Modellbasierte Konsistenzprüfung von Produktlinien in Automatisierungssystemen[1]

Matthias Riebisch, Yibo Wang

Universität Hamburg, FB Informatik
{riebisch|wang}@informatik.uni-hamburg.de

Abstract: Bei der Entwicklung von Automatisierungssystemen stellen Produktlinien eine wichtige Technologie dar. Dazu werden Modellierungssprachen benötigt, die sowohl Variabilität und domänenspezifische Aspekte darstellen als auch Abhängigkeiten zwischen den verschiedenen Elementen von Anlage und Automatisierungssystem ausdrücken können. Vorhandene Modellierungssprachen erfüllen nur einzelne dieser Anforderungen - ein Vorgehen zur Entwicklung und Zusammenführung in einem Modell fehlt noch. Der Kurzbeitrag stellt die Entwicklung eines modellbasierten und an Produktlinien orientierten Entwicklungsvorgehens für Automatisierungssysteme im Überblick vor. Seine Evaluierung wird im industriellen Einsatz im Rahmen eines Verbundprojekts erfolgen.

1 Einleitung

Bei der Entwicklung von komplexen Systemen haben Wünsche nach kurzfristigen kundenspezifischen Anpassungen eine hohe Priorität. Bei Automatisierungssystemen besteht eine wichtige Forderung in hoher Flexibilität bezüglich der Optimierung von gesteuerten Prozessen der Verfahrens- und Fertigungstechnik. Techniken von Produktlinien beziehungsweise Systemfamilien erfüllen diese Forderungen. Diese Techniken ermöglichen außerdem die Verringerung der Entwicklungsdauer des einzelnen Automatisierungssystems, da die benötigte Variabilität bereits bei der Entwicklung der gemeinsamen Basis der Produktlinien vorgesehen wird. Variabilität wird dazu benutzt, um Entscheidungen auf eine spätere Entwicklungsphase zu verschieben werden.

Aufgrund vieler Risiken bei der Automatisierung wie Produktionsstörungen oder Umweltschäden sind frühzeitige Prüfungen erforderlich, um die Risiken zu verringern. Eine Beschreibung von Zusammenhängen und Wechselwirkungen mittels Modellen unterstützt diese Ziele. Die Komplexität, die aus den Anforderungen an zu automatisierende Prozesse und aus heterogenen Systemarchitekturen resultiert, wird durch Variabilität weiter gesteigert und erfordert daher ein modellbasiertes Vorgehen bei der Entwicklung. Jedoch sind vorhandene Vorgehensweisen und Modellierungssprachen noch nicht ausreichend entwickelt, um diese Anforderungen zu erfüllen.

Dieser Kurzbeitrag bietet einen ersten Überblick über ein Forschungsvorhaben, bei dem ein modellbasiertes und an Produktlinien orientiertes Entwicklungsvorgehen für Auto-

[1] Diese Arbeiten werden teilweise gefördert durch das BMBF unter dem Kennzeichen 01M3204C im Rahmen des Verbundprojekts „Entwurfsmethoden für Automatisierungssysteme mit Modellintegration und automatischer Variantenbewertung (EfA)“

matisierungssysteme entwickelt wird. Der wissenschaftliche Beitrag der Arbeiten besteht in der Entwicklung von domänenspezifischen Modellierungssprachen und deren Zusammenwirken sowie einem Vorgehen bei Entwicklung, Konfiguration und Verifikation von Automatisierungssystemen. Das Forschungsvorhaben ist Teil des Verbundprojekts „Entwurfsmethoden für Automatisierungssysteme mit Modellintegration und automatischer Variantenbewertung“ (EfA).

2 Stand der Technik

Für das Forschungsvorhaben sind vor allem folgende Gebiete und Vorarbeiten relevant: *Produktlinien-Ansätze* im Software Engineering haben das Featuremodell eingeführt. Features sind für den Nutzer relevante Eigenschaften eines Systems. Das Featuremodell bietet eine abstrakte Repräsentation der variablen Features sowie der Abhängigkeiten zwischen ihnen [CL01]. Auf Basis des Featuremodells ist eine kompakte Repräsentation aller Produkte einer Software-Produktlinie auf einem hohen Abstraktionsniveau für Anforderung und Grobplanung möglich [RU12]. Variabilität auf einer detaillierteren Ebene kann zum Beispiel im Familienmodell [PU04] oder durch sogenannte Assets im DOPLER-Decision-Modell [DH11] dargestellt werden. Weitere Feature-Eigenschaften und die Beziehungen zwischen Features und Artefakten können nicht dargestellt werden, so dass für Konsistenzbewertung weitere Modelle erforderlich sind, wie das Orthogonal-Variability-Model [PO05]. Codegenerierung wird in der Automatisierungstechnik meist nicht benötigt

Modellierungssprachen mit Beschreibung von Variabilität wurden zunächst für die Software- und Systementwicklung entwickelt. Für eingebettete Systeme existieren Erweiterungen bezüglich Variabilität für Modellierungssprachen wie Simulink [PO09] und SysML [HO12]. Beide Ansätze stellen Abhängigkeiten und deren Auswirkung auf die Konsistenz von Produkten einer Produktlinie nur auf der Ebene von Features dar, nicht jedoch auf feingranularer Ebene für Bestandteile von Systemen. Eine durchgängige Konsistenzprüfung über verschiedene Modelle und Abstraktionsebenen hinweg, wie für Feature-Modell, Funktionsnetz, Funktionsbaustein und generierten Quelltext, wurde in derartigen Ansätzen nicht komplett behandelt. Unser Forschungsansatz soll dies erlauben, und so eine modellbasierte Entwicklung von Automatisierungssystemen zu erreichen.

Vorgehensweisen für Entscheidungsfindung und Konfiguration wurden ebenfalls zunächst für Software-Produktlinien entwickelt, beispielsweise in [VI10] für durchgängige Konsistenzprüfung von Feature-Modell über Komponenten-Modell bis zum Code. Für Automatisierungssysteme fehlen solche Vorgehensweisen noch.

Für die Konfiguration einer Produktlinie wird in [CH11] zwischen regelbasierten, modellbasierten und Fall-basierten Konsistenzprüfungen unterschieden. In unserem Vorhaben wird der regelbasierte Ansatz verfolgt, denn nur dieser ermöglicht es, Konsistenzregeln auf allen Ebenen im gleichen Format wie zum Beispiel mit RuleML[2] oder mit

[2] RuleML – Rule Markup Language auf Basis XML http://wiki.ruleml.org/index.php/RuleML_Home

EMFTrace[3] zu definieren. Zudem sind formalisierte Regeln nach diesen Ansätzen sowohl für Menschen als auch maschinell lesbar.

3 Forschungsansatz

Für die *Modellierung von Variabilität* werden Featuremodelle angewendet. Abbildung 1 zeigt ein Beispiel aus dem Projekt EfA. Featuremodelle dienen vor allem dazu, Features und die Beziehungen zwischen ihnen (alternative, or, requires, excludes) sowie ihre Variabilität (mandatory oder optional) darzustellen. Sie werden für die Definition und die Konsistenzprüfung von Produkten auf Basis der Produktlinie angewendet. Die Modellierung von Variabilitätspunkten erfolgt nicht nur innerhalb der Featuremodelle, sondern mittels *domänenspezifischer Modellierungssprachen* auch innerhalb der Modelle für Anlage sowie Automatisierungssystem jeweils für Struktur und Verhalten.

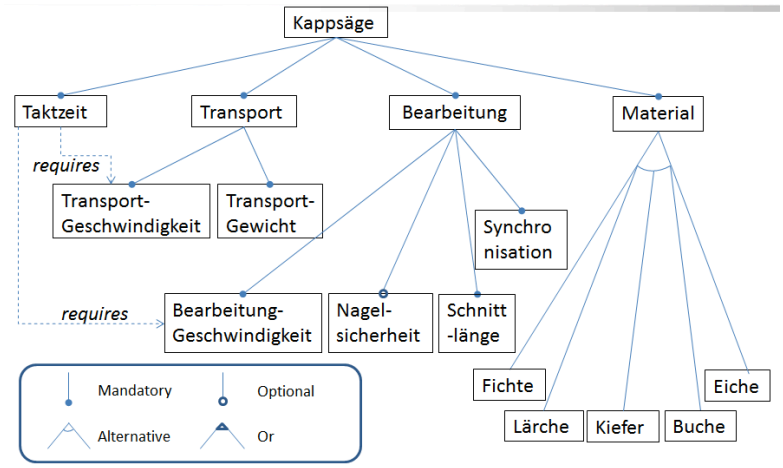

Abbildung 1: Featuremodell der Kappsäge im Projekt EfA

Entsprechende Modellierungssprachen sind in der Praxis etabliert, allerdings müssen sie um Ausdrucksmittel für Variabilitätspunkte erweitert werden. Außerdem müssen *Verknüpfungen* zwischen den verschiedenen Modellen und mit dem Featuremodell hergestellt werden, um Abhängigkeiten ausdrücken zu können, die für die Konsistenzprüfung ausgewertet werden. Zur *Zusammenführung von Modellen* bei der Entwicklung von Automatisierungslösungen wird AutomationML [AU13] als neutrales, XML-basiertes Datenformat genutzt, um die Speicherung und den Austausch zwischen verschiedenen Beteiligten und Werkzeugen bei der Planung von Automatisierungsanlagen zu ermöglichen. Abbildung 2 zeigt schematisch die mittels AutomationML beschreibbaren Aspekte. Im Projekt EfA wird Variabilität für die Aspekte Topologie und Logik betrachtet und in AutomationML modelliert. Dazu soll eine Kopplung zwischen Modellelementen von AutomationML und Features des Featuremodells vorgenommen werden. Diese Kopplung soll, wie auch bei anderen beteiligten Modellierungssprachen, über Verweise implementiert werden, die zwischen den XML-Repräsentationen der Modellierungsspra-

[3] EMFTrace – Repository zum Zusammenführen von Modellen http://sourceforge.net/projects/emftrace/

chen etabliert werden. AutomationML wurde um neue Elemente Variation Point und Variant sowie um Attribute für Variabilität erweitert.

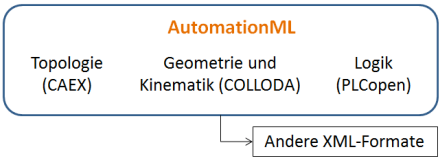

Abbildung 2: Von AutomationML abgedeckte Aspekte der Planung und Entwicklung von Automatisierungssystemen

Die Entscheidung für Art der Produktlinien-Implementierung wie 150%-Ansatz oder Delta-Ansatz [SC10] hängt vor allem von Lösungsbausteinen und Technologie der Implementierung ab. Unser Konzept sieht die Unterstützung beider Ansätze vor.

Da AutomationML keine Unterstützung für Variantenmodellierung bietet, muss es um Variationspunkte, Attribute für Varianten und Bezüge zu Features in Featuremodellen erweitert werden. Für den im Projekt angestrebten Entwurf von Automatisierungssystemen auf Basis der Produktlinie wird ein Konfigurationswerkzeug entwickelt, dass auf Basis des Featuremodells die passenden Bibliothekskomponenten und zulässige Optionen für Konfigurationen anbietet. Darüber hinaus wird bei der Varianten-Konfiguration, auf Basis der vordefinierten Regeln, die Konsistenzprüfung auf jeder Detaillierungsstufe durchgeführt, um die ungültige Variantenkombination frühzeitig ausschließen zu können.

Der beim Entwurf und der Planung gewünschte Grad der Werkzeugunterstützung bestimmt den Grad der *Formalisierung der Modelle*. In semiformalen Beschreibungen wie Blockdiagrammen von SysML oder Objekt-Bäumen von AutomationML (siehe Abbildung 3) wird ein wesentlicher Teil der Semantik üblicherweise durch Bezeichner und erläuternde Texte ausgedrückt, die in eine Struktur eingebettet werden, welche durch eine formal definierte Syntax definiert ist. Ein Beispiel hierfür ist die Verbindung der Signalschnittstelle „Channel01“ zu der Schnittstelle „Start“ in Abbildung 3. Eine Prüfung, ob die Verbindung dieser Schnittstellen korrekt ist, wäre erst durch Einbeziehen von weiteren Beschreibungen zur Semantik der Blöcke möglich.

Werkzeugunterstützung zur Konsistenzprüfung setzt jedoch formalisierte Ausdrucksmittel voraus. Sind beispielsweise Konsistenzprüfungen bezüglich des Verhaltens erforderlich, werden entsprechende formalisierte Modelle für Verhalten und Logik, wie etwa erweiterte State-Charts, Petri-Netze oder Entscheidungstabellen, angewendet. Soweit der Formalisierungsgrad erhöht werden soll, werden natürlich-sprachliche Ausdrücke durch Referenzen und definierte Typen ersetzt – in unserem Forschungsvorhaben für Anlagenstruktur und logische Abhängigkeiten. Aussagen zur Semantik innerhalb anderer Modelle, wie beispielsweise zur Bedeutung eines, sollen im Rahmen des Forschungsvorhabens nicht formalisiert werden.

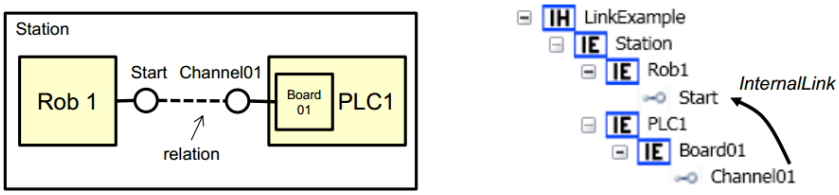

Abbildung 3: Beispiel für Blockdiagramm in SysML (links) und Objekt-Bäume in AutomationML (rechts) [AU13]

Bei Entwurfsentscheidungen im Planungsprozess für Automatisierungslösungen werden die Modelle ausgewertet, um werkzeuggestützte *Konsistenzprüfungen* durchzuführen. Im Rahmen eines Produktlinien-Ansatzes erfolgt der Entwurf überwiegend durch die Konfiguration von variablen Elementen einer Produktlinie. Konsistenzprüfungen beziehen sich dabei beispielsweise auf Beschreibungen von Attributen und Schnittstellen. Für solche Prüfungen werden Regeln entwickelt und formalisiert, die sich zum Beispiel auf Typen und Parameter von Funktionsblöcken und auf Wertebereiche von Eingangs- und Ausgangssignalen beziehen.

Als Ergebnis des Forschungsvorhabens werden die *Modelle und Regeln* eingesetzt, um Anwenderunterstützung beim Entwurf von Automatisierungslösungen auf Basis einer Produktlinie bereitzustellen. Entwickler werden beispielsweise bei Entscheidungen unterstützt, indem nur gültige Lösungsvarianten zur Auswahl angeboten und Plausibilitäts- und Konsistenzprüfungen der Entwurfsergebnisse automatisch durchgeführt werden. Der Nutzen besteht in Effizienzsteigerung sowie in Fehlervermeidung beim Entwurf von Automatisierungslösungen. Abbildung 4 stellt eine Regel für das Kappsäge-Beispiel dar. Sie prüft, ob der Markensensor in Bezug auf das Bearbeitungsaggregat korrekt platziert wurde. Durch Verweis auf einen Regel-Katalog wird gekennzeichnet, dass es sich um Eine Regel für die Dimension Solution Domain und um Konsistenzprüfung bezüglich Topologie handelt. Eine Zuordnung zu Regler/Regelstrecke gibt es hier nicht.

```
<Rule RuleID="Rule01" Description="Der Markensensor muss vor der
 Startposition des Bearbeitungsaggregats platziert werden">
         <Elements Type="Sensor" Alias="e1"/>
         <Elements Type="Werkzeug" Alias="e2"/>
         <Actions ActionType="ReportConsistencyViolation"
       ResultType="" SourceElement="e1" TargetElement="e2"
       ImpactedElement=""/>
                  <Conditions>
                           <BaseConditions Type="ValueGreaterThan"
                            Source="e1::position"
                            Target="e2::position"/>
                  </Conditions>
</Rule>
```

Abbildung 4 Beispiel einer Konsistenzregel in EMF-Trace

Die Bewertung und Entscheidungsunterstützung wird in das *Vorgehen und die Methodik* des Entwurfs der Automatisierungssysteme integriert. Domänenspezifische sowie betriebliche Vorgaben fließen ebenfalls ein. Das Vorgehen und die Methodik definieren den Umfang und Anwendungsbereich der zu erstellenden Modelle, deren Abstimmung auf die beteiligten Rollen der verschiedenen Disziplinen, die Entwicklungsphasen von Produktlinien- und Produktentwicklung.

4 Weitere Schritte

Das hier vorgestellte Forschungsvorhaben wird seit Herbst 2012 durchgeführt und hat zu ersten Ergebnissen geführt; sein Abschluss ist für den August 2015 geplant. Als nächste Schritte werden Abhängigkeiten zwischen Elementen der benötigten Modellierungssprachen ermittelt und als Traceability-Beziehungen zwischen verschiedenen Modellen beschrieben. Weiterhin wird eine Methode zur Variantenbewertung entwickelt, die ähnlich wie die beschriebene Konsistenzprüfung Informationen aus den formalisierten Modellen ableitet. Als Implementierung dieser Konzepte wird ein prototypisches Werkzeug entwickelt. Innerhalb des Verbundprojekts EfA wird dieses Werkzeug einen Bestandteil der Werkzeugkette bilden, die beim Anforderungsmanagement auf der Sprache ReqIF[4] basiert. Für die Anlagen-Modellierung baut sie auf AutomationML [AU13] und den daran angebundenen Planungswerkzeugen auf, für die Traceability-Links auf das Repository EMFTrace. Für das Variantenmanagement wird das Werkzeug pure::variants[5] eingesetzt.

Literaturverzeichnis

[AU13] -: AutomationML Whitepaper Part 1 - Architecture and general requirements. AutomationML e.V., 2013. auf https://www.automationml.org/ am 15.01.2014

[CL01] Clements, P.; Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley Professional, 2001; S. 114

[CH11] Brecher, C., Karlberger, A., & Herfs, W.: Synchronisation of Distributed Configuration Tools using Feature Models. In 4th International Conference on Changeable, Agile, Reconfigurable und Virtual Production, 2011, S. 340-345

[DH11] Dhungana, D., Grünbacher, P. & Rabiser, R.: The DOPLER meta-tool for decision-oriented variability modeling: a multiple case study. Automated Software Engineering 18, 2011; S. 77-114

[HO12] Hoyos, H.; Casallas, R.; Jimenez, F.: Model-Based Framework for Embedded System Product Line. In Proc. 38th Annual Conference on IEEE Industrial Electronics Society (IECON), 2012; S. 3101-3106

[PO05] Pohl, K.; Böckle, G.; Van der Linden, F.: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer. 2005; S. 85

[PO09] Polzer, A.; Kowalewski, S.; Botterweck, G.: Applying Software Product Line Techniques in Model-based Embedded Systems Engineering. In Proc. Model-based Methodologies for Pervasive and Embedded Software (MOMPES), 2009; S. 2-10

[PU04] -: Technical White Paper Variantenmanagement mit pure::variants. pure-systems GmbH, 2004. auf http://www.pure-systems.com/fileadmin/downloads/pv-whitepaper-de-04.pdf am 18.02.2014

[RU12] Russell, J.; Cohn, R.: Feature Model, Book on Demand, 2012

[SC10] Schaefer, I.; Bettini, L.; Damiani, F.; Tanzarella, N.: Delta-oriented programming of software product lines. In Proceedings of the 14th international conference on Software product lines: going beyond (SPLC'10), Jan Bosch and Jaejoon Lee (Eds.). Springer-Verlag, Berlin, Heidelberg 2010; S. 77-91

[VI10] Vierhauser, M.; Grünbacher, P.; Egyed, A.; Rabiser, R.; Heider, W.: Flexible and scalable consistency checking on product line variability models. In Proc. of the IEEE/ACM international conference on Automated software engineering (ASE), 2010; S. 63-72

[4] Requirements Interchange Format (ReqIF) http://www.omg.org/spec/ReqIF/

[5] pure::variants - Werkzeug zum Variantenmanagement http://www.pure-systems.com/pure_variants.49.0.html