Vergleich unterschiedlicher Konfigurationsmethoden im Hinblick auf die Nutzbarkeit von Wissens über das Zustandsverhalten der Konfigurationsobjekte

Christian Kühn

DaimlerChrysler AG
Forschung und Technologie
T721, D-70546 Stuttgart
christian.kuehn@dcx.com

Kurzfassung. Während bestehende Methoden zum wissensbasierten Konfigurieren in erster Linie auf Wissen über strukturelle Zusammenhänge basieren, kann es von Vorteil sein, auch Wissen über das Verhalten der Konfigurationsobjekte bzw. des zu konfigurierenden Systems für die Problemlösung zu nutzen. Das vorliegende Papier zeigt Möglichkeiten auf, wie Ansätze zum regel-, struktur-, constraint-, ressourcen- und fallbasierten Konfigurieren so erweitert werden können, dass vorliegendes Verhaltenswissen zur Problemlösung beiträgt und bewertet die Konfigurationsmethoden hinsichtlich ihrer Eignung für ein verhaltensbasiertes Konfigurieren.

1 Einleitung

Für die wissensbasierte Konfigurierung technischer Systeme existiert eine Reihe von Methoden, deren Problemlösungsmechanismen in erster Linie auf Wissen über strukturelle Zusammenhänge, jedoch kaum auf Wissen über das Verhalten des zu konfigurierenden Systems basieren. Dagegen kann die Berücksichtigung des Systemverhaltens bei der Konfigurierung eine große Rolle spielen, insbesondere bei Anwendungen, bei denen dem Benutzer das Verhalten des konfigurierten Systems wichtiger ist als sein struktureller Aufbau. Dieses kann z.B. bei der Konfigurierung softwarebasierter Systeme aus vorgefertigten Softwarebausteinen der Fall sein (vgl. [21], [22]).
Mit der Einbindung von verhaltensbasierten Methoden in bestehende Konfigurationsmethoden verfolgen wir das Ziel einer tieferen Modellierung und damit einer möglichst weitgehenden Unterstützung des Konfigurierungsprozesses durch das Verhaltenswissen. Dieses setzt voraus, dass

- neben dem Wissen über die Struktur des zu konfigurierenden Systems bzw. seiner Komponenten auch Wissen über dessen bzw. deren dynamisches Verhalten adäquat modelliert werden kann.
- das Verhaltenswissen (ebenso wie das Strukturwissen) Konfigurationsentscheidungen ermöglicht, also Schlussfolgerungen auf die Struktur der Konfiguration zulässt.

Das vorliegende Papier zeigt Möglichkeiten auf, wie verschiedene bestehende Konfigurierungsmethoden um verhaltensbasierte Ansätze erweitert werden können, und stellt einen Vergleich der Konfigurationsmethoden im Hinblick auf die Eignung für solche Erweiterungen an. Verhalten kann in unterschiedlichster Form vorliegen bzw. modelliert werden (z.B. kontinuierliches Verhalten in Form von mathematischen Gleichungen bzw. Differentialgleichungen, Interaktionsdiagramme, Simulationsmodelle, diskretes Zustandsverhalten in endlichen Automaten oder Pertrinetzen). Wir beschränken uns hier auf ein Verhaltensmodell in Form von *Statecharts*, das in der Praxis, z.B. bei der Entwicklung und Konfiguration eingebetteter Systeme, eine große Rolle spielt.
Nach einer kurzen Beschreibung der anwendungsseitigen Ausgangsbasis und einer Behandlung der Begriffe *Konfigurieren* und *verhaltensbasiertes Konfigurieren* folgt in Abschnitt 2 eine Kurzübersicht über die untersuchten Konfigurationsmethoden. Anschließend (Abschnitt 3) findet eine Diskussion unterschiedlicher Möglichkeiten zur Ausnutzung von Verhaltenswissen für den Konfigurierungsprozess in den vorgestellten Konfigurationsmethoden statt. Das Papier endet mit einer Zusammenfassung und Gesamtbewertung (Abschnitt 4).

1.1 Ausgangsbasis: Verhaltensmodellierung mit Statecharts

Die folgenden Betrachtungen fokussieren insbesondere auf den Anwendungsbereich der Konfiguration von Software für eingebettete bzw. reaktive Systeme. Typische Eigenschaften solcher Systeme sind u.a. (vgl. [4], [14]):

- Interaktionen zwischen mehreren Prozessen, die parallel ablaufen, sowie zwischen Teilsystem und Systemumgebung (insbesondere Sensoren, Aktuatoren, andere eingebettete Systeme).
- Ein System hat viele mögliche Betriebsszenarios (abhängig vom inneren Zustand).
- Begrenzte Ressourcen (Rechenzeit und Arbeitsspeicher) aus Kostengründen.
- Einsatz oftmals in sicherheitskritischen Bereichen.
- Es bestehen häufig zeitliche Anforderungen an den Betrieb des Systems und die Reaktion auf Inputs.

Für die Modellierung von eingebetteten Systemen werden oftmals Statecharts (vgl. [13], [14]) eingesetzt, die herkömmliche endliche Automaten insbesondere um Techniken zur Repräsentation von Parallelität und Interaktion, aber auch um hierarchische Modularisierung, echtzeitliche Bedingungen und Historie erweitern.
Im folgenden gehen wir von einem Anwendungsszenario aus, in dem ein reaktives System konfiguriert werden soll, für dessen Komponenten Wissen über das reaktive Verhalten bekannt ist und in Form von speziellen Statecharts vorliegt. Abbildung 1 zeigt z.B. das Zustandsverhalten eines Fahrzeug-Softwaremoduls „Geschwindigkeitsabhängige Fenstersteuerung“. Hier wird das Zustandsverhalten mit Zustandsvariablen modelliert, wobei angenommen wird, dass im zeitlichen Ablauf in der Betriebsphase immer genau ein Zustandswert aus einer Menge von vorgegebenen Werten für diese Zustandsvariable aktiv ist. Transitionen zwischen den Zustandswerten sind ereignisbedingte Übergänge, wobei unterschieden wird zwischen internen Ereignissen (also Ereignisse, die durch Zustandswertwechsel anderer Zustandsvariablen ausgelöst sind) und externen Ereignissen, welche im Modell nicht detaillierter abgebildet werden

(hier liegt also die Modellgrenze). Der Zustand einer Komponente zu einem Zeitpunkt ist das Tupel aller aktiven Zustandswerte dieser Komponente, der Zustand des Gesamtsystems (oder des jeweilig konfigurierten Teilsystems) ist das Tupel der Zustände der Systemkomponenten. Solange keine Transitionsbedingung erfüllt ist, bleibt ein Zustand aktiv.
Zusätzlich zu dem Wissen über das Zustandsverhalten der Komponenten liegt Wissen über die Funktion (also das zu erzielende Verhalten) des zu konfigurierenden Gesamtsystems vor, in Form von einzuhaltenden Bedingungen an das Zustandsverhalten, wie z.B. erlaubte oder verbotene Transitionen (näheres s. [22]). Weiterhin kann restriktives Wissen über globale Abhängigkeiten von Zustandsvariablen voneinander vorliegen, die nicht explizit einer Komponente zugeordnet sind.

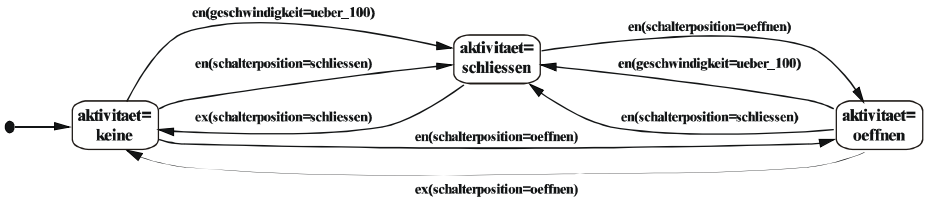

Abb. 1. Modelliertes Verhalten zu einem Softwaremodul „Geschwindigkeitsabhängige Fenstersteuerung“

Echtzeitliche Aspekte des Zustandsverhalten können durch zusätzliche zeitliche Transitionsbedingungen (neben internen und externen), wie z.B. durch Timeout-Bedingungen modelliert werden. Auf die Betrachtung von expliziten zeitlichen Zusammenhängen wird im folgenden verzichtet, das hier angewendete Zeitmodell beruht auf einer impliziten Zeitfortschreitung durch die *Abfolge* von Zustandswechseln; wir gehen davon aus, dass das hier betrachtete Anwendungsfeld mit diesen Mitteln ausreichend modelliert werden kann[1].

1.2 Verhaltensbasiertes Konfigurieren

Unter dem Begriff *Konfigurieren* verstehen wir die schrittweise Zusammensetzung und Ausprägung (Parametrierung) von Komponenten zu einem Gesamtsystem (Konfiguration) unter Einhaltung vorgegebener Restriktionen und vorgegebener Ziele (vgl. [12], [32]). Die Grundlage für das Konfigurieren bildet eine Wissensbasis, welche Wissen über die Komponenten (Domänenobjekte) des Anwendungsbereiches mit ihren Eigenschaften (Parametern), Relationen zwischen den Komponenten (z.B. taxonomische und kompositionelle Beziehungen) und Restriktionen zwischen den Komponenteneigenschaften, sowie Wissen über das Vorgehen und über Aufgabenstellungen beinhaltet.

---

[1] So ist es z.B. möglich, anstelle von Timeout-Bedingungen interne Übergänge zu verwenden, die von einer explizit modellierten Zeitgeberkomponente abhängen, deren Zustandsvariable den aktuellen Zeitwert repräsentiert. Dieser Zeitwert kann durch ein externes Ereignis (ausgelöst durch eine angenommene „Uhr“) wechseln.

Unter dem *verhaltensbasierten Konfigurieren* verstehen wir einen Spezialfall des Konfigurierens, wobei Wissen über das Verhalten der Konfigurationsobjekte oder des Gesamtsystems zur Problemlösung beiträgt. Als Verhalten eines Objekts bzw. eines Systems bezeichnen wir hier die Zustandsänderungen seiner veränderlichen Eigenschaften über der Zeit. Somit zählen zum verhaltensbasierten Konfigurieren prinzipiell auch solche Ansätze, bei denen Simulation zum Einsatz kommt (z.B. [2], [11], [23], [31]), jedoch setzt ein simulationsbasiertes Konfigurieren voraus, dass die zu simulierenden Verhaltensmodelle (wenn auch nur einem Teilsystem einer Konfiguration zugeordnet) jeweils vollständig bekannt sind. Demgegenüber betrachten wir hier eine Form des verhaltensbasierten Konfigurierens, die auf der Basis einer generischen Repräsentation von Verhalten auch auf unvollständig bekanntem Verhaltenswissen Konfigurationsentscheidungen treffen kann und damit in wesentlich früheren Phasen als die simulationsbasierten Ansätze zur Problemlösung beiträgt. Wenn wir nachfolgend von *verhaltensbasiertem Konfigurieren* sprechen, so beschränken wir uns auf eine solche Methoden und schließen simulationsbasierte Ansätze aus der Betrachtung aus.

2 Methoden zum Konfigurieren

Im folgenden wird kurz eine Auswahl von Methoden zum Konfigurieren vorgestellt, die anschließend hinsichtlich der Erweiterbarkeit um ein verhaltensbasiertes Vorgehen untersucht werden.. Die vorgestellten Methoden unterscheiden sich wesentlich in der Art der Modellierung ebenso wie in der Weise, wie Konfigurationsentscheidungen getroffen werden. Der Einsatz dieser Problemlösungsmethoden in ihrer reinen Form ist eher selten zu finden, dagegen bietet sich häufig eine Kombination der Methoden an. Einen weiteren Überblick über Konfigurationsmethoden, sowie -anwendungen und -Tools liefern [10], [12], [29], [32].

2.1 Regelbasiertes Konfigurieren

Grundprinzip ist die Formulierung von Wissen in Form von assoziativen Regeln, bestehend aus Bedingungs- und Aktionsteil. Regelbasierte Systeme für die Konfigurierung verwenden i.a. einen vorwärtsverkettenden, also datenorientierten Inferenzmechanismus. Für das Vorgehen bei der Problemlösung, insbesondere die Auswahl einer Regel aus einer Konfliktmenge an aktuell anwendbaren Regeln, existierten unterschiedliche Strategien, sowohl domänenunabhängige (z.B. Auswahl der aktuellsten oder speziellsten Regel) als auch auf Domänenwissen basierende (z.B. Prioritäten, Bewertungsfunktionen, Metaregeln).

2.2 Strukturbasiertes Konfigurieren

Die Domänenobjekte (mit Parametern) werden häufig in taxonomischen Hierarchien beschrieben, kompositionelle Abhängigkeiten zwischen den Objekten in Zerlegungshierarchien. Eine solche Begriffshierarchie beschreibt generisch die Menge der mög-

lichen Lösungen (Konfigurationen), die der Konfigurationsprozess bis zum Erreichen der Lösungskonfiguration schrittweise einschränken soll. Beim strukturbasierten Konfigurieren orientiert sich der Problemlösungsvorgang an der Struktur des Domänenmodells, wie z.B. beim Skelett-Konstruieren [28] oder bei der begriffshierarchie-orientierten Kontrolle [7].

2.3 Constraint-basiertes Konfigurieren

Die Modellierung und Verarbeitung von ungerichteten Abhängigkeiten zwischen Domänenobjekten kann mithilfe von Constraints erfolgen. Besonderheiten bei der Konfiguration bestehen u.a. darin, dass mit dem schrittweisen Aufbau der Lösung erst nach und nach das Constraint-Netz aufgebaut werden kann. Hierzu wurden – gegenüber herkömmlichen Methoden der Constraint Satisfaction (CSP) – weitergehende Ansätze entwickelt, wie z.B. die Dynamic CSPs (s. [24], [30], [32]), der dreistufige Constraint-Ansatz in den Systemen PLAKON [3], KONWERK [8], [9] und EngCon [1] sowie die bedingte Propagation in ConBaCon (s. [18], [19]).

2.4 Ressourcenorientiertes Konfigurieren

Das ressourcenorientierte Ansatz basiert auf einem Domänenmodell, bei dem Beziehungen zwischen Komponenten durch den Austausch von Ressourcen (abstrakten Leistungen) beschrieben werden (s. [5], [15], [16], [20]). Dabei wird von dem Prinzip ausgegangen, dass eine Komponente eine Menge von Ressourcen fordert bzw. anbietet. Der Problemlösungsprozess beruht auf initialen Ressourcenforderungen als Aufgabenstellung, die sukzessive in einem iterativen Prozess durch Hinzunahme weiterer Komponenten ausgeglichen werden (Bilanzierung).

2.5 Fallbasiertes Konfigurieren

Beim fallbasierten Konfigurieren werden bereits erstellte Konfigurationen in einer Fallbibliothek konserviert und zur späteren Lösung von ähnlichen Konfigurationsaufgaben herangezogen (s. [12], [27], [33]). Wesentliche Schritte sind dabei die Auswahl eines ähnlichen Falls (z.B. durch Generalisierung [33] oder mittels induzierten Problemklassen [26], [27]) sowie die Übernahme und Anpassung einer Lösung an die aktuelle Aufgabenstellung.

3 Erweiterung der bestehenden Konfigurationsmethoden um verhaltensbasiertes Schließen

In diesem Abschnitt werden Überlegungen angestellt, wie eine Modellierung und Auswertung von Wissen über das Verhalten der Konfigurationsobjekte gewinnbringend in die zuvor beschrieben Konfigurationsmethodiken eingebunden werden kann, und anschließend bewertet.

3.1 Regelbasiertes Konfigurieren mit Verhaltenswissen

Dass das modellierte Verhaltenswissen in Statecharts mit bedingten Transitionen vorliegt und die Transitionen damit direkt auf Bedingung-Aktion-Regeln abgebildet werden können, könnte nahe legen, die Regelmenge an Konfigurationsregeln einfach um solche „Verhaltensregeln“ zu erweitern. Anders als beim Planen jedoch, wo solche Regeln zum Einsatz kommen (z.B. als Operatoren im Problemlöser *STRIPS*, s. [17]), geht es hier nicht darum, eine Lösungssequenz von Handlungsoperatoren oder *eine* Zustandsfolge zu finden, sondern das Wissen über *potenzielles* Verhalten für die Auswertung von Konfigurationsregeln zu nutzen.
Stattdessen schlagen wir vor, Verhaltensinformationen (z.B. eine oder mehrere Transitionen, im folgenden als *Modellfragmente* bezeichnet) als Wissenseinheiten zu behandeln, die mit anderen Wissensarten oder mit weiteren Verhaltensinformationen in Konfigurationsregeln verknüpft werden können. Solche Modellfragmente können sowohl im Bedingungs- als auch im Aktionsteil von Konfigurationsregeln verwendet werden, so dass sowohl von Verhalten auf Elemente der Lösungskonfiguration als auch von Konfigurationsinformationen auf Verhalten geschlossen werden kann. Beispiele für solche Regeln sind:

```
WENN Modellfragment_23 DANN Software-Modul
  Geschwindigkeitsabhängige_Fenstersteruerung verwenden
mit Modellfragment_23:
  Transition von offen nach geschlossen bei
  en(Geschwindigkeit=über_100)
```

Entsprechende Modellfragmente können auch in domänenabhängigen Metaregeln benutzt werden, um Verhaltenswissen für Kontrollentscheidungen (z.B. Auswahl einer Regel aus einer Konfliktmenge) auszunutzen.
Eine Schwierigkeit einer solchen Verwendung von Modellfragmenten in Regeln besteht darin, dass während des Konfigurationsprozesses für ein Modellfragment, das nicht Bestandteil der aktuell gültigen Datenmenge ist, nicht bekannt ist, ob es später noch in die gültige Datenmenge aufgenommen werden kann oder ob seine Aufnahme auszuschließen ist (dahinter steht die Annahme der Closed-World-Assumption). Für den zweiten Fall müssten entsprechend auch auszuschließende Modellfragmente explizit modelliert werden, was den Aufwand der Modellierung erhöht. Besonderen Aufwand verursacht aber die Modellierung der Abhängigkeiten zwischen Modellfragmenten selbst. Hier müssten die den Statechart-Transitionen zugeordneten Bedingungen in explizite Regelform überführt werden.

Fazit: Der Einsatz von Verhaltenswissen, das generisch repräsentiert und ggf. nur unvollständig bekannt ist, erscheint beim regelbasierten Konfigurieren nur bedingt geeignet. Dagegen kann aber die Verwendung von Simulationsmodellen (unter Trennung des Verhaltenswissen von den Konfigurationsregeln) sinnvoll eingesetzt werden, um Konflikte zu erkennen und z.B. die Rücknahme von Entscheidungen auszulösen bzw. als Bewertungsfunktion für die Auswahl einer Regel aus einer Konfliktmenge.

3.2 Strukturbasiertes Konfigurieren mit Verhaltenswissen

Ziel ist es, mithilfe des Wissens über das Zustandsverhalten von Konfigurationsobjekten während des Konfigurationsprozesses Entscheidungen über die Struktur der Lösungskonfiguration zu unterstützen, insbesondere Spezialisierungsschritte, Zerlegungsschritte und Schritte zur Integration von Lösungsobjekten in die Lösungshierarchie.
Wir schlagen dazu vor, das Verhaltens eines Domänenobjekts als zusätzliche Eigenschaft des Objekts zu modellieren. Dazu wird einem Domänenobjekt z.B. eine Menge von Zustandsvariablen (vgl. Abschnitt 1.1) und Statecharts zugeordnet. Als eine geeignete Integration in eine Begriffshierarchie sehen wir die Beschreibung des Verhaltens mit Objektdeskriptoren, für die Subsumptionsbeziehungen definiert sind, so dass das Verhalten selbst spezialisierbar ist. Bei dem Ansatz *ABACUS* (*A behavior-based configuration using statecharts*, s. [22]) werden aus den Statecharts der betroffenen Domänenobjekte Zustandsübergangsgraphen abgeleitet, ähnlich zu Envisionment-Graphen in einigen Ansätzen der qualitativen Simulation (vgl. [6], [25]). Zusätzlich werden für die Konfiguration die Kanten mit einem M- („muss“), V- („verboten“) oder K- („kann“) Label versehen. Dabei genügt es, z.B. nur die M- und K-Relationen aufzuzählen, da eine nicht existierende Kante automatisch als V-Kante betrachtet werden kann. K-Kanten können zu M- oder V-Kanten spezialisiert werden. Auf diese Weise können Verhaltensalternativen in der gesamten Begriffshierarchie behandelt werden und somit Vergleichsoperationen und Spezialisierungen durchgeführt werden.
Durch die Verwendung von solchen Verhaltensdeskriptoren kann das Verhalten einer Lösungsinstanz gegenüber dem entsprechenden Domänenobjekt eingeschränkt sein. Ursache für solche Einschränkungen können z.B. in einer Aufgabenstellung vorgegebene Anforderungen[2] oder Constraints sein. Diese Einschränkung von Instanzverhalten ermöglicht Spezialisierungsentscheidungen, d.h. aufgrund des Verhaltens von Instanzen kann ggf. geschlossen werden, dass die Spezialisierung nur zu einem oder einigen der potenziellen Unterkonzepte zulässig ist. Ebenso kann bei einer Integration aufgrund des Instanzverhaltens ggf. ein zuzuordnendes Aggregat bestimmt werden. Des weiteren ist es möglich, aus den Verhaltensmodellen auf Instanzebene die Notwendigkeit der Existenz bestimmter Domänenobjekte in der Lösungskonfiguration abzuleiten, die in diesem Fall instantiiert und zu einem späteren Zeitpunkt in die Lösung integriert werden können.

Fazit: Die Modellierung von Verhalten als Eigenschaft von Objekten einer Begriffshierarchie kann häufig eine adäquate Form der Wissensrepräsentation darstellen. Schlussfolgerungsverfahren zur Ableitung von strukturellen Konfigurationsentscheidungen aus diesem Verhaltenswissen können die Effektivität des Konfigurationsprozesses erheblich verbessern.

---

[2] Anforderungen an das Verhalten können ähnlich spezifiziert werden wie Constraints für das Verhalten (s. Abschnitt 3.3). Die Unterscheidung zwischen Verhaltens-Constraints und -anforderungen ist vergleichbar mit kontextunabhängigen und kontextabhängigen Constraints in [18], [19].

3.3 Constraint-basiertes Konfigurieren mit Verhaltenswissen

Zur Repräsentation von Restriktionen an das Verhalten der Konfigurationsobjekte sind Beschreibungsmittel notwendig, die die Constraints heutiger Konfigurationssysteme i.a. nicht unterstützen. Constraints bezogen auf das Zustandsverhalten, die zur Modellierung einer Anwendungsdomäne notwendig sein können, sind z.B.:

- Constraints, die eine *Transition* zwischen zwei Zuständen fordern oder verbieten, falls eine vorgegebene Bedingung erfüllt ist.
- Constraints, die das *Verlassen eines Zustands* fordern oder verbieten, falls eine vorgegebene Bedingung erfüllt ist.
- Constraints, die das *Eintreten eines Zustands* fordern oder verbieten, falls eine vorgegebene Bedingung erfüllt ist.
- Constraints, die das *gleichzeitige Auftreten zweier oder mehrerer Zustände* fordern oder verbieten.

Mit diesen Verhaltens-Constraints[3] werden somit keine vollständigen Statecharts beschrieben, sondern Bedingungen an das Zustandsverhalten der Instanzen einer Teilkonfiguration. Dabei können die Verhaltens-Constraints Restriktionen an das Gesamtverhalten des Systems sein, nicht nur an das lokale Verhalten einer einzelnen Komponente.[4]

Ebenso wie die Struktur-Constraints können die Verhaltens-Constraints eine passive und eine aktive Rolle haben: Die passive Rolle zur Prüfung der Konsistenz von Verhaltens-Constraints gegenüber dem (auch unvollständigen) Lösungsverhalten. Die aktive Rolle ist die Einschränkung des potenziellen Verhaltens einer oder mehrerer zur aktuellen Teilkonfiguration gehörenden Instanzen. Bezogen auf den im vorigen Abschnitt genannten Ansatz bedeutet dies die Reduzierung des Zustandsübergangsgraphen. Dabei können K-Transitionen zu V- oder M-Transitionen soweit eingeschränkt werden, wie noch alle Verhaltens-Constraints zu dem Zustandsverhalten der aktuellen Teilkonfiguration konsistent sind.

Grundsätzlich sind auch Constraints denkbar, die Restriktionen gleichzeitig an strukturelle sowie an Verhaltenseigenschaften darstellen bzw. als Kombination von Verhaltens- und Struktur-Constraints gebildet werden (z.B. als Compositional Constraints in [18], [19]). Ggf. müsste man dann auf eine Trennung von Verhaltens- und Struktur-Constraint-Netzen und -Propagationszyklen verzichten. Diese kann aber sehr nützlich sein, um z.B. die Häufigkeit der unterschiedlich komplexen Propagationszyklen geeignet einzustellen.

Auch constraintbasierte Konfigurierungsmethoden können somit geeignet durch verhaltensbasierte Modellierungs- und Schlussfolgerungsverfahren ergänzt werden bzw. als eigenständige Verhaltens-Constraint-Systeme eingeführt werden. Geeignet erscheint insbesondere die Kombination mit den in Abschnitt 3.2 beschriebenen Techniken.

---

[3] Im folgenden sprechen wir von „Verhaltens-Constraints“ und bezeichnen andere Arten von Constraints beim Konfigurieren als „Struktur-Constraints“.

[4] Dieses entspricht im Prinzip *globalen Kriterien* zur Filterung von qualitativem Verhalten (vgl. [25]).

3.4 Ressourcenorientiertes Konfigurieren mit Verhaltenswissen

Die Modellierung von Verhalten beim ressourcenbasierten Konfigurieren kann z.B. durch einen eigenen Ressourcentyp für das Verhalten erfolgen. So könnte das Zustandsverhalten aus Abb. 1 z.B. eine Ressource darstellen. Während auf diese Weise solche Verhaltens-Ressourcen durch die Zustandsvariablen und Statecharts beschrieben werden, kann das Verhalten in den Ressourcenanforderungen – wie die Anforderungen bei dem strukturbasierten Ansatz (Abschnitt 3.2) bzw. die Constraints (Abschnitt 3.3) – wieder unvollständig beschrieben sein. Die Bilanzierungsfunktion lässt sich nach dem in den vorigen zwei Abschnitten beschriebenen Prinzip realisieren, das zu den Anforderungen ein jeweils konsistentes Zustandsmodell liefert. Ein Beispiel für die Zuordnung von Verhalten-Ressourcen zu Komponenten zeigt Abb. 2.

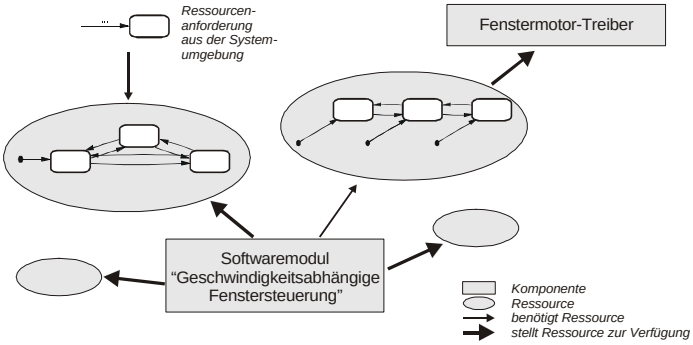

Abb. 2. Beispiel, wie von der Forderung nach einer Funktionalität, die ab einer vorgegebenen Fahrgeschwindigkeit alle Fenster im Fahrzeug schließt, über Verhaltens-Ressourcen auf notwendige Softwaremodule geschlossen wird.

Ein Problem besteht darin, dass das lokale Verhalten einer Komponente (also die Verhaltensressource) von dem Verhalten anderer Komponenten abhängen kann. Dieses lässt sich dadurch lösen, dass jede Zustandsvariable als eigene Ressource modelliert wird. Dahinter steht die Betrachtungsweise, dass eine Zustandsvariable eine teilbare (also nicht-exklusive) „Informationsressource“ ist. So kann eine Komponente Informationen über das Zustandsverhalten einer Zustandsvariablen liefern und dafür Informationen über das Zustandsverhalten anderer Zustandsvariablen benötigen. Entsprechendend dem Prinzip der Minimalität [20] kann auch hier eine Heuristik angewandt werden, die bei der Auswahl einer Komponente zur Befriedigung einer Ressourcenanforderung diejenige auswählt, die die wenigsten Referenzen auf andere Zustandsvariablen aufweist (also am wenigsten andere Informationsressourcen benötigt). Dieses entspricht dem „allgemeinsten“ Verhalten bei den Verhaltensdeskriptoren in Abschnitt 3.2.

Bewertung: Auch der ressourcenbasierte Konfigurierungsansatz kann sinnvoll durch eine verhaltensbasierte Methodik ergänzt werden. Hat man es jedoch mit einer Vielzahl von Restriktionen zu tun, die das Gesamtverhalten des Systems – nicht nur das lokal in *einer* Ressource beschriebene Verhalten – betreffen, so ist die Adäquatheit der ressourcenbasierten Methodik in Frage zu stellen.

3.5 Fallbasiertes Konfigurieren mit Verhaltenswissen

Verhaltenswissen kann sowohl die Fallauswahl als auch die Übernahme und Adaption von Fällen unterstützen.
Ist jedem Fall einer Fallbibliothek ein Verhaltensmodell zugeordnet, so kann dieses bei der Auswahl als „Index“ dienen: Es ist der Fall zu ermitteln, dessen Verhalten vorgegebene Anforderungen erfüllt bzw. bestmöglich erfüllt. Für die Auswahl „ähnlicher“ Fälle muss eine Metrik für „ähnliches Verhalten“ aufgestellt werden, falls möglich domänenabhängig (z.B. durch Zuordnung von Gewichten zu Verhaltensanforderungen), aber es sind auch domänenunabhängige Metriken möglich (z.B. die Bildung des Quotienten aus Anzahl an erfüllten und verletzten Verhaltensanforderungen).
Die Zuordnung von Verhalten zu den Fällen setzt nicht voraus, dass diese Fälle „verhaltensbasiert“ konfiguriert wurden. Das Fallverhalten kann z.B. zu einem späteren Zeitpunkt, nach Erzeugung des Falles, manuell modelliert werden, oder durch Simulation erzeugt werden, wie dieses von einigen Konfigurationstools unterstützt wird (z.B. [2], [11], [31]).
Durch die Einführung einer Subsumptionsbeziehung für Zustandsverhaltensmodelle (s. Abschnitt 3.2) ist auch die Generalisierung von Objektverhalten möglich, so dass eine Fallauswahl durch Generalisierung unterstützt wird.

Bei der Übernahme eines Falls (Lösungsübernahme) kann Verhaltenswissen zur Durchführung von Anpassungsoperationen dienen: z.B. Hinzufügen eines Lösungsobjekts mit benötigtem Verhalten, Löschen eines Objekts, dessen Verhalten inkonsistent zur Aufgabenstellung ist, oder Ersetzen eines Objekts durch ein anderes Objekt mit konsistentem Verhalten. Diese Operationen können analog zu den Konfigurationsschritttypen Spezialisieren, Zerlegen und Integrieren aufgestellt werden (vgl. Abschnitt 3.2).
Wird eine Fallübernahme nach dem Prinzip des „vorgangsbasierten Wissenstransfers“ vorgenommen, wird also der Kontrollablauf übernommen, so kann dieses einen kombinierten verhaltens- und strukturbasierten Konfigurationsvorgang erheblich verbessern: Der übernommene Kontrollablauf kann bestimmen, wann die evtl. zeitlich aufwendige Auswertung von Verhaltenswissen durchzuführen ist und wann diese zu unterlassen ist und damit die Effizienz des Konfigurationsablaufs verbessern.

Insgesamt kann die Nutzung von Verhaltenswissen beim fallbasierten Konfigurieren als sehr geeignet gesehen werden. Insbesondere kann Verhaltenswissen zur Lösung des grundsätzlichen Problems beim fallbasierten Konfigurieren beitragen, dass Anforderungen oftmals nicht vollständig explizit vom Benutzer als geschlossene Aufgabenstellung vorgegeben werden, sondern viele Anforderungen als Benutzerentscheidungen in den Konfigurationsprozess einfließen, ohne dass diese für eine spätere Fallauswahl verwendet werden können. Verhaltensmodelle können hier die Möglichkeiten für eine Anforderungsmodellierung verbessern und damit auch die Effektivität beim fallbasierten Vorgehen erhöhen.

4 Zusammenfassung und Gesamtbewertung

Ausgehend von der Prämisse, dass man zusätzlich zum Konfigurationswissen Wissen über das Zustandsverhalten der Konfigurationsobjekte und Anforderungen an das Verhalten vorliegen hat, wurde untersucht, inwieweit dieses Verhaltenswissen für den Konfigurationsprozess ausgenutzt werden kann. Für unterschiedliche Konfigurierungsmethoden wurden Ansätze vorgeschlagen. So können z.B. Fragmente von Verhaltensmodellen in Regeln so verwendet werden, dass sowohl von Informationen über das Verhalten einer Konfiguration auf Informationen über die Konfigurationslösung geschlossen werden kann und umgekehrt. Für das strukturbasierten Konfigurieren wurde die Modellierung von Zustandsverhalten als eine besondere Eigenschaft von Domänenobjekten vorgeschlagen, wobei die Einführung eines Objektdeskriptors, der die Spezialisierbarkeit von Verhaltens definiert, strukturelle Entscheidungen im Konfigurationsprozess unterstützt. Während die Constraint-Mechanismen jetziger Konfigurationssysteme kaum Restriktionen an das Verhalten von Konfigurationsobjekten unterstützen, können solche als zusätzliche Constraints eingeführt werden, die sowohl die Konsistenz von (Teil-)Konfigurationen prüfen als auch das potenzielle Verhalten einer Teilkonfiguration – analog zu strukturellen Eigenschaften (Parametern) von Konfigurationsobjekten – einschränken. In das ressourcenbasierte Konfigurationsmodell lässt sich Zustandsverhalten als Ressourcentyp einführen, zusätzlich muss eine Bilanzierungsfunktion zur Bilanzierung von angefordertem und angebotenem Verhalten definiert werden. Beim fallbasierten Konfigurieren kann Verhaltenswissen sowohl die Fallauswahl als auch die Übernahme und Adaption von Fällen unterstützen.

Zwar soll an dieser Stelle keine Pauschalempfehlung für *eine* anzuwendende Methodik gegeben werden, denn letztendlich hängt es vom Einzelfall ab, in welcher Form das Konfigurationswissen der jeweiligen Domäne vorliegt. Dennoch bewerten wir die verschiedenen Methoden grundsätzlich – obwohl verhaltensbasierte Inferenztechniken prinzipiell in alle der untersuchten Konfigurationsmethoden integriert werden können – für eine solche Integration als unterschiedlich geeignet. Am wenigsten geeignet erscheint der regelbasierte Konfigurationsansatz mit einer expliziten Modellierung von jeweils gültigem und auszuschließendem Verhaltenswissen und einem dadurch hohen Modellierungsaufwand. Als besser geeignet erscheint der ressourcenbasierte Ansatz, jedoch kann die Berücksichtigung von globalen Restriktionen an das Verhalten (also an das Gesamtverhalten einer Konfiguration) zu Problemen führen. Als sehr vorteilhaft sehen wir dagegen die Integration verhaltensbasierter Methoden in die strukturbasierte, constraintbasierte und fallbasierte Konfigurationsmethodik: Der strukturbasierte Ansatz liefert einen guten Rahmen, um Verhaltenswissen adäquat Komponenten zuzuordnen und durch die Anordnung in Begriffshierarchien Schlussfolgerungen vom Verhalten der Objekte auf die Struktur einer Konfiguration zu ermöglichen. Constraints ermöglichen die Formulierung und Auswertung von Restriktionen an das Systemverhalten – sowohl an das lokale Verhalten von Komponenten als auch an das globale Systemverhalten. Als besonders geeignet erscheint eine kombinierter Ansatz aus struktur-, constraint- und verhaltensbasiertem Vorgehen. Beim fallbasierten Konfigurieren können Verhaltensmodelle zur Beschreibung von Benutzeranforderungen sowie Systemverhalten der Falllösungen einen wesentlichen Beitrag sowohl für die Fallauswahl als auch für die Fallübernahme liefern.

Natürlich hängt es von der Art des vorliegenden Wissens ab, ob ein verhaltensbasierter Ansatz gewählt werden sollte. Dabei ist das verhaltensbasierte Konfigurieren weniger als alleinige Problemlösungsmethode zu sehen, sondern eher als eine Methode, die mit einer der oben untersuchten Konfigurationsmethoden – bzw. mit mehreren – geeignet zu kombinieren ist.

Literatur

1. Arlt, V.; Günter, A.; Hollmann, O.; Wagner, T.; Hotz, L.: *EngCon - Engineering & Configuration.* Friedrich, G. (Hrsg.). *Configuration – Papers from the AAAI Workshop,* Orlando, Florida, AAAI Press, California (1999)
2. Brinkop, A.: *Variantenkonstruktion durch Auswertung der Abhängigkeiten zwischen den Konstruktionsbauteilen*. Infix, St. Augustin (1999)
3. Cunis, R.; Günter, A.; Strecker, H. (Hrsg.): *Das PLAKON Buch - Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen*. Springer, (1991)
4. Douglass, B. P.: *Real-Time UML: Developing Efficient Objects for Embedded Systems*. Addison-Wesley, Reading, MA u.a. (1998)
5. Emde, W.; Rahmer, J.; Voß, A.; Beilken, C.; Börding, J.; Orth, W.; Petersen, U.; Schaaf, J.; Spenke, M.; Wrobel, S.: *Interactive Configuration in KIKon.* Mertens, P. and Voss, H. (Hrsg.). *Expertensysteme 97,* Bad Honnef am Rhein, Infix (1997)
6. Forbus, K. D.: *Qualitative Reasoning*. Tucker, A. B. (Hrsg.). *CRC Computer Science and Engineering Handbook.* CRC Press, Boca Raton, Florida (1996)
7. Günter, A.: *Flexible Kontrolle in Expertensystemen für Planungs- und Konfigurierungsaufgaben in technischen Domänen*. Infix, St. Augustin (1991)
8. Günter, A.: *KONWERK - ein modulares Konfigurierungswerkzeug.* Maurer, F. and Richter, M. M. (Hrsg.). *Expertensysteme 95,* Kaiserslautern, Infix (1995)
9. Günter, A. (Hrsg.): *Wissensbasiertes Konfigurieren - Ergebnisse aus dem Projekt PROKON*. Infix, St. Augustin (1995)
10. Günter, A.; Kreuz, I.; Kühn, C.: *Kommerzielle Software-Werkzeuge für die Konfigurierung von technischen Systemen*. Künstliche Intelligenz 13 (3) (1999) 61-65
11. Günter, A.; Kühn, C.: *Einsatz der Simulation zur Unterstützung der Konfigurierung von technischen Systemen.* Mertens, V. (Hrsg.). *Expertensysteme '97 - Beiträge zur 4. Deutschen Tagung Wissensbasierte Systeme (XPS-97),* Bad Honnef am Rhein, Infix (1997)
12. Günter, A.; Kühn, C.: *Knowledge-Based Configuration - Survey and Future Directions.* Puppe, F. (Hrsg.). *XPS-99: Knowledge Based Systems - Survey and Future Directions, 5th Biannual German Conference on Knowledge Based Systems,* Springer (1999)
13. Harel, D.: *Statecharts: a Visual Formalism for Complex Systems*. Science of Computer Programming, 8 (1987) 231-274
14. Harel, D.; Politi, M.: *Modeling Reactive Systems with Statecharts - The Statemate Approach*. McGraw-Hill, New York u.a. (1998)
15. Heinrich, M.: *Ressourcenorientiertes Konfigurieren*. Künstliche Intelligenz 7 (1) (1993) 11-15
16. Heinrich, M.; Jüngst, E. W.: *Konfigurieren technischer Einrichtungen ausgehend von den Komponenten des technischen Prozesses: Prinzip und erste Erfahrungen*. F. Puppe, A. Günter (Hrsg.) Expertensysteme 93 (1993) 98-111
17. Hertzberg, J.: *Planen - Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*. B. I. Wissenschaftsverlag, Mannheim u.a. (1989)
18. John, U.; Geske, U.: *Constraint-logische Modellierung und Bearbeitung technischer Konfigurationsprobleme - Das System ConBaCon.* Köhler, J. (Hrsg.). *Workshop Planning and Configuration (PuK), 5th Conference on Knowledge-Based Systems,* Würzburg (1999)

19. John, U.; Geske, U.: *Reconfiguration of Technical Products Using ConBaCon.* Friedrich, G. (Hrsg.). *Configuration – Papers from the AAAI Workshop,* Orlando, Florida, AAAI Press, California (1999)
20. Jüngst, W. E.; Heinrich, M.: *Using Resource Balancing to Configure Modular Systems*. IEEE Intelligent Systems Jul/Aug 98 (1998) 50-58
21. Kühn, C.: *Requirements for Configuring Complex Software-Based Systems.* Friedrich, G. (Hrsg.). *Configuration – Papers from the AAAI Workshop,* Orlando, Florida, AAAI Press, California (1999)
22. Kühn, C.: *Modeling Structure and Behavior for Knowledge-Based Software Configuration.* Sauer, J. (Hrsg.). *Proceedings Workshop Planen und Konfigurieren, 14th European Conference on Artificial Intelligence (ECAI),* Berlin (2000)
23. Kühn, C.; Günter, A.: *Combining Knowledge-Based Configuration and Simulation.* Büning, H. K. (Hrsg.). *Workshop "Simulation in Wissensbasierten Systemen" (SiWiS-98), report tr-ri-98-194,* Universität-GH Paderbon, Germany (1998)
24. Mittal, S.; Falkenhainer, B.: *Dynamic constraint satisfaction problems*(Hrsg.). *AAAI Conference* (1990)
25. Montag, M.; Struß, P.: *Qualitatives und modellbasiertes Schließen*. Görz, G. (Hrsg.). *Einführung in die künstliche Intelligenz.* Addison-Wesley, Berlin (1995) 87-128
26. Pfitzner, K.: *Die Auswahl von Bibliothekslösungen mittels induzierter Problemklassen*(Hrsg.). *Workshop Planen und Konfigurieren,* FAW Ulm (1990)
27. Pfitzner, K.: *Fallbasierte Konfigurierung technischer Systeme*. Künstliche Intelligenz 7 (1) (1993) 24-30
28. Puppe, F.: *Problemlösungsmethoden in Expertensystemen*. Springer, Berlin u.a. (1990)
29. Sabin, D.; Weigel, R.: *Product Configuration Frameworks - A Survey*. IEEE Intelligent Systems Jul/Aug 98 (1998) 50-58
30. Sabin, M.; Freuder, E. C.: *Detecting and Resolving Inconsistency and Redundancy in Conditional Constraint Satisfaction Problems.* Friedrich, G. (Hrsg.). *Configuration – Papers from the AAAI Workshop,* Orlando, Florida, AAAI Press, California (1999)
31. Stein, B. M.: *Functional Models in Configuration Systems*, Universität-GH Paderborn(1995)
32. Stumptner, M.: *An Overview of Knowledge-Based Configuration*. AI Communications, 10 (2) (1997) 111-126
33. Vietze, T.: *Ein Konzept für fallbasiertes Konfigurieren auf der Basis von Generalisierungen*. Günter, A. (Hrsg.). *Wissensbasiertes Konfigurieren - Ergebnisse aus dem Projekt PROKON*. Infix, St. Augustin (1995) 193-200