquivalent zu den Klassen beim CLS-Algorithmus.
- Beschreibungssprache: Beispiele der Lernmenge und evt. auch Teile des Systemwissens werden in Termen einer bestimmten Beschreibungssprache ausgedrückt.
- Hintergrundwissen: Mit diesem Wissen werden zusätzliche Designbeschreibungen (unabhängig von den Konzepten) aus dem Design abgeleitet. Diese so gewonnene Gesamtmenge der Designbeschreibungen wird auch als Fakten über das Design bezeichnet.
- Hypothese: Die Hypothese stellt eine Verbindung zwischen den Interpretationen und Fakten des Designs her, normalerweise in Form von Regeln. Zu Begin ist die Hypothese leer und entwickelt sich im Laufe des Lernprozesses.
- Generalisierungswissen: Dieses Wissen verursacht die Transformation und Entwicklung der Hypothese, üblicherweise ausgedrückt in Form von Lernregeln.
- Kontrollwissen: Dieses Wissen kontrolliert die Suche nach einer Hypothese im Hypothesenraum d.h. die Anwendung und den Einsatz des Generalisierungswissens. Die Form und der Wert der Endhypothese hängt wesentlich von der Anwendung der Generalisierungsregeln ab

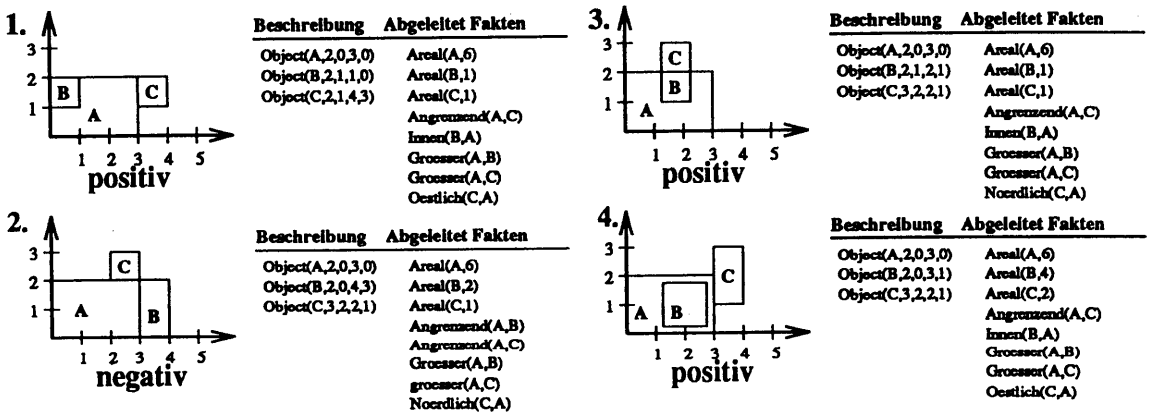


ABBILDUNG 5.4. Lernmenge mit Beschreibung und weiteren abgeleiteten Regeln

und der Hypothesenraum ist oftmals so groß, daß zusätzliches Wissen benötigt wird um das Lernverfahren effizient zu machen.

Das folgende Beispiel soll diese Klassifizierung illustrieren: Figur 5.4 stellt die Lernmenge dar. Das System soll nun herausfinden, was die positiven Beispiele auszeichnet und die Negativen abgrenzt. Die Beispiele sind in Termen von Objekten beschrieben, um zusätzliche Fakten abzuleiten, steht dem System Hintergrundwissen zur Verfügung. Die folgenden Regeln stellen das Generalisierungswissen dar:

REGEL1: IF [die Hypothese ist leer]

THEN [nehme das erste Beispiel als Starthypothese]

REGEL2: IF [es liegt ein Statment im positiven Beispiel vor, daß einer Aussage der Hypothese widerspricht]

THEN [lösche das Statement oder seine Negation aus der Hypothese]

REGEL3: IF [zwei Statements enthalten verschiedene Werte für das gleiche Attribut]

THEN [die Statments widersprechensich] (siehe Regel 2)

REGEL4: IF [es liegt ein Statment in der Hypothese vor, das gleich ist zu einem Stament in einem negativen Beispiel]

THEN [lösche dieses Statmentaus der Hypothese]

REGEL5: IF [Ein Attribut tritt sowohl im positiven Beispiel als auch in der Hypothese auf und ist gleich bis auf einige Werte oder Objekte]

THEN [Ändere die Namen dieser Werte oder Objekte in der Hypothese zu Variablen]

Diese Regeln sind eher heuristischer Natur und unvollständig. In diesem einfachen Beispiel mögen sie ausreichen, aber bei einer modifizierten Lernmenge können sie fehlschlagen. Zusätzlich könnte man noch Meta-Regeln (Kontrollwissen) angeben, die die Anwendung obiger Regeln steuern, dies ist jedoch in diesem einfachen Beispiel nicht nötig. Zunächst werden mit Hilfe des Hintergrundwissens für alle Beispiele weitere Fakten erzeugt und nachdem Regel 1 auf das erste Beispiel angewendet wurde ergibt sich folgendes Bild:

Object(A,2,0,3,0), Object(B,2,1,1,0), Object(C,2,1,4,3)
Areal(A,6), Östlich(C,A), Angrenzend(A,C), Innen (B,A)
Areal(B,1), Östlich(C,B), Areal(C,1)
Grösser(A,B), Grösser(A,C)

Wird Beispiel 2 eingegeben, so feuert Regel4 und es ergibt sich:

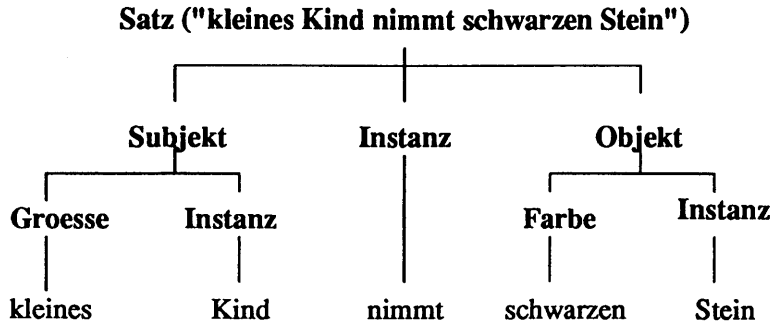


ABBILDUNG 5.5. Zerlegung eines einfachen Satzes

Object(B,2,1,1,0), Object(C,2,1,4,3)
Östlich(C,A), Innen(B,A)
Areal(B,1), Östlich(C,B)

Wird Beispiel 3 eingegeben, so feuert Regel2:
Areal(B,1), Innen(B,A)

Nach Eingabe von Beispiel 4 und feuern von Regel2:
Innen(B,A)

Dieses Faktum beschreibt die positiven Beispiele und schließt die Negativen aus und resultiert in der Regel: IF [Innen(B,A)] THEN [Positive(Beispiel)]. Diese Regel enthält keine Variablen; um Regel5 zum feuern zu bringen müßten wir z.B. ein weiteres positives Bsp. haben, welches ein Fakt Innen(X,A) für X verschieden von B.

Dieses Beispiel soll zeigen, wie Interpretationswissen eine Hypothese formt. Eine andere Menge von Lernwissen kann andere Regeln erzeugen, die ebenso gelten. Außerdem spielt die Reihenfolge, in der Beispiele eingegeben werden eine entscheidende Rolle.

5.3 Ableiten von syntaktischen Wissen

Im allgemeinen wird syntaktisches Wissen durch einen Lernprozeß aus Beispielen erzeugt (z.B. aus Sätzen einer Sprache wird eine Grammatik abgeleitet). Dieser Prozeß wirft aber einige Schwierigkeiten auf. Bei der Generierung durch eine Grammatik können gleiche Sätze (bzw. Designs) aus verschiedenen Sequenzen und verschiedenen Mengen von Aktionen entstehen. Umgekehrt heißt dies: Beim Erzeugen grammatikalischer Regeln aus Designs müssen wir festlegen, wie diese Designs geparsed werden sollen, d.h. wir müssen die Reihenfolge der Aktionen festlegen. Eine Vorgehensweise verfolgt einen Ansatz mit Ähnlichkeit zum kognitiven Verfahren. Ein Satz wird in der geichen Weise 'geparsed' wie er vom Menschen verstanden wird, d.h. in der gleichen Reihenfolge wie ein Mensch einen Satz in immer kleinere Untereinheiten zerlegt. Figur 5.5 zeigt, wie ein einfacher Satz zerlegt wird.

Wie gestaltet sich der Lernprozeß beim Menschen?

Der Grammatikableitungsprozeß wird bei Kindern zunächst sehr einfach gemacht, indem nur leicht zerlegbare Sätze gebildet werden. Nomen sind Objekte und Prädikate sind Verben. Die Muster Subjekt, Prädikat, Objekt sind einfache Kategorien, die von einfachen Sätzen generalisiert wurden. Der Prozeß ist ähnlich dem der Bildung einer Hypothese durch Generalisierungs-Regeln. Komplexere Sätze werden langsam eingeführt, damit Schrittweise eine Verfeinerung der Kategorien erfolgen kann. Ein Fehler bzw. eine Berichtigung führt zur Verwerfung der alten Hypothese und der Bildung einer Neuen.

Normalerweise resultieren viele Aktionssequenzen im einem einzigen Design. Man kann [illegible]

Raumplan dadurch erzeugen, daß man mit einem großen Raum beginnt und ihn in immer kleinere Einzelräume unterteilt, oder dadurch, daß man mit einem einzelnen kleinen Raum beginnt und andere Räume seitlich anbaut. Das Resultat beider Prozesse kann das gleiche sein. Es sollte also Wissen in das System einbracht werden, daß eine Kontrollfunktion übernimmt und die Anzahl der Möglichkeiten begrenzt.

Zur Verdeutlichung des Generierungs-Prozesses folgendes Beispiel aus dem Designbereich für die Unterteilung eines Hauses:

1. Starte mit der kompletten Hauszone.
2. Teile die Hauszone in eine Esszone und eine Wohnzone.
3. Teile die Eßzone in Küche und Esszimmer.
4. Ersetze die Wohnzone durch ein Wohnzimmer.

Zonen sind die syntaktischen Kategorien in Sprachgrammatiken (z.B. Verphrase, Nomenphrase usw.) während Räume wie Küche oder Wohnzimmer Terminalwörter sind. Ein legales Design darf nur aus Terminalwörtern bestehen. In diesem einfachen Beispiel gibt es nicht viele Möglichkeiten, wie ein Design geparsed werden kann, aber in Komplizierteren sind ausgefeilte Heuristiken nötig, die die Anzahl der möglichen Parse-Trees beschränkt.

5.4 Schließen über Ähnlichkeitsvergleiche in Designsystemen

Bei diesem alternativen Ansatz ist die Vorgehensweise die, daß man die Repräsentationen der eingegebenen Beispiele aus der Lernmenge speichert, um aus ihnen später über Ähnlichkeitsvergleiche Schlüsse zu ziehen. Ein Vorteil dieser Methode ist, daß nicht wie bei den vorherigen Ansätzen durch das Verwerfen der Lernmenge Informationen verloren gehen können. Es sollen hier zwei Ansätze verfolgt werden: Im ersten soll mit gespeicherten Design-Beschreibungen Übereinstimmungen gesucht werden. Im zweiten Ansatz soll aus gespeicherten Beschreibungen von Entscheidungsvorgängen während früherer Designs-Prozesse geschlossen werden.

5.4.1 Schliessen aus Design-Beispielen

Auf dieser Basis wurde ein System zur Erzeugung der korrekten Aussprache von Wörtern entwickelt. Das System setzt eine ausgefeilete Metrik ein, um z.B. zu erkennen, daß das "g" in "gypsum" ähnlicher zu dem "g" in "gypsy" ist als zu dem "g" in "guzzle" Die Metrik beachtet dabei nicht nur den einzelnen Buchstaben sondern auch die anderen Buchstaben, ihre Übereinstimmungen und ihre Lage zueinander. Der Vorteil des Systems besteht darin, daß seine Leistungsfähigkeit durch hinzufügen einzelner selbst eingeordneter Wörter immer weiter steigt. Unglücklicherweise gibt es keine Universal-Metrik für String-matching Probleme, sondern gerade in solchen Fällen muß eine sehr speziell ausgewählte Metrik konzipiert werden.

Die hier behandelte Vorgehensweise mit Ähnlichkeitsschlüssen läßt sich nicht nur an solch einfachen Beispielen einsetzen, sondern auch ganz allgemein ausweiten. Neuronale Netzwerke oder PDP-Modelle basieren auf dem mathematischen Modell neuronaler Vorgänge. Diese Modelle beruhen ebenfalls auf einer Speicherung vergangener Vorgänge und in einem späteren matching in einer geschlosseneren Form. Davon mehr im nächsten Abschnitt.

5.4.2 Schliessen aufgrund von Entscheidungserklärungen früherer Designprozesse

Die hier verfolgte Idee wurde erstmals 1987 in einem System mit dem Namen CYCLOPS implementiert. Als Anwendungsbeispiel wurde die Konstruktion eines Hauses an einem Abhang gewählt. Dies wirft einige Schwierigkeiten auf, die z.B. dadurch gelöst werden können, daß das Haus auf Stützen errichtet oder Terassen aufgeschüttet werden. Für diesen Anwendungsbereich wurde das System lediglich mit Anfängerwissen ausgestattet, jedoch ein Zugang zu der Wissensbasis aus anderen Bereichen ermöglicht, wie z.B. Wissen über den Bau einer Thai-Siedlung in einem überfluteten

Gebieten. Diese Wissensbasis war aus Erklärungungen über die Vorgehensweise in füheren Designbereichen ausgestattet, in Form folgender Regeln:

- IF [Baugrund ist überflutet]
  THEN [Baugrund ungeeignet]
- IF [Die Hütte soll auf einem bestimmten Baugrund errichtet werden und dieser Boden ist ungeeignet]
  THEN [Stelle die Hütte auf Pfähle]

Will nun das System oben angesprochenes Problem lösen, so kommt es an einen Punkt, an dem das Anfängerwissen keine weitere Aktion zur Verfügung stellt. Um Abhilfe zu schaffen kann das System über den Vorgang des analogen Schließens zwei Wege bestreiten: Zunächst suche nach analogen Situationen, die auf unsere Objekte angewendet werden können. Dies führt hier jedoch nicht zum Ziel. Als nächstes Suche nach analogen Objekten (die Thai Hütten werden als analog zu dem Haus erkannt), um anschließend bei dem ersten Punkt, dem Suchen nach analogen Situationen wieder aufsetzen zu können. Ist im Anfängerwissen etwa folgenden Regel gespeichert:
IF [Der Baugrund liegt an einem Abhang] THEN [der Baugrund ist ungeeignet]
so ist das System in der Lage die Schlußkette bei obiger Regel 2 weiterzuführen und eine Lösung für das Problem zu finden. Der besprochene Vorgang, das Schließen aufgrund von Entscheidungserklärungen, ist letztendlich eine Suche nach Übereinstimmugen in Interferenzbäumen.

5.5 Das Neuronen-Netzwerkmodell

Im neuronalen Netzwerk-Modell ist das Wissen über Designs als pattern repräsentiert und als Gewichte und Schwellwerte in einem neuronalen Netz gespeichert. Netzwerkknoten werden als Units bezeichnet. Input-units, output-units und hidden-units werden über gerichtete Kanten/Verbindungen verbunden. Input-units empfangen die Information, output-units senden sie nach außen, hidden units empfangen und senden nur innerhalb des Netzes. Die Parameter des Netzwerks sind die Gewichte der Kanten, die Schwellwerte der Knoten bzw. Units und die Input- und Output-Werte. Die Ausgabefunktion jeder einzelnen Einheit (Wert 1 oder 0) ist die Summe der gewichteten Ausgangswerte der einzelnen angeschlossenen Vorgänger-Units. Das Neuron feuert (Ausgang '1'), falls der Wert der Ausgangsfunktion gleich oder größer als der Schwellwert ist, ansonsten nicht (Ausgang '0'). Diese Operation wird durch das gesamte Netzwerk von den Eingängen zu den Ausgängen propagiert. Lernen bedeutet in diesem Zusammenhang die automatische Anpassung der Gewichte und Schwellwerte für eine Menge von geg. Input/Output-Pattern-Paaren (Lernmenge). Nach der Lernphase soll das Netz in der Lage sein, alle Eingabepattern auf die zugehörigen Ausgabepattern zu reproduzieren.

5.5.1 Ein einfaches Modell der Pattern-Assoziation

Das Neuronen-Netzwerkmodell soll zunächst an einem einfachen Beispiel ohne hidden-units verdeutlicht werden. Das Netzwerk besteht aus 3 Eingabeneuronen und einem Ausgabeneuron. Im Initialzustand sollen alle Gewichte mit '1' und der Ausgabeschwellwert mit '0' belegt sein. (Figur 5.6a) Man sieht leicht, daß dieses Netz immer dann '1' am Ausgang liefert, wenn mindestes ein Eingang mit '1' belegt ist. Die geforderte Aufgabe besteht nun darin, das Netzwerk so zu trainieren, daß es z.B. bei einem Eingabepattern von (0 1 0) eine '0' am Ausgang liefert. Folgender einfacher Algorithmus bewältigt Probleme dieser Art:

1. Berechne zunächt den Ausgangs-Istwert für das zu erlenende Eingabepattern
2. Vergleiche mit dem Ausgabesollwert
3. Führe folgende Operation für alle Eingänge durch:
   (a) IF [Eingabewert '1' und Ausgabewert '1' und Sollwert '0']
   THEN [subtrahiere 1 von dem Gewicht zugehörigen Kante]

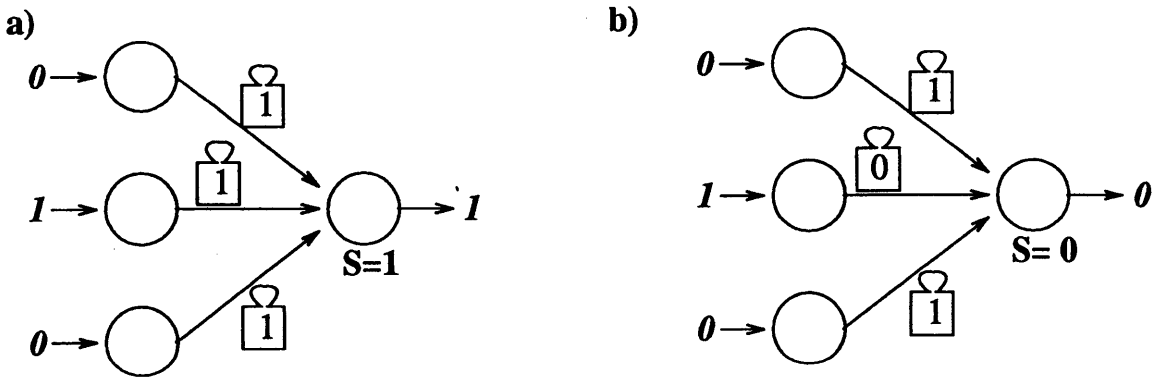


ABBILDUNG 5.6. Einfaches neuronales Netz

(b) IF [Eingabewert '1' und Ausgabewert '0' und Sollwert '1']
THEN [addiere 1 zu dem Gewicht zugehörigen Kante]

Nach Anwendung des Lernalgorithmus ergibt sich Figur 5.6b.
Beim Betrieb eines neuronalen Netzes kann man zwei Phasen unterscheiden:

1. Die Lernphase, in der dem System Paare von Input/Output-Pattern vorgegeben und die Gewichte und Schwellwerte nach einem bestimmten Lernalgorithmus justiert werden.
2. Die Simulations- oder Matchingphase. Hier werden dem System Input-Pattern vorgegeben und man erwartet eine Ausgabe, quasi die Betriebsphase.

Die Lernphase gestaltet sich normalerweise so, daß dem Netzwerk immer wieder die in der Reihenfolge permutierte Lernmenge vorgegeben wird, da eine Veränderung der Gewichte und Schwellwerte zu einem bestimmten Zeitpunkt wieder Teile des vorher Erlernten vergessen läßt. Ein System, daß nur aus Input- und Output-Units besteht ist nicht besonders leistungsfähig. Eine Vielzahl von Lernproblemen sind sogar damit prinzipell nicht erlernbar. Durch zusätzliche Einbringung von Hidden-Neuronen werden auch solche Lernprobleme bewältigbar, da die Anzahl der Parameter im System steigt. Allerdings gestalten sich die Lernalgorithmen wesentlich schwieriger. (z.B. Backpropagation-Algorithmen)

5.5.2 Pattern-Vervollständigung

Interpretiert man die Pattern, die das neuronale Netz verarbeiten soll, nicht als eindimensinal sondern zweidimensonal, so gelangt man zu einer Darstellung die für den Designbereich interessant wird. Man kann sich zweidimensionale Pattern als auf einem Gitter aufgetragene Designentwürfe vorstellen. In der vorangegangen Diskussion wurden Lernpattern als Paare von Input- und Output-Pattern aufgefaßt. Gibt man diesen Ansatz auf und läßt Input- und Output-Units zusammenfallen, so leistet ein neuronales Netz während der Simulationsphase eine Pattern-Vervollständigung. Figur 5.8 demonstriert dies. 9 Pattern werden dem System vorgegeben (Figur 5.7). Nach der Lernphase vervollständigt das System ein vorgegebenes Teilpattern. Zunächst wird nur ein unvollständiges Bild des Pattern mit Rauschanteil geliefert. Aber nach einigen Rückkopplungen erhält man das Originalbild (Figur 5.8)

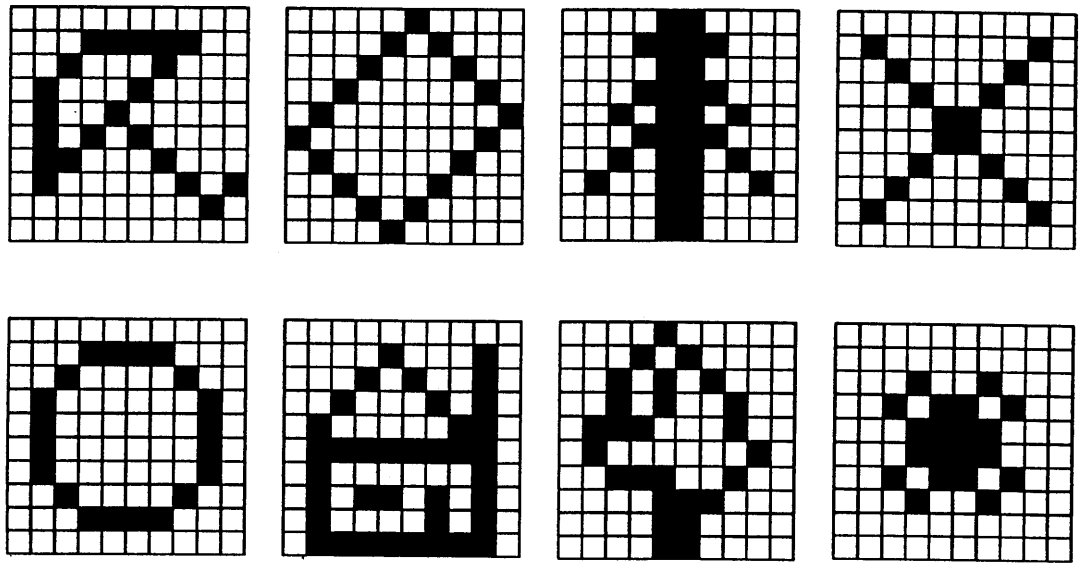


ABBILDUNG 5.7. Lernmenge von Pattern

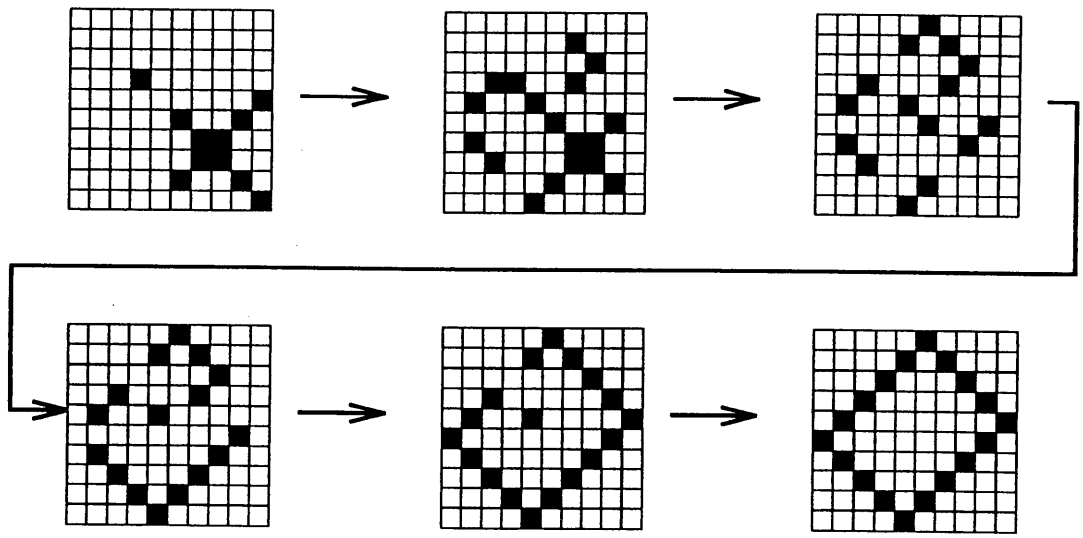


ABBILDUNG 5.8. Beispiel für eine Pattern-Vervollständigung

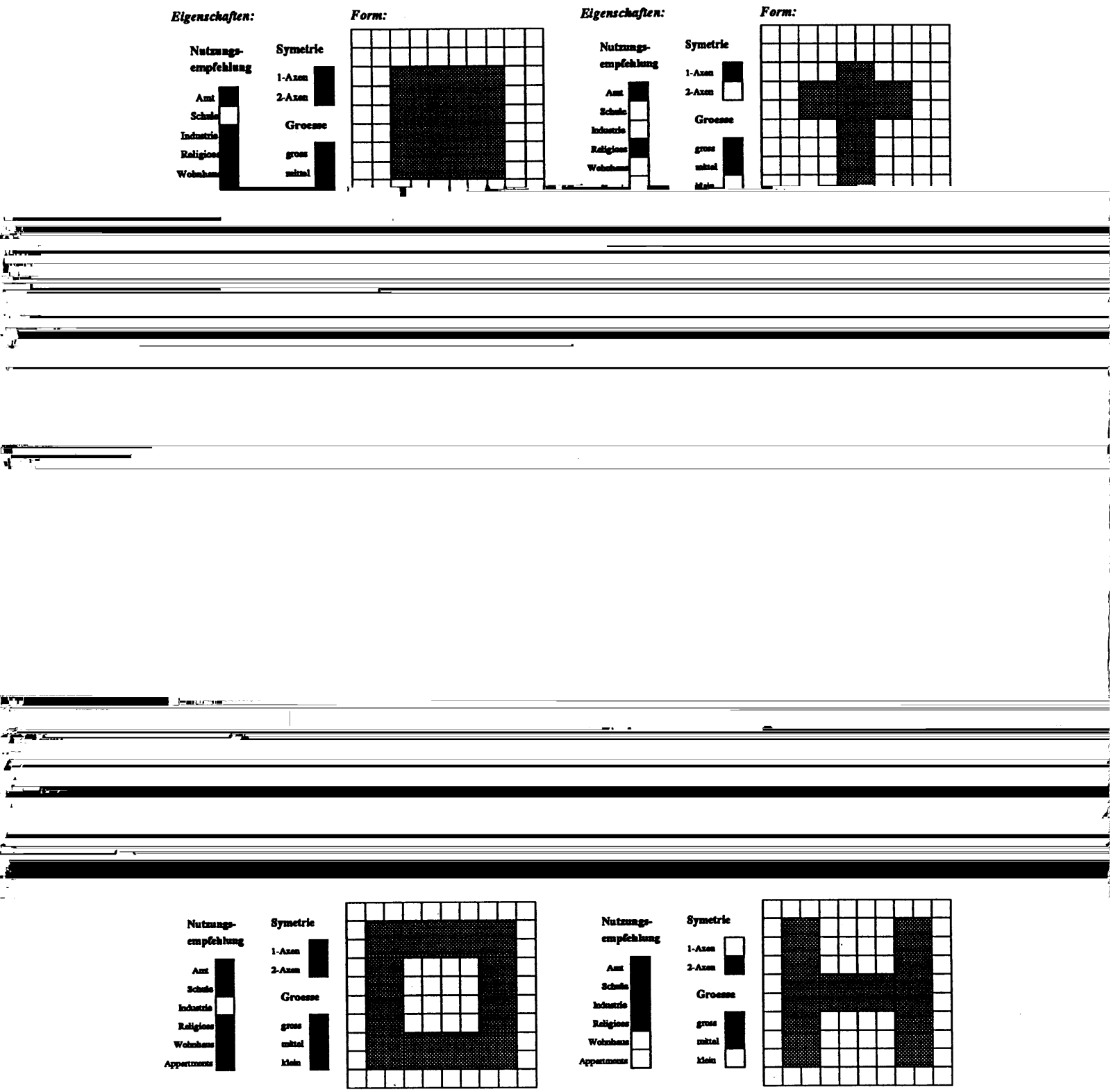


ABBILDUNG 5.9. Lernmenge aus Gebäude-Design-Pattern mit Assoziationsmatrix

Für das neuronale Netz besteht natürlich kein Unterschied zwischen der Matrix für die gerasterte Gebäudedarstellung und der Assoziationsmatrix. Gibt man eine Gebäudedarstellung vor so erzeugt das Netz via Patternvervollständigung zugehörige Bewertungen. Wir haben somit ein System das Interpretationswissen repräsentiert. Gibt man umgekehrt bestimmte Eigenschaftswerte vor so wird ein Design erzeugt, d.h. das implizit im Netz gespeicherte Wissen kann als Generierungswissen interpretiert werden. Zu beachten ist, daß beide Eingaben nicht vollständig vorgegeben werden müssen, um akzeptale Ergebnisse zu liefern. Gerade hier liegt die Stärke des Neuronalen-Netz-Modells. Figur 5.10 zeigt die Stadien eines solchen Rückkopplungsprozesses.

Es ist in vielen Anwendungsfällen sicherlich interessant eine Kombination verschiedener Pläne

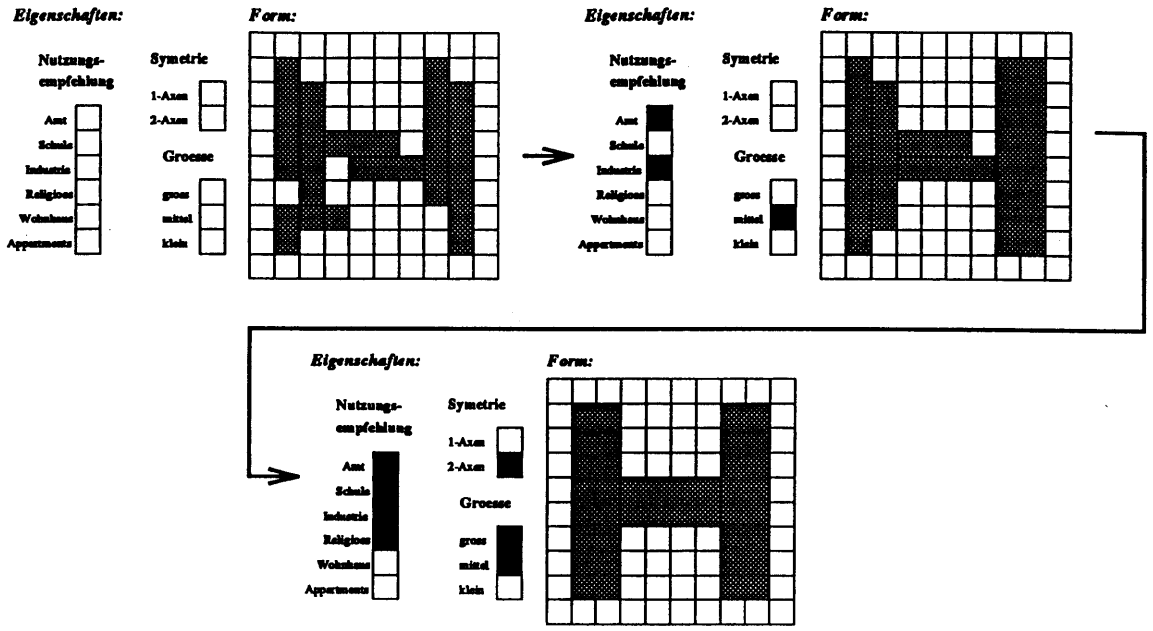


ABBILDUNG 5.10. Vervollständigung und Interpretation eines Gebäudedesigns

5.6.1 Produktkreativität

Man kann drei Bemessungskriterien zur Bewertung der Produktkreativität ansetzen. Den Grad an Innovativität, der dazu in Beziehung stehende Wert und das Maß an Interpretierbarkeit:

1. Innovativität: Weicht ein Produkt eines Designers von den Produkten anderer Designer ab und war dieses Produkt in einer solchen Form und Funktion vorher nicht verfügbar, so kann es in gewissem Sinne als kreativ bezeichnet werden. Es läßt sich leider nur schwer eine Bemessungsgröße für die Inovativität angeben, die es erlauben würde Designer, Produkte und Designsysteme zu vergleichen. Die Einschätzung der Kreativität kann auch nicht global erfolgen, sondern muß im räumlichen und zeitlichen Zusammenhang mit dem Umfeld gesehen werden. So wäre z.B. die Erfindung der Dampfmaschine in der heutigen Zeit nichts besonderes, wohl aber die Herleitung der Relativitätstheorie durch einem Schüler, der nur Grundkenntnisse der Mathmatik und Physik besitzt.

2. Wert: Eine andere Sicht der Kreativität ist die Betrachtung des Wertes eines Produktes, wie immer dieser auch definiert sein mag, als Schönheit, Einfachheit, Nutzbarkeit oder Vermarktbarkeit.

3. Interpretierbarkeit: Kann ein Produkt in vielen Einsatzbereichen benutzt werden, für die es eigentlich nicht entwickelt wurde, so besitzt es einen großen Interpretationsreichtum

5.6.2 Produktionskreativität

Anhand der Gesichtspunkte Entropie, Effizienz und Variierbarkeit soll nun die Kreativität des Designprozesses d.h. des Designsystems betrachtet werden:

- Entropie: Ein System kann als kreativ bezeichnet werden, wenn es eine komplexe Produktbeschreibung aus einer kleinen 'Saat-Information' ableitet. So ist z.B. die Neuerfindung einer Dampfmaschine durch einen Menschen, der kein Wissen über Mechanik und Physik besitzt kreativer als durch einen Ingeneur. Ein Designsystem, daß ein Gebäudedesign aus einer großen Datenbank von Designs ableitet ist sicherlich weniger kreativ als eines, daß mit anderen, artverwandten Daten dies tut. Es besteht also ein enger Zusammenhang zwischen Kreativität und Generalisierung. Je genereller die Information ist, die dem System zur Verfügung steht,

desto größer ist die kreative Leistung bei der Ableitung eines Designs einzustufen. Die Entropie kann ausgedrückt werden, als Betrag an Organisation und Struktur in einem System zwischen den Komponenten. Eine große Entropie drückt eine geringe Organisation aus und ein großes kreatives Potential.

- Effizienz: Kommt ein System nicht in endlicher Zeit zu einem Ergebnis kann es trotz hoher Entropie sicherlich nicht als kreativ bezeichnet werden.
- Variierbarkeit: Ist ein System in einem weiten Einsatzfeld einsetzbar, d.h. ist es in der Lage auf vielen Gebieten Designs zu erzeugen, so kann es eher als kreativ bezeichent werden als ein System, welches nur begrenzt eingesetzt werden kann. Hierunter fällt auch der Gesichtpunkt der Adaptierbarkeit.

5.7 Kreativität in wissensbasierten Designsystemen

In diesem Abschnitt soll die Definition des Begriffs Kreativität im Zusammenhang mit wissensbasierten Designsystemen weiterentwickelt werden. Die wesentliche Erkenntnisse lieferten dabei Beobachtungsstudien am Menschen. Die Tatsache, daß diese Sachverhalte nicht alle rational erfaßbar sind, wirft Schwierigkeiten bei der Computer-Portierung auf.

5.7.1 Dynamik der Design-Spaces

Kreative Individuen sind in der Lage aus konventionellen Formen des Denkens auszubrechen und neue Wege zu bestreiten. In der Begriffswelt der wissensbasierten Designsysteme können wir sagen, Kreativität bedeutet über die Grenzen des bekannten Wissens hinauszugehen und die Design-Spaces zu erweitern. Das Designwissen ist als Ganzes normalerweise nicht wohlgeformt und die Suche in dem Gesamtraum nicht gerade einfach. Die Menge aller Designs erhält auch nicht durch äußeren Faktoren ein klar umrissene Grenze. Die Suche selbst modifiziert die Gestalt und die Grenzen dieses Raums. Die Grenzen sind dynamisch und der Designprozeß beschränkt sich nicht nur auf die Suche von Designs innerhalb dieses Raumes, sondern vielmehr auf die Ableitung und Modifikation von Wissen, das diesen Raum definiert. Um dies bewerkstelligen zu können muß ein Designsystem über verschiedene vorher besprochenen Schlußmechanismen verfügen.

5.7.2 Menschliche Kreativität

Aufgrund der engen Verknüpfung von Psychologie und KI erlaubt die psychologische Sicht einige hilfreiche Einblicke bei der Entwicklung leistungsfähiger Problemlösemechanismen. Zunächst soll auf die Voraussetzungen der Kreativität beim Menschen eingegangen werden:

- Intelligenz: Intelligenz gibt Personen die Möglichkeit Informationen und Wissen in hilfreicher Art und Weise zu organisieren und anzuwenden.
- Ausbildung: Für Experten, die in ihrem Gebiet erfolgreich sein wollen ist ein fundiertes Breiten- und Tiefenwissen von Nöten

Diese beiden Faktoren stellen aber eher notwendige und nicht hinreichende Bedingungen für Kreativität dar. Zusätzlich sind noch andere Eigenschaften wichtig:

- Autonomie: Dies meint, Autonomie in Denkweisen und vorallem in der Bewertung neuer Ideen, unabhängig von der Einschätzung neuer Gedanken durch das Umfeld.
- Fähigkeit mit Ideen zu spielen: Kreative Menschen haben eine Begabung, Konzepte zu verändern und den Erfordernissen anzupassen. Aufdecken, Umordnen, Verändern und Kombinieren sind hier Schlagworte. Diese Veränderungen geschehen jedoch nicht orientierugslos, sondern werden von meschlichen Merkmalen gesteuert, die sowohl auf rationaler, wie auch auf emotionaler Ebene angesiedelt sind.
- Ästhetische Sensibilität: Schönheit von Mustern, Strukturen, Ideen und Theorien sind zudem Leitmotive für kreative Menschen.

Neben diesen Faktoren gibt es noch einige wichtige, jedoch weniger gut faßbare Charaktereigenschaften kreativer Menschen wie: Orginalität, Freude, Durchhaltevermögen usw. Gerade die letztgenannten Eigenschaften lassen sich nur sehr rudimentär oder gar nicht auf einem Computer modellieren, sind aber gleichwohl sehr wichtige, vielleicht die wichtigsten Eigenschaften für kreativen Erfolg.

5.7.3 Modellierung menschlicher Strategien der Kreativität

In diesem Abschnitt sollen Methoden des kreativen Denkens erörtert werden. Ein vielversprechender Weg ist die Portierung bekannter meschlicher Denktechniken auf den Computer. Im folgenden werden fünf solcher Techniken und ihre Beziehung zur KI vorgestellt:

1. Trial and Error: Diese Methode wird in weiten Bereichen eingesetzt, sowohl zur Lösung von Routineproblemen als auch solchen, die kreatives Handeln verlangen. Trial and error meint die Generierung einer Lösung und der anschließende Test auf die Korrektheit. Schlägt der Test fehl, so wird eine neue Lösung evt. unter Zuhilfenahme des Fehlers erzeugt und erneut getestet. Die Computervariante nennt sich "generate and test". Dieser Prozeß besitzt zwei Komponenten, eine die die Lösungen erzeugt und eine zweite die diese Lösungen auf ihre Korrektheit überprüft. Der gravierende Unterschied besteht darin, daß ein Computer gar nicht oder nur sehr rudimentär aus dem vorangegangenen Fehler lernen kann. Auf dem Gebiet der KI wird deshalb an diesem Problem des Lernens gearbeitet.

2. Brainstorming: Dies ist eine Technik für Gruppenarbeit. Menschen mit verschiedenem Wissen über eine Sache und verschiedenem Backroundwissen treffen sich zu einem Meeting in dem sie ihre Ideen und Vorstellungen ohne Beschränkungen und Kritik vortragen können, quasi eine Ideensammlung. Jede Person kann die Ideen einer anderen aufgreifen modifizieren und weiterentwickeln, um zusammen zu einer guten Lösung zu kommen in die möglichst viel verschiedenes Wissen einfließt. Auf einen Computer übertragen bedeutet dies: Nach jeder Problemspezifizierung wird in der Wissensbasis Wissen zu diesem Problem zusammengetragen. Dies alleine reicht natürlich nicht aus. Es kommt entscheidend darauf an, daß dieses Wissen weiterentwickelt und kombiniert wird und so weitere Beziehungen zu existierendem Wissen hergestellt werden kann.

3. Mutation: Mutation meint eine wohlüberlegte Veränderung von Attributen oder Teilen eines Objektes oder Konzeptes in einer unkonventionellen Art und Weise. Die Änderung soll nicht durch die bestehenden Regeln oder Bedingungen beschränkt sein. Die Mutation soll den Objekten neue Eigenschaften, Funktionen oder Bedeutungen verleihen. Hier sind alle möglichen Modifikationen denkbar. Der Prozeß soll letztendlich Prototypen erzeugen und ist somit besonders wichtig für die Kreativität. Es ist allerdings sehr schwierig für eine bestimmte Aufgabe eine wirklich leistungsfähige Mutationsfunktion und einen Kontrollmechanismus zu finden, der angibt wann mutiert werden soll. Diese Methode scheint trotzdem sehr verheißungsvoll.

4. Analogien: Der Analogie-Ansatz ist hilfreich zur Lösung nicht vertrauter neuer Probleme ohne adequates, direkt nutzbares Wissen. Bei diesem Ansatz wird versucht Beziehungen zwischen dem neuen Problem und altem Wissen aufzubauen. Diese alten Erfahrungen können in die neue Situation plaziert werden, um sie besser zu verstehen, oder eine Lösung zu liefern. Durch Ähnlichkeiten wird Raum geschaffen für Vorstellungen, eben Kreativität. Voraussetzung ist natürlich, daß eine Menge Wissen und Erfahrungen zur Verfügung stehen und ein guter Mechanismus zur Auffindung der Analogien. Gerade auf diesem Gebiet liegen die größten Schwierigkeiten und Forschungsschwerpunkte. Das Schließen mit Analogien kann in zwei Klassen eingeteilt werden: 1. Transformierte Analogien versuchen das neue Problem auf die bekannten alten zu transformieren und deren Lösungsansätze zu verwenden. 2. Abgeleitete Analogien versuchen die alten, bekannten Erfahrungen den neuen Problemen anzupassen.

5. Inversion (Umkehrung): Bei dieser Technik wird eine Invertierung des Ausgangsgrundes und Effektes eines Phenomens versucht. z.B. Bewegt man einen Leiter in einem Magnetfeld, so wird in ihm ein Strom induziert; Umgekehrt schickt man einen Strom durch den Leiter, so erzeugt dieser ein Magnetfeld. Um die Inversionsmethode an einer Wissensbasis anzuwenden

muß diese aus Paaren von Gründen/Bedingungen und Aktionen aufgebaut sein. Zusätzlich ist oft noch weitergehendens Wissen über die Beziehungen zwischen Bedingungen und Aktionen nötig für die Umkehrung. Es wird z.B. an Designsystemen für die Physik gearbeitet die auf dieser Methode aufbauen.

5.8 Schlußbetrachtungen

Alle vorgestellten Ansätze sind in bestimmten Bereichen sehr aussichtsreich und Gegenstand der aktuellen KI-Forschung. Eine Verknüpfung erfolgte bisher jedoch nur sehr unzureichend. Gerade hier müssen jedoch die zukünftigen Forschungsschwerpunkte liegen, nicht zuletzt, weil das erfolgreichste Designer, der Mensch selbst, vorallem auf dem Prinzip der Nutzung aller Verfahren aufbaut. Ein solches Designssysstem müßte allerdings eine höhere Kontroll- oder Ordnungsfunktion zur Koordinierung der einzelnen Komponenten enthalten und genau an diesem Punkt existieren noch keine erfolgversprechenden Ansätze. Die unzureichende Möglichkeiten der Fassung und Implementierung von Kreativität lassen darauf schließen, daß es nicht möglich sein wird ein leistungsfähiges autonomes Designsystem zu entwerfen. Somit konzentriert sich die Forschung im wesentlichen darauf, designunterstützende Systeme zu kreieren, die ebenfalls ohne eine Umsetzung kognitiver Ansätze keine Leistung entfalten.

5.9 Literaturverzeichnis

[CRR+90] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge-Based Design Systems.* Addison-Wesley Publishing Company, 1990.

[Ric89] M.M. Richter. *Prinzipien der Künstlichen Intelligenz.* Teubner Verlag, 1989.

[Ric93] M. M. Richter. Lernende Systeme II: Konnektionismus. Vorlesungsmanuskript, Universität Kaiserslautern, 1993.

[RW93] M.M. Richter and O. Wendel. Lernende Systeme I: Symbolische Methoden. Vorlesungsmanuskript, Universität Kaiserslautern, 1993.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or per anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49
Christoph Klauck, Ralf Legleitner, Ansgar Bernardi:
Heuristic Classification for Automated CAPP
15 pages

RR-92-50
Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-53
Werner Stephan, Susanne Biundo:
A New Logical Framework for Deductive Planning
15 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55
John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-56
Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58
Franz Baader, Bernhard Hollunder:
How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59
Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
13 pages

RR-92-60
Karl Schlechta: Defaults, Preorder Semantics and Circumscription
19 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist:
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi:
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-04
Christoph Klauck, Johannes Schwagereit:
GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-05
Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-06
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: On Skolemization in Constrained Logics
40 pages

RR-93-07
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: Concept Logics with Function Symbols
36 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer: COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-09
Philipp Hanschke, Jörg Würtz:
Satisfiability of the Smallest Binary Program
8 Seiten

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf: Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-11
Bernhard Nebel, Hans-Juergen Buerckert:
Reasoning about Temporal Relations:
A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12
Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13
Franz Baader, Karl Schlechta:
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen:
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman:
Corporate Agents
14 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman:
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman:
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-33
Bernhard Nebel, Jana Koehler:
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster:
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.):
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf:
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen:
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt: Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-46
Philipp Hanschke: A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

DFKI Technical Memos

TM-92-01
Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh:
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

TM-93-01
Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

TM-93-04
Hans-Günther Hein: Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

TM-93-05
Michael Sintek: Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

DFKI Documents

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

D-93-01
Philipp Hanschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

D-93-03
Stephan Busemann, Karin Harbusch(Eds.):
DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

D-93-05
Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster:
PPP: Personalized Plan-Based Presenter
70 pages

D-93-06
Jürgen Müller (Hrsg.):
Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993
235 Seiten
Note: This document is available only for a

D-93-11
Knut Hinkelmann, Armin Laux (Eds.):
DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein:
RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-15
Robert Laux: Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt: Design & KI
74 Seiten

D-93-20
Bernhard Herbig:
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-21
Dennis Drollinger:
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken

Klaus-Peter Gores, Rainer Bleisinger:
Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-08

D-93-22
Andreas Abecker: Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

Design & KI

Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt (Hrsg.)

D-93-16
Document