Konfigurierung eingebetteter Systeme mit Wissen über die Struktur und das Zustandsverhalten der Konfigurationsobjekte

Christian Kühn

DaimlerChrysler AG, Forschung und Technologie, T721, 70546 Stuttgart
christian.kuehn@daimlerchrysler.com

Kurzfassung. Mit zunehmender Komplexität und zunehmendem Funktionsumfang von konfigurierbaren Produkten steigt die Notwendigkeit, Wissen über Funktionen und Verhalten im Domänenmodell für eine wissensbasierte Konfigurierung zu berücksichtigen. Dies gilt insbesondere für viele Anwendungsgebiete der Konfigurierung eingebetteter Systeme. Es wird hier eine Methodik vorgestellt, die den Einsatz von Wissen über das Zustandsverhalten der Komponenten und des Gesamtsystems für die wissensbasierte Konfigurierung technischer Systeme ermöglicht.

1 Einleitung

1.1 Motivation

Bei Zunahme des Automatisierungsgrades in vielen Anwendungsgebieten, wie Produktions- und Anlagentechnik, Gebäudetechnik, Verkehrstechnik, ebenso wie Kraftfahrzeug-, Bahn- und Luftfahrttechnik, ist gleichzeitig eine Zunahme sowohl der Komplexität dieser Systeme als auch des Variantenreichtums solcher Systeme zu erkennen. Der zunehmende Reichtum an Varianten bedeutet zum einen eine steigende Vielfalt an verbaubaren Komponenten und deren Ausprägungen (strukturelle Vielfalt), zum anderen aber auch eine steigende Vielfalt an realisierbarer Funktionalität (funktionale Vielfalt). Letzteres bedeutet nicht nur eine höhere Anzahl von Funktionen, sondern auch erweiterte und neue Möglichkeiten, Funktionen zu strukturieren und durch verteilte und vernetzte Komponenten zu realisieren.

Gerade bei steigender funktionaler Vielfalt in den o.g. Anwendungsbereichen steigt gleichzeitig ein Bedarf, funktionales Wissen und Wissen über das Verhalten der zu erstellenden Systeme in die Konfigurierung einzubeziehen. Dies gilt insbesondere für die Konfigurierung Software-basierter Systeme aus vorgefertigten Softwarebausteinen (vgl. [11]).

Der Begriff *Konfigurierung* meint in diesem Zusammenhang die schrittweise Zusammensetzung und Ausprägung (Parametrierung) von Komponenten zu einem Gesamtsystem (Konfiguration) unter Einhaltung vorgegebener Restriktionen und vorgegebener Ziele (vgl. [3], [6], [15], [18]). 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.

Einige Konfigurierungsansätze verfolgen eine Modellierung von Funktionen auf begrifflicher Ebene. Hierzu zählt insbesondere die Feature-basierte Software-Konfigurierung (vgl. [9], [10], [16]). Eine explizite Modellierung von zeitlichem Verhalten wird von diesen Ansätzen bislang nicht unterstützt. Die Abbildung von Funktion und Verhalten auf Feature-Begriffe kann in vielen Fällen ausreichend sein, wenn das Verhalten dem Benutzer eindeutig bekannt ist. Besteht in einer Domäne jedoch eine besondere Anwendersicht auf das Verhalten und hat man es insbesondere mit vielfältigem Verhalten der betrachteten Objekte zu tun, so sind Features nicht mehr ausreichend und eine tiefere Modellierung der zeitlichen Zusammenhänge ist notwendig. Für ressourcenbasierte Ansätze gilt dasselbe, zumal Ressourcen i.a. noch weniger Möglichkeiten zur Strukturierung bieten als Features.

1.2 Anwendungsfeld

Die vorliegenden Betrachtungen sind primär auf das Gebiet der Konfigurierung eingebetteter Systeme gerichtet. Prinzipiell kann auch in anderen Bereichen Bedarf nach einer verhaltensbasierten Konfigurierung bestehen, wie z.B. bei der Konfigurierung mechanischer, hydraulischer oder pneumatischer Systeme.

Die Konfigurierung eingebetteter Systeme bedeutet nicht Entwurf und Design, Entwicklung oder Programmierung dieser Systeme. Vielmehr geht es darum, ein Produkt, für das eine Vielzahl an Auslegungsvarianten existiert, für einen Kunden oder Anwender individuell auszulegen. Die hohe Zahl an Auslegungsvarianten kommt durch eine hohe Zahl zur Verfügung stehender Komponenten zustande, die alternativ und in unterschiedlichen Kombinationen einsetzbar sind und unterschiedlich ausgeprägt sein können. Die Konfiguration setzt also erst nach abgeschlossener Produkt- und Komponentenentwicklung ein. Es geht somit nicht um eine Entwicklungsunterstützung; hierzu sind andere Technologien aus dem Bereich Hardware-/ Software-Co-Design heranzuziehen, wie z.B. Ansätze zur Systemmodellierung, Code-Generierung und Programmierung, Testverfahren, Validierung und Verifikation mit Simulation (z.B. Harware-in-the-loop-, Software-in-the-loop-Simulation).

Derzeit sind in vielen unterschiedlichen Anwendungsfeldern von eingebetteten Systemen folgende Trends erkennbar:

- *Komplexer werdende Produkte:* Sowohl Hardware- als auch Software-Komponenten werden mit steigendem Leistungsumfang immer komplexer.
- *Steigender Anteil an Software:* Während für die Hardware meist Standard-Komponenten produkt- und herstellerübergreifend zum Einsatz kommen, sind die wesentlichen Teile der Produkt-Software für das Produkt spezifisch entwickelt [19].
- *Rekonfigurierbarkeit von Produkten:* Modifikation der Produktfunktionalität durch den Kunden bzw. Benutzer selbst oder z.B. durch einen Servicetechniker. Austausch oder Hinzunahme von Hardware-Modulen (z.B. Speichererweiterung) oder Software-Modulen (z.B. Software-Update mittels Internet-Download).

Diese Trends entsprechen dem Wunsch von Kunden nach immer mehr Funktionalität, nach immer größerer Produktindividualität und persönlicher Produktkonfiguration sowie nach immer größerer Produktanpassung durch Funktionen-Updates.

Ein Beispielszenario ist die Konfigurierung von Komfort-Funktionen in zukünftigen Kraftfahrzeugen: Auf der Basis von individuellen Kundenwünschen soll eine Zusammenstellung und Ausprägung von Software-Modulen (sowohl High-Level-Modulen als auch Basis-Modulen und Hardware-Treibern) gefunden werden, die diese Anforderungen erfüllen.

1.3 Ziele und Anforderungen

Der hier beschriebene Ansatz verfolgt das Ziel, den Vorgang der wissensbasierten Konfigurierung von Systemen mit variantenreicher Funktionalität – hier schwerpunktmäßig eingebetteten Systemen – durch eine tiefere, funktionale Modellierung wirksam zu unterstützen. Eine solche Methodik soll es ermöglichen, Konfigurationen zu finden, die ein gewünschtes und zulässiges Verhalten aufweisen.

Hierzu gelten die folgenden Anforderungen:

- *Funktionale Modellierung* mit einer expliziten Repräsentation von dynamischem Verhalten und zeitlichen Zusammenhängen (im Folgenden kurz als Verhaltenswissen bezeichnet), sowohl für Komponenten als auch für das Gesamtsystem (Konfiguration).
- *Zustandsmodellierung auf Basis von Statecharts* bzw. an Statecharts angelehnter Formalismen zur Erfüllung spezifischer Anforderungen eingebetteter Systeme, wie z.B. die Modellierung von reaktivem Verhalten, parallelen Prozessen, Ereignissen, Interaktionen und zeitlichen Abhängigkeiten.
- *Funktionale Anforderungen:* (Unvollständige) Beschreibung des gewünschten bzw. geforderten Verhaltens eines zu konfigurierenden Systems.
- *Verhaltensbasierter Problemlösemechanismus:* Basierend auf dem zur Verfügung stehenden Verhaltenswissen soll ein erweiterter Problemlösemechanismus Konfigurationsentscheidungen treffen können und damit zu einem höheren Grad an Automatisierung des Konfigurierungsprozesses beitragen.
- *Umgang mit unvollständig bekanntem Verhaltenswissen:* Ein verhaltensbasiertes Problemlösen als integraler Bestandteil des Konfigurierungsprozesses erfordert Inferenzen auch auf Basis von unvollständig bekanntem Verhaltenswissen.
- *Beitrag zur Validierung und Verifikation von Konfigurationen* auf Grundlage des modellierten Wissens über zeitlich dynamisches Verhalten in der Domäne.

Insgesamt wird hier nicht das Ziel verfolgt, eine Konfigurationsmethodik zu finden, die alleinig auf Verhaltenswissen basiert, sondern beabsichtigt ist die Erweiterung eines bestehenden strukturbasierten Konfigurationsansatzes um die zusätzliche Modellierung und Auswertung von Wissen über Verhalten. Dieses Wissen über Verhalten kann quasi als eine zusätzliche Dimension für modellierbares Wissen im Domänenmodell betrachtet werden. Das grundlegende Ziel besteht immer noch darin, eine Konfiguration und damit eine Lösungs-*Struktur* zu finden.

2 Bestehende verhaltensbasierte Konfigurierungsansätze und ihre Grenzen

Für die wissensbasierte Konfigurierung besteht bereits eine Reihe von Ansätzen, die zeitliche Zusammenhänge in der Domäne direkt in das Wissensmodell einbeziehen. Hierbei handelt es sich insbesondere um Simulationsmodelle, die meist zur Evaluierung einer Lösung oder Teillösung dienen (s. z.B. [1], [5], [14], [17]). Den simulationsbasierten Ansätzen ist gemeinsam, dass

- sie prinzipiell analytisch sind (d.h. eher bewertend, als dass sie konstruktiv zur Einschränkung des Lösungsraums beitragen),
- das (strukturelle) Konfigurationswissen weitestgehend unabhängig vom Simulationsmodell beschrieben wird (und damit wenige Wechselwirkungen aufeinander betrachtet werden) sowie
- die Verhaltensmodelle i.a. einen relativ hohen Detaillierungsgrad aufweisen.

Zwar wird der modulare Aufbau von Simulationsmodellen und die Generierung von Modellvarianten gemäß einer Konfiguration unterstützt, jedoch kann eine Simulation erst dann stattfinden, wenn die zugrunde liegende Modellvariante (und damit die jeweilige Konfiguration oder der betroffene Teil der Konfiguration) vollständig ist. Das vorliegende Verhaltenswissen kann somit kaum für frühe Konfigurationsentscheidungen genutzt werden.

3 Konfigurieren mit Wissen über Zustandsverhalten

Im Folgenden wird die Methodik ABACUS (*A Behavior-Based Configuration Approach using Statecharts*, s. auch [12], [13]) zur verhaltensbasierten Konfigurierung vorgestellt. Die ABACUS-Methodik ist eine Gesamtmethodik, die strukturbasiertes Konfigurieren mit Mechanismen zur Durchführung von Inferenzen auf Verhaltenswissen kombiniert. Dabei basiert die Methodik auf dem begriffshierarchieorientierten Konfigurierungsansatz (vgl. [2], [3], [4]), d.h. auch in der ABACUS-Methodik spielt eine Modellierung des Domänenwissens in Form von Domänenobjekten in taxonomischen und kompositionellen Hierarchien, ebenso wie Constraints und eine flexible, begriffshierarchieorientierte Kontrolle, eine zentrale Rolle.

3.1 Verhaltensmodelle

Für die Modellierung und Verwaltung des Wissens über dynamisches Zustandsverhalten von Objekten und (Teil-)Systemen finden unterschiedliche Modelle Einsatz: Auf Statecharts basierende Modelle zur Beschreibung von Komponentenverhalten und Restriktionen an das Verhalten sowie Verhaltensdeskriptoren zur Repräsentation der Verhaltensmodelle in der Begriffshierarchie und zur Beschreibung des Lösungsverhaltens (s. Abbildung 1).

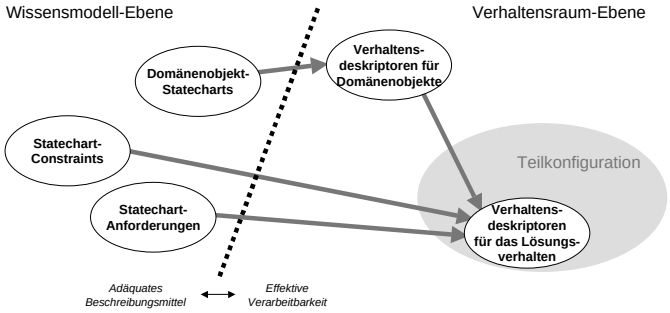


Abb. 1. Unterschiedliche Verhaltensmodelle in ABACUS

3.2 Wissensrepräsentation

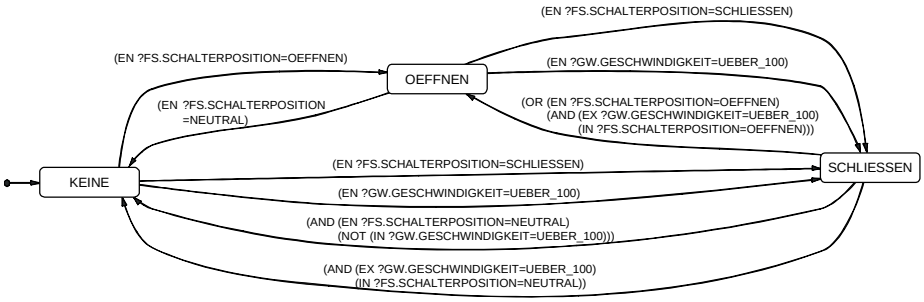
Es werden – analog zu den verschiedenen Arten von Wissen für das strukturbasierte Konfigurieren – vier Arten von modellierbarem Verhaltenswissen unterschieden:

- Wissen über das Zustandsverhalten von Domänenobjekten (Domänenobjekt-Statecharts).
- Wissen über die Zulässigkeit von Zustandsverhalten (Statechart-Constraints).
- Anforderungen an das Zustandsverhalten einer Konfiguration (Statechart-Anforderungen).
- Wissen zur Steuerung der Verhaltensauswertung (Strategien).

Das Grundprinzip der Modellierung von dynamischem Objektverhalten basiert darauf, dass jedem Domänenobjekt ein Modell über sein Zustandsverhalten als eine besondere Eigenschaft zugeordnet werden kann. Damit wird das Prinzip einer komponentenorientierten Verhaltensmodellierung verfolgt, die unabhängig von der genauen Kenntnis des strukturellen Aufbaus vom betrachteten Gesamtsystem (Konfiguration) ist. Die Modellierung des Zustandsverhaltens von Domänenobjekten erfolgt mit speziellen, für die Konfigurierung angepassten Statecharts, im folgenden als ABACUS-Statecharts oder Domänenobjekt-Statecharts bezeichnet.

Mit Statecharts werden generalisierte ereignisbasierte Zustandsübergangsgraphen beschrieben, die Nebenläufigkeit, Verkapselung und Synchronisation unterstützen (vgl. [7], [8]). Anders als bei Systementwicklung und -implementation dient die Verwendung von Statecharts bei der Konfigurierung der *abstrakten* Modellierung des Zustandsverhaltens. Entsprechend wird hier auf eine Reihe von Beschreibungsmitteln der Harelschen Statecharts (wie z.B. Hierarchisierung von Zuständen, Historie-Verwaltung und realzeitliche Bedingungen) verzichtet. Dagegen ist der hier verwendete Statechart-Formalismus um spezifische Anforderungen für die Konfigurierung erweitert (Beschreibung von alternativ gültigen Zustandsmodellen für Domänenobjekte in der taxonomischen Hierarchie, Referenzierung des Zustandsverhaltens von Domänenobjekt-Instanzen durch Domain Patterns, mengenwertige Referenzierung von Zustandswerten für eine vor der Konfigurierung unbekannte Menge von

Domänenobjekt-Instanzen, etc.). Abbildung 2 zeigt als Beispiel den ABACUS-Statechart für ein Domänenobjekt *Geschwindigkeitsabhängige Fenstersteuerung im Fahrzeug*.



Domain-Pattern:

Variable	Domänenobjekt	Bedingungen
?TSG	TUER-SG	ZUGEORDNETES-SW-MODUL ?SELF
?GW	GESCHWINDIGKEIT-WERTGEBER	ZUGEORDNETES-SG ?KSG
?FS	FENSTERSCHALTER-TREIBER	ZUGEORDNETES-SG ?TSG
…	…	…

Abb. 2. Beispiel für ein Domänenobjekt-Statechart

Zur Repräsentation von Restriktionen an das Gesamtverhalten einer Konfiguration (also an das Zustandsverhalten der Instanzen einer Teilkonfiguration) werden sog. Verhaltens-Constraints verwendet. Diese orientieren sich an den Statechart-Modellen. Solche Constraint-Relationen können Zusammenhänge in den Verhaltensmodellen der Teil-Konfigurationen fordern oder verbieten. Beispiele für entsprechende Constraint-Relationen sind:

- Relationen, die das Auftreten (also die Erreichbarkeit) eines Zustandswertes oder einer Kombination von Zustandswerten fordern bzw. verbieten.
- Relationen, die die Erreichbarkeit eines Zustandswertes oder einer Kombination von Zustandswerten von jedem Zustand aus fordern (wiederkehrende Erreichbarkeit).
- Relationen, die die Aktivierung bzw. das Verlassen eines Zustandswertes unter einer angegebenen Bedingung fordern bzw. verbieten.
- Relationen, die das Auftreten einer Transition zwischen zwei Zustandswerten unter einer angegebenen Bedingung fordern bzw. verbieten.

Zur Lösung einer konkreten Konfigurierungsaufgabe soll es dem Benutzer möglich sein, neben strukturellen Anforderungen ebenso Anforderungen an das Lösungsverhalten der zu erzielenden Konfigurationslösung vorgeben zu können, in Form von einschränkenden Vorgaben an das Verhalten. Hierfür stehen die gleichen Beschreibungsmittel wie für Verhaltens-Constraints zur Verfügung.

Da die Verwaltung der Verhaltensmodelle während der Konfigurierung und die Auswertung des Verhaltens für den Konfigurierungsprozess sehr aufwendig sein kann, ist es notwendig, diese Verfahren gezielt zu steuern. Ebenso wie auf die Verarbeitung von strukturellem Wissen kann auch auf die Verarbeitung von Verhaltens-

wissen durch Kontrollwissen Einfluss genommen werden, das in Strategien definiert wird. Entsprechende Vorgaben sind die Berechnung und Auswertung von Verhaltensmodellen gezielt in vordefinierten Phasen, Abbruchkriterien für die Berechnung von Verhaltensmodellen, die Beschränkung des Bearbeitungsverfahrens Verhaltensauswertung (s. Abschnitt 3.4) auf bestimmte Konfigurationsschritttypen, Vorgaben für die Propagierung von Verhaltens-Constraints (s. Abschnitt 3.5), etc.

3.3 Verhaltensdeskriptoren

Für die Verarbeitung des Verhaltenswissen bei der Konfigurierung werden sog. Verhaltensdeskriptoren eingesetzt. Im Gegensatz zu den Domänenobjekt-Statecharts repräsentieren Verhaltensdeskriptoren alternative Verhaltensmodelle (aufgrund alternativer Strukturen in der Begriffshierarchie) geschlossen in einem Modell. Eine definierte Subsumptionsrelation ermöglicht die Vergleichbarkeit und Spezialisierbarkeit und damit eine Behandlung von Verhaltensdeskriptoren in der Begriffshierarchie.

Es wird zwischen Verhaltensdeskriptoren für Domänenobjekte und Verhaltensdeskriptoren der aktuellen Lösungskonfiguration (Teilkonfiguration) unterschieden. Verhaltensdeskriptoren für Domänenobjekte werden aus den Domänenobjekt-Statecharts und der taxonomischen Struktur abgeleitet. Mit der Instantiierung von Domänenobjekten während des Konfigurierungsprozesses findet gleichzeitig eine Instantiierung der Verhaltensdeskriptoren für das Lösungsverhalten statt. Diese Repräsentation des Verhaltens eines teilweise konfigurierten Systems kann im Laufe der Konfigurierung durch Eingrenzen von Alternativen im Verhalten konkretisiert werden (analog zur Einschränkung von Wertebereichen für Parameter oder Relationen). Die Verhaltensdeskriptoren liefern die Ausgangsbasis für die Durchführung von Operationen für Konfigurationsentscheidungen (verhaltensbasiertes Bearbeitungsverfahren) und die Anwendung von Verhaltens-Constraints, wie im Folgenden dargestellt.

3.4 Verhaltensauswertung als Bearbeitungsverfahren

Eine Auswertung des Verhaltenswissens für die Konfigurierung erfolgt durch ein spezielles Bearbeitungsverfahren, das Konfigurierungsschritte durch Schlussfolgerungen vom Verhalten auf die Struktur ermöglicht. Die Aktivierung des Bearbeitungsverfahrens wird durch die Konfigurationskontrolle vollzogen, die Zulässigkeit einer Verhaltensauswertung als Bearbeitungsverfahren sowie weitere Feineinstellungen des Bearbeitungsverfahrens können in Strategien definiert werden.

Das Bearbeitungsverfahren Verhaltensauswertung ist definiert für die Konfigurationsschritttypen Spezialisieren, Zerlegen, Integrieren und Parametrieren. Ausgangsbasis für die Durchführung von Konfigurationsschritten ist ein Look-ahead-Vorgehen: Spezialisieren bedeutet den Ausschluss von Unterkonzepten, deren Verhalten inkonsistent zum aktuellen Lösungsverhalten ist. Zerlegen findet durch Modifikation der Unter- bzw. Oberschranke der Kardinalität einer Relation statt, indem Kardinalitäten ausgeschlossen werden, die zu inkonsistentem Lösungsverhalten führen würden. Integrieren führt zum Ausschluss von potentiellen Aggregat-Instanzen,

für die eine Integration ein inkonsistentes Lösungsverhalten bedeutet. Und Parametrieren schließt Wertebereiche aus, die in Transitionsbedingungen von Zustandsübergängen zu inkonsistentem Verhalten führen würden.

3.5 Auswertung von Constraints und Anforderungen

Prinzipiell bestehen viele Gemeinsamkeiten zwischen Verhaltens-Constraints einerseits und Struktur- und Parameter-Constraints andererseits. Dennoch ist es sinnvoll, die Mechanismen zur Auswertung beider Arten von Constraints voneinander zu trennen, um eine gezielte Steuerung der möglicherweise sehr aufwendigen Auswertung beider Arten von Constraints vornehmen zu können.

Die Anwendung von Verhaltens-Constraints auf die aktuelle Teilkonfiguration basiert auf Domain-Patterns, über die Abhängigkeiten gegenüber der Struktur der aktuellen Teilkonfiguration abgeprüft werden. Die Einschränkung des potenziellen Zustandsverhaltens der aktuellen Teilkonfiguration beutet eine Konkretisierung der Aussage darüber, welche Zustände und Transitionen als Bestandteil des Verhaltensmodells auftreten. Folglich stellt die Verhaltens-Constraint-Propagierung das Komplement zum Bearbeitungsverfahren Verhaltensauswertung dar, das Schlussfolgerungen vom aktuellen Lösungsverhalten auf Änderungen an der Struktur ausführt (s.o.). Die Behandlung von Anforderungen an das Verhalten erfolgt analog zu den Verhaltens-Constraints.

Die Einbettung der Auswertung von Constraints und Anforderungen an das Verhalten in den Gesamtablauf wird im Folgenden beschrieben.

3.6 Konfigurierungsvorgang

Während des Konfigurierungsprozesses wird das aktuelle Lösungsverhalten als Bestandteil einer Teilkonfiguration verwaltet und ist parallel zum Konfigurierungsprozess weiterzuentwickeln, d.h. an die aktuelle Lösungsstruktur anzupassen. Diese Anpassung bedeutet die Aufnahme neuer Verhaltensdeskriptoren zu Objektinstanzen in das Lösungsverhalten und die Bindung von Domain-Pattern-Variablen der Verhaltensdeskriptoren an Instanzen der aktuellen Teilkonfiguration.

Die Anwendung des Bearbeitungsverfahrens Verhaltensauswertung erfolgt mit Durchführung eines Konfigurierungsschrittes und kann somit relativ leicht in den bestehenden Kontrollfluss der strukturbasierten Konfigurierung integriert werden. Die Ableitung des aktuellen Lösungsverhaltens aus der Struktur einer Teilkonfiguration und die Propagierung von Verhaltens-Constraints sind – ebenso wie die Propagierung von Struktur- und Parameter-Constraints – den einzelnen Konfigurierungsschritten nachgelagert.

Während die Struktur- und Parameter-Constraint-Propagierung unabhängig vom aktuellen Lösungsverhalten ist, eine Weiterentwicklung des Lösungsverhaltens aber potenziell nach jeder Änderung in der Struktur der jeweiligen Teilkonfiguration notwendig ist, ist die geeignete Reihenfolge für die Einzelschritte zur Nachbearbeitung eines Konfigurierungsschritts wie in Abbildung 3 dargestellt. Ggf. sind die Schritte zwei und drei solange zu iterieren, bis beide Verfahren zu keiner Ein-

schränkung des aktuellen Lösungsverhaltens mehr beitragen oder eine Abbruchbedingung erfüllt ist. Beide ergänzten Schritte sind optional, ihre Anwendung ist durch die zur Zeit aktive Strategie bestimmt. Bei der Weiterentwicklung des Lösungsverhaltens und bei der Verhaltens-Constraint-Propagierung können bei Erkennen von Inkonsistenzen im aktuellen Lösungsverhalten Konflikte identifiziert werden.

Übersteigt der Aufwand für die Auswertung des Verhaltenswissens die in der aktuellen Strategie zulässigen Vorgaben, so ist jederzeit ein Abbruch des erweiterten Ablaufs möglich. Dies betrifft sowohl die Anwendung des Bearbeitungsverfahrens Verhaltensauswertung, als auch Aufbau und Weiterentwicklung des aktuellen Teilkonfigurations-Verhaltensdeskriptors und die Durchführung der Verhaltens-Constraint-Propagierung.

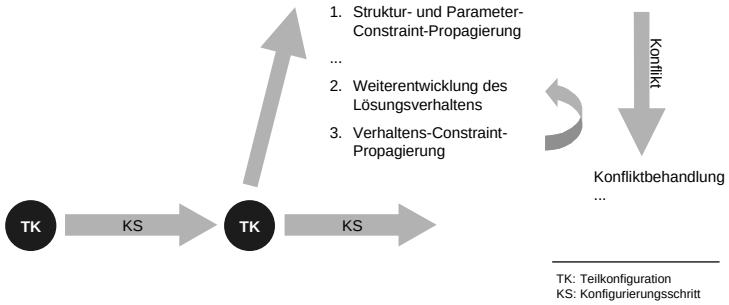


Abb. 3. Konfigurierungsprozess mit Verhaltensauswertung

4 Bewertung und Ausblick

Insgesamt bietet die Möglichkeit zur Spezifikation von Wissen über die Struktur, über das Verhalten und insbesondere über Wechselwirkungen zwischen beidem die Grundlage für eine tiefere Modellierung. Die Modellierung basiert auf Statecharts, die aufgrund ihrer klaren und realitätsnahen Ausdrucksform im Ingenieurumfeld vielfach etabliert sind.

Der Vorteil des hier vorgestellten Verfahrens besteht in der Möglichkeit, aus dem in der Domäne vorliegenden Wissen über das dynamische Verhalten der Objekte Konfigurationsentscheidungen abzuleiten und damit den Konfigurierungsprozess konstruktiv zu unterstützen.

Aufgrund eines hohen Aufwands für Verwaltung und Auswertung der Zustandsmodelle (der exponentiell zur Anzahl der voneinander abhängigen Zustandsgrößen ist) spielt eine gezielte Steuerung der Verhaltensauswertung unter Vorgabe entsprechenden Kontrollwissens eine große Rolle.

Für die wissensbasierte Konfigurierung eingebetteter Systeme kann die ABACUS-Methodik einen wesentlichen Beitrag liefern, sofern Varianten von Funktionen in der Anwendungsdomäne eine herausragende Rolle spielen und eine Modellierung von Verhaltensalternativen auf einer ausreichend abstrakten Ebene möglich ist.

Referenzen

1. Brinkop, A.: Variantenkonstruktion durch Auswertung der Abhängigkeiten zwischen den Konstruktionsbauteilen. Infix, St. Augustin (1999)
2. Cunis, R., Günter, A., Strecker, H. (Hrsg.): Das PLAKON Buch – Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen. Springer, Berlin u.a. (1991)
3. Günter, A.: Flexible Kontrolle in Expertensystemen für Planungs- und Konfigurierungsaufgaben in technischen Domänen. Infix, St. Augustin (1991)
4. Günter, A. (Hrsg.): Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON. Infix, St. Augustin (1995)
5. 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)
6. 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)
7. Harel, D.: Statecharts: a Visual Formalism for Complex Systems. Science of Computer Programming, **8** (1987) 231-274
8. Harel, D., Politi, M.: Modeling Reactive Systems with Statecharts – The Statemate Approach. McGraw-Hill, New York u.a. (1998)
9. Hein, A., MacGregor, J., Thiel, S.: Configuring Software Product Line Features. Pulvermüller, E., Speck, A., Coplien, J. O., D'Hondt, M., DeMeuter, W. (Hrsg.). Proceedings Workshop Feature Interaction in Composed System, 15th European Conference on Object-Oriented Programming (ECOOP), Budapest, Ungarn (2001)
10. Krebs, T., Hotz, L., Günter, A.: Knowledge-based Configuration for Configuring Combined Hardware/Software Systems. Sauer, J. (Hrsg.). Proceedings 16. Workshop "Planen, Scheduling und Konfigurieren, Entwerfen", Freiburg (2002)
11. Kühn, C.: Requirements for Configuring Complex Software-Based Systems. Friedrich, G. (Hrsg.). Configuration – Papers from the AAAI Workshop, Orlando, Florida, AAAI Press, California (1999)
12. 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)
13. Kühn, C.: Vergleich unterschiedlicher Konfigurationsmethoden im Hinblick auf die Nutzbarkeit von Wissen über das Zustandsverhalten der Konfigurationsobjekte. Sauer, J. (Hrsg.). Proceedings Workshop AI in Planning, Scheduling, Configuration and Design (PuK), Joint German/Austrian Conference on AI (KI-2001), Wien (2001)
14. 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, Univ.-GH Paderborn (1998)
15. Sabin, D., Weigel, R.: Product Configuration Frameworks – A Survey. IEEE Intelligent Systems Jul/Aug 1998 (1998) 50-58
16. Schlick, M., Hein, A.: Knowledge Engineering in Software Product Lines. Stumptner, M., Tatar, M., Zdrahal, Z. (Hrsg.). Proceedings Workshop on Knowledge-Based Systems for Model-Based Engineering, 14th European Conference on Artificial Intelligence (ECAI), Berlin (2000)
17. Stein, B. M.: Functional Models in Configuration Systems, Univ.-GH Paderborn (1995)
18. Stumptner, M.: An Overview of Knowledge-Based Configuration. AI Communications, **10** (2) (1997) 111-126
19. Vierhaus, H. T.: Eingebettete Systeme. Spektrum der Wissenschaft – Dossier Software (1999) 68-71