

KONWERK – ein modulares Konfigurierungswerkzeug

Andreas Günter¹

Universität Hamburg, FB Informatik
Hamburger Informatik Technologie-Center e.V.,
Vogt-Kölln-Str. 30, 22527 Hamburg
e-mail: guenter@informatik.uni-hamburg.de

Kurzfassung

Konzeption und Architektur des modularen Konfigurierungswerkzeuges KONWERK werden beschrieben. Die Module von KONWERK sind Problemlösungsbausteine aus denen speziell für eine Domäne Anwendungssysteme zusammengefügt werden können. Schwerpunkt des Beitrages sind die Grundkonzeption von KONWERK, die Modulstruktur und die Schnittstellen zwischen den Modulen sowie den jeweiligen domänenspezifischen Oberflächen.

1. Einleitung

Die Entwicklung von Expertensystemen wird durch die Bereitstellung von speziellen Programmiersprachen und Werkzeugen unterstützt. In letzter Zeit werden verstärkt Werkzeuge entwickelt, deren Anwendungsbereich auf eine Anwendungs- oder Problemklasse beschränkt ist (vgl. [Harmon93]). Zentrales Ziel dieser Arbeiten ist es, durch die Bereitstellung von Mechanismen – welche speziell auf einen Bereich zugeschnitten sind – die Entwicklungszeit von Anwendungssystemen deutlich zu verkürzen.

Im vorliegenden Beitrag wird ein sogenanntes *Problemklassenspezifisches Werkzeug* für den Bereich der Konfigurierung vorgestellt (KONWERK). Ähnliche Entwicklungen sind auch in anderen Problemklassen erkennbar: z.B. die Planners-Workbench (s. [Appelrath94], [Sauer93]), die Diagnose-Werkbank MOLTKE (s. [Althoff93]), die Behavior-Workbench zur modellbasierten Diagnose (s. [Behavior93]) oder der Diagnosebaukasten D3 (s. [Puppe94]). Den Ansätzen ist gemeinsam, daß jeweils Problemlösungsverfahren oder Methoden in einem Rahmensystem zur Verfügung gestellt werden. Allgemeine Untersuchungen zu Verfahren und deren einheitliche Darstellung finden sich in [Chandrasekaran88] über *Generic Tasks* und in [Puppe90] zu *Problemlösungsmethoden* in Expertensystemen.

Ausgangspunkt für das Konzept von KONWERK waren auch die Erfahrungen bei der Entwicklung und dem Einsatz des Expertensystemkerns PLAKON (vgl. [Cunis91], [Günter93]).

¹ Das diesem Beitrag zugrundeliegende Vorhaben wird mit Mitteln des Bundesministers für Forschung und Technologie (Förderkennzeichen ITW9101A6, Verbundvorhaben: PROKON) gefördert. Die Verantwortung für den Inhalt liegt bei dem Autor.

Basierend auf dem in Abschnitt 1.1 vorgestellten Verständnis von Konfigurierungsproblemen werden vier zentrale Aufgaben für Konfigurierungssysteme identifiziert. Für diese Aufgaben stellt KONWERK sogenannte Unterstützungs-, Basis- und Erweiterungsmodule bereit (Abschnitt 2 - 5). Die Zusammenstellung dieser Module zu KONWERK-Varianten und Anwendungssystemen wird in Abschnitt 6 und 7 geschildert. Die Anbindung von domänen-spezifischen Oberflächen wird in Abschnitt 8, einige Aspekte der Vorgehensweise bei der Entwicklung und Integration der Module in Abschnitt 9 beschrieben. Der Beitrag schließt mit einer kurzen Beschreibung von Meta-KONWERK (Abschnitt 10) sowie einer Diskussion und Zusammenfassung (Abschnitt 11).

1.1. Konfigurieren

Konfigurieren wird als Syntheseaufgabe verstanden, bei der Objekte aus der Domänen zu einer Konfiguration zusammengefügt werden. Dabei sind gegeben:

- Eine Spezifikation der Aufgabe (Konfigurationsziele), die insbesondere angibt, welche Anforderungen die zu erzeugende Konfiguration erfüllen soll.
- Eine Menge von Objekten und deren Eigenschaften.
- Eine Menge von Relationen und Restriktionen zwischen den Objekten. Dabei sind für die Konfigurierung insbesondere die kompositionellen Beziehungen von Bedeutung.
- Wissen über die Vorgehensweise bei der Konfigurierung.

Eigenschaften können auch als Relationen aufgefaßt werden (wie z.B. in terminologischen Systemen), bei der hier vorgenommenen Unterteilung wird unterschieden zwischen

- Relationen zwischen mehreren Objekten der Domäne und
- den Eigenschaften (im Sinne von Parametern) eines Domänenobjektes.

Es gibt verschiedene Einteilungen von Syntheseaufgaben; die für das Konfigurieren interessanteste ist die Klassifizierung von Designaufgaben von Brown und Chandrasekaran [Brown89]. Die folgenden Bemerkungen beziehen sich auf deren Einteilung in drei Design-Klassen. Ausgehend von dem allgemeinen Modell kann das sogenannte *Routine-Konfigurieren* (vergleichbar mit der Design-Klasse 3: Routine-Design) als eine Problemstellung beschrieben werden, bei der die Spezifikation der Objekte, ihre Eigenschaften und ihre kompositionelle Struktur bereits vorgegeben sind und die Lösungsfindung auf einer ebenfalls bekannten Vorgehensweise basiert. Die Aufgabe lautet dann, aus den bekannten Objekten und nach einem vorgegebenen Ablaufplan eine Konfiguration zusammenzubauen, welche die Aufgabenspezifikation erfüllt. Ein Beispiel hierfür ist die Angebotserstellung für technische Systeme auf der Basis einer vollständigen Beschreibung von Komponenten, deren Eigenschaften, einer kompositionellen Struktur und Kombinationsrestriktionen.

Sind die genannten Einschränkungen nicht in der Domäne gegeben, dann kann von *Innovativem Konfigurieren* (vgl. Design-Klasse 2) oder noch weitergehend von *Kreativem*

Konfigurieren (vgl. Design-Klasse 1) gesprochen werden. Dabei sind die Übergänge zwischen den verschiedenen Konfigurierungsklassen fließend, insofern ist eine eindeutige Zuordnung selten möglich und zwischen dem Routine-Konfigurieren und dem Kreativen Konfigurieren könnten mehrere Abstufungen eingeführt werden. Aus diesem Grunde spreche ich von einer Skala, die vom Routine-Konfigurieren bis zum Kreativen Konfigurieren reicht. Ein Beispiel für Kreatives Konfigurieren ist der "künstlerische" Architekturstwurf [Carrara94]. Zu diskutieren bleibt, ob in diesem Fall der Begriff *Konfigurieren* noch zutrifft. In [Tank91] wird z.B. Konfigurieren auf den Bereich des Routine-Konfigurierens eingeschränkt. Übertragen auf den Maschinenbau und die Konstruktionslehre kann hier das Spektrum von der Prinzipkonstruktion, Variantenkonstruktion, Anpassungskonstruktion bis hin zur Neukonstruktion als Vergleich herangezogen werden (vgl. [Breiing91], [Kratz91]).

2. Architektur von KONWERK

Eine zentrale Anforderung an Werkzeugsysteme ist, daß tragfähige Konzepte für unterschiedliche Domänen gegeben sein müssen, aber auch keine unnötig komplexen Mechanismen bereitgestellt werden (d.h. alle notwendigen Mechanismen, aber auch nur diese). Unser Lösungsansatz sieht hierzu eine Unterteilung in aufeinander aufbauende Problemlösungsmodule vor. KONWERK wird als Modulbaukasten realisiert. Für die folgenden Aufgaben von Konfigurierungssystemen werden in KONWERK Module bereitgestellt:

- zur Repräsentation von Domänenobjekten und deren Eigenschaften,
- zur Repräsentation und Verarbeitung von Relationen und Restriktionen,
- zur Formulierung von Zielspezifikationen (Aufgabenstellung) und
- zur Steuerung (Kontrolle) des Konfigurierungsvorganges.

Für die Aufgaben werden sowohl einfache Standardmechanismen (Basismodule) als auch konzeptionell weiterreichende Methoden in Form von Erweiterungsmodulen in einem gemeinsamen Rahmensystem angeboten. In Anlehnung an [Syska91] entsprechen obige Aufgaben den Modularten und die verschiedenen Module den Modulausprägungen.

Abbildung 1 zeigt die Gesamtarchitektur von KONWERK. Auf die Basis-, Erweiterungs- und Unterstützungsmodule wird detaillierter eingegangen (Abschnitt 3-5). Als Hardwarebasis werden SUN- und MAC-Workstations eingesetzt. Aufgrund der Verwendung der Standards von CommonLisp, CommonLispObjectSystems (CLOS) und CommonLispInterfaceManager (CLIM) ist die Portabilität gewährleistet. Insbesondere bei CLIM gab der Standardisierungsvorteil den Ausschlag gegenüber teilweise wesentlich effizienteren Sprachen (s. Abschnitt 8). Bei der domänenunabhängigen Oberfläche des Werkzeuges KONWERK wird unterschieden zwischen der Wissensakquisition und der eigentlichen Konfigurierung. KONWERK unterstützt auch eine weitgehend interaktive Konfigurierung. Hierfür und für eine Erklärungskomponente sind eigene Module vorgesehen.

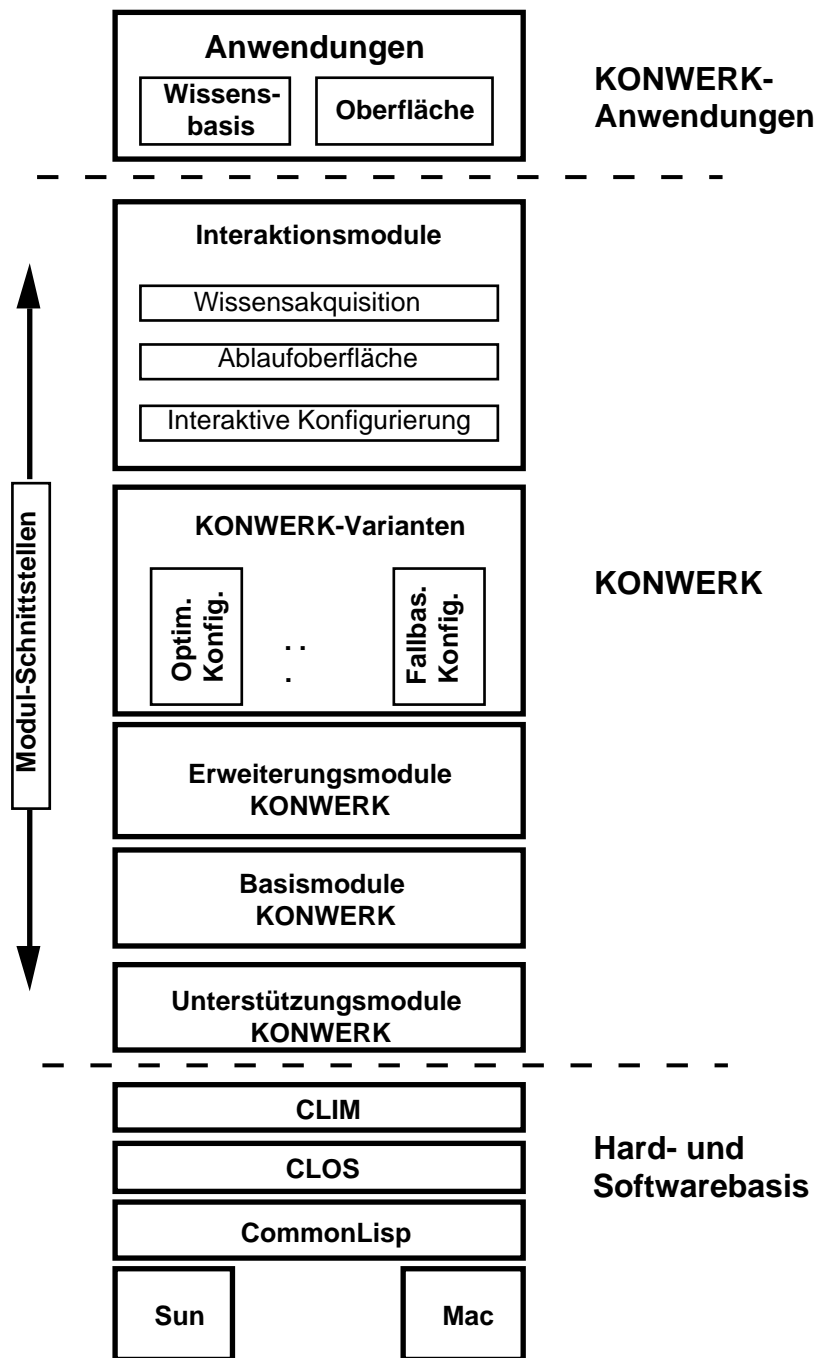


Abbildung 1: Architekturschema von KONWERK

Die Anwendungen von KONWERK (Abschnitt 7) bestehen jeweils aus einer Wissensbasis und in den meisten Fällen aus einer domänenspezifischen graphischen Oberfläche. Für eine KONWERK-Anwendung werden entweder

- die geeigneten Basismodule und Erweiterungen ausgewählt und zu einem speziellen Konfigurierungssystem zusammengesetzt, oder
- eine der vordefinierten problemspezifischen Konfigurierungs-Varianten ausgewählt und ggf. durch Erweiterungsmodule ergänzt.

Den Kern von KONWERK bilden die in den nächsten Abschnitten erläuterten Module und die Schnittstellen zwischen ihnen.

3. Basismodule

Die Basismodule (BM) bilden die Standardkomponenten von KONWERK für die zentralen Aufgaben von Konfigurierungssystemen. In Tabelle 1 findet sich eine Auflistung der Basismodule und ihre Zuordnung zu den Aufgaben von Konfigurierungssystemen.

Aufgabe	Basismodule
Repr. von Domänenobjekten (DO)	• Begriffshierarchie
Repr. und Auswertung von Relationen	• Constraints
Steuerung / Kontrolle	• Gesteuerte Tiefensuche mit Backtracking
Zielspezifikation	• Zielobjekt, weitere DO und Eigenschaften von DO

Tabelle 1: Basismodule von KONWERK

Die Basismodule von KONWERK im einzelnen:

- Begriffshierarchie zur Repräsentation von Domänenobjekten
Domänenobjekte (DO) und ihre Eigenschaften werden in einer Spezialisierungshierarchie repräsentiert (vgl. [Cunis91]). Die Wertebereiche der Eigenschaften werden durch sogenannte Objektdeskriptoren vordefiniert. Dabei existiert eine ausgezeichnete Relation zur Beschreibung der kompositionellen Hierarchie (has-parts/part-of).
- Constraints zur Repräsentation von Restriktionen
Die Repräsentation von Konfigurierungsrestriktionen und ihre Verwaltung während des Konfigurierungsvorganges erfolgt in einem Constraint-Netz. Unser dreistufiges Constraint-Modell besteht aus Constraint-Relationen, konzeptuellen Constraints und den im Constraint-Netz instantiierten Constraints (vgl. Constraintsystem von PLAKON in [Cunis91]). Zur Auswertung wird Constraint-Propagierung eingesetzt.
- Gesteuerte Tiefensuche mit Backtracking als Ablaufsteuerung
Der Ablauf basiert auf einer gesteuerten Tiefensuche mit chronologischem Backtracking. Das Kontrollwissen wird in Form von Strategien repräsentiert, welche die Reihenfolge der Konfigurierungsschritte steuern, die aus einer Agenda ausgewählt werden. Weiterhin kann auf Teilaspekte fokussiert und die Anwendung von Bearbeitungsverfahren gesteuert werden. Die Anwendung der Strategien wird durch Bedingungen aktiviert. Die Instanzen der Teilkonfigurationen werden in einem Viewpoint-Mechanismus verwaltet, der es insbesondere ermöglicht, zu früheren Teilkonfigurationen zurückzugehen (Backtracking). Die Steuerung entspricht einer einfachen Version der Begriffshierarchie-orientierten Kontrolle von PLAKON (s. [Günter91]).
- Zielspezifikation
Eine Aufgabe wird durch die Auswahl eines Zielobjektes und optional durch die Vorgabe von weiteren Domänenobjekten (Komponenten) und der Einschränkung ihrer Eigenschaften (Slots) spezifiziert.

Die Basismodule von KONWERK sind vergleichbar mit einer einfachen Version von PLAKON. Auf der Ebene der Implementierung gibt es allerdings deutliche Unterschiede, insbesondere auch durch den modularen Aufbau von KONWERK bedingt (siehe Abschnitt 5).

4. Erweiterungsmodule

Ergänzungen zu den Basismodulen werden als *Erweiterungsmodule* (EM) bezeichnet. In Tabelle 2 wird ein Überblick gegeben², dabei werden die Module jeweils den Aufgaben von Konfigurierungssystemen zugeordnet. Eine Kurzbeschreibung aller EM und Referenzen auf die jeweils zugrundeliegenden Konzepte finden sich in [Günter94].

Beispielhaft sei das Prinzip des Modulbaukastens an den Modulen zur Steuerung des Konfigurierungsvorganges und hier der Auflösung von Konfigurationskonflikten erläutert. Im Basismodul "Kontrolle" wird zur Auflösung von Konfigurationskonflikten nur chronologisches Backtracking verwendet. Dabei wird immer zur zeitlich letzten Konfigurierungsentscheidung zurückgegangen und diese modifiziert. Ergänzungen zu dieser "einfachen" Form der Auflösung von Konfigurationskonflikten werden in Form von Erweiterungsmodulen bereitgestellt (Details zu den Konzepten der verschiedenen Konfliktauflösungsverfahren finden sich in [Günter93a]):

- Wissensbasiertes Backtracking

Im Gegensatz zum chronologischen Backtracking kann zu anderen Entscheidungen (TK) zurückgegangen werden, wodurch Konfigurierungsschritte übersprungen werden. Dieses Vorgehen kann durch domänenspezifisches Wissen gesteuert werden. Wissensbasiertes Backtracking liefert eine der früheren Teillösungen (Teilkonfigurationen) als Ausgangspunkt für die weitere Konfigurierung. Dabei kann ein Vorschlag für die erneute Durchführung eines Konfigurierungsschrittes mit vorgegeben werden, auch in Form einer Tendenzangabe.

- Wissensbasiertes Backtracking mit Datenübernahme

In Erweiterung zum wissensbasierten Backtracking unterstützt dieses Modul die Übernahme aller nicht vom Konflikt betroffenen Entscheidungen. Es wird nicht zu einer "alten" (früheren) Teilkonfiguration (TK) zurückgegangen, sondern eine neue TK erzeugt und in diese die Domänenobjekte der "alten" TK und die vom Konflikt unabhängigen Entscheidungen eingetragen. Bei diesem Verfahren werden unnötige Mehrfachberechnungen vermieden; allerdings entsteht zusätzlicher Aufwand durch die Verwaltung der Abhängigkeiten zwischen den Konfigurierungsschritten in einer Begründungsverwaltung (TMS).

- Reparaturanweisungen

Allgemein wird beim Backtracking zu einer früheren TK zurückgegangen und von dort aus erneut ein Lösungsversuch gestartet. Den Reparaturanweisungen liegt ein anderes Konzept

² Die aufgeführten Konzepte können in dieser Übersicht nur Schlagwortcharakter haben. Anhand einer Anwendung und der Module zur Auflösung von Konflikten wird detaillierter auf einige der Module eingegangen.

zugrunde, sie modifizieren die konfliktbehaftete TK direkt, d.h. ohne Bezug auf die Konfigurierungshistorie, solange bis sie widerspruchsfrei ist. Hierbei entstehen typischerweise nicht-monotone Veränderungen an der Teilkonfiguration.

Ergebnis dieser Reparaturen ist eine neue – reparierte – Teilkonfiguration. Diese gilt als "Startpunkt" des Konfigurierungsvorganges nach einem Konflikt.

Aufgabe	Erweiterungsmodule
Repräsentation von Domänenobjekten	<ul style="list-style-type: none"> • Begriffshierarchie mit Sichten und Mixins • Begriffshierarchie mit unscharfen Eigenschaften • Begriffshierarchie mit linguistischen Werten • Räumliche Eigenschaften (2D und 3D) • Unscharfe Vererbungspfade (Fuzzy-Begriffshierarchie) • Maßeinheiten
Repräsentation und Auswertung von Relationen	<ul style="list-style-type: none"> • Berechnungsfunktionen • Regeln zur Repräsentation von Konfigurierungsheuristiken • Constraints mit unscharfen Werten • Räumliche Relationen (2D und 3D) • Spezifikation von Relationen als Gleichungen • Auswertung von taxonomischen Informationen • Constraint-Relaxation
Steuerung (Kontrolle)	<ul style="list-style-type: none"> • Fallbasiertes Konfigurieren • Exploratives Konfigurieren • Interaktives Konfigurieren • Optimierungs-Grund-Modul • Optimierungsmodul MADM • Lineare Optimierung • Nicht-lineare Optimierung mit Constraints • Branch-and-Bound-Optimierung • Wissensbasiertes Backtracking • Wissensbasiertes Backtracking mit Datenübernahme • Reparaturanweisungen • Vollständige Suche über einen expliziten Lösungsraum • ATMS-basiertes Konfigurieren
Zielspezifikation	<ul style="list-style-type: none"> • Anforderungsmodellierung (funktionale und unscharfe Anforderungen, Formulierung von Optimierungszielen)
Spezielle Erweiterungen	<ul style="list-style-type: none"> • Integration von Simulationsverfahren • Kompilation von Constraints • Begründungsverwaltung (TMS) • Schnittstellen zu Datenbanken • Schnittstellen zu CAD-Systemen

Tabelle 2: Erweiterungsmodule von KONWERK

Daneben gibt es weitere Module zur Konfigurationssteuerung (siehe Tabelle 2), welche mit den hier aufgeführten kombiniert werden können. Allerdings existieren zwischen den Modulen auch Abhängigkeiten in Form von Modulanforderungen und dem Verbot von Modulkombinationen. Z.B. erfordert das Modul zum "Wissensbasierten Backtracking mit Datenübernahme" das Modul TMS, da die Abhängigkeiten zwischen den Konfigurationsentscheidungen verwaltet werden müssen.

Die angebotenen Verfahren zur Konfliktauflösung (einschließlich einer interaktiven Variante) haben Vor- und Nachteile. Zentraler Vorteil der hier vorgestellten Modularisierung ist die

Auswahl eines Moduls je nach den Erfordernissen der Anwendung. Auch das Testen von Verfahren für eine Anwendung ist ein Vorteil der modularen Struktur.

5. Module zur Unterstützung der Implementierung

Basis- und Erweiterungsmodule sind Module auf der Problemlösungsebene, die unabhängig von den Details der konkreten Implementation sind. Daneben entstand bei der Entwicklung von KONWERK auch ein Bedarf an Mechanismen und Funktionen zur Unterstützung bei der Implementierung obiger Module. In KONWERK werden für diese Zwecke eine Reihe von sogenannten Unterstützungsmodulen angeboten:

- Für die Implementation von KONWERK notwendige objektorientierte Erweiterungen von CLOS,
- Definition von sogenannten Schnittstellenklassen und -methoden,
- Slot-Kontexte (Viewpoints) zur Unterstützung von Backtracking,
- Slot-Optionen und
- Funktionen für einen Strukturvergleich.

Diese Module sind aus konzeptioneller Sicht auf der Ebene der Programmierung angesiedelt. Von besonderer Bedeutung für KONWERK ist das Modul über Schnittstellenklassen und -methoden (vgl. [Hotz94]). Aufgrund der Kommunikation und des Zusammenspiels der Module untereinander besteht die Notwendigkeit in KONWERK Objekte einzuführen, welche von mehreren Modulen spezialisiert, erweitert, instantiiert und mit entsprechenden Methoden versehen werden können. Dies muß geeignet unterstützt werden unter der Voraussetzung, daß die Erweiterungs- und Basismodule auf der Implementationsebene weitgehend unabhängig voneinander sein sollen und somit die entsprechenden Erweiterungen der Objektklassen in den Erweiterungsmodulen nicht bekannt sind.

Aufgrund obiger Anforderungen wurde das Konzept der Schnittstellenklassen eingeführt. Es ermöglicht die Spezialisierung einer Klasse ohne Kenntnis der bisher "speziellsten" Unterklasse. Z.B. wird im Basismodul zur Ablaufsteuerung die Schnittstellenklasse `tiefensuche-strategie` mit Hilfe des Konstruktes `(def-interface-class ...)` eingeführt. In verschiedenen Erweiterungsmodulen können nun Erweiterungen dieser Strategie eingeführt werden, ohne Kenntnis voneinander. Bei Verwendung von `(def-extended-interface-class ...)` wird die neue Klasse automatisch unter die bisher speziellste Klasse von `tiefensuche-strategie` eingehängt. Dies geschieht ohne Kenntnis der entsprechenden Klasse, bei den Definitionen in den Erweiterungsmodulen muß nur die Schnittstellenklasse `tiefensuche-strategie` angegeben werden. Auch bei der Methodenselektion wird bei Angabe der Schnittstellenklasse jeweils die speziellste Methode selektiert. Mit `(call-next-method)` können – bzw. müssen – dann die anderen Methoden ausgeführt werden.

6. KONWERK-Varianten

KONWERK-Varianten bezeichnen eine Zusammenstellung von mehreren Basismodulen und Erweiterungen, um spezielle Anforderungen und Konfigurierungsmethodiken geeignet zu unterstützen. Eine Variante setzt sich aus mehreren Modulen zusammen, die integriert als eine *problemspezifische Konfigurierungs-Variante* angeboten werden. Ein Beispiel ist das *Optimierungsbasierte Konfigurieren*, es besteht aus allen BM, dem Modul zur Spezifikation von Relationen als Gleichungen, allen EM zur Optimierung (siehe Tabelle 2, vgl. [Funke94]) und der Anforderungsmodellierung.

Eine Variante entspricht den Problemlösungsmethoden wie sie bei Puppe [Puppe90] eingeführt werden. Dort wird z.B. die Methode *vorschlagen-und-vertauschen* beschrieben. Die zentralen Aspekte dieser Methode finden sich in den Steuerungsmodulen "Gesteuerte Tiefensuche" und "Wissensbasiertes Backtracking" wieder, daneben werden für ein Anwendungssystem Module für die anderen Aufgaben (z.B. Repräsentation und Auswertung von Restriktionen) benötigt. Wichtig ist, daß die anderen Module weitgehend unabhängig von den Steuerungsmodulen sind. So können z.B. beliebige Module zur Repräsentation von Domänenobjekten hiermit kombiniert werden.

Auch die *Generic Tasks* nach Chandrasekaran (s. [Chandrasekaran88]) lassen sich auf die Varianten abbilden. Bei den *Generic Tasks* werden sowohl Wissensrepräsentation als auch Inferenzverfahren zusammen für ein Verfahren spezifiziert.

Übertragen auf das Modulkonzept von KONWERK stellen sowohl die Problemlösungsmethoden als auch die *Generic Tasks* somit eine abgeschlossene KONWERK-Variante dar, welche einem Knowledge Engineer zur Verfügung gestellt wird. Die einzelnen Bestandteile einer solche Variante sind aber im Gegensatz zu KONWERK nicht beeinflussbar und änderbar.

7. KONWERK-Anwendungen

Die Strukturierung als Modulbaukasten erlaubt eine flexible Zusammenstellung von Modulen für konkrete KONWERK-Anwendungen. Das Anwendungsbeispiel der Hamburger-Projektgruppe ist die Konfigurierung von Passagierkabinen des AIRBUS A340 [Kopisch92]. Ein Kabinenlayout spezifiziert die Passagierkabine eines Verkehrsflugzeuges, indem es die Anzahl, die Anordnung, die Art und die Lage der Einrichtungsgegenstände angibt. Einrichtungsgegenständen sind hier die Objekte, die zum Einbau in die Passagierkabine zur Verfügung stehen, wie Passagiersitze, Küchen, Toiletten, Flugbegleitersitze, Stauschränke etc. Deren Gesamtheit definiert die Beförderungskapazität, den Komfort etc. der Passagierkabine. Das Problem bei der Erstellung eines Kabinenlayouts besteht darin, daß eine große Zahl von Vorgaben (Wünsche der Fluggesellschaft, gesetzliche Zulassungsbestimmungen und technische Anforderungen) gleichzeitig erfüllt werden müssen. Außerdem wird Optimalität bezüglich verschiedener Kriterien angestrebt (z.B. Anzahl der Sitze, Komfort, Kosten, Lieferzeit). Diese

Kriterien müssen von Auftrag zu Auftrag mit unterschiedlichen Gewichten berücksichtigt werden.

Unter dem Namen XKL wurde mit PLAKON bereits für diese Anwendung ein Prototyp realisiert [Kopisch92]. Für das neue XKL mit KONWERK sind folgende Module vorgesehen:

- Die Unterstützungs- und Basismodule von KONWERK
- Räumliche Eigenschaften
Hiermit werden die räumliche Eigenschaften der Domänenobjekte repräsentiert. So hat z.B. eine Küche bestimmte Maße.
- Maßeinheiten
Insbesondere für die räumlichen Eigenschaften sind Maßangaben notwendig, aber auch z.B. für Gewichts- und Preisangaben.
- Begriffshierarchie mit Sichten und Mixins
Wird von dem Modul zur Repräsentation von räumlichen Eigenschaften benötigt und stellt insbesondere die Möglichkeit zur Vererbung über mehrere Pfade zur Verfügung.
- Räumliche Relationen
Spezielle Verfahren zur effizienten Auswertung von räumlichen Relationen zwischen den Domänenobjekten mittels Constraint-Propagation.
- Anforderungsmodellierung
Dieses Modul dient zur Spezifikation von funktionalen Anforderungen an ein Kabinenlayout und zur Angabe von Optimierungszielen.
- Optimierungs-Grundmodul
Dieses Modul bildet die Schnittstellen zu den speziellen Optimierungsmethoden, es wird bei jeder Optimierung benötigt (s. [Funke94]).
- Nicht-Lineare-Optimierung mit Constraints
Die Optimierung erfolgt mit nicht-linearen Optimierungsverfahren auf der Basis von mit Hilfe von Constraints formulierten Restriktionen und Gleichungen.
- Reparaturanweisungen
Nach einem Konfigurationskonflikt kann die Teillösung durch direkte Modifikationen korrigiert werden (auch interaktiv). Ist dieses Verfahren nicht erfolgreich oder nicht einsetzbar, dann wird chronologisches Backtracking durchgeführt.

Im PROKON-Projekt werden darüberhinaus von den anderen Projektpartnern folgende Anwendungen mit KONWERK realisiert:

- Auswahl und Auslegung von Gleitlagern
- Konfigurierung und Auslegung von Gelenkwellen
- Erstellen eines optimalen Layouts für ein Logistiksystem
- Entwurf von Flüssigkristallen

Die günstigste Modulkombination für XKL und die anderen Anwendungen wird sich erst nach empirischen Tests ergeben. Die Möglichkeiten des Testens von verschiedenen Wissensrepräsentationsformen und Problemlösungsmethoden wird durch die Konzeption als Modulbaukasten wesentlich unterstützt. Neben der Funktionalität der Module ist auch deren Effizienz und Interaktivität bei der letztendlichen Modulkombination von Bedeutung.

8. Anbindung von Benutzerschnittstellen

KONWERK als ein Werkzeugsystem besitzt domänenunabhängige Benutzerschnittstellen. Es steht eine einfache Terminalversion (ascii) zur Verfügung und eine mit CLIM realisierte graphische Oberfläche. Die Module von KONWERK sollen vollständig unabhängig von einer konkreten Benutzerschnittstelle sein. Bereits bei den beiden Versionen erfordert dies die Spezifikation von Interaktionen und Ausgaben zwischen KONWERK und einem Benutzer unabhängig von einer konkreten Oberfläche. Insbesondere aber zur Anbindung von domänenspezifischen Oberflächen ist eine derartige Spezifikation notwendig.

Für KONWERK wurde das Konzept der *Interaktionszustände* entwickelt. Sie spezifizieren, welche Interaktionen in einer bestimmten Problemlösungssituation möglich und notwendig sind (s. [Buhr94]). In welchen Interaktionszustand jeweils gewechselt wird, entscheidet KONWERK. KONWERK teilt der Oberfläche mit, in welchem Interaktionszustand es sich befindet, daraufhin werden von der Oberfläche die Methoden für den jeweiligen Interaktionszustand ausgeführt. Die Ablaufsteuerung erfolgt durch KONWERK, dies ist bedingt durch die zentrale Anforderung nach Unabhängigkeit des Problemlösungsverfahrens von der Benutzerschnittstelle³.

Ein Interaktionszustand ist unabhängig von einer konkreten Oberfläche definiert. Aufgrund der objektorientierten Programmierung kann ein Entwickler ein eigenes Oberflächenobjekt definieren und dafür entsprechende Methoden (zu den Interaktionszuständen) schreiben. Somit kann spezifiziert werden, ob, wie und welche Informationen in einem Interaktionszustand dem Benutzer präsentiert wird. Von KONWERK sind alle zulässigen Interaktionszustände vordefiniert und für den Großteil gibt es Methoden in den beiden Oberflächenversionen, diese können von einem Entwickler (sukzessive) spezialisiert werden.

Es ist auch möglich eine graphische Schnittstelle in einer anderen Sprache als CLIM zu spezifizieren. Voraussetzung ist allerdings, daß die "Logik" weiterhin in Lisp enthalten ist und nur die Präsentation und das Erfassen der Interaktionen in einer anderen Sprache abläuft. Auch wegen Effizienzproblemen von CLIM ist dies ein interessanter Punkt. Es wurden bereits Teilimplementationen in C realisiert und an Lisp angebunden.

³ Allerdings hat ein Benutzer die Möglichkeit KONWERK durch verschiedene Kommandos zu steuern, somit kann ein Benutzer über die Oberfläche auch (indirekt) die Arbeit des KONWERK-Kerns lenken.

Der Ablauf zwischen KONWERK und Interaktionszuständen läßt sich wie folgt darstellen:

1. KONWERK erkennt, daß eine Interaktion notwendig ist.
2. KONWERK setzt den Interaktionszustand und übergibt die definierten Parameter.
3. Ausführen der Methoden. Je nach Anwendungsoberfläche können an dieser Stelle dem Benutzer unterschiedliche Interaktionsformen und Darstellungen präsentiert werden.
4. In Abhängigkeit vom Interaktionszustand ggf. warten auf eine Benutzerentscheidung oder ein Kommando.
5. Weiterer Ablauf der Problemlösung.

Insbesondere bei der Darstellung und Anzeige von Ergebnissen oder allgemeinen Informationen über die Problemlösung wird es deutliche Unterschiede zwischen den einzelnen Anwendungssystemen geben. Hingegen sind die Unterschiede bei der Wissensakquisition geringer.

9. Vorgehensweise bei der Entwicklung

Ein zentraler Aspekt bei der Konzeption war die verteilte Softwareentwicklung (5 Standorte) und die weitgehende Unabhängigkeit der Module. Um das Problem zu umgehen, fertige Programmpakete nachträglich zusammenfügen zu müssen, wurden zunächst die Modulschnittstellen definiert und für die Schnittstellenobjekte und -methoden einfache Implementationen realisiert. D.h. als erstes wurde die Integration der Module konzipiert und auch programmiert.

Die Vorgehensweise bei der Integration wird am Beispiel der Module für das Backtracking verdeutlicht. Im Basismodul Kontrolle ist chronologisches Backtracking in der generischen Funktion `bestimme-aufsetzpunkt` realisiert, die über die Schnittstellenklasse der aktuellen Strategie selektiert wird. Im Basismodul gibt es entsprechend eine Methode für die tiefensuche-strategie (siehe Abschnitt 5), welche ein chronologisches Backtracking beinhaltet, indem zur letzten Teilkonfiguration zurückgegangen wird. Die Integration des EM für das wissensbasierte Backtracking wurde realisiert, indem eine speziellere Klasse `wb-tiefensuche-strategie` und die entsprechende Methode von `bestimme-aufsetzpunkt` definiert wurden. Die Methode – welche ein wissensbasiertes Backtracking realisieren soll – wählte zunächst zufällig eine Teilkonfiguration aus. Dies ist natürlich kein wissensbasiertes Backtracking, aber es wurde getestet, ob die Methode richtig aufgerufen wurde sowie eine korrekte (in diesem Falle allerdings nicht sinnvolle) Rückgabe lieferte und diese von KONWERK richtig interpretiert und verarbeitet wurde. Die Realisierung der eigentlichen Funktionalität erfolgt nach der erfolgreichen Integration in KONWERK. Im Beispiel wird auch ein Vorteil einer durchgängigen objektorientierten Programmierung deutlich. Falls die Methode für das wissensbasierte Backtracking nicht erfolgreich ist, dann kann mit `(call-next-method)` die nächst allgemeinere Methode – in diesem Beispiel für chronologisches Backtracking – aufgerufen werden.

Zum (verteilten) Testen der Module entwickelt jeder Projektpartner eine Test-Wissensbasis. Diese besteht aus exemplarischem Domänenwissen, wobei insbesondere die "Spezialitäten" der jeweiligen EM berücksichtigt werden. Beim Laden aller KONWERK-Module müssen diese Wissensbasen jeweils korrekt bearbeitet werden.

Zum aktuellen Zeitpunkt sind von KONWERK alle UM, BM und ein Großteil der EM implementiert. Zu jeder Anwendung existiert ein mit KONWERK realisierter Prototyp. XKL und KONWERK werden im Rahmen der Expertensystemausstellung auf der XPS-95 vorgeführt.

10. Meta-KONWERK

Das Zusammenfügen einer KONWERK-Anwendung aus einzelnen Modulen ist eine Konfigurierungsaufgabe. Somit bietet es sich an, diese Aufgabe durch KONWERK zu unterstützen. *Meta-KONWERK* ist eine KONWERK-Anwendung, die die Konfigurierung von Konfigurierungssystemen unterstützt. Die Domänenobjekte von Meta-KONWERK sind die KONWERK-Module, die Konfigurierungsrelationen sind die Restriktionen der Module untereinander, die Aufgabenstellung ist die Spezifikation eines Konfigurierungsproblems und das Kontrollwissen ist das Wissen über die Vorgehensweise eines Konfigurierungs-Spezialisten.

Es existiert eine Meta-KONWERK-Wissensbasis. Sie enthält alle Modulbeschreibungen und die Restriktionen zwischen den Modulen. D.h. insbesondere, daß sich Module gegenseitig ausschließen und Module einander bedingen. Z.B. schließen sich die Module "Exploratives Konfigurieren" und "Auswertung von taxonomischen Informationen" gegenseitig aus, da sie von unterschiedlichen Annahmen über die Abgeschlossenheit der Wissensbasis ausgehen.

Meta-KONWERK ermöglicht bereits die interaktive Konfigurierung von KONWERK-Anwendungen, die allen Restriktionen genügen. Sie sind somit korrekt und enthalten alle notwendigen Module. Was fehlt – und was vermutlich auch noch länger fehlen wird – ist eine Formalisierung der Anwendungscharakteristika und deren Abbildung auf geeignete Module. Es können zwar KONWERK-Anwendungen interaktiv "fehlerfrei" konfiguriert werden, aber ob diese Konfiguration das gegebene Problem löst, ist vom Sachverstand des Benutzers abhängig.

11. Diskussion und Zusammenfassung

KONWERK besteht aus Problemlösungsmodulen. Es wird unterschieden zwischen Basismodulen (Standardkomponenten) und Erweiterungsmodulen, die entsprechend den Aufgaben bei der Konfigurierung strukturiert sind. Die für eine Anwendung benötigte Funktionalität wird durch "Konfigurieren" der Module zu einem Anwendungssystem bereitgestellt. Der modulare Aufbau von KONWERK wird unterstützt durch Schnittstellenklassen, die Anbindung von domänenspezifischen Oberflächen durch das Konzept der Interaktionszuständen.

Der Entwicklung von KONWERK liegt die Annahme zugrunde, daß es keinen allgemeinen Mechanismus gibt mit dem alle (oder ein Großteil) der Konfigurierungsprobleme effizient und adäquat bearbeitet werden können. Es gibt somit kein monolithisches allgemeines Verfahren für Konfigurierungsaufgaben. In dieser Situation erscheint ein modulares System, welches das Erstellen, eines für die jeweilige Anwendung geeigneten Systems aus einzelnen Modulen ermöglicht, eine vielversprechende Alternative. Dieser Punkt wird auch durch Erfahrungen mit PLAKON belegt (s. [Cunis91], [Günter93]). Obwohl der eigentliche Kernmechanismus von PLAKON relativ einfach ist, wurden in PLAKON zur Erfüllung immer weiterer Anforderungen aus unterschiedlichen Anwendungen die Mechanismen letztendlich zu komplex und unübersichtlich. Dies zeigte sich bei der Realisierung von einfachen Konfigurierungsaufgaben und bei der Einarbeitung von Anwendern oder neuen Mitarbeitern. In KONWERK gibt es – auch aus diesem Grunde – mit den Basismodulen eine Grundversion für Konfigurierungsaufgaben. Die Basismodule von KONWERK entsprechen in etwa einer einfachen PLAKON-Version.

PLAKON kann als Vorläufer von KONWERK bezeichnet werden. Die Erweiterungsmodule gehen aber weit über die Funktionalität von PLAKON hinaus. Aus meiner Sicht hätte auch keine Möglichkeit bestanden, die Funktionalität der Erweiterungsmodule in PLAKON zu integrieren. Der Fortschritt kann u.a. auch an der Anwendung XKL (Abschnitt 7, [Kopisch92]) aufgezeigt werden. Zentrale Verbesserungen gegenüber der PLAKON-Variante sind:

- die Repräsentation und Verwaltung von räumlichen Eigenschaften und Relationen,
- die Einbeziehung von Optimierungszielen,
- die Modellierung von funktionalen Anforderungen und
- die (interaktive) Reparatur oder Korrektur von (Teil-) Lösungen.

Auch in PLAKON wurde ein modularer Aufbau bereits konzipiert und in Ansätzen realisiert (s. [Syska91]). Im Gegensatz hierzu lassen sich die Module von KONWERK größtenteils auf einer "höheren" Ebene einordnen⁴, sie abstrahieren von Operatoren oder Parametern auf der Wissens- und Problemlösungsebene. Bei Syska findet sich keine Unterteilung in BM und EM, nicht die Anbindung von Benutzerschnittstellen und die Definition von Schnittstellenklassen zwischen den Modulen. Die grundsätzliche Idee einer derartigen Modularisierung von Werkzeugen war aber auch bereits der Ausgangspunkt von Syska.

Die hier vorgestellte Strukturierung in Problemlösungsmodule liegt auf der technischen Ebene, d.h. nicht auf der Ebene des menschlichen Problemlösens. Eine derartige Strukturierung steht z.B. bei den konzeptuellen Modellen von KADS im Vordergrund. Für die Konfigurierung existieren hier allerdings bisher relativ wenige operationalisierbare Verfahren (vgl. [Bauer94]). Mir erscheint es in diesem Zusammenhang sinnvoller, eine Abbildung auf die Module der technischen Ebene vorzugeben als ein Werkzeugsystem aufbauend auf den menschlichen

⁴ Das gilt nicht für Unterstützungsmodule

Problemlösemethoden zu konzipieren. Dies ist von Vorteil für die Realisierung, kann aber Nachteile bei der Erklärbarkeit und Adäquatheit beinhalten. Für einige Module, z.B. ATMS-basiertes Konfigurieren, existieren auch keine äquivalenten Vorgehensweisen. Trotzdem ist dies ein interessantes und vielversprechendes Verfahren. Die meisten Werkzeuge zur Entwicklung von Expertensystemen bieten Module auf der Ebene der Programmierung an, z.B. Constraints in Babylon (s. [Christaller89]). Die Module von KONWERK lassen sich demgegenüber in Anlehnung an [Syska91] der Funktionsebene zuordnen, d.h. sie stellen eine Funktionalität zur Verfügung unabhängig von der Implementierung und dem menschlichen Vorgehen. Eine Ausnahme bilden die Unterstützungsmodule, sie sind der Ebene der Programmierung zuzuordnen. Die Basis- und Erweiterungsmodule stellen eine Funktionalität auf der Ebene der Problemlösung von Konfigurierungsaufgaben bereit. Sie unterscheiden sich damit von den Modulen

- der Wissensebene, wie sie z.B. in KADS spezifiziert werden und
- der Programmierenebene, wie sie von typischen XPS-Werkzeugen bereitgestellt werden.

Den Problemlösungsmethoden von Puppe [Puppe90], den Generic Tasks [Chandrasekaran88] und der Modularisierung von Syska [Syska91] ist gemeinsam, daß jeweils nur der Bereich des Routine-Konfigurierens betrachtet wird. Nach unserer Meinung ist aber gerade bei innovativen Konfigurierungsaufgaben eine flexible und variable Zusammenstellung von verschiedenen Methoden notwendig. Ein typisches Problem ist, daß eine Anwendung nur in seltenen Fällen mit genau einer Methode erfolgreich realisiert werden kann. Stattdessen sind jeweils mehrere Repräsentationsformen und Lösungsmethoden notwendig. Ein Anwendungsproblem kann somit nicht eindeutig einer Methode zugeordnet werden. KONWERK ermöglicht auch die Realisierung von hybriden Systemen, die verschiedene Repräsentationsformen und Problemlösungsmechanismen beinhalten und zwischen diese Methoden wechseln können.

Im Sinne von Puppe und Chandrasekaran stellen die KONWERK-Varianten *starke Problemlösungsmethoden* bzw. *Generic Tasks* für Konfigurierungsaufgaben dar (s. Abschnitt 6). Die Module von KONWERK lassen sich auf die Bestandteile dieser Verfahren abbilden. Aufgrund der offenen Modulstruktur von KONWERK können die einzelnen Module aber – unabhängig von ihrer Zugehörigkeit zu Varianten – miteinander kombiniert werden, dies ist bei den Problemlösungsmethoden und den Generic Tasks nicht vorgesehen.

Ein Entwickler kann bei der Verwendung von KONWERK genau die für seine Domäne notwendigen Module zu einem Anwendungssystem zusammenstellen. Dies erfordert allerdings – wie in Abschnitt 10 erwähnt – Erfahrung und Kompetenz im Bereich Konfigurierungsexpertensysteme. Der aus Anwendersicht wichtigste Vorteil ist, daß unnötige Komplexität eines Anwendungssystems und daraus resultierende Effizienz-, Wartungs- und Akzeptanzprobleme weitgehend vermieden werden. Allerdings wird bedingt durch die Möglichkeiten der Modulkombination, ein KONWERK-Anwendungssystem naturgemäß komplexer sein, als eine

direkte Implementation der benötigten Funktionalität. Andererseits können durch die Entwicklung wiederverwendbarer KI-Software-Bausteine in einem Modulbaukasten die Konfigurierungsmethoden wirtschaftlicher zur Anwendung gebracht werden. Bei Verwendung von KONWERK kann somit der Entwicklungsaufwand deutlich verringert werden.

Ein modularer Aufbau und eine Werkzeugentwicklung wie hier beschrieben, ist nicht nur im Expertensystembereich sinnvoll; dazu ein abschließendes Zitat von Wirth [Wirth94]:

“Ein weiterer Grund (Anm.: Ursache für komplexe Softwaresystemen) liegt im üblichem monolithischen Design. Beim Bau eines Systems werden alle erdenklichen Features sogleich berücksichtigt und eingebaut. Jeder Kunde muß für sie alle bezahlen, er hat keine Auswahl, obwohl er typischerweise nur wenige dieser Features benutzen wird. Die vernünftige Lösung besteht im Bau eines Systemkerns, der nur die wichtigen und unabdingbaren Features anbietet, jedoch leicht erweiterbar ist. Jeder Kunde kann dann jene Erweiterungen auswählen, die seinen spezifischen Bedürfnissen entsprechen.”

Literatur

- [**Althoff93**] Althoff, K.-D.:
Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme
Dissertation Uni Kaiserslautern; auch KI-Dissertationsreihe Nr. 23, infix-Verlag (1993)
- [**Appelrath94**] Appelrath, H.-J.; Bruns, R.; Henseler, H.; Sauer, J.:
Planner's Workbench - eine Werkzeugumgebung zur Modellierung und Konfigurierung von Ablaufplanungssystemen
in: Beiträge zum 8. Workshop "Planen und Konfigurieren", Kaiserslautern, SEKI Working Paper SWP-94-01, S. 3-9 (1994)
- [**Bauer94**] Bauer, C.; Löckenhoff, C.:
Das ESPRIT-Projekt KADS-II
in: KI 3/94, Interdata-Verlag (1994)
- [**Biundo93**] Biundo, S.; Günter, A.; Hertzberg, J.; Schneeberger, J.; Tank, W.:
Planen und Konfigurieren
in: Görz, G. (Hrsg.) Einführung in die Künstliche Intelligenz, Addison-Wesley (1993)
- [**Behavior93**] Behavior-Projektpartner:
BEHAVIOR - Modellbasierte Wissensrepräsentation und Schlußfolgerungsverfahren im Bereich dynamischer technischer Systeme
in: G. Wolf (Hrsg.) Statusseminar des BMFT Künstliche Intelligenz, S. 29-55 (1993)
- [**Breiiing91**] Breiiing, A.; Flemming, M.:
Theorie und Methoden des Konstruierens
Springer (1991)
- [**Brown89**] Brown, D.C.; Chandrasekaran, B.:
Design Problem Solving
Pitman, Research Notes in AI (1989)
- [**Buhr94**] Buhr, F.:
Benutzerschnittstelle(n) von KONWERK
PROKON-Memo Nr. 54, Uni Hamburg (1994)
- [**Carrara94**] Carrara, G.; Kalay, Y.E. (Hrsg.):
Knowledge-Based Computer-Aided Architectural Design
Elsevier (1994)
- [**Chandrasekaran88**] Chandrasekaran, B.:
Generic Tasks as Building Blocks for Knowledge-Based Systems: the Diagnosis and Routine Design Examples
in: The Knowledge Engineering Review, Vol. 3, No.3, S.183-210 (1988)
- [**Christaller89**] Christaller, T. u.a. (Hrsg.):
Die KI-Werkbank Babylon
Addison-Wesley, (1989)
- [**Cunis91**] Cunis, R.; Günter, A.; Strecker, H. (Hrsg.):
Das PLAKON-Buch
Springer, (1991)
- [**Funke94**] Funke, B.:
Das Optimierungs-Grundmodul - Beschreibung und Einbindung in KONWERK
PROKON-Memo Nr. 53, RWTH Aachen (1994)
- [**Günter91**] Günter, A.:
Flexible Kontrolle in Expertensystemen für Planungs- und Konfigurierungsaufgaben
Dissertation Uni Hamburg (1991); auch KI-Dissertationsreihe Nr. 3, infix-Verlag (1992)
- [**Günter91a**] Günter, A.; Dörner, H.; Gläser, H.; Neumann, B.; Posthoff, C.; Sebastian, H.-J.:
Das Projekt PROKON - Problemspezifische Werkzeuge für die wissensbasierte Konfigurierung
PROKON-Bericht Nr. 1 (1991)
- [**Günter93**] Günter, A.; Cunis, R.:
PLAKON - Ergebnisse einer Entwicklung
in: KI 1/93, S.: 51-56 (1993)

- [**Günter93a**] Günter, A.:
Verfahren zur Aulösung von Konfigurationskonflikten in Expertensystemen
in: KI 1/93, S.: 16-22 (1993)
- [**Günter94**] Günter, A. (Hrsg.):
Kurzbeschreibung der KONWERK-Module,
PROKON-Memo Nr. 48, Uni Hamburg (1994)
- [**Harmon93**] Harmon, P.:
The Market for Expert Systems-Building Tools
in: Intelligent Software Strategies, Vol. 9, No. 2, Feb. 93 (1993)
- [**Hotz93**] Hotz, L.; Vietze, T.:
Über Spezialisierungshierarchien, Mehrfachvererbung, Mixins, Sichten und ein neues Konzept ihrer Verwendung in Konfigurierungssystemen
in: Proc. 7. Workshop "Planen und Konfigurieren" GMD-Arbeitspapier, S. 143-153 (1993)
- [**Hotz94**] Hotz, L.:
Schnittstellenklassen und -Methoden von KONWERK
internes Papier, Uni Hamburg (1994)
- [**Kopisch92**] Kopisch, M.; Günter, A.:
Configuration of a Passenger Aircraft Cabin based on Conceptual Hierarchy, Constraints and flexible Control
in: Proc. IEA/AIE-92, Springer, S. 421-430 (1992)
- [**Kopisch93**] Kopisch, M.:
Räumliche Beziehungen beim Konfigurieren von Passagierkabinen des AIRUBS A340
in: Proc. Expertensysteme 93, Springer, S. 84-97 (1993)
- [**Kratz91**] Kratz, N.:
Architektur eines wissensbasierten Systems zur Unterstützung der Konzeptionsphase in der Konstruktion
Dissertation Uni Kaiserslautern (1991)
- [**Puppe90**] Puppe, F.:
Problemlösungsmethoden in Expertensystemen
Springer (1990)
- [**Puppe94**] Puppe, F.; u.a.:
Wiederverwendbare Bausteine für eine konfigurierbare Diagnostik-Shell.
in: KI 2/94, S. 13- 18, Interdata-Verlag (1994)
- [**Sauer93**] Sauer, J.:
Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken
Dissertation Uni Oldenburg; auch KI-Dissertationsreihe Nr. 37, infix-Verlag (1993)
- [**Syska91**] Syska, I.:
Modulare Architekturen für Konstruktionssysteme
Dissertation Uni Hamburg (1991); auch KI-Dissertationsreihe Nr. 4, infix-Verlag (1992)
- [**Tank91**] Tank, W.:
Modellierung von Expertise über Konfigurierungsaufgaben
Dissertation TU Berlin (1991); auch KI-Dissertationsreihe Nr. 5, infix-Verlag (1992)
- [**Wirth94**] Wirth, N.:
Gedanken zur Software-Explosion
in: Informatik Spektrum, Band 17, Heft 1, S.: 5-10, Springer (1994)