Modellorientiertes Variantenmanagement

Christian Bartelt, Sebastian Herold

AG Softwarearchitektur
TU Kaiserslautern - FB Informatik
Erwin-Schrödinger-Straße
67653 Kaiserslautern
{bartelt, herold}@informatik.uni-kl.de

Abstract: Ein probates Mittel zur Beschreibung von komplexen Sachverhalten ist das Entwickeln geeigneter Modelle. Häufig entsteht während der Entwicklung von Systemen oder Prozessen nicht nur ein Modell, sondern mehrere Modellvarianten. Ein Ansatz zur Beschreibung von Variabilität ist der Produktlinienansatz, der versucht, eine Menge von Produktausprägungen in einem geeigneten Modell zu beschreiben. Dadurch entstehen häufig sehr komplexe und schwer verständliche Modelle. Zudem werden Modellvarianten nicht berücksichtigt, wie sie bspw. in der verteilten, modellbasierten Entwicklung entstehen. Ein modellorientiertes Variantenmanagement löst die beschriebenen Probleme durch die Bildung von Varianten auf Modellebene. Eine Herausforderung der resultierenden Variantenvielfalt ist ihre Verwaltung. Beispiele für Verwaltungsoperationen ist die Analyse von Unterschieden zwischen Varianten und das Zusammenführen von Varianten. Unterstützung im Management der Variabilität beliebiger Modelle soll das hier vorgestellte modellorientierte Variantenmanagement bieten, dabei wird der aktuelle Stand der Forschung zu diesem Thema vorgestellt, die relevanten Anforderungen an ein modellorientiertes Variantenmanagement identifiziert und Lösungsideen für die unterschiedlichen Problemstellungen aufgezeigt.

1 Einleitung

Modelle dienen in vielen Bereichen als Abstraktion von komplexen Sachverhalten. Insbesondere während der Entwicklung von Systemen oder Prozessen entsteht nicht nur ein Modell, sondern eine ganze Reihe von Modellvarianten. Die Ursachen für diese Variantenbildung sind vielfältig. So entstehen neue Varianten (a) bei der geplanten Anpassung von Produkten an die Nutzer (Customization), (b) bei der unabhängigen Bearbeitung von Modellen verschiedener Teams und (c) bei der räumlich verteilten Entwicklung von Modellen. Ein Beispiel für die Variantenbildung in Bezug auf die geplante Anpassung von Produkten an die Nutzer ist der Produktlinienansatz. Die unter (b) genannte Variantenbildung ist z. B. in der Open-Source-Softwareentwicklung bei Programmcode zu finden. Hier bilden sich häufig verschiedene Entwicklungsstränge auf Basis einer gemeinsamen Softwareversion. Bei der unter (c) genannten Ursache entstehen Varianten eines Modells durch Änderungen, die an einer lokalen Kopie des Modells durch die Modellbearbeiter

gleichzeitig vorgenommen werden. Diese Variantenbildung wirft Fragen danach auf, wie die Modellvarianten wieder zusammen geführt oder Unterschiede zwischen den Modellen festgestellt werden können. Diese Fragen hängen nicht zuletzt von Charakteristika von Modellen und Modellierungssprachen ab, wie bspw. Grad der Formalisierung der Syntax. In den folgenden Abschnitten wird das Themengebiet Variabilitätsmanagement von Modellen strukturiert, relevante Problemstellungen erörtert und Lösungsansätze vorgestellt.

Abschnitt 2 motiviert an einem Beispiel das modellbasierte Variantenmanagement. Der Abschnitt 3 erläutert genauer die Grundlagen eines modellbasierten Variantenmanagements. Im Abschnitt 4 wird auf die Grundlagen von Graphen als eine geeignete Modellierungsmöglichkeit für das Variantenmanagement eingegangen, anschließend werden in Abschnitt 5 Möglichkeiten für ein modellorientiertes Variantenmanagement auf der Basis von Graphen beschrieben. Zum Abschluss wird in Abschnitt 2 ein Beispiel aus der Praxis beschrieben, in dem die zuvor erläuterten Probleme bei der Variantenbildung von Modellen auftreten.

2 Beispiel: Variabilität des Prozessmodells V-Modell XT

In der Einleitung wurde das Themengebiet Variabilität von Modellen allgemein umrissen. In diesem Abschnitt wird anhand eines konkreten Szenarios aus der Praxis die Problemstellung bei der Variantenbildung von Modellen heraus gestellt.

Das Vorgehensmodell V-Modell XT [KMS+05] gibt seit seiner Veröffentlichung den verbindlichen Standard in allen Bundesbehörden der Bundesrepublik Deutschland vor. Mit Hinblick auf eine umfassende Werkzeugunterstützung wurde ein Großteil der Struktur des V-Modell XT von vorne herein durch ein formales Metamodell beschrieben. Zusätzlich zu den Struktur beschreibenden Teilen des V-Modell XT enthält es eine Reihe von Informationen, die keine formale Syntax besitzen, wie zum Beispiel Texte, die das Modell in natürlicher Sprache beschreiben, oder Grafiken. Das V-Modell-XT-Metamodell [KMS+05] beinhaltet in der aktuellen Version sowohl die Struktur des Modells als auch dessen Dokumentation.

Im Hinblick auf ein Variantenmanagement des gesamten Modells interessiert uns in diesem Abschnitt ausschließlich die organisationsspezifische Anpassbarkeit des V-Modell XT. Besonders hervorzuheben ist, dass das V-Modell XT zur Zeit fast keine Einschränkungen für eine Anpassung festlegt. Der Rahmen für Anpassungen besteht im Wesentlichen nur aus der geforderten Konformität zum V-Modell-XT-Metamodell. Somit wird die Variabilität des V-Modell XT hinsichtlich organisationsspezifischer Anpassungen nicht explizit im Modell (wie bspw. mit Hilfe von Variationspunkten) beschrieben. Organisationen, Unternehmen und öffentliche Einrichtungen passen die offiziellen Releases an und bilden dadurch organisationsspezifische Varianten des Modells.

Mit dem englischen V-Modell XT kommt eine neue Variante einer textuellen Repräsentation des Modells hinzu. Varianten der Sprachrepräsentation eines Modells, wie in diesem Beispiel aus der Praxis, haben die besondere Eigenschaft, dass es i.d.R. keine automatische Transformation zwischen Modelltexten unterschiedlicher Sprachen geben kann, sondern

nur der Mensch allein die Konsistenz der Inhalte zwischen den Varianten beurteilen kann.

Bei der organisationsspezifischen Anpassung des V-Modell XT besteht die Möglichkeit sowohl Texte (Modellteile mit informeller Syntax) als auch die Struktur (mit formaler Syntax) zu ändern. Um die Variabilität bei der verteilten Entwicklung des Modells zu beherrschen, werden Diff- und Merge-Operationen zur Zeit auf dem in XML dargestellten Modell mit Hilfe eines CVS-Systems organisiert [MRS06]. Für die textuellen Anteile des Modells hat sich dieses Vorgehen in der Praxis als erfolgreich erwiesen, strukturelle Änderungen werden allerdings unbefriedigend berücksichtigt.

3 Grundlagen des modellorientierten Variantenmanagements

Zum besseren Verständnis wird zunächst der Begriff des modellorientierten Variantenmanagements erläutert. Das *modellorientierte Variantenmanagement* ist ein Ansatz zur Beherrschung der Vielfalt unterschiedlicher Modelle, die aufgrund von Veränderungen eines gemeinsamen Ursprungsmodells entstanden sind. Das hier betrachtete Variantenmanagement bezieht sich ausschließlich auf die Analyse und Bearbeitung der Modellsyntax. Von der konkreten Modellsyntax wird hier abstrahiert, um Lösungsansätze für beliebige Modelle zu diskutieren. In diesem Kontext ist eine Reihe von Fragen interessant:

- Wie sieht die Differenz zwischen den unterschiedlichen Modellvarianten aus? Gibt es korrespondierende Modelländerungen in den Varianten und wie lassen sich diese erkennen?
- Ist ein (teil)automatisiertes Zusammenführen der Modellvarianten möglich und wenn ja, wie sieht dies aus? Wie werden Konflikte beim Zusammenführen dargestellt und aufgelöst?

Ziel der hier vorgestellten Ansätze ist es, ein Konzept für die Variantenverwaltung von Modellen zu entwickeln. Die Schwierigkeiten liegen dabei in der Allgemeinheit der angestrebten Lösung. So sind die Formen für Modellrepräsentationen vielfältig, außerdem ist der Formalisierungsgrad der verwendeten Modelle unterschiedlich hoch. Nicht zuletzt kann der Einsatz eines Variantenmanagementsystems unterschiedlich motiviert sein (Modellanpassung, verteilte Entwicklung, Synchronisation von Varianten unterschiedlicher Entwicklungszweige, etc.) und dadurch unterschiedlichen Anforderungen und Rahmenbedingungen unterliegen. Im folgenden Abschnitt werden mögliche Features näher erläutert.

3.1 Eigenschaften des modellorientierten Variantenmanagements

Die Syntax von Modellen ist nicht immer vollständig formal, in vielen Modellen ([KMS+05, IBM03]) findet man neben formal strukturierten Informationen informell beschriebene Modellteile (als Texte, Grafiken, etc.). An ein praxistaugliches Variantenmanagement stellt

sich somit die Anforderung, sowohl formal beschriebene Modellinformationen als auch Informationen mit informeller Syntax (bspw. natürliche Sprache) zu verwalten.

Umfangreiche Änderungen der Modellvarianten Der Umfang von Änderungen vom Ursprungsmodell zu den einzelnen Modellvarianten kann unterschiedlich groß sein. So pflegen Benutzer von Versionsmanagementsystemen wie CVS ([Ced93]), die auf einem gemeinsamen Repository arbeiten, i.d.R. Änderungen in feingranularen Schritten ein. Operationen zum Zusammenführen von Modellversionen und Konfliktbereinigung werden hier auch thematisch als Operationen eines Variantenmanagements eingeordnet. In Projekten, die sich bspw. mit organisationsspezifischen Anpassungen eines Standardmodells beschäftigen, werden im großen Umfang Erweiterungen und Änderungen des ursprünglichen Modells vorgenommen, ohne dass zwischenzeitlich das organisationsspezifische Modell mit dem Standard synchronisiert wird.

Dezentrale Verwaltung Im Gegensatz zu Versionsverwaltung wie CVS arbeiten Benutzer eines Variantenmanagements im Allgemeinen nicht auf einem gemeinsamen Repository. Demzufolge kann im Allgemeinen auch nicht davon ausgegangen werden, dass die Bearbeiter der zu verwaltenden Modelle Synchronisationsinformationen zu anderen Modellen (wie bspw. zu einem Master-Modell) lokal gespeichert haben, sondern, dass die Modellvarianten vollständig unabhängig voneinander sind.

Mustererkennung in Modellstrukturen Mehrere Modellvarianten, auf denen mit Hilfe eines Variantenmanagements Operationen ausgeführt werden sollen, haben laut Definition ein gemeinsames Ursprungsmodell. Durch die Speicherung des Ursprungsmodells, der Änderungshistorie oder durch die Verwendung von eindeutigen Bezeichnern für alle Modellelemente ist es möglich, die Modellteile des Ursprungsmodells in den Modellvarianten zu erkennen. Allerdings ist es möglich, dass in den Modellvarianten ähnliche oder äquivalente Erweiterungen vorgenommen wurden. Durch Analysen der Struktur dieser Modellerweiterungen ist es möglich, ähnliche Muster in den Varianten zu erkennen. Einen Ansatz dafür findet man in [AAAN+05].

3.2 Eigenschaften der zu verwaltenden Modelle

Die Ansätze zum Management von Modellvarianten, hängen wesentlich von den Charakteristika der Modellbeschreibungen ab. Außerdem werden Modelle häufig unterschiedlich repräsentiert, wobei es nicht in jedem Fall möglich ist, die Modellinformationen automatisch in allen Repräsentationen abzugleichen. In den folgenden Abschnitten werden diese Eigenschaften genauer erläutert.

Varianten der Modellrepräsentation sind Variationen eines Modells, deren modellierte Information identisch ist, bei denen allerdings die Modellinformationen unterschiedlich repräsentiert werden. Ein Beispiel ist die Repräsentation von Modellinformationen als Text in unterschiedlicher Sprache (vgl. Abschnitt 2). Die Problematik bei diesem Szenario ist

die Konsistenzsicherung der Modellinformationen zwischen den Sprachen. So entstehen potentiell bei jeder Änderung der Modellsyntax in einer Sprachvariante Inkonsistenzen in der Semantik zwischen den Sprachvarianten. Die Entscheidung, ob es sich wirklich um Inkonsistenzen handelt, kann nur durch den Menschen getroffen werden. Allerdings ist es möglich, den Benutzer rechnergestützt auf eventuelle Inkonsistenzen hinzuweisen, wenn die Änderungshistorie gespeichert ist.

Modelle sind unterschiedlich formal strukturiert. Beispiele für unstrukturierte Dokumente sind reine Textbeschreibungen von Modellen, wohingegen eine vollständige formale Spezifikation den höchsten Strukturierungsgrad aufweist. Modellinformationen ohne formale Syntax sind in der Regel Modellbeschreibungen als Text oder Grafik ohne eine formal spezifizierte Struktur. Textanalysetechniken, die schon in der Praxis effektiv in Versionsverwaltungssystemen wie im populären CVS eingesetzt werden, basieren in der Regel auf [Fel69]. Ein ganzheitlicher Ansatz zum Variantenmanagement sollte auch unstrukturierte Modellinformationen berücksichtigen können.

Um ein möglichst automatisches Management von Modellvarianten zu verwirklichen, ist es notwendig, Modelle in einer maschinenverarbeitbaren, eindeutigen Form zu beschreiben. Die Grundlage dafür bildet eine formale Syntax der Modelle. Wie in Abschnitt 5 beschrieben wird, bieten sich Graphen besonders gut für formale Modellbeschreibungen an.

4 Modellierung

Die Bedeutung formaler Modelle bei der Entwicklung von Systemen ist von großer Bedeutung. Ziel dabei ist, den Entwickler in allen Phasen des Lebenszyklus des Systems durch Modelle zu unterstützen. In der industriellen Praxis spiegelt sich dies u.a. in der wachsenden Bedeutung von modellbasierten Entwicklungsansätzen wie bspw. der *Model Driven Architecture (MDA)* [Gro] wider. Zusätzlich wird der Entwicklungsprozess selbst immer häufiger durch *Prozessmodelle* festgelegt

In diesem Abschnitt wird beschrieben, wie mittels einer verallgemeinerten Formalisierung durch Graphen ein möglichst generischer Ansatz zum Management von Modellvarianten auf Basis einer formalen Syntax realisiert werden kann.

4.1 Graphen als Modellierungsgrundlage

Um eine automatische Analyse und Bearbeitung eines Modells im Rahmen eines allgemeinen Variantenmanagements durchführen zu können, ist eine formale Definition der Modelle und der Modellierungssprachen erforderlich. Die MDA bedient sich dazu der Metamodellierungsmechanismen der *Meta Object Facility (MOF)*. Durch Metamodelle wird die abstrakte Syntax der Modellierungssprache beschrieben und somit eine eindeutig definierte Struktur der Modelle festgelegt. Die abstrakte Syntax eines Modells kann auch als Graph aufgefasst werden. Der Vorteil der Betrachtung als Graph liegt in der Möglichkeit,

Änderungen und Transformationen formal und deklarativ ausdrücken zu können, wie in Graphersetzungssystemen zu sehen (vgl. [Sch91]). Zudem existieren einige Ansätze zur Realisierung von Operationen zur Feststellung von Differenzen zwischen Graphen sowie zur Zusammenführung von Graphen. Diese werden im Folgenden näher betrachtet.

4.2 Ansätze zum Vergleichen und Zusammenführen von graphähnlichen Strukturen

Im Folgenden werden einige Ansätze zum Vergleichen und Zusammenführen von graphähnlichen Strukturen verglichen und auf ihre Anwendbarkeit für das Variantenmanagement untersucht. Die betrachteten Ansätze lassen sich bzgl. der Modelleigenschaften bzw. der Einbettung in das umgebene System kategorisieren:

Modellstruktur: Im Allgemeinen bildet die abstrakte Syntax eines Modells zwar einen Graphen, je nach Modellart oder betrachteter Domäne können bspw. Sichten hierarchisch strukturiert sein, wie die Modulsicht auf die statische Struktur eines Systems. Diese Hierarchie spiegelt sich in großen, baumähnlichen Teilgraphen im abstrakten Syntaxgraphen wider.

Eindeutige Bezeichner: Das Vergleichen und Vereinigen von Graphen kann als Graphmatchingproblem aufgefasst werden, welches im Allgemeinen ein NP-vollständiges Problem ist. Unter einigen Annahmen kann das Matching von Graphen jedoch effizient durchgeführt werden. Bspw. führt die Annahme von eindeutigen Bezeichnern zu Vergleichsalgorithmen mit polynomialen Laufzeitverhalten.

Aufzeichnung von Modelländerungen: Modellveränderungen können in einer Änderungshistorie aufgezeichnet werden. Die Identifikation korrespondierender Elemente kann so u.U. durch Rückverfolgung der Historie vereinfacht werden.

Benutzerinteraktion: Die Ansätze unterscheiden sich ebenfalls im Grad der unterstützten Benutzerinteraktion. Dies betrifft sowohl Benutzerentscheidungen bei der Auflösung von Konflikten während des Zusammenführens von Graphen, als auch die Berücksichtigung manuell gematchter Strukturen.

In [DBDK04] wird ein Ansatz vorgestellt, der das Matching von Graphen mit eindeutigen Knotenbezeichnern ohne Attribute durchführt. Das Matchingproblem wird dort auf die Schnittmengenbildung der Knotenbezeichner reduziert und somit effizient lösbar. Auf ähnliche Weise wird die Berechnung einer *graph edit distance* effizient durchgeführt. Sie beschreibt eine minimale Folge von elementaren Graphoperationen, die notwendig zum Transformieren eines Graphen in einen anderen ist. Auf dieses Maß für die Ähnlichkeit von Graphen können andere Ansätze zum eigentlichen Vereinigen von Graphen aufsetzen. Ein ähnlicher Ansatz findet sich in [AAAN$^+$05].

[ZWR01] und [AP03] schlagen Ansätze vor, in dem das Matching über identische Knotenbezeichner in den beiden beteiligten Graphen realisiert wird. Beim Vereinigen wird davon ausgegangen, dass die Graphen sich aus einem gemeinsamen Ursprungsgraphen entwickelt haben. Modelländerungen werden also nicht aufgezeichnet, sondern zurückge-

rechnet. Sofern die Änderungen nicht dieselben Knoten betreffen, lässt sich eine Vereinigung einfach durch die Ergänzung der jeweils anderen Änderung durchführen (s. Abb. **??**). Mögliche Konfliktauflösung und sinnvolle Benutzerinteraktionen werden nicht erwähnt.

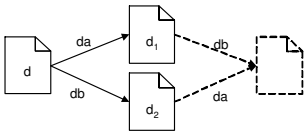

Abbildung 1: 3-Wege-Vereinigung

In [MGMR02] (ähnlich in [KWN05]) wird ein allgemeiner Graphmatchingalgorithmus vorgestellt, der auf der Annahme basiert, dass die Ähnlichkeit zweier Knoten in unterschiedlichen Graphen von der Ähnlichkeit ihrer adjazenten Knoten abhängt und über diese ermittelt wird *(similarity flooding)*. Ausgehend von einem beliebigen initialen, ggf. anwendungsspezifisch bestimmten Mapping wird ein Mapping zwischen Knoten mit möglichst hohen Ähnlichkeitswerten berechnet. Benutzerinteraktion ist explizit vorgesehen. Zudem kann das Verfahren gut durch Anpassung der Ähnlichkeitspropagation parametrisiert werden. Eindeutige Knotenbezeichner sind nicht erforderlich.

5 Graphbasiertes Variantenmanagement von Modellen

In Abschnitt 4.1 bzw. 4.2 wurde erläutert, wie allgemeine Modelle durch Graphen repräsentiert werden können, und Modellvergleiche und -zusammenführungen (s. Abschnitt 3) auf Graphen realisiert werden können. Dieser Abschnitt betrachtet Eigenschaften von Graphen als Repräsentationen von Modellvarianten und beschäftigt sich mit den Anforderungen an die Operationen zum Zusammenführen von Varianten.

5.1 Eigenschaften von Graphen als Repräsentationen von Modellvarianten

Strukturvarianten von Modellen zeichnen sich durch ein größeres, in beiden Modellen existierendes Modellfragment aus, dass bei disjunkten Ergänzungen dem Ursprungsmodell, im Allgemeinen aber einem Teilmodell dessen entspricht. Die Modellvarianten besitzen folglich einen gemeinsamen Teilgraphen, der anhand von eindeutigen Knotenbezeichnern ermittelt werden kann.

Überlappende strukturelle Änderungen sind ebenfalls in beiden Modell- bzw. Graphvarianten enthalten, können anders als Teile des Ursprungsmodells aber nicht über eindeutige Knotenbezeichner im Graphen identifiziert werden. Für die Operationen auf Graphen im Rahmen eines Variantenmanagements ergeben sich somit einige Anforderungen. Wie beschrieben ist die Verwendung von eindeutigen Knotenbezeichnern in Graphen für Modellvarianten nützlich. Im Allgemeinen handelt es sich bei den zu verwaltenden Strukturen um

Graphen, so dass ein Variantenmanagement auf diesem Strukturtyp aufsetzen sollte. Eine Änderungshistorie der Modelle wird nicht mitgeführt. Da nicht davon ausgegangen werden kann, dass bestehende Werkzeuge zur Erstellung von Modellen oder Modelle selbst eine Änderungshistorie erstellen, müsste diese aufwändig a-posteriori integriert werden.

Der Nutzen eines Variantenmanagements ist für den Benutzer nach Meinung der Autoren am höchsten, wenn er beim Zusammenführen von Varianten direkt interaktiv eingreifen kann. Zudem ist eine sinnvolle Zusammenfassung und Abfolge von Benutzerentscheidungen aufgrund der Langläufigkeit von Varianten und die daraus entstehende Komplexität der Modelldifferenzen notwendig.

5.2 Systemidee

Ein Variantenmanagementsystem soll die Analyse, das Erstellen und Verwalten von Varianten unterstützen. Abbildung **??** zeigt ein Beispiel für die Verwendung eines solchen Systems.

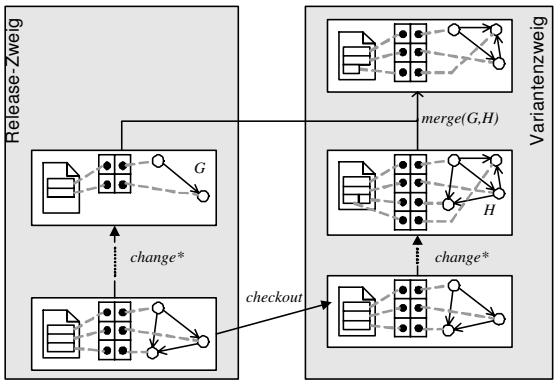

Abbildung 2: Arbeitsweise eines Variantenmanagementsystems

Dargestellt ist die Entwicklung eines Modells in einem Release-Zweig, der bspw. für den offiziellen Entwicklungszweig eines Vorgehensmodells steht. Das Variantenmanagement verwaltet zusätzlich zum eigentlichen Modell eine interne Graphrepräsentation des Modells sowie ein Mapping zwischen Modellelementen und Knoten über deren eindeutige Bezeichner. Eine Organisation möchte dieses Modell nun zu einer organisationsspezifischen Variante verändern und *erstellt* eine Kopie der aktuellen Release *(new_variant)*.

Beide Modelle werden nun verteilt und nebenläufig *verändert*. Zu beliebigen, diskreten Zeitpunkten können aktuelle Versionen der Modelle in das System eingepflegt werden *(change)*, wobei die Konsistenz zwischen Modell und Graphrepräsentation wieder hergestellt wird. Werden nun Modellvarianten M und N bzw. ihre Graphdarstellungen G_M und G_N betrachtet, so können unterschiedliche Teilstrukturen identifiziert werden:

- $M' \subseteq M$ und $N' \subseteq N$ sind *korrespondierende* Strukturen, wenn ihre Graphrepräsentationen $G_{M'} \subseteq G_M$ bzw. $G_{N'} \subseteq G_N$ bzgl. eines Ähnlichkeitsmaßes als ähnlich erkannt werden.
- Existiert für eine Teilstruktur keine korrespondierende Struktur in der jeweils anderen Modellvariante, so gilt sie als *variantenspezifisch*.

Sollen Änderungen im offiziellen Release des Modells in die Variante übernommen werden, muss das Variantenmanagement die Änderungen einpflegen. Das Zusammenführen wird auf den Graphrepräsentationen ausgeführt und lässt sich in drei Phasen unterteilen:

1. Erkennung vorhandener und entfernter Teile des Ursprungsmodells: Über das Mapping auf eindeutige Knotenbezeichner können in beiden Graphen vorhandene und dem Ursprungsmodell entstammende Teile identifiziert werden. Dazu können effiziente Verfahren, wie in [DBDK04] beschrieben, eingesetzt werden (s. Abschnitt 4.2).
2. Ergänzte, korrespondierende Anteile der Modellvarianten können nicht über Knotenbezeichner ermittelt werden, ebenso wie Anteile, für die die Zuordnung der modellinternen Bezeichner auf eindeutige Knotenbezeichner nicht mehr korrekt ist. Deswegen bieten sich Verfahren wie [MGMR02] an, die strukturbasierte Ähnlichkeiten erkennen.
3. Einfügen variantenspezifischer Strukturen: Variantenspezifische Strukturen besitzen keine korrespondierenden Strukturen und können einfach übernommen werden.

Für alle Schritte gilt, dass der Benutzer mit manuellen Entscheidungen eingreifen können muss. Hier ist eine Lenkung des Benutzers durch die notwendigen Entscheidungen erforderlich, bspw. durch zeitliche Nähe oder Bündelung notwendig. Bisherige Ansätze bieten hierzu keine Lösung. Elementare Änderungsoperationen auf den Graphen der Modelle (Knoten löschen/einfügen, Attribute ändern, etc.) können bspw. zu anwendungsspezifischen und semantisch höher liegenden Operationen zusammengefasst werden (bspw. das Einfügen einer Klasse mit Attributen und Methoden in UML-Klassendiagrammen) und bei Konflikten als solche betrachtet werden. Zur Beschreibung bieten sich Graphtransformationen an (s. bspw. [Roz97]).

6 Zusammenfassung und Ausblick

In diesem Artikel werden Ansätze für ein modellorientiertes Variantenmanagement vorgestellt, das die verteilte Entwicklung von Modellen durch entsprechende Verwaltungsoperationen wie bspw. das Zusammenführen von Varianten unterstützen soll. Insbesondere das Zusammenführen von Modellen mit formaler Syntax wird betrachtet. Graphen als zugrunde liegende Struktur der Syntax von Modellen sind eine geeignete Grundlage zur Realisierung der gewünschten Funktionalität.

Nachdem in diesem Artikel ein Reihe von Ansätzen zur Realisierung eines modellorientierten Variantenmanagements erläutert wurden, sollen diese in Zukunft detailliert ausgearbeitet, in einem System implementiert und während der verteilten Weiterentwicklung und Anpassung des V-Modell XT erprobt werden.

Literatur

[AAAN+05] Marwan Abi-Antoun, Jonathan Aldrich, Nagi Nahas, Bradley Schmerl und David Garlan. Differencing and Merging of Architectural Views. Bericht, Institute for Software Research International School of Computer Science Carnegie Mellon University, 2005.

[AP03] Marcus Alanen und Ivan Porres. Difference and Union of Models. Bericht, TUCS Turku Centre for Computer Science, 2003.

[Ced93] P. Cederqvist. *CVS Manual: Version Management with CVS*. Signum Support, 1993.

[DBDK04] Peter J. Dickinson, Horst Bunke, Arek Dadej und Miro Kraetzl. Matching graphs with unique node labels. *Pattern Analysis & Applications*, 7(3):243–254, 2004.

[Fel69] A.B. Fellegi, Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64:1163–1210, 1969.

[Gro] OMG (Object Management Group). Model Driven Architecture.

[IBM03] IBM. *Rational Unified Process, Version 2003.06.13*. 2003. http://www-306.ibm.com/software/awdtools/rup.

[KMS+05] TU Kaiserslautern, TU München, 4Soft, EADS, IABG und SIEMENS. *V-Modell XT Release 1.1*. Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, 2005. http://www.v-modell-xt.de.

[KWN05] Udo Keller, Jürgen Wehren und Jörg Niere. A Generic Difference Algorithm for UML Models. In *Software Engineering 2005*, Seiten 105–116, 2005.

[MGMR02] Sergey Melnik, Hector Garcia-Molina und Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *18th Intl. Conference on Data Engineering (ICDE)*, 2002.

[MRS06] Michael Meisinger, Andreas Rausch und Marc Sihling. 4everedit Team-Based Process Documentation Management. *Software Process: Improvement and Practice*, erscheint 2006.

[Roz97] Grzegorz Rozenberg, Hrsg. *Handbook of Graph Grammars and Computing by Graph Transformation, Foundations*, Jgg. 1. World Scientific, 1997.

[Sch91] Andy Schürr. *Operationales Spezifizieren mit programmierten Graphersetzungssystemen*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 1991.

[ZWR01] Albert Zündorf, Jörg P. Wadsack und Ingo Rockel. Merging graph-like object structures. In *Proceedings of the Tenth Workshop on Software Configuration Management*, 2001.