Dem Menschen abgeschaut: Fallbasiertes Lösen von Problemen

Nutzbarmachung eines mächtigen Paradigmas in der Informatik

Der Einsatz fallbasierter Techniken bei der rechnergestützten Lösung von Problemen hat sich bewährt. Insbesondere in Gebieten, in denen die Vorgehensweise des Menschen nur unzureichend verstanden wird oder nur schlecht nachgebildet werden kann, weisen fallbasierte Ansätze Erfolge auf.

Fallbasiertes Schließen bzw. fallbasiertes Problemlösen ist ein dem Menschen ureigenes, altbewährtes Prinzip. Seine Übertragung auf den Rechner bedeutet die Umsetzung von Konzepten wie *Fallähnlichkeit, Fallspeicherung* oder *Fallanpassung*.

Das Prinzip aller fallbasierten Systeme ist gleich – die Herausforderungen bei der Entwicklung eines solchen Systems und die Grenzen seiner Leistungsfähigkeit hängen jedoch ausschließlich davon ab, wie gut die Umsetzung der genannten Konzepte für ein aktuelles Problem gelingt.

Es ist schon eine Weile her, als wir wieder mal vor dem Problem standen, den klemmenden Schraubverschluß einer Saftflasche zu lösen. Kurz bevor wir aufgeben wollten, fiel uns ein, was wir bei jemandem beobachtet hatten, der vor der gleichen Aufgabe stand: Er drehte die Flasche um, schlug mit dem Handballen kurz auf den Boden der umgedrehten Flasche und konnte anschließend mühelos den Deckel abdrehen. Es gab nichts zu verlieren und so machten wir das nach, was wir beobachtet hatten – es funktionierte.

Obwohl etwas versteckt, handelt es sich bei der geschilderten Situation um ein Beispiel für fallbasiertes Problemlösen. Dabei besteht ein Fall aus einer Problembeschreibung („*Schraubverschluß klemmt*“) und einer Lösung („*auf den Flaschenboden schlagen*“).

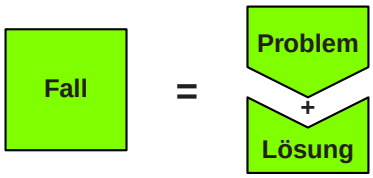

Abbildung 1: *Beim fallbasierten Schließen besteht jeder Fall aus einer Problembeschreibung mit seiner zugehörigen Lösung.*

Die grundsätzliche Annahme hinter jedem fallbasierten Ansatz ist, daß aus der Ähnlichkeit zweier Probleme auf die Ähnlichkeit ihrer Lösungen geschlossen werden darf: Zu dem offenen Problem wird ein möglichst ähnliches, in der Vergangenheit schon gelöstes Problem gesucht. Dessen Lösung wird angepaßt und dient als Lösung des offenen Problems (siehe Abbildung 2).

Das Saftflaschenbeispiel illustriert einige Aspekte des fallbasierten Schließens:

- *Erinnerung.* Man erinnert sich an eine vergleichbare Situation.
- *Anpassung.* Wahrscheinlich handelt es sich in der aktuellen Situation um eine andere Saftflasche als im gespeicherten Fall. Durch Fallanpassung (Austausch der Flasche) ist man in der Lage, die gespeicherte Lösung an die neue Situation anzupassen.
- *Anwendung.* Die Lösung kann angewandt werden, auch wenn man nicht verstanden hat, wie sie funktioniert.[1]

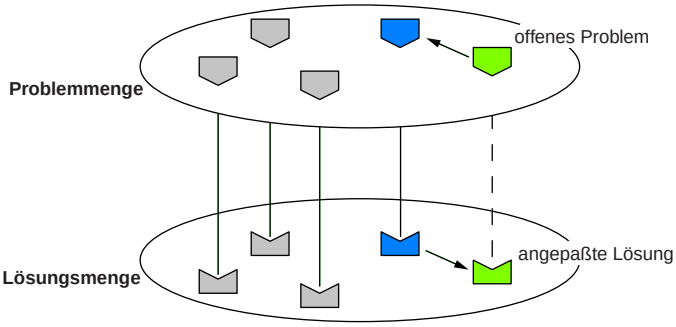

Abbildung 2: *Grundannahme des fallbasierten Schließens: Aus der Ähnlichkeit zweier Probleme kann auf die Ähnlichkeit ihrer Lösungen geschlossen werden [9].*

Geschichte

Die theoretischen Ursprünge des fallbasierten Schließens (engl.: CBR für Case-Based Reasoning) stammen aus der Kognitionswissenschaft: Schank und Abelson verglichen das dem Menschen typische allgemeine Verständnis über Situationen mit einer Art Gedächtnisschema, das sie als Skript bezeichneten. Skripte ermöglichen es uns, stereotypische Ereignisse zu beschreiben, bestimmte Erwartungen zu haben oder Schlußfolgerungen zu ziehen [11, 12].

Eine weitere wichtige Wurzel des CBR ist der Bereich des analogen Schließens, das sich mit der Übertragung von Lösungsprinzipien aus einem gut verstandenen Problembereich auf eine ganz neue Situation beschäftigt.

Fallbasiertes Schließen versucht, ungelöste Probleme auf der Basis schon gelöster Probleme *desselben* Aufgabenbereiches in den Griff zu bekommen. Fallbasiertes Schließen kann also als eine Spezialisierung des analogen Schließens aufgefaßt werden [17].

Als erste Realisierung eines fallbasierten Systems wird Janet Kolodners System CYRUS aus dem Jahr 1983 bezeichnet, das Fragen über Begebenheiten aus der Tätigkeit des ehemaligen amerikanischen Außenministers Cyrus Vance beantworten kann [7]. Mittlerweile befinden sich eine Reihe von fallbasierten Systemen erfolgreich im Einsatz, sowohl in der Industrie als auch im Dienstleistungsbereich. Sie machen Vorschläge bei der Beschickung

[1] Wie funktioniert der Saftflaschentrick? Durch den Schlag auf die Flasche erfährt der Saft einen Impuls, der wiederum zu einer Kraft und somit zu einer (elastischen) Verformung des Deckels führt. Durch die entstandene Druckwelle wird etwas Flüssigkeit durch die Dichtung gequetscht, der die potentielle Verklebung zwischen Deckel und Flasche löst. Eventuell kann zusätzlich durch das kurzzeitige Abheben des Deckels ein (teilweiser) Druckausgleich zwischen Flasche und Umgebung stattfinden.

großer Brennöfen, steuern die Herstellung von komplizierten Aluminiumdruckgußteilen, unterstützen Diagnoseaufgaben in Call-Centern oder suchen nach Präzedenzfällen in juristischen Datenbanken [16, 2].

Umsetzung des fallbasierten Paradigmas als wissensbasierte Technik

Der mentale Prozeß des fallbasierten Schließens beim Menschen kann in vereinfachter Weise als ein Zyklus aufgefaßt werden, der aus vier Schritten besteht [1]:

1. *Retrieve.* Ein oder mehrere Fälle, die relevant für das zu lösende Problem sind, werden aus einer Fallbasis gesucht.
2. *Reuse.* Nach der Durchführung von eventuell notwendigen Adaptionen wird ein ausgewählter Fall wiederverwendet.
3. *Revise.* Nach einer Evaluierung des adaptierten Falls werden gegebenenfalls weitere „Reparaturen“ durchgeführt.
4. *Retain.* Der durch die vorangegangenen Schritte neu gewonnen Fall (bestehend aus Problem + Lösung) wird in der Fallbasis abgelegt.

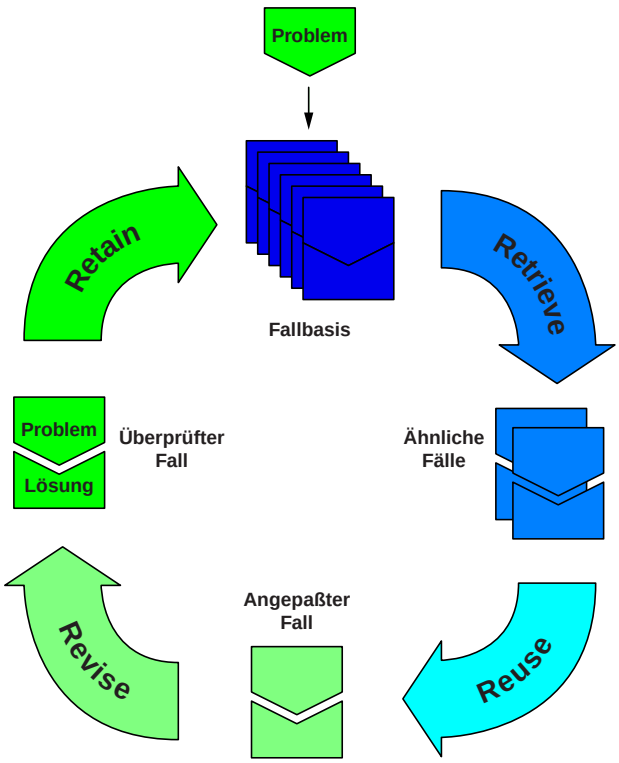

Abbildung 3: *Der Zyklus des fallbasierten Schließens nach Aamodt und Plaza [1].*

Jeder dieser vier Schritte enthält wieder spezielle Probleme, für die teilweise ausgereifte Lösungen, teilweise aber auch nur Lösungsideen oder vage Ansätze vorliegen. U. a. stellen sich folgende Fragen:

- ❑ Wie werden Probleme und Lösungen in Fällen geeignet repräsentiert?
- ❑ Was bedeutet und wie quantifiziert man Ähnlichkeit?
- ❑ Wie funktioniert das Wiederverwenden einer ähnlichen Lösung?

Bei der Repräsentation von Fällen reichen die Ansätze von einfachen Attribut-Wert-Repräsentationen bis hin zu Strukturen, die ein Gedächtnisschema nachbilden sollen [8]. Unabhängig von der Art der Repräsentation muß überlegt werden, ob es sinnvoll ist, die Lösung selbst oder den Lösungs*weg* zu speichern [3].

Jeder fallbasierte Problemlöseansatz startet mit einer Suche, hier Retrieve oder Retrieval genannt. Das Herz des Retrieve-Schrittes ist die Definition der Fallähnlichkeit. Man unterscheidet zwischen Ähnlichkeitsmaßen, die durch ein Prädikat, durch eine Präferenzrelation oder durch einen berechnungsorientierten Ansatz realisiert sind [17, 10]. Zum Beispiel bedeutet die Präferenzrelation $R(x, y, u, v)$:
„Fall x ist zu Fall y mindestens so ähnlich, wie Fall u zu Fall v.“
Der berechnungsorientierte Ansatz drückt die Ähnlichkeit implizit durch ein Maß „sim“ aus. Für jeweils zwei Fälle x und y liefert $sim(x, y)$ einen Wert aus dem Intervall $[0; 1]$, und je größer der Wert von sim ist, desto ähnlicher sind sich x und y. Um dem menschlichen Verständnis von Ähnlichkeit nahe zu kommen, sollten Ähnlichkeitsmaße zwei Eigenschaften erfüllen: (a) Reflexivität oder Selbstähnlichkeit, d. h. für einen Fall x muß $sim(x, x) = 1$ gelten. (b) Symmetrie, d. h. für zwei Fälle x und y muß $sim(x, y) = sim(y, x)$ gelten.

Im Retrieve-Schritt wird die offene Problemstellung mit den gespeicherten Fällen unter Zugrundelegung des Ähnlichkeitsmaßes verglichen. Mittlerweile ist eine Vielzahl von Retrieval-Verfahren entwickelt worden, die sich hinsichtlich der Organisation der Fallbasis, der Konstruktion eines Index oder der Verfolgung bestimmter Strategien unterscheiden.

Im Reuse-Schritt geschehen notwendige Anpassungen des ähnlichsten Falls in Hinblick auf das offene Problem. Danach kann der modifizierte Fall als Lösung eingesetzt werden; er kann aber auch – im Rahmen eines zusätzlichen Revise-Schrittes – überprüft und weiter angepaßt werden. Anpassungen bzw. Adaptionen, seien sie automatisch, unüberwacht, überwacht oder vollständig vom Menschen durchgeführt, spielen also eine wichtige Rolle [15, 5].

Die Leistungsfähigkeit eines fallbasierten Systems wird maßgeblich durch die Komplexität und die Qualität der durchführbaren Adaptionen bestimmt. Hier reicht das Spektrum von der Nulladaption (keine Anpassung) über die Parameteradaption (einfache Werteanpassung), der modellgestützten Adaption (Rückgriff auf kausale Modelle) bis hin zur Adaption durch Kombination mehrerer Fälle [16, 17]. Hinsichtlich der Formulierung von Adaptionswissen ist die Verwendung von sogenannten Reparaturregeln besonderes beliebt; allgemeiner, wenngleich auch schwieriger ist die Aufstellung und Lösung eines Constraint-Problems.

Der Retain-Schritt stellt die Lernkomponente eines fallbasierten Systems dar. Der neu gewonnene Fall bzw. eine Überarbeitung dieses Falles wird der Fallbasis hinzugefügt. Sinnvoll ist eine Erweiterung der Fallbasis eventuell dann, wenn die durchgeführten Adaptionen von struktureller Natur oder sehr aufwendig sind. Wann, wie oft oder um welche Fälle eine Fallbasis erweitert werden soll, ist nicht allgemein entscheidbar und stellt ein spannendes Problem der Forschung dar.

Fallbasiertes Schließen kritisch betrachtet

Fallbasierten Systemen, oder allgemeiner, dem fallbasierten Schließen werden eine Reihe von Vorteilen zugesprochen [9, 8]. Folgende Argumente werden häufig genannt:

- ❑ Lösungen müssen nicht von Grund auf neu entwickelt wer-

den.

- Lösungen können vorgeschlagen und eingesetzt werden, auch wenn sie nicht vollständig verstanden sind (siehe Saftflaschenbeispiel).
- Lösungen in Form von Fällen sind nützlich bei der Interpretation unvollständig beschriebener Probleme.
- Durch das Konzept einer Fallbasis ist der Erwerb und die Integration neuen Wissens einfach.

Doch die scheinbare Einfachheit des fallbasierten Ansatzes birgt ihre Tücken.

Allein die Entwicklung eines sinnvollen Ähnlichkeitsmaßes bedeutet in der Praxis eine große Hürde. So sehr der menschliche Geist talentiert ist, ein Urteil bzgl. der Ähnlichkeit von zwei Problemstellungen abzugegeben, so problematisch stellt sich die *Quantifizierung* dieser Urteilsfähigkeit in Form eines Ähnlichkeitsmaßes dar.

Auch die Aufstellung einer ausreichend großen Fallsammlung, dem „Gedächtnis“ eines fallbasierten Systems, stellt sich oft schwieriger heraus, als erwartet. Zudem möchte man diesem Gedächtnis möglichst wenig Ballast in Form von unnützen oder gar schlecht gelösten Fällen mitgeben. Dabei ist es sehr schwierig, ja oft unmöglich, den Beitrag, den ein neuer Fall zur Verbesserung der Lösungsqualität beiträgt, zu quantifizieren.

Für die Falladaption gibt es keine allgemeine Theorie (es kann auch keine geben): Die Durchführung von Adaptionen erfordert tiefes Verständnis aus dem Anwendungsbereich, und so stellt die Entwicklung von Adaptionsalgorithmen jedesmal eine neue Herausforderung dar. Folglich sind Problemstellungen, die eine umfangreiche Nachbearbeitung einer existierenden Lösung erfordern, ungeeignet für den fallbasierten Ansatz.

Erfordert die Problemstellung die Bestimmung einer optimalen Lösung, so ist ein fallbasierter Lösungsansatz in der Regel zu unflexibel, weil er immer von einer bestehenden und somit festen Lösung ausgeht.

Im Umkehrschluß kann man festhalten: Wenn ausreichendes Wissen über einen Problembereich vorliegt, die notwendigen Zusammenhänge gut verstanden sind und effizient verarbeitet werden können, sind die Standardtechnologien der Wissensverarbeitung dem fallbasierten Ansatz überlegen.

Auf größere Entwurfsprobleme treffen die genannten Eigenschaften nur eingeschränkt zu; und so tut sich die klassische Wissensverarbeitung schwer mit deren Lösung.

Lösung von Entwurfsproblemen – eine starke Domäne des Menschen

Entwerfen bedeutet die Schaffung eines Systems oder einer Systembeschreibung entsprechend einer Menge von Vorgaben. Bei dem System kann es sich um ein technisches Gerät, aber auch um ein Gebäude oder um ein Waschmittel handeln. Die entsprechende Systembeschreibung wäre ein Bauplan, eine Zeichnung oder eine Rezeptur. Der Mensch ist besonders leistungsfähig bei der Lösung von Entwurfsproblemen.

Auf dem Rechner werden Entwurfsprobleme durch Suche gelöst, und je mehr Entwurfswissen für einen Bereich vorliegt, umso effizienter kann man diese Suche gestalten. Trotz dieser wissensbasierten Herangehensweise bleiben die Suchräume bei Entwurfsproblemen sehr groß und schwer beherrschbar. Hinzu kommt, daß die menschliche Problemlösefähigkeit hier nur zum Teil verstanden ist, was sich bei vielen Entwurfsproblemen u. a. darin äußert, daß sich die Begriffe „guter Entwurf“ oder „schlechter Entwurf“ einer formalen Definition entziehen. So kommt zu dem Problem des großen Suchraums auch noch das Problem, daß man nicht exakt spezifizieren kann, wonach man sucht.

Aus Sicht der Informatik stellt sich ein Entwurfsproblem wie in Abbildung 4 skizziert dar: Gegeben ist eine Anforderungsspezifikation D aus der Menge $\mathcal{D}$ aller möglichen Anforderungenspezifikationen. Ziel ist es, D zu einer Systembeschreibung S zu transformieren, die alle Wünsche, Erwartungen und Randbedingungen aus D erfüllt [4, 13].

Abbildung 4: *Entwerfen heißt, eine Anforderungsmenge D auf ein System S abzubilden.*

Soll z. B. ein neues Auto entworfen werden, könnte D Anforderungen bzgl. des Verbrauchs, des Platzangebots und des Preises enthalten; wäre ein neuer Müsli-Riegel zu entwerfen, so könnte D Vorgaben bzgl. des Geschmacks, der Lagerfähigkeit und der Form machen.

Jede gelöste Entwurfsaufgabe enthält Entwurfswissen. Der größte Teil dieses Wissens ist lediglich implizit vorhanden: es ist versteckt oder komprimiert in der Entwurfslösung an sich. Zur Zeit ist man nur bedingt in der Lage, dieses Entwurfswissen explizit zu machen und z. B. in der Form von „Entwurfsregeln“ zu formalisieren.

An dieser Stelle kommt die Idee des fallbasierten Schließens ins Spiel. Es schafft einen Rahmen, alte Entwürfe zur Lösung neuer Entwurfsaufgaben wiederzuverwenden und versucht so, implizites Entwurfswissen nutzbar machen.

Fallbasierte Entwurfsautomatisierung am Beispiel hydraulischer Antriebe

Hydraulik kommt überall dort zum Einsatz, wo große Kräfte auf kleinem Raum erzeugt werden müssen. Hydraulische Antriebe verrichten Manipulations- und Fertigungsaufgaben in der Industrie, realisieren Hebeaufgaben bei jeder Art von Bühnentechnik und bewegen Roboter und Fahrzeuge. Zylinder sind die Aktuatoren eines hydraulischen Antriebs; Ventile wie Drossel-, Druckbegrenzungs-, Rückschlag- oder Proportionalventile steuern Druck und Fluß des hydraulischen Mediums. Pumpen stellen die hydraulische Energie zur Verfügung.

Die Anforderungsspezifikation D eines zu entwerfenden hydraulischen Antriebs enthält im wesentlichen Fahrprofile für den Verlauf der Wege, Kräfte und Geschwindigkeiten von Zylindern; hinzu kommen Wertebereichsbeschränkungen für physikalische Größen, Toleranzforderungen und andere Vorgaben. Das Ergebnis S des Entwurfsprozesses ist der Schaltplan eines Antriebs, der D erfüllt.

Der Entwurf eines neuen Antriebs, d. h. die Abbildung $D \longrightarrow S$, ist ein kreativer Prozeß, der von einem Hydraulikingenieur geleistet wird. Die Abbildung gelingt in der Regel nicht in einem Schritt, und so entsteht ein Entwurfszyklus mit den Schritten *Konstruktion*, *Simulation* und *Anforderungsvergleich*. Insbesondere für den Simulationsschritt existieren Verfahren und Werkzeuge, die den Entwerfer unterstützen [13]. Wegen der im Konstruktionsschritt erforderlichen Kreativität ist eine vollständige Automatisierung des Entwurfsprozesses zur Zeit noch in weiter Ferne.

Natürlich läßt sich die Kreativität des menschlichen Geistes auch nicht mit einem fallbasierten Ansatz nachbilden. Fallbasierte Techniken eröffnen aber die Möglichkeit, schwierigste Teile des Entwurfsprozesses – wie hier den anspruchsvollen Konstruktionsschritt – zu unterstützen oder teilweise zu automatisieren. Wie, das ist im Folgenden kurz beschrieben.

Aus Sicht des Entwerfers realisiert jeder hydraulische Antrieb eine komplexe Funktion, die wiederum durch das Zusammenspiel verschiedener Teilfunktionen entsteht. Die baulichen Entsprechungen dieser Teilfunktionen sind Ventil-Zylinder-Kombinationen, sogenannte hydraulische Achsen. Abbildung 5 zeigt einige Beispiele. In einem komplexen Antrieb sind mehrere hydraulische Achsen miteinander gekoppelt.

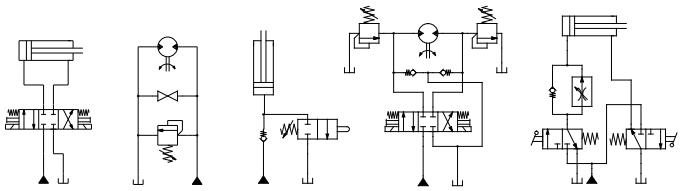

Abbildung 5: *Beispiele für hydraulische Achsen.*

Kernidee des von uns entwickelten Ansatzes zur Automatisierung des Entwurfs ist das Prinzip der „funktionellen Komposition“ [14]. Es besagt,

1. daß jede Anforderungsspezifikation D in eine Menge von Teilfunktionen $F = \{f_1, \ldots, f_n\}$ zerlegt werden kann,
2. daß jeder Teilfunktion $f \in F$ genau eine hydraulische Achse des Antriebs zugeordnet werden kann und
3. daß die Art der Kopplungen der hydraulischen Achsen (seriell, parallel, mit Rückführung, etc.) aus D ableitbar ist.

Während der erste Punkt praktisch konform mit der Realität ist, unterstellen Punkt 2 und 3, daß keine Teilfunktion f durch die Kombination verschiedener Achsen oder durch spezielle konstruktionstechnische Seiteneffekte realisiert wird.

Das heißt, jede Anforderungsspezifikation D kann zu einem hydraulischen Antrieb S transformiert werden, indem für jede Teilfunktion $f \in D$ eine hydraulische Achse A entworfen wird und, in einem zweiten Schritt, diese Achsen geeignet verschaltet werden.

Das Prinzip der funktionellen Komposition stellt eine Vereinfachung des Entwurfsprozesses dar, weil es die ganzheitliche Sicht des menschlichen Entwerfers vernachlässigt. Zwangsläufig muß dieses Prinzip zu suboptimalen Entwurfsergebnissen führen. Es spricht aber einiges für dieses Vorgehen:

1. Auch der menschliche Entwerfer benutzt, wenn auch nicht ausschließlich, das Prinzip der funktionellen Komposition.
2. Das Prinzip der funktionellen Komposition macht eine Automatisierung des Entwurfs möglich.
3. Ob ein automatisch erzeugter Entwurf die Anforderungsspezifikation D realisiert, kann automatisch festgestellt werden (durch eine Simulation).
4. Ein automatisch erzeugter Entwurf kann als Ausgangspunkt für Entwurfsüberlegungen des Menschen dienen.

Mit diesen Überlegungen stellt sich der Entwurfsprozeß wie in Abbildung 6 dar.

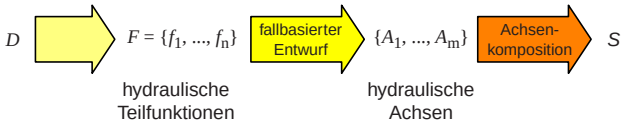

Abbildung 6: *Automatisches Entwerfen durch funktionelle Komposition: Die Anforderungsspezifikation D wird zerlegt in eine Menge von Teilfunktionen. Für jede Teilfunktion wird fallbasiert eine hydraulische Achse entworfen, die anschließend zu einem Antrieb verschaltet werden.*

In dem so modifizierten Entwurfsprozeß kann der Schritt $F \longrightarrow \{A_1, \ldots, A_m\}$ fallbasiert durchgeführt werden – vorausgesetzt, ein Ähnlichkeitsmaß für hydraulische Funktionen sowie Konzepte zur Anpassung ähnlicher hydraulischer Achsen existieren. Für beide Herausforderungen wurden Lösungen in unserer Arbeitsgruppe entwickelt, auf die hier nur illustrativ eingegangen wird [6].

Hydraulische Funktionen werden typischerweise durch Fahrprofile definiert. Ein Ähnlichkeitsmaß für hydraulische Funktionen muß also in der Lage sein, ein gewünschtes Sollfahrprofil mit vorhanden Fahrprofilen aus der Fallbasis zu vergleichen. Hierzu werden die Fahrprofile in Phasen zerlegt, zeitlich skaliert und entsprechend der Anpaßbarkeit der Phasen bewertet (siehe Abbildung 7).

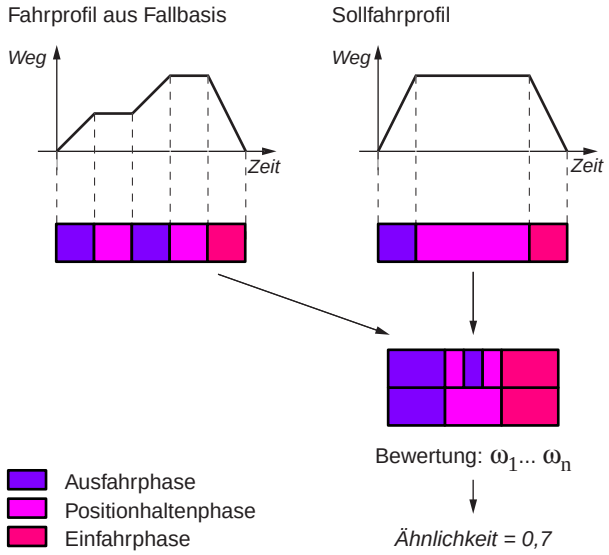

Abbildung 7: *Ein Ähnlichkeitsmaß für Fahrprofile (Prinzipskizze).*

Ein Fahrprofil ist anpaßbar, wenn die zugrundeliegende hydraulische Achse so modifiziert werden kann, daß sie das Sollfahrprofil realisiert. Idealerweise spiegelt sich der für die Anpassung notwendige Aufwand direkt im Ähnlichkeitsmaß wider.

Die Anpassung von hydraulischen Achsen geschieht mit Hilfe von Skalierungs- und Modifikationsregeln. Sie kodieren Wissen über physikalische Zusammenhänge, Baugrößen und andere Randbedingungen. Das nachfolgende Beispiel zeigt eine Skalierungsregel, die für die geometrische Veränderung des Zylinders verantwortlich ist, falls die Kraft angepaßt werden muß.

```
Name        FORCE-SCALING-RULE
Actions     AR := (F * AR) / (AK * PMAX)
            AK := F / PMAX
Qualifiers  F > 0
            AK * PMAX < F
```

Realisierung und Ausblick

Der hier vorgestellte Ansatz zur Entwurfsautomatisierung hydraulischer Antriebe wurde in Form eines Entwurfsassistenten umgesetzt. Der Entwurfsassistent besitzt eine Fallbasis mit hydraulischen Achsen und komplexen Antrieben; er realisiert das beschriebene Ähnlichkeitsmaß, die Skalierungs- und Modifikationsregeln und ein Kompositionsschema, um Achsen zu koppeln und zu einem Antrieb zu vervollständigen. Der Entwurfsassistent ist in der Lage, zu einer gegebenen Anforderungsspezifikation automatisch einen Antrieb zu konzipieren und den zugehörigen Schaltplan zu zeichnen.

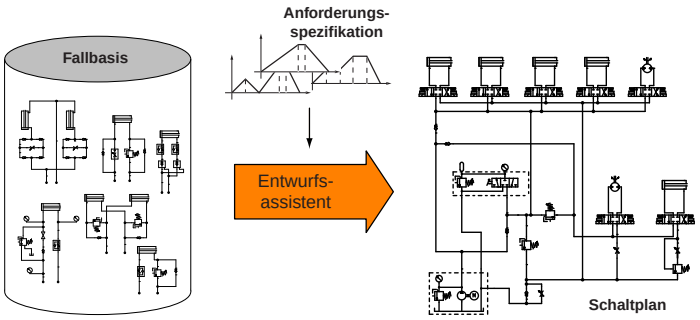

Abbildung 8: *Der Entwurfsassistent generiert zu einer Anforderungsspezifikation einen Schaltplan.*

Ingenieure haben uns bestätigt, daß die automatisch erzeugten Entwürfe sinnvoll sind und vom Menschen als Ausgangspunkt bei der Lösung komplexer Entwurfsprobleme genutzt werden können. Für einen umfangreichen Praxistest ist der Entwurfsassistent zur Zeit nicht gerüstet; hierfür müssen die Fallbasis, die Skalierungs- und die Modifikationsregeln zusammen mit Spezialisten für einen konkreten Einsatzbereich zugeschnitten werden.

Die Entwicklung und Umsetzung des Entwurfsassistenten zeigt, daß mit fallbasierten Techniken anspruchsvolles Entwurfswissen implementiert werden kann. Sie zeigt aber auch, daß der fallbasierte Ansatz hierfür kein Patentrezept liefert, sondern mit hohem technischen Sachverstand für die jeweilige Aufgabe spezialisiert werden muß.

Literatur

[1] A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICOM*, Seite 39–59, 1994.

[2] R. Barletta and D. Hennessy. Case Adaptation in Autoclave Layout Design. In K. J. Hammond, Editor, *Proceedings: Case-Based Reasoning Workshop*, Seite 203–207. Morgan Kaufmann Publishers, 1989.

[3] J. G. Carbonell. Derivational Analogy: a Theory of Reconstructive Problem Solving and Expertise Acquisition. In R. Michalski, J. Carbonnel, and T. Mitchell, Editoren, *Machine Learning: an Artificial Intelligence Approach*, Band 2, Seite 371–392, Los Altos, CA, Morgan Kaufmann Publishers, 1996.

[4] J. S. Gero. Design Prototypes: A Knowledge Representation Scheme for Design. *AI Magazine*, 11:26–36, 1990.

[5] T. R. Hinrichs and J. L. Kolodner. The Roles of Adaptation in Case-Based Design. In *Proceedings AAAI-91*. Cambridge, MA: AAAI Press / MIT Press, 1991.

[6] M. Hoffmann. *Zur Automatisierung des Designprozesses fluidischer Systeme*. eingereichte Dissertation, Universität Paderborn, Fachbereich Mathematik und Informatik, 1999.

[7] J. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(4), 1983.

[8] J. Kolodner. *Case-Based Reasoning*. San Mateo, CA, Morgan Kaufmann Publishers, 1993.

[9] D. B. Leake. Case-Based Reasoning: Issues, Methods, and Technology, 1995.

[10] M. M. Richter. Introduction to CBR. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Weß, Editoren, *Case-Based Reasoning Technology. From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400, Seite 1–15. Springer-Verlag, Berlin, 1998.

[11] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale, New Jersey, 1977.

[12] R. C. Schank. *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, Cambridge, UK, 1982.

[13] B. Stein. *Functional Models in Configuration Systems*. Dissertation, Universität Paderborn, Fachbereich Mathematik und Informatik, 1995.

[14] B. Stein. Optimized Design of Fluidic Drives—Objectives and Concepts. Technical Report tr-ri-97-189, Universität Paderborn, Fachbereich Mathematik und Informatik, Aug. 1996.

[15] B. Stein and M. Hoffmann. On Adaptation in Case-Based Design. In *Proceedings of the third International ICSC Symposia on Soft Computing (SOCO '99)*. ICSC Academic Press, 1998.

[16] I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann Publishers, 1997.

[17] S. Weß. *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik / Grundlagen, Systeme und Entscheidungen*. Dissertation, Universität Kaiserslautern, 1995.