Konzepte zur Absicherung der Variantenkonfiguration von eingebetteter Fahrzeugsoftware*

Hannes Holdschick †
Daimler AG
hannes.holdschick@daimler.com

Abstract: Eine breite Fahrzeugpalette, spezifische Produktanforderungen aus globalen Märkten und eine Vielzahl softwarebasierter Ausstattungsvarianten führen in der Automobilindustrie zu einer immer größeren Produktvielfalt. Um die entstehende Komplexität beherrschen zu können, wird die Methode des merkmalbasierten Variantenmanagements eingesetzt.

Der bei der Daimler AG verfolgte Ansatz sieht vor, dass variable Komponenten in Entwicklungsartefakten, wie beispielsweise einem Matlab / Simulink-Modell, mithilfe von aussagenlogischen Formeln über Merkmalen konfiguriert werden. Fehler innerhalb dieser Konfigurationsformeln bzw. des Merkmalmodells können bei der Konfiguration zu inkonsistenten Systemvarianten führen. Wir präsentieren daher Konsistenzbedingungen, um die Konfiguration abzusichern. Diese basieren auf SAT-Analysen des Merkmalmodells und der Konfigurationsformeln. Erste Implementierungen zeigen, dass die Performanz solcher Analysen auf industriellen Variantenmodellen praxistauglich ist.

1 Einleitung

In der Automobilindustrie wird das Konzept der Software-Produktlinien [CN01] eingesetzt, um der steigenden Variantenkomplexität in softwarebasierten Systemen zu begegnen. Erreicht dabei die Variabilität in Entwicklungsartefakten, wie z.B. einem Simulink-Modell oder einem Anforderungsdokument, eine hinreichende Komplexität, so wird sie in einem merkmalbasierten Variantenmodell dokumentiert. Darin werden die gemeinsamen und unterschiedlichen funktionalen Eigenschaften der System-Varianten in einem Merkmalmodell [KCH+90] festgehalten. Um auch die Konfiguration des Artefaktes zu ermöglichen, muss zusätzlich das Mapping zwischen Merkmalen und den variablen Bestandteilen des Artefaktes erstellt werden, welches im sogenannten Konfigurationsmodell abgebildet wird.

Ist ein Variantenmodell im Einsatz, entwickelt es sich stetig weiter. Dadurch entsteht die Herausforderung, dessen Konsistenz im Evolutionsprozess zu gewährleisten, welche auch schon in anderen Arbeiten thematisiert wurde [LSB+, Hol12]. Als einen Lösungsansatz

†Dieser Beitrag wurde durch das BMBF im Projekt SPES 2020_XT (Förderkennz: 01IS12005) gefördert.

dafür präsentieren wir Analysen des Variantenmodells, um nach einer Änderung prüfen zu können, ob Inkonsistenzen bzw. Fehlmodellierungen aufgetreten sind. Durch die aussagenlogische Fundierung des Merkmalmodells können diese Konsistenzbedingungen auf die Frage der Erfüllbarkeit einer aussagenlogischen Formel (SAT) zurückgeführt werden.

2 Beschreibung des Variantenmodells

In diesem Abschnitt wird näher auf die beiden Teilmodelle des Variantenmodells und den Konfigurationsprozess eingegangen.

2.1 Das Merkmalmodell

Im Merkmalmodell sind die funktionalen Eigenschaften des Systems hierarchisch als Baum strukturiert [KCH+90].

Definition: Sei zunächst $\mathcal{M} = (M, H \cup C, r)$ ein 3-Tupel bestehend aus:

- einer endlichen Menge M,
- der Menge H der hierarchischen Beziehungen:

  $$H := opt \ \dot\cup \ obl \ \dot\cup \ alt \ \dot\cup \ dis \ \ \subset M \times M,$$

- der Menge C der Cross-Tree-Constraints:

  $$C := req \ \dot\cup \ con \ \ \subset M \times M$$

- und einem ausgezeichnetem Wurzelelement $r \in M$

Wir nennen $\mathcal{M}$ ein **Merkmalmodell**, falls gilt:

(M1) Für jedes $m \in M \setminus \{r\}$ existiert genau ein $x \in M$ mit $(x, m) \in H$

(M2) Es existiert kein $x \in M$ mit $(x, r) \in H$

(M3) Für jedes $x \in M$ existieren $x_1, \ldots, x_n \in M$ mit $(r, x_1), \ldots, (x_n, x) \in H$

Mit anderen Worten: Interpretiert man M als Knotenmenge und H als Kantenmenge, so ist (M, H) ein Baum. Die Relationen in H beschreiben dabei die Variationstypen der Merkmale. Wir unterscheiden in Anlehnung an bestehende Ansätze [Bat05, CE00] die folgenden Typen: optional (opt), obligatorisch (obl), alternativ (alt) und disjunktiv (dis). Ist ein Paar von Merkmalen in einer der Relationen enthalten, so stehen diese in einer Vater-Kind-Beziehung und das Kindmerkmal hat den entsprechenden Variationstyp. Zum Beispiel bedeutet $(x, y) \in opt$, dass x das Vatermerkmal des optionalen Merkmals y ist. Dadurch definiert H eine Hierarchie auf der Menge M der Merkmale.

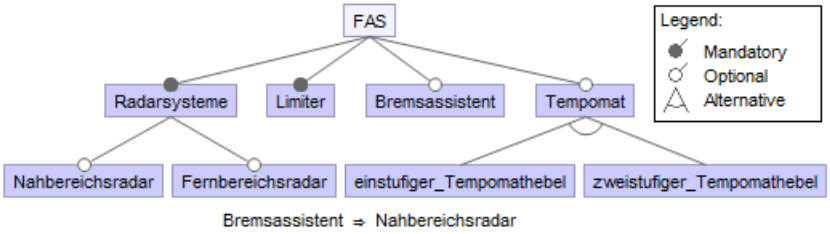

Abbildung 1: Beispiel für ein Merkmalmodell

Mithilfe der Cross-Tree-Constraints lassen sich weitere Beziehungen zwischen Merkmalen definieren. Wir beschränken uns hier auf die beiden Typen *requires* (req) und *conflicts* (con), da sie in der Praxis am häufigsten verwendet werden. In Abb. 1 sieht man ein stark vereinfachtes Beispiel für ein Merkmalmodell der Fahrerassistenzsysteme (FAS). Entsprechend der obigen Definition hat dieses Modell die folgende Struktur:

- opt = {(FAS, Tempomat), (FAS, Bremsassistent) (Radarsysteme, Nahbereichsradar), (Radarsysteme, Fernbereichsradar)}
- obl = {(FAS, Limiter), (FAS, Radarsysteme)}
- alt = {(Tempomat, einstufiger_Tempomathebel), (Tempomat, zweistufiger_Tempomathebel)}
- req = {(Bremsassistent, Nahbereichsradar)} bzw. $dis = con = \emptyset$

Ausgehend von der obigen Beschreibung des Merkmalmodells lässt sich nun dessen aussagenlogische Repräsentation definieren. Mit $F(M)$ bezeichnen wir dabei hier und auch im Weiteren die Menge aller aussagenlogischen Formeln über der Merkmalmenge M.

Definition: Sei $\mathcal{M} = (M, H \cup C, r)$ ein Merkmalmodell mit $H = \{opt, obl, alt, dis\}$ und $C = \{req, con\}$. Dann bezeichnen wir mit $\phi(\mathcal{M}) \in F(M)$ die **Formel des Merkmalmodells** $\mathcal{M}$. Sie ist definiert durch:

$$\phi(\mathcal{M}) \stackrel{\text{def}}{=} r \tag{1}$$

$$\wedge \bigwedge_{(x,y)\in opt} (y \rightarrow x) \tag{2}$$

$$\wedge \bigwedge_{(x,y)\in obl} (x \leftrightarrow y) \tag{3}$$

$$\wedge \bigwedge_{\substack{x\in M \\ (x,y_1),\ldots,(x,y_k)\in alt}} ((x \leftrightarrow \bigvee_{i=1}^{k} y_i) \wedge \bigwedge_{i<j} \neg(y_i \wedge y_j)) \tag{4}$$

$$\wedge \bigwedge_{\substack{x \in M \\ (x,y_1),\ldots,(x,y_k) \in dis}} (x \leftrightarrow (y_1 \vee \ldots \vee y_k)) \tag{5}$$

$$\wedge \bigwedge_{(x,y) \in req} (x \rightarrow y) \tag{6}$$

$$\wedge \bigwedge_{(x,y) \in con} (\neg x \vee \neg y) \tag{7}$$

Die Idee, das Merkmalmodell mithilfe der Aussagenlogik zu formalisieren, wurde bereits in verschiedenen Arbeiten beschrieben [Bat05, CW07]. Die Struktur der hier dargestellten Formel ist angelehnt an die Darstellung in [SLB+11]. Nach dieser allgemeinen Definition werfen wir nun einen Blick auf die Formel unseres Beispiel-Merkmalmodells. Sei $\mathcal{M}$ das Merkmalmodell aus Abb. 1. Dann ist

$$\begin{aligned}
\phi(\mathcal{M}) = \quad & FAS && (8)\\
& \wedge (Tempomat \rightarrow FAS) && (9)\\
& \wedge (Bremsassistent \rightarrow FAS) && (10)\\
& \wedge (Nahbereichsradar \rightarrow Radarsysteme) && (11)\\
& \wedge (Fernbereichsradar \rightarrow Radarsysteme) && (12)\\
& \wedge (FAS \leftrightarrow Limiter) && (13)\\
& \wedge (FAS \leftrightarrow Radarsysteme) && (14)\\
& \wedge (Tempomat \leftrightarrow (einstufiger_TH \vee zweistufiger_TH))^{1} && (15)\\
& \wedge \neg (einstufiger_TH \wedge zweistufiger_TH) && (16)\\
& \wedge (Bremsassistent \rightarrow Nahbereichsradar) && (17)
\end{aligned}$$

Wir können nun das Merkmalmodell mithilfe einer aussagenlogischen Formel beschreiben. Ist diese Formel $\phi(\mathcal{M})$ erfüllbar, so existiert eine gültige Variante für das Merkmalmodell $\mathcal{M}$. Genauer gesagt ist jede Belegung, die $\phi(\mathcal{M})$ erfüllt, eine gültige Variante des Merkmalmodells. Dies führt uns zum Begriff der *Selektion*.

Definition: Sei $\mathcal{M} = (M, H \cup C, r)$ ein Merkmalmodell mit $M = \{m_1, \ldots, m_n\}$. Dann nennen wir $S \in F(M)$ eine **Selektion** von $\mathcal{M}$, falls gilt:

$$S = s_1 \wedge \ldots \wedge s_n \text{ mit } s_i = m_i \text{ oder } s_i = \neg m_i \text{ für } 1 \leq i \leq n.$$

Eine Selektion $S \in F(M)$ heißt **gültig**, falls die Formel $S \wedge \phi(\mathcal{M}) \in F(M)$ erfüllbar ist.

[1] Für eine übersichtlichere Darstellung der Formel haben wir in diesem Beispiel die Namen der Merkmale „*einstufiger_Tempomathebel*“ und „*zweistufiger_Tempomathebel*“ durch „einstufiger_TH“ bzw. „zweistufiger_TH“ ersetzt.

Variation Point	Assignment
VAR_Tempomat	
0 (Leermodul)	NOT(Tempomat)
1 (TMPH einstufig)	einstufiger_Tempomathebel
2 (TMPH zweistufig)	zweistufiger_Tempomathebel

Abbildung 2: Ausschnitt aus einem Konfigurationsmodell

2.2 Das Konfigurationsmodell

Im Konfigurationsmodell ist der Zusammenhang zwischen Merkmalen und den variablen Komponenten im Entwicklungsartefakt dokumentiert. Dies kann zum Beispiel ein optionales Subsystem in Simulink oder eine variable Anforderung in einem Lastenheft sein. Dabei sind für uns vor allem die aussagenlogischen Konfigurationsformeln relevant, die formal beschreiben, welche Merkmalselektion zu welcher Systemvariante führt.

Definition: Sei $\mathcal{M}$ ein Merkmalmodell. Ein zugehöriges *Konfigurationsmodell* ist eine endliche Menge von Variationspunkten $KM = \{VP_1, \ldots, VP_N\}$. Dabei besteht jeder *Variationspunkt* $VP = ((V_1, KF_1), \ldots, (V_p, KF_p))$ aus einer endlichen Menge von *Variationen* $V_1, \ldots, V_p$. Jeder Variation V_i ist eine *Konfigurationsformel* $KF_i \in F(M)$ zugeordnet. Das Tupel $VM = (\mathcal{M}, KM)$ aus Merkmalmodell und Konfigurationsmodell nennen wir ein *Variantenmodell*.

In Abb. 2 sieht man einen beispielhaften Ausschnitt aus einem Konfigurationsmodell zu dem Merkmalmodell in Abb. 1. Es enthält den Variationspunkt VP_1 =*VAR_Tempomat* mit den Variationen
$V_1 = (Leermodul, \neg\,(Tempomat))$,
$V_2 = (TMPH\ einstufig, einstufiger\_Tempomathebel)$ und
$V_3 = (TMPH\ zweistufig, zweistufiger\_Tempomathebel)$.

VP_1 hat damit vergleichsweise einfache Konfigurationsformeln, da sie jeweils nur aus einem Literal bestehen. Grundsätzlich sind jedoch Konfigurationsformeln beliebiger Größe möglich und auch realistisch, wodurch die Komplexität entsteht, die eine manuelle Analyse deutlich erschwert bzw. unmöglich macht. VP_1 könnte beispielsweise die Modellierung eines Subsystems in Simulink sein, welches, abhängig von einem Parameter, entweder die eingehenden Signale ohne Verarbeitung weitergibt (V_1) oder einen Tempomat mit ein- bzw. zweistufigem Tempomathebel implementiert (V_2 bzw. V_3).

2.3 Der Konfigurationsprozess

Wie dieser Parameter ermittelt wird, beschreibt der Konfigurationsprozess. Soll dabei eine neue Systemvariante konfiguriert werden, wählt man dafür zunächst die gewünschten Merkmale aus und definiert damit eine Selektion. Abhängig von dieser Selektion wird

für jeden Variationspunkt einzeln entschieden, welche seiner Variationen in das System integriert wird.

Definition: Sei dazu $VP = ((V_1, KF_1), \ldots, (V_p, KF_p)) \in KM$ ein Variationspunkt und $S \in F(M)$ eine Selektion. Ist dann $S \wedge KF_i$ erfüllbar für ein $1 \leq i \leq p$, so nennen wir V_i *die an dem Variationspunkt* VP *für die Selektion* S *gültige Variation*. In diesem Fall wird im weiteren davon die Rede sein, dass die Selektion S die Konfigurationsformel KF_i erfüllt.

Wären für eine Premium-Variante der Fahrerassistenzsysteme z.B. unter anderen die Merkmale *Tempomat* und *zweistufiger_Tempomathebel* ausgewählt, so würde der Konfigurationsprozess am Variationspunkt VP_1 die Variation 2 als *gültig* bestimmen. Auf diese Art und Weise erhält man für jeden Variationspunkt genau eine Variation, welche mithilfe des bereits erwähnten Parameters codiert wird. Diese Liste an Variationspunkt-Parameter-Paaren wird anschließend in das Entwicklungsartefakt exportiert (z.B. per xml-Datei), um die noch offene Variabilität zu binden. In unserem Beispiel würde dabei in das Simulink-Subsystem die Implementierung des Tempomaten mit zweistufigem Tempomathebel eingefügt werden.

Für den Konfigurationsprozess ist es dabei essentiell, dass für eine gegebene Selektion S an jedem Variationspunkt *genau eine* Variation gültig wird. Ist an einem Variationspunkt diese Eindeutigkeit nicht gegeben, so kann dessen Parameter gar nicht oder ggf. nur falsch ermittelt werden. Die Folge davon ist ein inkonsistentes bzw. falsch konfiguriertes Artefakt. Im nächsten Abschnitt werden Konsistenzbedingungen beschrieben, die solche Fehlkonfigurationen verhindern können.

3 Definition der Konsistenzbedingungen

In diesem Abschnitt werden Konsistenzbedingungen für das Variantenmodell vorgestellt. Diese beziehen sich vor allem auf die Konfigurationsformeln, da diese in der Regel händisch eingegeben werden, wodurch es leicht zu Fehlern kommen kann. Außerdem wird auch die Formalisierung des Merkmalmodells einbezogen, womit eine Analyse aller theoretisch möglichen (also gültigen) Selektionen sichergestellt werden kann. Somit können nicht nur die bestehenden, sondern auch zukünftige Varianten abgesichert werden. In den folgenden Unterabschnitten wird jeweils die Konsistenzbedingung zuerst formuliert (*Bedingung*) und danach kurz erklärt (*Beschreibung*). Anschließend wird dargestellt, wie die Bedingung mithilfe der Aussagenlogik geprüft wird (*Prüfung*) und schlussendlich ein Beispielmodell angegeben, welches die Bedingung verletzt (*Gegenbeispiel*).

3.1 KB 1: Kontradiktionen in Konfigurationsformeln

Bedingung: Keine Konfigurationsformel steht im Widerspruch zu der Modellierung im Merkmalmodell.

Variation Point	Assignment
VAR_VP_1	
1 (Variation 1)	NOT(Tempomat)
2 (Variation 2)	einstufiger_Tempomathebel AND zweistufiger_Tempomathebel
VAR_VP_2	
1 (Variation 1)	Limiter
2 (Variation 2)	Tempomat

Abbildung 3: inkonsistentes Konfigurationsmodell

Beschreibung: Jede Konfigurationsformel muss auch unter den Bedingungen des Merkmalmodells erfüllbar sein. Andernfalls wird die betroffene Konfigurationsformel von keiner gültigen Selektion erfüllt. Dadurch ist die zugehörige Variation nie Teil einer Systemvariante und damit überflüssig.

Prüfung: Bei dieser Konsistenzbedingung wird für jede Konfigurationsformel KF im Konfigurationsmodell einzeln geprüft, ob der folgende Ausdruck erfüllbar ist:

$$\phi(\mathcal{M}) \wedge KF \tag{18}$$

Ist diese Formel nicht erfüllbar, gibt es also keine Belegung die $\phi(\mathcal{M})$ *und* KF erfüllt, so existiert keine gültige Selektion, die die Konfigurationsformel KF erfüllt. Sie steht somit im Widerspruch zum Merkmalmodell.

Gegenbeispiel: Betrachten wir das Variantenmodell, welches aus dem Merkmalmodell der Fahrerassistenzsysteme von oben und dem Konfigurationsmodell in Abb. 3 besteht. Hier verletzt der Variationspunkt *VAR_VP_1* die beschriebene Konsistenzbedingung. Die Konfigurationsformel von Variation 2 wird bei einer gültigen Selektion immer zu *false* evaluieren, da die Merkmale *einstufiger_Tempomathebel* und *zweistufiger_Tempomathebel* im Merkmalmodell als Alternativen modelliert sind und somit nicht beide Merkmale in einer Selektion ausgewählt werden dürfen.

3.2 KB 2: Tautologien in Konfigurationsformeln

Bedingung: Keine Konfigurationsformel wird von allen gültigen Selektionen erfüllt.

Beschreibung: Diese Konsistenzbedingung soll verhindern, dass man eine Konfigurationsformeln formuliert, die von jeder gültigen Selektion $S \in F(M)$ erfüllt wird, die also eine Tautologie bzgl. der Merkmalmodellierung ist.

Prüfung: Bei dieser Konsistenzbedingung wird für jede Konfigurationsformel KF im Konfigurationsmodell einzeln geprüft, ob der folgende Ausdruck erfüllbar ist:

$$\phi(\mathcal{M}) \wedge (\neg KF) \tag{19}$$

Ist diese Formel nicht erfüllbar, so existiert für die Konfigurationsformel KF keine gültige

Selektion, die ihre Negation erfüllt. Somit wird diese Formel von jeder gültigen Selektion erfüllt, ist also eine Tautologie im Bezug auf die Merkmalmodellierung.

Gegenbeispiel: Betrachten wir wieder das Variantenmodell, welches aus dem Merkmalmodell der Fahrerassistenzsysteme von oben und dem Konfigurationsmodell in Abb. 3 besteht. Hier verletzt der Variationspunkt *VAR_VP_2* die beschriebene Konsistenzbedingung. Die Konfigurationsformel von Variation 1 wird von jeder gültigen Selektion erfüllt, da das Merkmal *Limiter* obligatorisch ist und daher in jeder gültigen Merkmalauswahl selektiert werden muss. Damit ist diese Konfigurationsformel eine Tautologie bzgl. des Merkmalmodells.

3.3 KB 3: unterbestimmte Variationspunkte

Bedingung: An jedem Variationspunkt muss für jede gültige Selektion mindestens eine Konfigurationsformel erfüllt sein.

Beschreibung: Wie bereits in Abschnitt 2 beschrieben, ist im Konfigurationsprozess wichtig, dass von einer gültigen Selektion genau eine Konfigurationsformel je Variationspunkt erfüllt wird. Diese und die nächste Konsistenzbedingung stellen diese Eindeutigkeit sicher.

Prüfung: Jeder Variationspunkt wird bei dieser Bedingung einzeln geprüft. Die Bedingung ist für den Variationspunkt $VP = ((V_1, KF_1), \ldots, (V_p, KF_p)) \in KM$ verletzt, falls die folgende Formel aus $F(M)$ erfüllbar ist:

$$\phi(\mathcal{M}) \wedge \bigwedge_{i=1}^{p} (\neg KF_i) \tag{20}$$

Denn dann existiert eine Selektion $S \in F(M)$, die $\phi(\mathcal{M})$ *und*

$$\bigwedge_{i=1}^{p} (\neg KF_i) \tag{21}$$

erfüllt. S ist somit gültig und erfüllt $\neg KF_i$ für alle $1 \leq i \leq p$. Diese gültige Selektion S erfüllt damit keine Konfigurationsformel an dem Variationspunkt $VP \in KM$.

Gegenbeispiel: Betrachten wir das Variantenmodell, welches aus dem Merkmalmodell der Fahrerassistenzsysteme von oben und dem Konfigurationsmodell in Abb. 4 besteht. Hier verletzt der Variationspunkt *VAR_VP_3* die beschriebene Konsistenzbedingung. Das Merkmal *Tempomat* ist im Merkmalmodell optional, kann damit an- und abgewählt werden. Beide Konfigurationsformeln werden jedoch nur von Selektionen erfüllt, in denen dieses Merkmal ausgewählt wurde. Dadurch kann für eine Fahrzeugvariante ohne Tempomat für diesen Variationspunkt keine Variation ermittelt werden, wodurch die Konfiguration fehlschlägt.

Variation Point	Assignment
VAR_VP_3	
1 (Variation 1)	Tempomat AND einstufiger_Tempomathebel
2 (Variation 2)	Tempomat AND zweistufiger_Tempomathebel
VAR_VP_4	
1 (Variation 1)	Tempomat
2 (Variation 2)	einstufiger_Tempomathebel

Abbildung 4: inkonsistentes Konfigurationsmodell

3.4 KB 4: überbestimmte Variationspunkte

Bedingung: An jedem Variationspunkt darf für jede gültige Selektion höchstens eine Konfigurationsformel erfüllt sein.

Beschreibung: Ergänzend zur vorherigen Konsistenzbedingung werden hier Variationspunkte gesucht, an denen für eine bestimmte Selektion mehr als eine Konfigurationsformel erfüllt werden. In diesem Fall könnte keine gültige Variation an dem Variationspunkt ermittelt werden, wodurch im Konfigurationsprozess eine fehlerhafte Systemvariante entsteht.

Prüfung: Jeder Variationspunkt wird bei dieser Bedingung einzeln geprüft. Die Konsistenzbedingung ist für den Variationspunkt $VP = ((V_1, KF_1), \ldots, (V_p, KF_p)) \in KM$ verletzt, falls die folgende Formel erfüllbar ist:

$$\phi(\mathcal{M}) \wedge (\bigvee_{i<j} (KF_i \wedge KF_j)) \tag{22}$$

Begründung: Ist der obige Ausdruck erfüllbar, so existiert eine Selektion $S \in F(M)$, sodass die Formel

$$S \wedge \phi(\mathcal{M}) \wedge (\bigvee_{i<j} (KF_i \wedge KF_j)) \tag{23}$$

erfüllbar ist. Damit ist insbesondere auch $S \wedge \phi(\mathcal{M})$ erfüllbar, woraus folgt, dass S eine gültige Selektion ist. Darüber hinaus ist außerdem damit auch

$$S \wedge (\bigvee_{i<j} (KF_i \wedge KF_j)) \tag{24}$$

erfüllbar. Daraus folgt, dass mindestens ein Paar von Indizes $1 \leq i, j \leq p$ existiert, sodass $S \wedge KF_i \wedge KF_j$ erfüllbar ist. Das bedeutet, dass die gültige Selektion S die beiden Konfigurationsformeln KF_i *und* KF_j erfüllt. Die oben genannte Inkonsistenz liegt damit an dem untersuchten Variationspunkt vor, da zwei verschiedene Konfigurationsformeln existieren, die von einer gültigen Selektion erfüllt werden.

Tabelle 1: Laufzeiten des Prototyps

Modell	KB1	KB2	KB3	KB4
Variantenmodell 1	0,143 Sek.	0,143 Sek.	0,133 Sek.	0,145 Sek.
Variantenmodell 2	0,146 Sek.	0,147 Sek.	0,143 Sek.	0,155 Sek.

Gegenbeispiel: Betrachten wir wieder das Variantenmodell, welches aus dem Merkmalmodell der Fahrerassistenzsysteme von oben und dem Konfigurationsmodell in Abb. 4 besteht. Hier verletzt der Variationspunkt *VAR_VP_4* die beschriebene Konsistenzbedingung. Da es durchaus möglich ist, eine gültige Selektion mit den Merkmalen *Tempomat* und *einstufiger Tempomathebel* zu definieren, würden in diesem Fall beide Konfigurationsformeln von dieser Selektion erfüllt werden. Für diesen Variationspunkt könnte damit keine eindeutige Variation ermittelt werden, wodurch die Konfiguration fehlschlägt.

4 Prototypische Implementierung und Evaluation

Um die beschriebenen Konsistenzbedingungen auch an bestehenden Variantenmodellen prüfen zu können, ist eine prototypische Implementierung entstanden. Dadurch kann ein beliebiges Modell analysiert und im Fehlerfall der betroffene Variationspunkt bzw. die betroffene Konfigurationsformel angegeben werden. Die Analysen wurden an zwei Variantenmodellen durchgeführt, welche in der industriellen Praxis entstanden sind. Variantenmodell 1 besteht aus einem Merkmalmodell mit 298 Merkmalen und 367 Cross-Tree-Constraints und einem Konfigurationsmodell mit 164 Variationspunkten und insgesamt 429 Variationen und demnach ebenso vielen Konfigurationsformeln. Die Größe des Modells liegt damit in einem Bereich, der für Domänen wie Motorsteuerung, Fahrerassistenz oder e-Drive üblich ist. Das zweite Variantenmodell befindet sich noch in der Entwicklung und ist daher etwas kleiner: 94 Merkmale, 43 Cross-Tree-Constraints, 14 Variationspunkte mit 36 Variationen.

Tabelle 1 zeigt die Zeit, die der SAT-Solver im Durchschnitt benötigt, um eine Anfrage zu bearbeiten. Da bei KB1 bzw. KB2 jede Konfigurationsformel einzeln analysiert wird, müssen bei unserem ersten Beispielmodell 429 Analysen durchgeführt werden. Die Prüfung des gesamten Modells benötigt daher 429 x 0,143 Sek. $\approx$ 61,3 Sekunden. Bei KB3 und KB4 wird dagegen nur für jeden Variationspunkt eine SAT-Analyse durchgeführt, wodurch sich die Laufzeit verringert. Mit durchschnittlich ca. 0.14 Sekunden sind die Laufzeiten des Solvers für uns zufriedenstellend, zumal auch nach zahlreichen Durchläufen keine Analyse länger als eine halbe Sekunde dauerte. Alle Laufzeiten wurden dabei auf einem Windows 7-PC mit 2,53 GHz und 4 GB RAM gemessen.

5 Verwandte Arbeiten

Auch andere Arbeiten beschäftigen sich mit dem Mapping zwischen Merkmalen und Komponenten. Reiser et al. präsentieren einen Ansatz für das Variantenmanagement hierarchischer komponentenbasierter Systeme [RKW09]. Dabei beschreiben Merkmalmodelle die Variabilität der jeweiligen Hierarchieebene, wobei mit sogenannten *configuration links* Konfigurationsinformationen zwischen den Modellen ausgetauscht werden können. Obwohl sich auch unsere Modellierung auf variable Komponenten bezieht, liegt der Unterschied darin, dass das gesamte System in *einem* Variantenmodell (und damit einem Merkmalmodell) abgebildet wird, auf das sich die vorgestellten Konsistenzbedingungen beziehen. In [MRM+12] wird ein mengentheoretischer Formalismus vorgestellt, um ein Tracing zwischen Merkmalen (Spezifikation) und Komponenten (Implementierung) zu beschreiben. Während in deren Mapping ein Merkmal stets von einer oder mehreren Teilmengen von Komponenten implementiert werden kann, wird bei uns der Zusammenhang mithilfe einer booleschen Formel ausgedrückt. Diese kann beliebig komplex sein, sodass eine Komponente (in unserem Fall eine Variation) von mehreren Merkmalen abhängen kann oder z.B. in das System integriert wird, sobald ein Merkmal oder eine Kombination von Merkmalen *nicht* ausgewählt wird.

Ein ähnlicher Ansatz wird in [MHP+07] verfolgt. Hier wird zwischen Produktlinien- und Softwarevariabilität unterschieden und eine separate Modellierung mit OVM- bzw. Merkmalmodellen vorgeschlagen. Auf einer formalisierten Beschreibung dieser Modelle und deren Beziehungen können anschließend Analysen automatisch ausgeführt werden. Teile dieser Analysen (z.B. nicht realisierbare Merkmalselektionen) können auf unsere Art der Modellierung übertragen werden. Der Großteil ist dagegen für uns in diesem Beitrag nicht relevant (z.B. fehlende Eindeutigkeit zwischen Produktlinien- und Softwarevariabilität, ungenutzte Merkmale), da sie keine direkte Gefährdung für den Konfigurationsprozess darstellen. Insgesamt zielen die vorgestellten Analysen eher darauf ab, ob die Architektur der Produktlinie flexibel genug ist, um die geplanten Produkte zu konfigurieren, wohingegen wir uns in diesem Paper auf Fehler in der Aussagenlogik der Konfigurationsformeln konzentrieren, um den Konfigurationsprozess an sich abzusichern.

Thaker et al. beschreiben einen Mechanismus für die „safe composition of product lines“ [TBKC07]. Auch dort werden SAT-Analysen vorgestellt, um zu prüfen, ob gültige Merkmalselektionen zu inkonsistenten Systemvarianten führen können. Während wir auf Inkonsistenzen innerhalb des Variantenmodells eingehen, werden dort Constraints beschrieben, die sich aus der Zusammensetzung von „feature modules“, also aus der Produktlinie selbst, ergeben.

6 Zusammenfassung

Die vorgestellten Analysen sollen sicherstellen, dass der Konfigurationsprozess fehlerfrei abläuft, damit eine gültige Merkmalselektion stets zu einer validen Systemvariante führt. Mit den Konfigurationsformeln stehen dabei gerade die Modellelemente im Fokus, die oft

händisch erstellt werden und daher eine potentielle Fehlerquelle darstellen. Die manuelle Prüfung der beschriebenen Bedingungen ist vor allem bei Modellen üblicher Größe nur mit sehr viel Aufwand oder gar nicht möglich. Da unser Prototyp die Analysen in angemessener Zeit ausführen kann, stellt er eine große Unterstützung für den Entwicklungsprozess des Variantenmodells dar.

Literatur

[Bat05] Don Batory. Feature models, grammars, and propositional formulas. In *Proceedings of the 9th international conference on Software Product Lines*. Springer-Verlag, 2005.

[CE00] Krzysztof Czarnecki und Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Reading, MA, USA, 2000.

[CN01] Paul C. Clements und Linda Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison-Wesley, August 2001.

[CW07] Krzysztof Czarnecki und Andrzej Wasowski. Feature diagrams and logics: There and back again. In *Software Product Line Conference, 2007. SPLC 2007. 11th International*, Seiten 23–34. IEEE, 2007.

[Hol12] Hannes Holdschick. Challenges in the Evolution of Model-Based Software Product Lines in the Automotive Domain. In *Proceedings of the 4th International Workshop on Feature-Oriented Software Development*, FOSD '12. ACM, 2012.

[KCH+90] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak und A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Bericht, Carnegie-Mellon University Software Engineering Institute, November 1990.

[LSB+] Rafael Lotufo, Steven She, Thorsten Berger, Krzysztof Czarnecki und Andrzej Wasowski. Evolution of the linux kernel variability model. In *Proceedings of the 14th international conference on Software product lines: going beyond*, SPLC 2010.

[MHP+07] Andreas Metzger, Patrick Heymans, Klaus Pohl, Pierre-Yves Schobbens und Germain Saval. Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis. In *RE*. IEEE, 2007.

[MRM+12] Swarup Mohalik, S Ramesh, Jean-Vivien Millo, Shankara Narayanan Krishna und Ganesh Khandu Narwane. Tracing SPLs precisely and efficiently. In *Proceedings of the 16th International Software Product Line Conference-Volume 1*. ACM, 2012.

[RKW09] M-O Reiser, Ramin Tavakoli Kolagari und Matthias Weber. Compositional variability-concepts and patterns. In *HICSS'09. 42nd Hawaii International Conference on System Sciences*. IEEE, 2009.

[SLB+11] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski und Krzysztof Czarnecki. Reverse engineering feature models. In Richard N. Taylor, Harald Gall und Nenad Medvidovic, Hrsg., *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011*, Seiten 461–470. ACM, 2011.

[TBKC07] Sahil Thaker, Don Batory, David Kitchin und William Cook. Safe composition of product lines. In *Proceedings of the 6th international conference on Generative programming and component engineering*, Seiten 95–104. ACM, 2007.