SCHRIFTENREIHE DES FACHBEREICHS MATHEMATIK

MOKON – Eine modellbasierte Entwicklungsplattform zur Konfiguration technischer Anlagen

Benno Stein und Jürgen Weiner

SM - DU - 178 1990

Universität Duisburg Gesamthochschule

This work was supported by the ‘‘KI-Verbund NRW’’, founded by the Ministery for Science and Research of North Rhine-Westphalia. A reprint of this paper can be found in the Report Series KI-NRW (Knowledge-Based Software Technology in North-Rhine Westphalia), number 90-18, 1990.

Inhaltsverzeichnis

1 Motivation

In diesem Artikel wird das wissensbasierte System MOKON vorgestellt, das die Konfiguration technischer Anlagen unterstützt. Kern des Systems ist eine einheitliche Konfigurationsmethodik, die verschiedene Aufgabenstellungen wie Neukonfigurierung, Ändern einer bereits installierten und Prüfen einer vorgegebenen Konfiguration integriert löst. Ziel ist der weitere Ausbau von MOKON zu einer Konfigurationsshell, welche die Wissensakquisition unterstützt und die Wissenspflege erleichtert.

Besonderes Merkmal der Konfigurationsmethodik in MOKON ist die Modellierung des Wissens über eine eigenschaftsorientierte Beschreibung der zu konfigurierenden Komponenten. Durch diese Beschreibung wird ein Abhängigkeitsnetz zwischen den Komponenten aufgebaut, das den Konfigurierungsprozeß steuert. Dieses Abhängigkeitsnetz drückt kausale Beziehungen zwischen den Komponenten aus, die als Angebote und Forderungen interpretiert werden.

Diese Konfigurationsmethodik ist besonders für Anwendungsdomänen geeignet, in denen solche Abhängigkeitsnetze den Hauptbestandteil des Konfigurationswissens darstellen. Beispiele für solche Anwendungsdomänen sind die Konfiguration von Computern und Telekommunikationsanlagen.

2 Charakterisierung der Aufgabenstellungen

Die hier diskutierten Aufgabenstellungen sind das Erstellen einer neuen, das Ändern einer installierten sowie das Prüfen einer Konfiguration. In Unterabschnitten dieses Kapitels beschreiben wir diese Aufgabenstellungen.

2.1 (Neu-)Konfiguration technischer Anlagen

Die Aufgabe der Konfiguration einer technischen Anlage wird in [Eich et al. 89] wie folgt beschrieben:

> *Unter Konfigurieren wollen wir einen planerischen Prozeß verstehen, der die Auswahl von Komponenten aus einer vorgegebenen Menge von Komponenten, . . . , und die Organisation dieser ausgewählten Komponenten zu einem einer Anforderungsdefinition entsprechenden Gesamtsystem zum Ziel hat. Das Resultat der Konfigurierung . . . wird im folgenden als Konfiguration bezeichnet.*

Diese abstrakte Beschreibung der Konfigurationsaufgabe wollen wir im folgenden konkretisieren. Zunächst beschreiben wir den Aufbau und die Struktur des Inputs (d. h. der Anforderungsdefinition) und des Outputs (d. h. der Konfiguration) einer Konfigurierung, bevor wir auf den Konfigurationsprozess eingehen.

Anforderungsdefinition

Eine Anforderungsdefinition besteht nach unserem Verständnis aus einer Auflistung von Funktionen, die das zu konfigurierende System erfüllen muß. Es handelt sich um eine *flache* Anforderungsdefinition, falls keine der Funktionen durch Unterfunktionen spezifiziert werden muß. Andernfalls sprechen wir von einer *strukturierten* Anforderungsdefinition.

Die allgemeine und formale Beschreibung unserer Anforderungsdefinition hat in der sogenannten Backus-Naur-Form folgende Gestalt:

$\langle$Anforderungsdefinition$\rangle \longrightarrow \{(\langle$Funktionskomplex$\rangle) \mid \langle$Einzelfunktion$\rangle\}_1^*$
$\langle$Funktionskomplex$\rangle \longrightarrow \langle$Beschreibung$\rangle\ \langle$Anforderungsdefinition$\rangle$
$\langle$Einzelfunktion$\rangle \longrightarrow \langle$Beschreibung$\rangle$
$\langle$Beschreibung$\rangle \longrightarrow \langle$Bezeichner$\rangle\ \{\langle$Wert$\rangle\}_0^1$

Diese Vorschrift beschreibt den syntaktischen Aufbau und wird ergänzt um Relationen, die zulässige Unterspezifikationen für Funktionskomplexe festlegen. Beispielsweise kann die Anforderungsdefinition an ein Rechner-Cluster folgendermaßen aussehen:

Rechner-cluster

- Rechner-Arbeitsplatz (Wert: 1)
  - Hardware
    - Hauptspeicher-Kapazität (Wert: 20 MB)
    - Platten-Kapazität (Wert: 520 MB)
    - VGA-fähig
    - ...
  - Software
    - Graphikpaket-XY
    - Betriebssystem
    - ...
- Rechner-Arbeitsplatz (Wert: 2)
  - Hardware
    - Hauptspeicher-Kapazität (Wert: 1 MB)
    - CGA-fähig
- ...

Konfiguration

In unserer Vorstellung besteht eine Konfiguration auf der untersten Ebene aus Einzelkomponenten. Einzelkomponenten können zu Baugruppen zusammengefaßt werden, welche (u. U. wiederum als Bestandteil übergeordneter Baugruppen) in die Konfiguration eingehen. Zerfällt eine Konfiguration ohne Zwischenstufen direkt in Einzelkomponenten, so sprechen wir von einer *flachen*, ansonsten von einer *strukturierten* Konfiguration.

Die Zusammensetzung einer Konfiguration aus Baugruppen und Einzelkomponenten kann wie in Abbildung 1 durch einen Gozinto-Graphen bzw. eine kompositionelle Hierarchie veranschaulicht werden (vgl. [Scheer 88]).

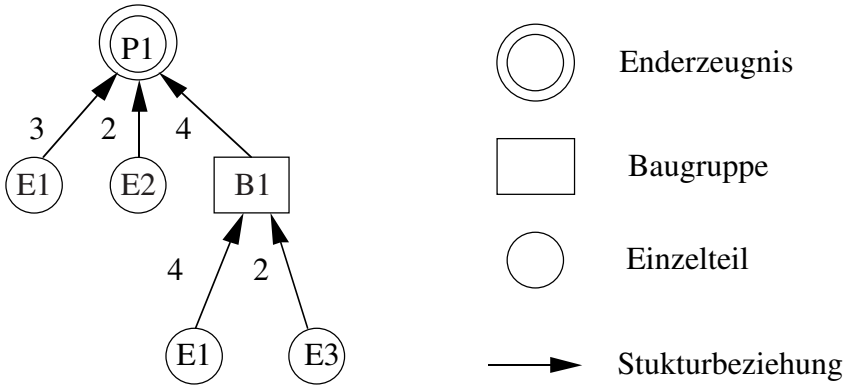

Abbildung 1: *Gozintograph einer Erzeugnisstruktur*

Abbildung 1 beschreibt zwar die Struktur einer Konfiguration, enthält jedoch nicht notwendigerweise alle Informationen zur räumlichen Anordnung. Diese Struktur können wir wie folgt formalisieren:

$\langle \text{Konfiguration} \rangle \longrightarrow \{(\langle \text{Baugruppe} \rangle) \mid \langle \text{Einzelkomponente} \rangle\}_1^*$
$\langle \text{Baugruppe} \rangle \longrightarrow \langle \text{Beschreibung} \rangle \langle \text{Konfiguration} \rangle$
$\langle \text{Einzelkomponente} \rangle \longrightarrow \langle \text{Beschreibung} \rangle$
$\langle \text{Beschreibung} \rangle \longrightarrow \langle \text{Bezeichner} \rangle \{\langle \text{Wert} \rangle\}_0^1$

Im folgenden werden wir ausgehend von der obigen Definition unterschiedliche Mengen von Konfigurationen charakterisieren.

Die Menge aller möglichen Kompositionen von Einzelkomponenten zu Baugruppen und von Baugruppen zu Konfigurationen bezeichnen wir als den *Kompositionsraum*.

Der *Konfigurationsraum* ist die Teilmenge des Kompositionsraumes, die nur *zulässige* Konfigurationen enthält. Dabei drückt *zulässig* die Realisierbarkeit einer Konfiguration unter konstruktiven, fertigungstechnischen, betriebswirtschaftlichen u. a. Randbedingungen aus.

Alle Konfigurationen des Konfigurationsraums, deren *Leistung* eine gegebene Anforderungsdefinition erfüllt, bilden den *Variantenraum*. Unter Leistung einer Konfiguration verstehen wir die Summe der Fähigkeiten, die zur Realisierung einer Anforderungsdefinition beitragen. Hiermit wollen wir verdeutlichen, daß eine Konfiguration

u. U. viel mehr leisten kann als ursprünglich gefordert wurde. Zum Beipiel können geforderte 60 Megabyte Plattenkapazität durch eine 100 Megabyte Platte befriedigt werden. Formal beschrieben bedeutet dies:

Kompositionsraum := $\{X \mid X$ ist eine Konfiguration$\}$
Konfigurationraum := $\{X \mid X \in$ Kompositionsraum $\wedge\ X$ ist zulässig$\}$
Variantenraum := $\{X \mid X \in$ Konfigurationsraum $\wedge\ X$ erfüllt Anforderungsdefiniton$\}$

Die Zusammenhänge dieser Räume werden in folgender Abbildung veranschaulicht:

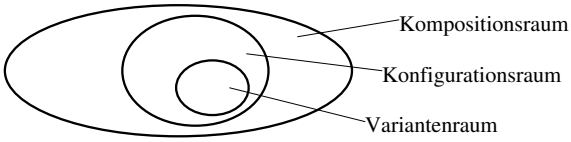

Abbildung 2: *Zusammenhang der Räume*

Konfigurationsprozeß

Unter Konfigurieren verstehen wir den Prozeß, der zu einer gegebenen Anforderungsdefinition den zugehörigen Variantenraum bestimmt.

Enthält der Variantenraum mehr als ein Element, so kann man versuchen, die Varianten mit Hilfe von Bewertungskriterien zu ordnen. Beispiele für Bewertungskriterien sind:

- Komponentenpreise
- Montagekosten
- Kosten für erwartete Änderungen (Opportunitätskosten)
- Lieferfristen

Ein Verfahren zur Lösung der Konfigurationsaufgabe ist *vollständig*, wenn alle Konfigurationen, die Element des Variantenraumes sind, gefunden werden. Ein Lösungsverfahren ist *korrekt*, wenn jede gefundene Konfiguration Element des Variantenraumes ist. Ein Lösungsverfahren heißt *exakt*, wenn es vollständig und korrekt ist (vgl. [Hein, Tank 90]).

2.2 Änderungskonfiguration

Bei unserem bisherigen Verständnis der Konfigurationsaufgabe sind wir von der Annahme ausgegangen, daß die Konfiguration von Grund auf neu erstellt wird.

Häufig steht man aber auch vor der Aufgabe, eine bereits konfigurierte und in der Regel schon installierte Anlage ändern zu müssen. Bei einer Änderungskonfigurierung wird also die Lösung nicht völlig neu, sondern in Anlehnung an eine gegebene

Konfiguration für bekannte Anforderungen bestimmt. Da die Leistung der gegebenen Konfiguration im allgemeinen die aktuellen Anforderungen nicht erfüllt, muß sie durch Hinzufügen, Wegnehmen oder Umordnen von Komponenten entsprechend modifiziert werden.

Auf unsere Begriffswelt übertragen vollzieht sich der Prozess einer Änderungskonfigurierung in folgenden Schritten:

- Zur neuen Anforderungsdefinition wird der Variantenraum bestimmt.
- Bestimmung der Variante dieses Variantenraumes, in die sich die installierte Konfiguration mit dem geringsten Aufwand transformieren läßt.
- Angabe der Operationen (Hinzufügen, Wegnehmen oder Umordnen von Komponenten), aus denen sich die Transformation zusammensetzt.

2.3 Konfigurationsprüfung

Die Aufgabe der Prüfung einer Konfiguration besteht darin, zu einer Konfiguration des Kompositionsraumes festzustellen, ob sie Element desselben ist, d. h. eine zulässige Konfiguration darstellt.

Falls eine Konfiguration nicht Element des Konfigurationsraumes ist, so besteht die Aufgabe darin, Vorschläge zu geben, wie diese Konfiguration in eine zulässige Konfiguration transformiert werden kann.

3 Vorstellung der Lösungsmethodik

In diesem Kapitel stellen wir unsere Lösungsmethodik vor. Zunächst beschreiben wir unsere allgemeinen Leitgedanken der Wissensrepräsentation. Der sich daran anschließende Abschnitt enthält die in MOKON verwirklichte Wissensrepräsentation und deren Verarbeitung.

3.1 Grundgedanken zur Wissensrepräsentation

Typische Ansprüche an wissensbasierte Systeme sind u. a. die Verwirklichung einer benutzergerechten Aktualisierungskomponente für das Domänenwissen und die Generierung aussagekräftiger Erkärungen einer gefundenen Lösung bzw. des Lösungsprozesses. Viele wissensbasierte Systeme werden diesen Forderungen nur bedingt gerecht. Die Modellierung des Wissens spielt eine bedeutende Rolle bei der Erfüllung dieser Ansprüche.

Den Begriff *Wissen* kann man unserer Meinung nach von drei grundsätzlich verschiedenen Standpunkten aus betrachten:

- Repräsentationstechnik
- Wissensebene
- Modellierungstiefe

Besondere Aufmerksamkeit richten wir auf die beiden letzten Punkte der Aufzählung.

Repräsentationstechnik

Sie beschreibt auf welche Art und Weise das Wissen auf dem Computer implementiert wird. Die wichtigsten Techniken sind Regeln, Objekte, Constraints usw. Wir gehen an dieser Stelle nicht ausführlicher darauf ein. Eine genaue Beschreibung dieser Techniken und eine Zuordnung zu Problemlösungstypen findet sich in [Puppe 88].

Wissensebenen

Wissen über einen Problembereich kann in unterschiedliche Ebenen klassifiziert werden [Wielenga, Breuker 86]:

> *The knowledge of the expert is modelled at several levels: the domain level, where mainly objects and relations are represented, and a number of higher levels which contain descriptions of the ways in which the domain knowledge can be used.*

Wir unterscheiden hier prinzipiell zwischen einer Metaebene und einer Objektebene. Das Wissen, das etwas *über* die Objektebene aussagt, bezeichnen wir als Metawissen.

Der Begriff Meta drückt aus, daß ein Sachverhalt in übergeordneter Beziehung zu einem anderen steht (vgl. [Richter 89]).

Abbildung 3 gibt eine weitergehende Verfeinerung der beiden genannten Wissensebenen an:

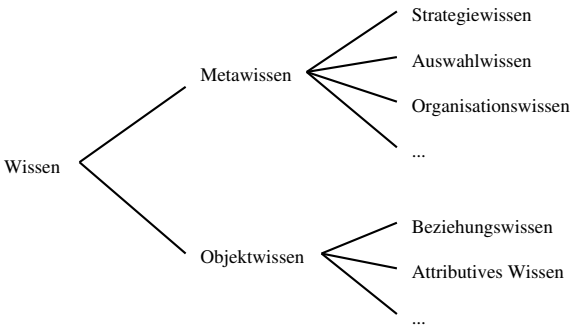

Abbildung 3: *Klassifizierung von Wissensarten*

Die Wissensarten aus obiger Abbildung können wie folgt charakterisiert werden:

- Strategiewissen beinhaltet Wissen über die Gestaltung der Lösungssuche (Tiefensuche, Breitensuche, heuristische Strategien usw.)
- Auswahlwissen enthält Wissen über Verwendung und Herkunft alternativer Wissensressourcen.
- Organisationswissen stellt Informationen darüber bereit, wie Wissen abgelegt ist und verfügbar gemacht werden kann.
- Beziehungswissen drückt aus, welche Relationen zwischen den Objekten der Anwendungsdomäne bestehen.
- Attributives Wissen beschreibt, welche Fähigkeiten oder Funktionalitäten den Objekten der Anwendungsdomäne zugeordnet sind.

Eine ungenügende oder fehlende Trennnung dieser Wissensebenen erschwert die Wartbarkeit (vgl. [Wielenga, Breuker 86]). Ein Beispiel hierfür ist das Konfigurationsexpertensystem R1/XCON (vgl. [Soloway et. al. 87]), in dem Objekt- und Metawissen innerhalb gleicher Regeln repräsentiert wird[1].

Modellierungstiefe

Häufig wird das gesamte Wissen eines Anwendungsbereiches auch als Modell bezeichnet. Der hier vorgestellte dritte Betrachtungsstandpunkt von Objektwissen beruht auf der Idee eines *variablen* Modellverständnisses. Variabel deshalb, weil die Abbildung eines Sachverhaltes von einem *flachen* bis zu einem *tiefen* Modell möglich ist. Stein beschreibt in [Stein 89] ein tiefes Modell wie folgt:

[1] Trotz dieser fehlenden Trennung arbeitet dieses System erfolgreich und produktiv in der betrieblichen Praxis.

Bei einem tiefen Modell handelt es sich um Wissen, das eine Problemstellung oder einen technischen Sachverhalt von ihrem Wesen her erklärt. Wichtiger Bestandteil dieses Wissen sind Ursache-Wirkungsbeziehungen, die es ermöglichen, Aussagen über den vorliegenden Problembereich zu machen.

Die Verarbeitung dieser Ursache-Wirkungsbeziehungen beruht auf deren Propagierung in diesem tiefen Modell. Weil die Lösung über die gleichen Ursache-Wirkungsbeziehungen abgeleitet werden, wie sie der Experte in seinem Gedankenmodell benutzt, können sie entscheidend zur Erklärung beitragen. Deshalb sprechen wir in diesem Zusammenhang auch von einem *kausalen* Modell. Dieses Modellverständnis kann die Aktualisierung dadurch unterstützten, daß die Konsistenz und Redundanz neuen Wissens an Hand des tiefen Modells automatisch geprüft wird.

Im Gegensatz dazu handelt es sich bei einem *flachen* Modell um einen assoziativen Beschreibungsansatz, der auf Oberflächenwissen beruht und überwiegend Heuristiken bzw. Daumenregeln zur Lösungsfindung heranzieht.

Eine tiefe Beschreibung eines Modells ist umfangreicher als eine assoziative; deshalb wächst die Komplexität der Lösungsfindung mit der Beschreibungsgenauigkeit (Tiefe) des Modells. Dieser Zusammenhang wird in folgender Abbildung veranschaulicht:

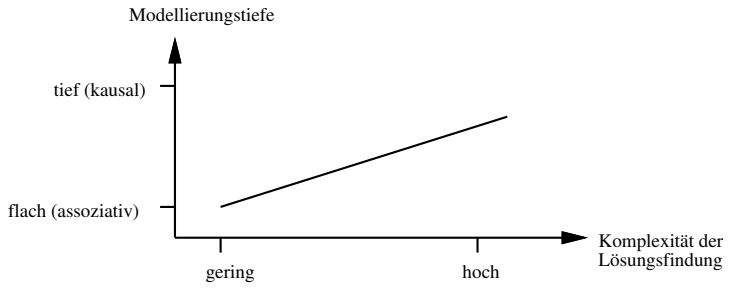

Abbildung 4: *Zusammenhang zwischen Modellierungstiefe und Komplexität*

Wir möchten hervorheben, daß eine Unterteilung in tiefe und flache Modelle nicht absolut ist, sondern vielmehr ein relatives Maß für die unterschiedliche Modellierungstiefe verschiedener Beschreibungsansätze darstellt.

Ein weiteren Aspekt der tiefen Modellierung beschreibt Steels in [Steels 90]:

Making deep knowledge explicitly available allows a system to fall back on underlying models and (weaker) problem-solving methods if no explicit surface knowledge is available to efficiently solve a problem.

Das bedeutet, daß tiefes Wissen u. a. zu einem flexiblen Problemlösungsverhalten beiträgt.

Aus dieser Diskussion lassen sich folgende Schlüsse für eine Wissensrepräsentation ableiten:

- Bei der Abbildung des Wissens sollte eine klare Trennung zwischen Meta- und Objektwissen angestrebt werden.

- Bei der Beschreibung eines Sachverhaltes sollte überlegt werden, inwiefern die Vorteile einer tiefen Modellierung genutzt werden können.

Der Grad der Modellierungstiefe kann jedoch kein eindeutiges Optimierungskriterium darstellen, weil er eng an die Performance gebunden ist (s. Abb. 4). Darüberhinaus sollte man bedenken, daß Anwendungsprobleme existieren, in denen das für eine tiefe Modellierung notwendige Wissen nicht oder nur schwer verfügbar gemacht werden kann.

3.2 Konzept MOKON

Grundidee des Konfigurierungsprinzips ist eine eigenschaftsorientierte Komponentenbeschreibung. Jede Komponente bietet eine oder mehrere Eigenschaften und fordert auch welche. Durch diese Eigenschaften sind Angebots- und Forderungsbeziehungen zwischen den Komponenten gegeben. Diese impliziten Beziehungen werden bei der Neukonfigurierung[2] wie folgt ausgenutzt:

Ausgangspunkt des Konfigurierungsprozesses ist eine flache Anforderungsdefinition. Sie besteht aus einer Auflistung von Eigenschaften, die das zu konfigurierende System erfüllen soll. Der Konfigurierungsprozeß besteht in der sukzessiven Abarbeitung von offenen Forderungen. Eine offene Forderung wird dadurch erfüllt, daß Komponenten bereitgestellt werden, welche die geforderte Eigenschaft anbieten. Dabei stellen die ausgewählten Komponenten in der Regel neue Anforderungen. Der Konfigurierungsprozeß ist beendet, wenn alle Forderungen erfüllt sind. Ergebnis dieses Prozesses ist eine unstrukturierte Auflistung von Komponenten, also in unserer Terminologie: eine flache Konfiguration. Das vorgestellte Verfahren für die Neukonfigurierung ist mit dem ressourcenorientierten Modell von Heinrich [Heinrich 89] vergleichbar.

Dieser Prozeß läßt sich mit dem Prinzip einer Bilanz, auf der Angebote und Forderungen saldiert werden, vergleichen. Wir sprechen daher auch von dem *Bilanzkonzept* in der Konfigurierung.

Folgende vereinfachte Abbildung zeigt Abhängigkeiten einiger Komponenten:

Die Steckkarten in Abbildung 5 fordern jeweils einen Steckplatz und das Mainboard stellt Steckplätze bereit. Das Netzteil bietet Strom an, die übrigen Komponenten fordern Strom.

Eine Konfiguration ist das Ergebnis der Propagierung solcher Angebots-Forderungsbeziehungen. Diese Angebots-Forderungsbeziehungen entsprechen in unserem Verständnis den Ursache-Wirkungsbeziehungen des zugrundeliegenden tiefen Modells. Denkt man die hier skizzierte Idee des tiefen Modells konsequent weiter, so impliziert man damit, daß das Verhalten des Systems als Ganzes ausschließlich durch das Einzelverhalten der Komponenten und ihrer wechselseitigen Beziehungen beschrieben werden kann [Stein 89].

[2]Die Realisierung der Änderungskonfigurierung bzw. der Konfigurationsprüfung mit diesem Prinzip wird in Abschnitt 4 beschrieben.

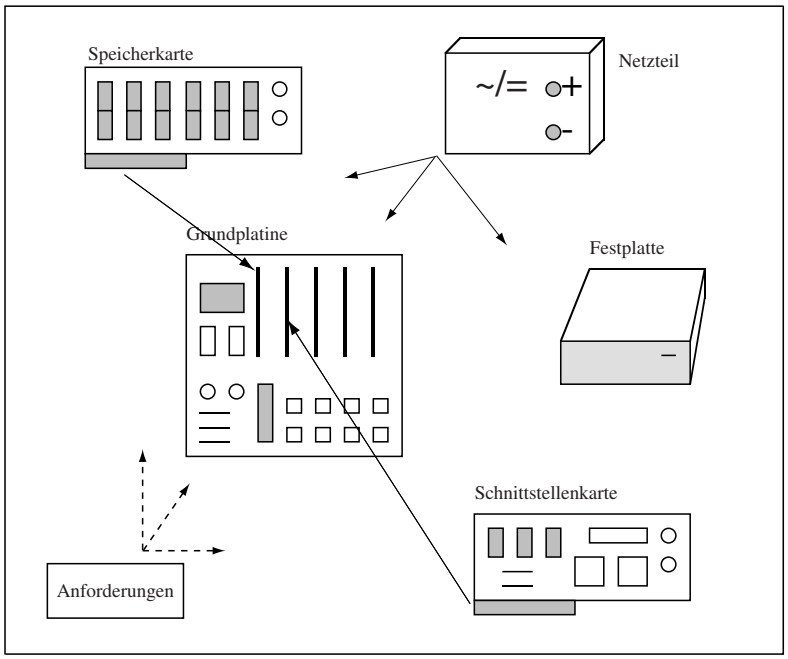

Abbildung 5: *Komponentenbeziehungen*

Diese Aussage ist eine Folgerung des No-Function-In-StructurePrinzips, das u. a. in den Arbeiten von [Bobrow 84], [De Kleer, Brown 84] und [De Kleer 84] diskutiert wird.

Die sich anschließenden Unterabschnitte gehen konkret auf die Wissensrepräsentation und -verarbeitung in MOKON ein.

3.2.1 Wissensrepräsentation

Die theoretisch in Abschnitt 3.1 beschriebenen Wissensarten finden sich in unserem Konzept wie folgt wieder:

Metawissen

- Priorität der Eigenschaften

  Den einzelnen Eigenschaften der Komponenten (z. B. Stromwert, Plattenkapazität) sind bestimmte Prioritäten zugeordnet. Sie legen fest, in welcher Reihenfolge offene Forderungen befriedigt werden sollen.

  Mit diesen Prioritätskennzahlen soll von Anfang an ein möglichst hoher Grad der Linearisierung des Konfigurationsprozesses erreicht werden, d. h. das Rücksetzen auf weit zurückliegende Entscheidungen und die damit verbundenen Neuberechnungen sollen vermieden werden.

- Auswahl unterschiedlicher Wissensquellen

  Metawissen ist hier Wissen darüber, wie konkurrierende Wissensquellen ausgewählt oder kombiniert werden sollen.

  Beispiel: Eine geforderte Plattenkapazität in Höhe von 90 Megabyte kann auf Basis des kausalen Modells auf mehrere Arten befriedigt werden: 3 Platten zu 30 MB, 2 Platten zu 50 MB usw. Wissen, das die – aus Sicht unseres kausalen Modells – gleichwertigen Alternativen bewertet, kann z. B. sein:

  – Optimiere lokal durch Auswahl der kostengünstigsten Alternative
  – Folge dem Erfahrungswissen des Experten:
    „Installiere nie mehr als 2 Festplatten“

  In diesem Beispiel wäre Metawissen, daß Erfahrungswissen (falls vorhanden) einer anderen Optimierungsstrategie vorzuziehen ist.

- Suchstrategien

  Die im obigen Beispiel aufgeführte Auswahlstrategie beruht auf folgendem (Meta-)Wissen: Eine lokale Optimierung aufgrund des Preises führt nicht unbedingt zum Optimum, stellt in der Regel jedoch eine gute Annäherung dar.

Objektwissen

- Attributives Wissen einer Komponente

  Dieses Wissen besteht zum einen aus funktionalem Wissen, wie den Angeboten und Forderungen, die sich aus der Funktionalität der Komponente ableiten lassen, zum anderen aus sonstigen Attributen wie Preis, Lagerbestand usw. Beispielsweise bietet ein Netzteil Strom an, eine Festplatte verbraucht Strom und stellt Speicherkapazität zur Verfügung.

- Attributives Wissen einer Eigenschaft

  Hierbei handelt es sich um Wissen, das eine dem kausalen Modell angemessene Behandlung der Eigenschaften ermöglicht. Die Behandlung einer Eigenschaft legt fest, wie aus einer Liste von Forderungen bzw. Angeboten eine Gesamtforderung bzw. ein Gesamtangebot für diese Eigenschaft generiert wird und in welcher Weise diese ermittelten Werte verglichen werden.

  Dieses eigenschaftsabhängige Verhalten wird folgendermaßen modelliert:

Name	Plattenkapazität	Lieferzeitpunkt	Stromwert
Typ	numerisch	datum	numerisch
Constraint	erfüllbar	unerfüllbar	erfüllbar
Priorität	70	100	10
Saldierungsoperator	>=	<=	>=
Angebot			
Kommunikationstyp	intern	intern	intern
Defaultwert	-	-	-
Wertebereich	0 - 300	-	-
Verarbeitung	+	Maximum	+
Forderung			
Kommunikationstyp	ask	ask	intern
Defaultwert	20	-	-
Wertebereich	-	-	-
Verarbeitung	+	Minimum	+

Abbildung 6: *Beispiele für Eigenschaftsspezifikationen*

In unserem Beispiel berechnet sich das Gesamtangebot an Plattenkapazität aus der Summe der Einzelkapazitäten der ausgewählten Festplatten (vgl. Abb. 6: Verarbeitung „+“). Dieses lineare Verhalten liegt u. a. nicht bei der Eigenschaft Lieferzeitpunkt vor, die jede Komponente als Angebot besitzt. Hier bestimmt der späteste aller angebotenen Lieferzeitpunkte, ob eine Konfiguration den geforderten Lieferzeitpunkt erfüllt. Die Spezifikation „Constraint“ legt fest, ob eine offene Forderung im Laufe des weiteren Lösungsprozesses nie mehr erfüllt werden kann. Das wäre zum Beispiel bei einem überschrittenen Lieferzeitpunkt der Fall.

Die hier für die Ermittlung des Gesamtangebots bzw. –forderung und deren Saldierung skizzierten Verarbeitungsfunktionen bezeichnen wir im folgenden als *abstrakte Operatoren*. Sie bilden einen wichtigen Bestandteil des zugrundeliegenden kausalen Modells.

- Taxonomisches Beziehungswissen

  In der Taxonomie sind die Komponenten entsprechend ihrer Eigenschaften angeordnet. Ausdrückt wird hier keine Has-Part-Beziehung, sondern eine funktionale Verwandschaft (Is-a-Beziehung). Zum Beispiel gehört eine Speichererweiterung zur Klasse der Karten (vgl. Abb. 7) und fordert wie jede Karte einen Steckplatz. Solche klassentypische Informationen werden in der Taxonomie möglichst weit oben angeordnet, um redundante Informationen zu vermeiden und die Aktualisierung zu erleichtern.

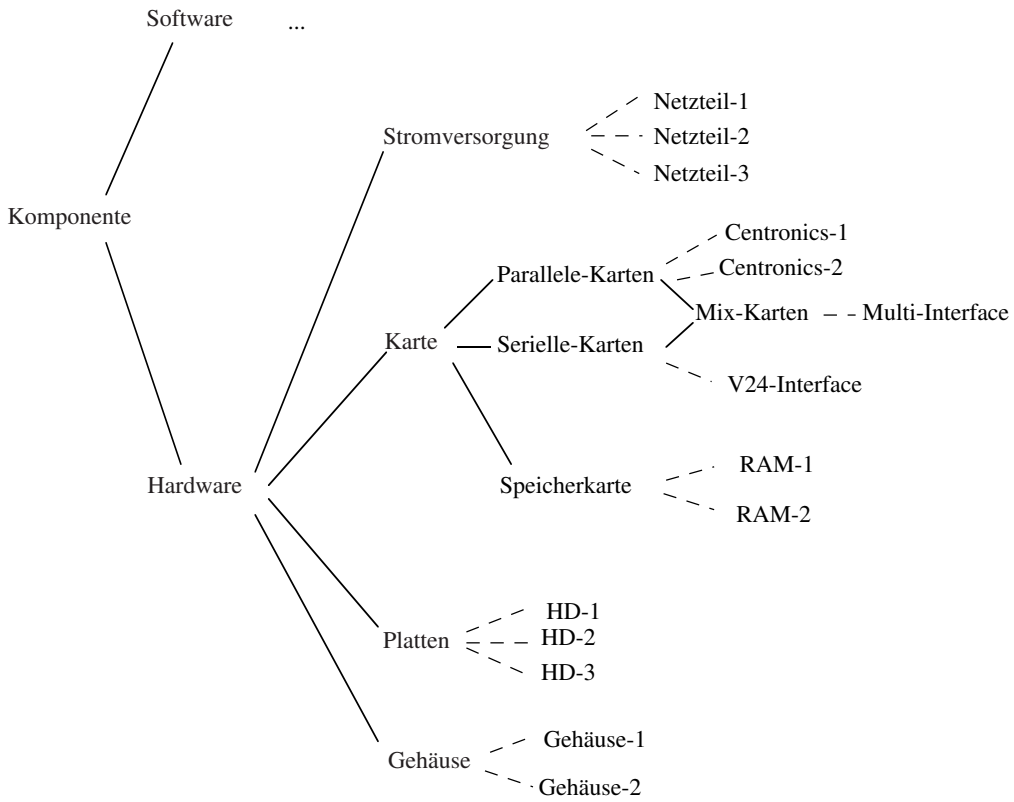

Abbildung 7: *Beispiel einer Komponententaxonomie*

- Kompositionelles Beziehungswissen

  Eine kompositionelle Hierachie (im Sinne von Has-Part-Beziehungen wie beispielsweise im System PLAKON [Cunis et al. 87]) liegt in diesem Modell nicht vor. Diese Informationen gewinnt das System über die funktionale Sicht des zugrundeliegenden kausalen Modells.

  Zusätzlich liegt kompositionelles Beziehungswissen in unserem Modell dann assoziativ vor, wenn das Vorhandensein einer Komponente in der Konfiguration nicht über Ursache-Wirkungsbeziehungen abgeleitet wird.

3.2.2 Wissensverarbeitung

Abbildung 8 enthält eine graphische Darstellung der Verarbeitung.

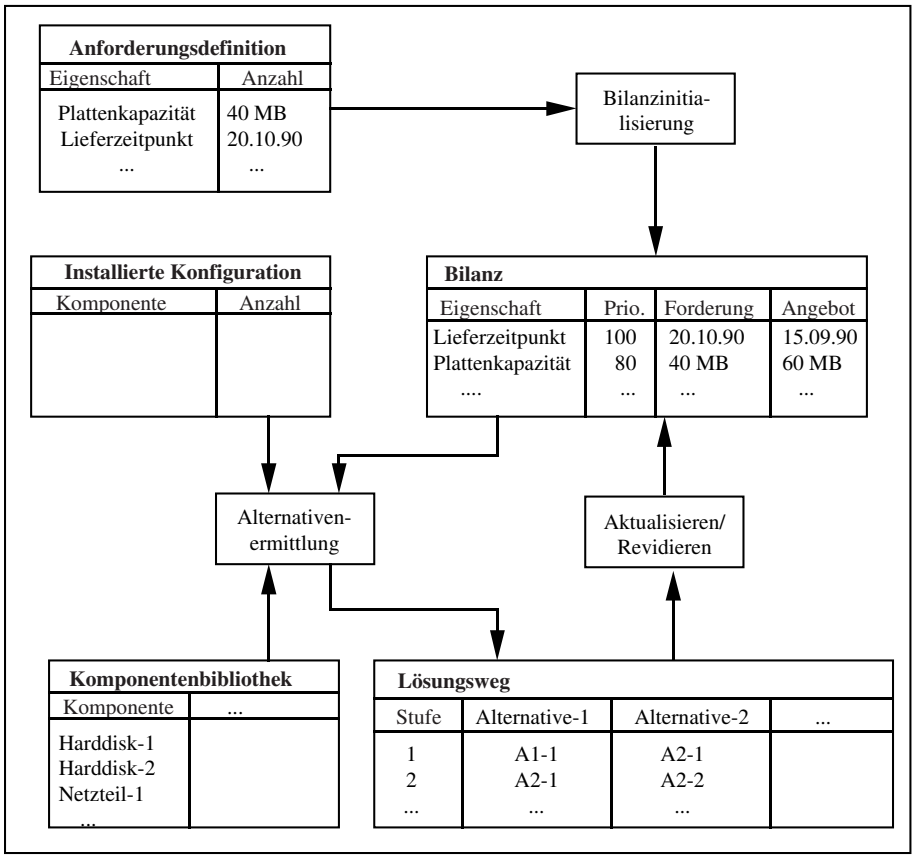

Abbildung 8: *Skizzierung der Verarbeitung*

Ausgangspunkt der Verarbeitung sind die Anforderungen eines Benutzers. Mit diesen Anforderungen und zusätzlichem assoziativen Wissen (das z. B. wie allgemeingültige Konfigurationsrestriktionen die Lösungssuche steuert) wird die Bilanz initialisiert. D. h. die Eigenschaften werden nach der Priorität geordnet in die Bilanz eingetragen. Nun startet der Abarbeitungsprozeß:

Sukzessiv wird für jede Eigenschaft, die in der Bilanz eingetragen ist, geprüft, ob das Angebot die Forderung erfüllt. Diese Prüfung erfolgt abhängig von den einer Eigenschaft zugeordneten abstrakten Operatoren (vgl. Abb. 9):

	Gesamtangebot	Saldierung	Gesamtforderung
numerisch	Summe Maximum Minimum Summe ...	>= <= > <	Summe Summe Maximum Durchschnitt ...
datum	spätestens ...	früher als	frühestens ...
string	Menge ...	Obermenge von	Menge ...

Abbildung 9: *Möglichkeiten der Eigenschaftsverarbeitung*

Erfüllt das Angebot nicht die Forderung, so wird überprüft, ob diese überhaupt

jemals erfüllt werden kann. Handelt es sich um eine erfüllbare Forderung, so wird die Alternativenermittlung angestoßen. Sie versucht nun eine Alternative aus Komponenten zusammenzustellen, welche die offene Forderung erfüllt. Liegt die Aufgabenstellung Änderungskonfiguration vor, werden die installierten Komponenten mitberücksichtigt.

Meistens existieren mehrere Alternativen, die eine offene Forderung befriedigen. Abbildung 10 zeigt, welche Alternativen für eine geforderte Plattenkapazität in Höhe von 160 Megabyte erzeugt werden.

Festplattenkapazität			
Forderung	Angebot	Festplatte 60MB	Festplatte 100MB
160	160	1	1
160	180	3	0
160	200	0	2

Abbildung 10: *Beispiel einer Alternativenmenge*

Abhängig von der aktuellen Strategie wird eine Alternative ausgewählt und die Bilanz mit dieser aktualisiert, d. h. die Konsequenzen, welche diese Alternative zur Folge hat, werden in der Bilanz nachvollzogen.

Handelt es sich bei der offenen Forderung um eine unerfüllbare, kann keine Alternativenmenge gebildet werden und in der Lösungssuche wird zur letzten Verzweigungsmöglichkeit zurückgegangen. Dort wird die aktuelle Alternative durch eine andere aus dieser Alternativenmenge ersetzt.

Der Prozeß des Überprüfens und Aktualisierens der Bilanz wird solange fortgesetzt, bis alle offenen Forderungen erfüllt sind oder festgestellt wird, daß keine Lösung existiert.

Dieses Vorgehen entspricht einem Generate & Test - Verfahren und kann folgendermaßen algorithmisch skizziert werden:

Suche-Loesung

```
begin
    if Alle Eigenschaften erfüllt
    then Lösung gefunden
    else if Eigenschaft erfüllbar
        then
            Ermittle Alternativen für Forderung;
            if Alternativenmenge leer
            then Backtrack
            else
                Aktualisiere Lösungsweg;
                Aktualisiere Bilanz;
                Suche Lösung;
            end
        else Backtrack
        end
    end
end
```

4 Integration der Aufgabenstellungen

4.1 Änderungskonfiguration

Im Gegensatz zur (Neu-)Konfiguration treten bei der Aufgabenstellung eine bereits installierte Konfiguration zu ändern zusätzlich folgende Probleme auf, die wir in diesem Artikel nicht ausführlich diskutieren, sondern nur kurz anreißen können:

- Zeitproblem

  Zwischen dem Zeitpunkt der Installierung einer Konfiguration und dem Zeitpunkt der Änderung einer Konfiguration kann sich – zum Beispiel durch die technische Weiterentwicklung in dieser Zeit – das konfigurationsrelevante Wissen verändert haben. Ziel einer technischen Weiterentwicklung ist es, zum einen die Funktionalität eines Systems zu erweitern und zum anderen Funktionen kostengünstiger zu realisieren. Beide Zielsetzungen resultieren darin, daß neue Komponenten entwickelt werden, die alte Komponenten ersetzen oder ergänzen.

  Dies bedeutet, daß unter Umständen viele Komponenten der zu ändernden Konfiguration zum Zeitpunkt der Änderung nicht mehr gefertigt werden oder lieferbar sind. Deshalb werden sie im (Neu-)Konfigurationsfall bei der Bestimmung des zu einer Anforderungsdefinition gehörenden Variantenraumes nicht mehr berücksichtigt.

  Um eine möglichst kostengünstige Konfiguration für den Änderungsfall bestimmen zu können, müssen aber alle bereits installierten Komponenten bei der Lösungsfindung mitberücksichtigt werden. Bei der Bestimmung des Variantenraumes zu einer vorgegebenen Anforderungsdefinition steht also im Falle einer Änderungskonfiguration in der Regel eine andere Ausgangsmenge von Einzelkomponenten zur Verfügung, auf deren Basis nun die zulässigen Konfigurationen ermittelt werden.

  Das Zeitproblem besteht im wesentlichen darin, das Wissen über eine sich in der Zeit ändernde Menge von Einzelkomponenten und deren Beziehungen in einer geeigneten Form zu repräsentieren und aus dieser Repräsentation den Variantenraum zu generieren.

  Diese Problematik versuchen wir in unserem Ansatz wie folgt zu lösen:

  – Zeitbehaftete Komponenten

    Die Komponenten werden aufgrund ihrer Funktionalität in die Taxonomie eingeordnet und darüberhinaus mit dem weiteren Attribut des Gültigkeitsdatums versehen.

  – Zeitlose Eigenschaftsbegriffe

    Die durch die Funktionalität (physikalische Größen, genormte Schnittstellen usw.) festgelegten Beziehungen werden als zeitlos betrachtet und überdauern mehrere Komponentengenerationen.

  - Filterung zulässiger Komponenten

    Obige Punkte definieren ein Gültigkeitskriterium, das in der Alternativenermittlung zugrundegelegt wird. Bei der Änderungskonfigurierung bilden neben den für eine Neukonfigurierung zulässigen auch die installierten Komponenten die Grundlage für die Alternativenermittlung. Deshalb unterscheidet sich in der Regel der entstehende Variantenraum vom Neukonfigurierungsfall.

- Transformationsproblem

  Das Transformationsproblem besteht darin, die Konfiguration des Variantenraumes zu bestimmen, in die sich die bereits installierte Konfiguration mit dem geringsten Aufwand transformieren läßt.

  Als Maß zur Abschätzung dieses Transformationsaufwandes bietet sich an, die Kosten für die hinzuzufügenden Komponenten und die Kosten für die Montage- bzw. Demontage von Komponenten heranzuziehen.

  Im allgemeinsten Fall spielt es bei der Berechnung der Montage- bzw. Demontagekosten eine wesentliche Rolle, von welcher Art die Verbindungen der einzelnen Komponenten sind und in welcher räumlichen Beziehung sie zueinander stehen.

  Die Art einer Verbindung (geschraubt, gesteckt, geschweißt, genietet usw.) kann in die Berechnung wie folgt eingehen: Man belegt die Auflösung und die Wiederherstellung einer Verbindung mit Kosten. Ist eine Auflösung einer Verbindung nicht mehr möglich, so setzt man dies mit unendlich hohen Auflösungskosten gleich.

  Die räumliche Anordnung von Komponenten kommt zum Tragen, wenn Montage- bzw. Demontagefolgen von der räumlichen Anordnung abhängen. So könnte beispielsweise aus der Information: „Komponente X verdeckt Komponente Y“ abgeleitet werden, daß um die Komponente Y demontieren zu können, zunächst auch die Komponente X demontiert werden muß und eventuell später wieder zu montieren ist.

  Die Komplexität dieser Aufgabenstellung läßt sich unter folgenden Annahmen erheblich reduzieren:

  A1: Montage- bzw. Demontagekosten sind für die Anwendungsdomäne vernachlässigbar.

  A2: Alle Verbindungen der zu ändernden Konfiguration sind auflösbar.

  Legt man Annahme A1 zugrunde, so sind die Kosten für die hinzuzufügenden Komponenten ein geeignetes Maß zur Abschätzung des Transformationsaufwandes. Unter der Annahme A2 lassen sich die hinzuzufügenden Komponenten durch eine einfache Vergleichsrechnung (+/− - Rechnung) der neuen mit einer alten Konfiguration ermitteln.

  Diese beiden Annahmen fließen in eine Heuristik zur Suchraumverwaltung ein, indem bereits installierte Komponenten mit dem Preis Null bewertet werden.

Die Idee hierbei ist, daß zunächst die Konfigurationen gefundenen werden, die möglichst viele bereits installierte Komponenten enthalten.

- Dokumentationsproblem

  Welche Informationen von einer Konfiguration als Basis für die Ermittlung vorzunehmender Änderungen benötigt werden, hängt von der Allgemeinheit der Lösung oben diskutierter Problemstellungen ab. Unter den vereinfachenden Annahmen A1 und A2 des vorgestellten Modells besteht diese Information aus einer Liste der installierten Komponenten.

Aus der Analyse obiger Problemstellungen wird deutlich, daß die Aufgabenstellung, eine bereits installierte Konfiguration zu ändern, sehr komplex ist. Durch das in diesem Artikel beschriebene Konfigurationssystem lösen wir bei weitem nicht diese Aufgabenstellung in der allgemeinsten Form. Wir haben jedoch aufgezeigt, daß diese Vorgehensweise unter gewissen vereinfachenden Annahmen, die zum Beispiel in der Anwendungsdomäne Telekommunikation durchaus zulässig sind, eine gute Unterstützung auch für diese Aufgabenstellung bietet.

4.2 Konfigurationsprüfung

Wir zeigen, wie das Bilanzkonzept auch für die Aufgabenstellung der Konfigurationsprüfung eingesetzt wird. Das Vorgehen entspricht der in [Neumann, Weiner 89] beschriebenen Idee der *statischen* Bilanzprüfung und ist in folgender Abbildung graphisch dargestellt:

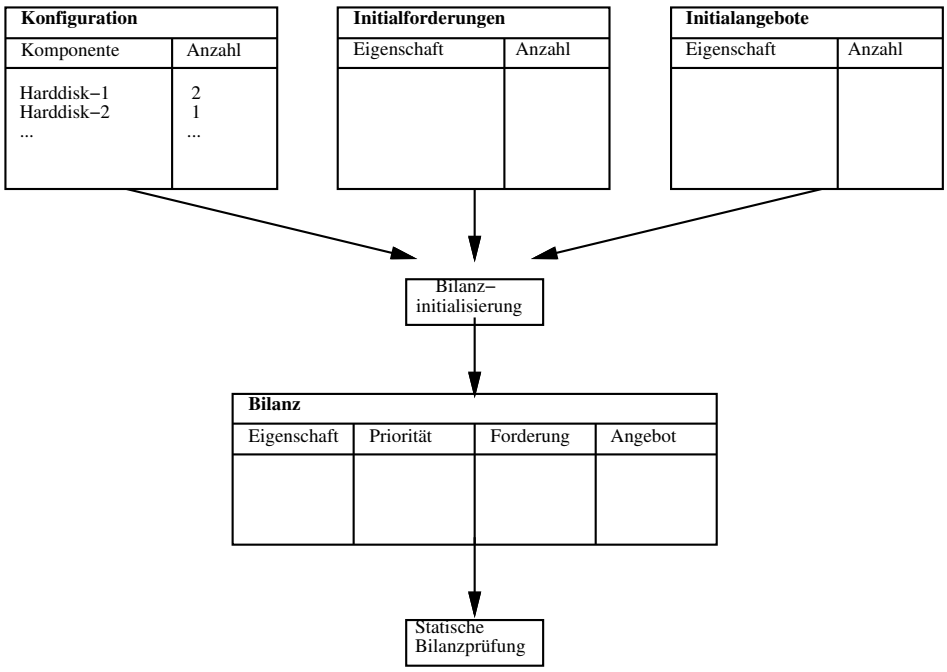

Abbildung 11: *Skizzierung der Konfigurationsprüfung*

Ausgangspunkt ist die zu prüfende Konfiguration sowie Initialangebote bzw. -forderungen, die allgemeingültige Konfigurationsrestriktionen repräsentieren. Mit diesen

Initialeigenschaften und den Eigenschaften der Konfigurationskomponenten wird die Bilanz initialisiert.

Nun werden für alle Eigenschaften die Forderungen überprüft. Für eine offene Forderung wird eine Fehlermeldung generiert. Eine Konfiguration ist zulässig, wenn alle Forderungen durch ihre zugehörigen Angebote erfüllt sind.

5 Einordnung von MOKON

MOKON basiert auf einer speziellen und vom Prinzip her einfachen Konfigurationsmethodik. Wir haben gezeigt, daß sich durch diese Methodik die Intergration unterschiedlicher Aufgabenstellungen erreichen läßt. Der Konfigurierungsprozeß wird durch ein funktionales Abhängigkeitsnetz, das zwischen den zu konfigurierenden Komponenten besteht, gesteuert. Deshalb ist diese Methodik in erster Linie für Anwendungsdomänen geeignet, in denen ein solches Abhängigkeitsnetz den Hauptbestandteil des Konfigurationswissens darstellt.

In diesem Kapitel beschreiben wir zunächst den derzeitigen Stand des Projektes. Desweiteren nehmen wir eine Abgrenzung zu ausgewählten Konfigurationssystemen vor. Abschließend geben wir einen Ausblick auf geplante Erweiterungen.

5.1 Stand des Projektes

Zur Zeit sind in unserem System folgende Funktionalitäten realisiert:

- Konfigurationsaufgaben

  Die folgende Konfigurationsaufgaben sind unter der Vorbedingung entwickelt, daß die Beschreibung einer Anforderungsdefinition bzw. einer Konfiguration flach, d. h. nicht strukturiert (vgl. Abschnitt 2.1) vorliegt. Durch diese Einschränkung können wir in der Anwendungsdomäne Rechnerkonfiguration nur *einen* Arbeitsplatz, jedoch kein Netz von sich gegenseitig beeinflußenden Systemen konfigurieren.

  Im einzelnen gestalten sich Eingabe und Ausgabe auf folgende Weise:

Neukonfiguration	Eingabe: flache Anforderungsdefinition Ausgabe: flache Konfiguration
Konfigurationsprüfung	Eingabe: flache Konfiguration Ausgabe: Fehlerprotokoll
Änderungskonfiguration	Eingabe: flache Anforderungsdefinition flache (installierte) Konfiguration Ausgabe: flache Konfiguration

- Suchraumverwaltung

  Zur Zeit wird eine heuristische und benutzergesteuerte Suchraumverwaltung unterstützt. Zwischen diesen Optionen kann während einer Konsultation jederzeit gewechselt werden.

Implementiert wurde das System in Lucid Common LISP und KEE auf einer IBM 6150 (RT). Dabei wurde das KEE Object System zur Wissensrepräsentation sowie KEE Pictures zur Realisierung der Benutzerschnittstelle verwandt. Abbildung 12

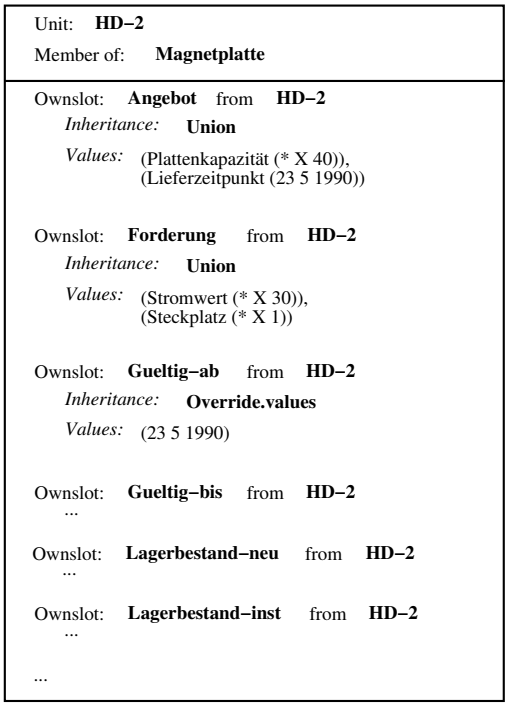

Abbildung 12: *Auszug einer Komponentenbeschreibung*

zeigt in vereinfachter Form die Komponente HD-2 unseres Anwendungsbeispiels in der KEE-spezifischen Objektbeschreibung.

Die Benutzerschnittstelle wird weitestgehend von einer Maus & Menü – Technik unterstützt.

Derzeit wird die vorgestellte Methodik an komplexeren Produkten aus der Praxis (Telekommunikationsbereich) verifiziert.

5.2 Abgrenzung zu anderen Systemen

Mittlerweile existiert eine große Anzahl von Systemen, die für unterschiedliche Aufgabenstellungen aus dem Bereich der Konfiguration bzw. zu deren Unterstützung entwickelt wurden. Eine Abgrenzung zu diesen Systemen kann deshalb nur für eine gewisse Auswahl erfolgen. Grundsätzlich teilen wir diese Auswahl in artverwandte und artfremde Systeme ein. Dabei bezieht sich diese Artverwandheit auf das Prinzip der Wissensverarbeitung (vgl. Abschnitt 3.2.2). Wir betrachten in Relation zu MOKON die folgenden Systeme:

- R1/XCON

  Dieses wissensbasierte System befindet sich im Praxiseinsatz und führt routinemäßig Konfigurationen für VAX-11/780-Computersysteme der Digital Equipment Corporation aus [McDermott 82].

  Hauptunterschied zu MOKON ist die uniforme Darstellung des Konfigurationswissens in Form von Regeln. Diese uniforme Darstellung erschwert besonders die Wissensakquisition und -pflege (vgl. [Soloway et. al. 87]).

- PLAKON

  PLAKON ist eine Entwicklungsplattform, die nicht auf eine spezielle Konfigurierungsaufgabe hin ausgerichtet ist. Es handelt sich um ein System, in dem eine Reihe komplexer Konfigurationsmethodiken implementiert sind. Wesentlicher Bestandteil von PLAKON ist ein hierarchisch strukturiertes Modell der Domäne wie es in [Cunis et al. 87] beschrieben wird.

  Im Gegensatz zu PLAKON soll MOKON kein System für beliebige Problemstellungen aus dem Bereich der Konfiguration darstellen. Vielmehr soll mit einer einfachen Konfigurationsmethodik eine adäquate Unterstützung für bestimmte Anwendungsdomänen angeboten werden.

- Heinrich-Ansatz

  Heinrich [Heinrich 89] bezeichnet seinen Ansatz als ein generisches Modell zur Konfiguration technischer Systeme aus modularen Komponenten. Grundidee ist die Gestaltung des Konfigurationsprozesses auf der Basis von Relationen zwischen Komponenten und Ressourcen. Dafür werden die Relationen „stellt-bereit“, „verbraucht“ und „erfordert“ eingeführt.

  Das Prinzip der Neukonfigurierung erfolgt in MOKON auf vergleichbare Weise, wurde aber um mächtigere Modellierungsmöglichkeiten wie das abstrakte Operatorenkonzept erweitert. Im Gegensatz zu MOKON ist der Heinrich-Ansatz auf die Neukonfigurierung begrenzt.

- AKON

  Hierbei handelt es sich um ein System zur Auftragsprüfung von Kommunikationsanlagen [Neumann, Weiner 89]. Es existieren zwei unterschiedliche Prüfungstechniken: Innerhalb von Basissystemen werden die Komponentenbeziehungen assoziativ modelliert; Beziehungen zwischen Basissystemen sind kausalorientiert beschrieben.

  In MOKON wurde AKON's Prinzip der statischen Bilanzverarbeitung für die Konfigurationsprüfung zu einer dynamischen Methodik für verschiedene Aufgabenstellungen ausgebaut.

Artverwandte Systeme sind das System AKON und der von Heinrich vorgestellte Ansatz, den wir im folgenden mit Heinrich-89 abkürzen.

Die oben skizzierten Unterschiede der vorgestellten Systeme werden wir im folgenden anhand ausgewählter Kriterien tiefergehend beschreiben.

Konfigurationsaufgabe, Ein- und Ausgabe

Wir unterscheiden hier zwischen den Aufgabenstellungen Neukonfigurierung, Ändern einer bereits installierten und Prüfen einer vorgegebenen Konfiguration.

R1/XCON: Prüfung vorgegebener Komponenten eines Computersystems auf Vollständigkeit und Korrektheit. Eingabe ist eine komponentenorientierte, flache Anforderungsdefinition. Die Ausgabe besteht aus der evt. korrigierten und ergänzten Anforderungsdefinition und einer Reihe von Diagrammen, die die räumliche Anordnung visualisieren.

PLAKON: Ausgelegt ist das System für ein möglichst breites Anwendungsfeld. Deshalb sind die mit PLAKON realisierten Aufgabenstellungen von der jeweiligen Anwendung abhängig. U.a. wurde das System XRAY zur Konfigurierung automatischer Röntgenprüfsysteme auf Basis des Kerns von PLAKON realisiert [Strecker 90]. Eingabe ist eine teilinstanziierte (strukturierte) Konzepthierarchie. Das Ergebnis ist eine Vervollständigung dieser Hierarchie.

Heinrich-89: Neukonfiguration am Beispiel modularer speicherprogrammierbarer Steuerungen (SPS). Eingabe ist eine flache Anforderungsdefinition. Die Ausgabe besteht aus einer (flachen) Komponentenliste.

AKON: Konfigurationsprüfung einfach strukturierter Telekommunikationsanlagen. Hier wird die Eingabe strukturiert beschrieben. Neben der strukturierten Ausgabe wird noch ein Fehlerprotokoll erstellt.

MOKON: Neukonfigurierung, Ändern einer installierten und Prüfen einer vorgegebenen Konfiguration am Beispiel eines Rechnersystems. Die Ein- und Ausgabeinformationen sind abhängig von der Aufgabenstellung im Unterabschnitt 5.1 beschrieben.

Wissensrepräsentation

Das Wissen betrachten wir von den Standpunkten der Repräsentationstechnik, der Unterscheidung von Ebenen und der Modellierungstiefe (vgl. Abschnitt 3.1).

R1/XCON: Die fast ausschließlich verwandte Repräsentationstechnik sind Regeln. Das Metawissen wird nicht von dem Objektwissen getrennt beschrieben. In R1/XCON liegt kein tiefes Modell der Anwendungsdomäne vor.

PLAKON: Repräsentationstechnik ist eine taxonomisch-kompositionelle Hierarchie, die um Constraints zur Beschreibung nicht-hierarchischer Relationen ergänzt werden kann. Metawissen kann von Objektwissen getrennt werden. Die Tiefe eines Modells kann jeweils nur bei einer konkreten Anwendung beurteilt werden.

Heinrich-89: Dieser Ansatz verwendet eine taxonomische Hierarchie zur Beschreibung der Komponenten. Zwischen Objekt- und Metawissen wird insofern unterschieden, daß Wissen über die Steuerung der Lösungsfindung getrennt beschrieben ist. Die Abarbeitung benutzt kausale Ursache-Wirkungsbeziehungen und basiert somit auf tieferem Verständnis der Anwendungsdomäne.

AKON: Kern der Wissensrepräsentation ist hier eine kompositionelle Hierarchie. Die kleinste Einheit in dieser Hierarchie ist ein Basiskonzept. Innerhalb eines Basiskonzeptes dienen explizite Relationen und Regeln zur Beschreibung von Konfigurationsvorgaben. Basiskonzeptübergreifende Beziehungen werden über kausale Abhängigkeiten auf einer Bilanz verwaltet. Aufgrund der Aufgabenstellung (Konfigurationsprüfung) und der einfachen Struktur des Modells ist kein Metawissen erforderlich.

MOKON: Die Konzepte zur Repräsentation sind in Unterabschnitt 3.2.1 dargestellt.

Wissensverarbeitung

Interessierende Aspekte bei der Wissensverarbeitung sind hier die Suchraumverwaltung und die Suchraumabarbeitung.

R1/XCON: Der Konfigurationsvorgang ist in sechs Phasen eingeteilt, in denen die zugehörigen Regeln vorwärtsverkettend abgearbeitet werden. Die Anordnungsproblematik z.B. bei der Positionierung von Steckkarten ist mit einem Backtrackingalgorithmus realisiert.

PLAKON: Für die Suchraumverwaltung existieren verschiedene Modi. Sie reichen von einem einfachen Konfigurationsprozeß ohne Backtracking bis hin zum einem durch ein ATMS verwalteten Suchraum. Der Konfigurationsprozeß wird durch eine Agenda (vergleichbar einer Blackboardarchitektur) koordiniert.

Heinrich-89: Kern der Abarbeitung ist ein Backtrackingalgorithmus.

AKON: Weil jeweils nur eine Konfiguration auf Zulässigkeit geprüft wird, muß kein Suchraum generiert und abgearbeitet werden.

MOKON: Kern der Abarbeitung ist ein Backtrackingalgorithmus.

Erklärung, Akquisition

Ein System erhält erst dann den Charakter einer Konfigurationsshell, wenn neben der Wissensrepräsentation und -verarbeitung Module für Erklärung und Wissensakquisition realisiert sind.

R1/XCON: Es werden Prüfungsprotokolle erzeugt. Die Sprache RIME unterstützt die Wissensakquisition dadurch, daß neues anwendungsspezifisches Wissen mit ihr eingegeben und automatisch nach OPS5 übersetzt werden kann.

PLAKON: Module für Erklärung und Wissensakquisition befinden sich in Vorbereitung.

Heinrich-89: Es existieren keine expliziten Module für Erklärung und Akquisition.

AKON: Die Erklärung besteht aus einem Fehlerprotokoll. Die Wissensakquisition wird durch Strukturediteren unterstützt.

MOKON: Es existieren keine expliziten Module für Erklärung und Akquisition. Bei der Konfigurationsprüfung werden bei Bedarf Fehlermeldungen generiert.

Integration

Integration bedeutet hier, inwieweit ein System in das organisatorische und technische Umfeld eingebettet ist bzw. eingebettet werden kann.

R1/XCON: R1/XCON ist vollständig in den betrieblichen Ablauf eingebunden.

PLAKON: Das auf dem PLAKON-Kern basierende XRAY befindet sich noch im Prototypstadium. Bei PLAKON stand die Entwicklung und Erprobung von Verfahren und nicht deren Intergration in das betriebliche Umfeld im Vordergrund. Da es sich um eine Stand Alone-Anwendung handelt, läßt sich eine Integration z.B. über eine lose Datenbankkopplung erreichen.

Heinrich-89: Das System befindet sich noch im Prototypstadium. Hierbei handelt es sich z.Z. auch um ein Stand Alone-System.

AKON: Bei der Konzeption von AKON spielte die Integration des Systems in das operative Umfeld die wichtigste Rolle. Sie wird u.a. durch die Abbildung der Wissensbasis auf eine zentral zugängliche Datenbank erreicht. Zur Zeit befindet sich das System in der Realisierungsphase.

MOKON: Vergleichbar mit PLAKON steht auch hier die Entwicklung und Verifizierung von Methoden im Vordergrund.

5.3 Ausblick

In Zukunft sind auf Basis des vorgestellten Modells von MOKON folgende Erweiterungen denkbar:

- Suchraumverwaltung

  Erweiterung des Systems um weitere heuristische Strategien der Suchraumbearbeitung und Einsatz zusätzlicher Techniken (wie Reason Maintenance System) zur Suchraumverwaltung.

- Strukturierte Anforderungsdefinition und Konfiguration

  Aufhebung oben genannter Vorbedingungen bzgl. der flachen Beschreibung. Diese Erweiterung soll den ersten Grundstein für die Konfigurierung verteilter Systeme bilden.

- Räumliches Wissen

Integration eines Moduls zur Berücksichtigung räumlicher Aspekte der Konfigurierung.

- Modellbasierte Wissensakquisitionskomponente

  Ausnutzung und Interpretation des zugrundeliegenden tiefen Modells zur Gestaltung einer auf dem kausalen Modell basierenden Wissensakquisitionskomponente.

- Modellbasierte Erklärungskomponente

  Ausbau des vorhandenen mehrstufigen Traces zu einer aussagenkräftigeren Erklärungskomponente.

Mit der Realisierung dieser Erweiterungen entsteht eine Konfigurationsshell für spezifische Anwendungsdomänen, die eine integrierte Verarbeitung verschiedener Aufgabenstellungen unterstützt.

6 Literaturverzeichnis

[Bobrow 84] D. G. Bobrow:
Qualitative Reasoning about Physical Structure, Artificial Intelligence, Band 24, 1984

[Cunis et al. 87] R. Cunis, A. Günter, I. Syska:
PLAKON - Ein übergreifendes Konzept zur Wissensrepäsentation und Problemlösung bei Planungs- und Konfigurierungsaufgaben, Expertensysteme '87, Konzepte und Werkzeuge, H. Balzert (Hrsg.), Teubner Verlag, Stuttgart 1987

[De Kleer, Brown 84] J. De Kleer, J. S. Brown:
A qualitative physics based on confluence, Artificial Intelligence Band 24, 1984

[De Kleer 84] J. De Kleer:
How circuits work, Artificial Intelligence Band 28 S.205–280, 1984

[Eich et al. 89] E. Eich, R. Klinger, J. Richter, M. Salfeldt:
Konfigurieren technischer Systeme mittels Expertensystemen, Automatisierungstechnische Praxis, atp31 (1989) 4, S.182-189

[Hein, Tank 90] M. Hein, W. Tank:
Einordnung der Konfigurationsaufgabe in die Welt der Design-Probleme, Beiträge zum 4. Workshop Planen und Konfigurieren, FAW-B-90008, Mai 1990

[Heinrich 89] M. Heinrich:
Ein generisches Modell zur Konfiguration technischer Systeme aus modularen Komponenten, GMD-Arbeitspapier Nr. 388, Mai 1989

[Marcus et al. 85] S. Marcus, J. McDermott, T. Wang:
Knowledge Acquisition for Constructive Systems, Proc. of the 9th Int. Joint Conference on Artificial Intelligence, Los Angeles, 1985

[McDermott 82] J. McDermott:
R1: A Rule-Based Configurer of Computer Systems, Artificial Intelligence 19 (1982) p.39-88

[Neumann, Weiner 89] B. Neumann, J. Weiner:
Anlagenkonzept und Bilanzverarbeitung – Ein Ansatz zur Konfigurationsprüfung, GMD–Arbeitspapier Nr. 388, Mai 1989

[Puppe 88] F. Puppe:
Einführung in Expertensysteme, Springer Verlag, 1988

[Richter 89] M. M. Richter:
Prinzipien der Künstlichen Intelligenz, B.G. Teubner Stuttgart, 1989

[Scheer 88] A.-W. Scheer:
Wirtschaftsinformatik - Informationssysteme im Industriebetrieb, Springer Verlag, 1988

[Soloway et. al. 87] E. Soloway, J. Buchant, K. Jensen:
Assessing the Maintainability of XCON–in–Rime: Coping with the Problems of a VERY Large Rule–Base, Proc. AAAI 87

[Steels 90] L. Steels:
Components of Expertise, AI Magazine, 1990

[Stein 89] B. Stein:
Objektorientierter Modellierungsansatz zur Generalisierung von Versorgungsnetzwissen am Beispiel eines Kupferbeschichtungsautomaten, Diplomarbeit, Universität Karlsruhe, 1989

[Strecker 90] H. Strecker:
Ein Expertensystem zur Konfigurierung automatischer Röntgenprüfsysteme, FAW-B-90008, Mai 1990

[Wielenga, Breuker 86] B. J. Wielenga, J. A. Breuker:
Models of Expertise, Proceedings of the ECAI86, North Holland, Amsterdam, 1986