

Differenzanalyse und Vereinigung von Modellen auf der Basis ihrer Metamodelle

- Positionspapier der ikv++ technologies ag zum VVUM '07 Workshop -

Einleitung

Modellgetriebene Softwareentwicklungstechnologien vereinfachen und verbessern viele Aspekte der Softwareentwicklung. So können nun Prozesse automatisiert werden, die zuvor nur in zeitintensiver und fehleranfälliger Handarbeit durchzuführen waren. Zwischen den Phasen von Entwicklungsprozessen können Modelltransformatoren den Output einer Phase automatisiert in den Input der nächsten Phase überführen, die automatische Erzeugung von Programmcode direkt aus detaillierten Implementierungsmodellen rationalisiert die Programmierarbeit. Dem gegenüber entstehen aber neue Probleme und Herausforderungen.

Ein Kernaspekt bei der Realisierung großer Softwareprojekte ist nach wie vor die parallele Entwicklung im Team. Traditionelle Entwicklungsumgebungen unterstützen die verteilte Erstellung von Programmcode sowie dessen spätere Zusammenführung (*Merge*). In der modellgetriebenen Softwareentwicklung müssen nun auch die Werkzeuge zur Modellierung diese Möglichkeiten paralleler Entwicklung bieten. Während das Zusammenführen von Code jedoch durch relativ einfache Textvergleiche und das Kopieren von Zeichenketten realisiert werden kann, sind der Vergleich von Modellen und die Vereinigung mehrerer Modelle zu einem neuen Zielmodell ungleich komplexer.

Hinzu kommt die Problematik, dass Modelltransformatoren Zielmodelle erzeugen, die von Entwicklern weiter verfeinert und detailliert werden. Werden nun nachfolgend die Quellmodelle von Modelltransformatoren verändert und die Transformatoren erneut benutzt, so können die Zielmodelle nicht einfach überschrieben werden, sondern müssen mit den bereits geleisteten Verfeinerungsarbeiten an der vorherigen Version des Zielmodells konsistent zusammengeführt werden.

Die ikv++ technologies ag ist in ihren Aktivitäten auf dem Gebiet der modellbasierten Optimierung und Automatisierung von Systementwicklungsprozessen den oben genannten Herausforderungen früh begegnet. Dementsprechend hat sie eine Technologie zur Versionierung von Elementen von Modellen, zur Analyse von Unterschieden zwischen Modellen und zum automatisierten Zusammenführen von unterschiedlichen Modellen entwickelt. Vor der Einführung dieser Technologie wurde in den Entwicklungsteams unserer Kunden ein manueller Prozess zur Differenzanalyse und zum Zusammenführen von Entwicklungsartefakten (Modellen, Dokumenten und Code) praktiziert. Da bei Modellen ein Zusammenführen auf der Grundlage von Textvergleichen nicht möglich war, haben sich die Teammitglieder in zeitaufwendigen Review-Treffen zusammengesetzt und Element für Element der zu synchronisierenden Artefakte verglichen, Unterschiede diskutiert und nachfolgend manuell neue Versionen erstellt. Die von der ikv++ technologies ag entwickelte Technologie basiert auf der Definition einer Modellierungssprache, dem sog. Metamodell. So können beispielsweise UML-Modelle miteinander auf der Basis des UML-Metamodells und kundenspezifische Modelle auf der Basis des kundenspezifischen Metamodells verglichen und zusammengeführt werden. Diese Technologie wird bereits zur Unterstützung von Teamarbeit, zum automatisierten Zusammenführen von Modellen nach Modelltransformationen sowie für das Importieren von Modellteilen in andere Modelle praktisch in Kundenprojekten eingesetzt¹.

Die folgenden Abschnitte beschreiben zunächst kurz die allgemeinen Aspekte der verteilten Modellbearbeitung. Im Anschluss daran wird der von der ikv++ technologies ag entwickelte Ansatz

¹ Unter http://www.ikv.de/index.php?option=com_content&task=view&id=49&Itemid=74&limit=1&limitstart=1 sind zwei Demonstrationsvideos verfügbar, die die hier beschriebene Technologie zum Modellvergleich und zum Zusammenführen von Modellen am Beispiel von *medini component modeler* näher bringen.

am Beispiel eines exemplarischen Werkzeugs zur Differenzanalyse und zur Vereinigung von Modellen erläutert. Die beschriebene Technologie ist Teil der *medini Basistechnologie* der ikv.

Teamwork und modellgetriebene Softwareentwicklung

Arbeiten innerhalb eines Entwicklungsprojekts mehrere Modellierer an den gleichen Modellen, so muss deren Arbeit zu verschiedenen Zeitpunkten der Projektdurchführung synchronisiert werden. Weiterhin kann es notwendig sein, extern entwickelte Modelle zu importieren und mit existierenden Modellen zu vereinigen. Neben den manuell erstellten Modellen spielen auch aus Modelltransformationen entstandene Modelle eine immer wichtigere Rolle – auch für sie muss eine Lösung hinsichtlich des Vergleichs und der Vereinigung gefunden werden.

Der Prozess des Zusammenführens zweier verschiedener Modellversionen ist ohne Werkzeugunterstützung sehr zeitraubend und komplex. Man kann ihn grob in drei Schritte unterteilen: Zuerst müssen die Unterschiede zwischen den beteiligten Modellen festgestellt werden, einschließlich eventueller Konflikte. Dafür müssen die Modelle aus dem Dateisystem oder einer eventuell vorhandenen Versionsverwaltung importiert werden. Im Anschluss daran ist ein Zielmodell zu erzeugen, welches alle benötigten Elemente der Ursprungsmodelle enthält und dabei die aufgetretenen Konflikte bestmöglich löst. Dieses Zielmodell wird im letzten Schritt wieder dem Dateisystem bzw. der Versionsverwaltung hinzugefügt, von wo aus es weiterbearbeitet werden kann.

Diesen Prozess gilt es so weit wie möglich zu vereinfachen und zu automatisieren. Entsprechend entwickelte die ikv++ ein automatisiertes Verfahren, bei dem man mit Hilfe eines Werkzeugs zur Differenzanalyse und Vereinigung von Modellen die Ursprungsmodelle auf zu vereinigende Elemente überprüfen, eventuelle Konflikte lösen und schließlich das Ergebnis automatisch in einem neuen Zielmodell speichern kann. Dieses Verfahren basiert zum einen auf der *Definition von Modelldifferenzen* im Kontext von Metamodellen, zum anderen auf einigen *Grundvoraussetzungen*. Sie werden im Folgenden erläutert.

Definition von Modelldifferenzen

Um Modelle sinnvoll vergleichen zu können, ist eine entsprechende Definition des Begriffs "Differenz" nötig. Im gegebenen Kontext gelten folgende Definitionen:

- Die kleinsten als Differenz zu betrachtenden Einheiten sind die Eigenschaften der Objekte (Attributwerte und Referenzen auf andere Objekte, wie im MOF Standard² beschrieben). Zwei Objekte sind verschieden, wenn deren Metaobjekt als Element des Metamodells der Objekte eine Eigenschaft (Property) definiert und die Objekte bzgl. dieser Eigenschaft unterschiedliche Werte enthalten oder unterschiedliche Vielfachheiten erklären (bei *multi-value* Attributen), oder wenn sie auf unterschiedliche Objekte verweisen (bei Eigenschaften vom Typ "classifier").
- Ein Metaobjekt (Element des Metamodells) kann sog. Referenzen definieren, Elemente eines Modells, die durch das Metaobjekt definiert sind, verweisen dann auf andere Elemente des Modells. Zwei solche Referenzen sind unterschiedlich, wenn sie unterschiedliche Objekte referenzieren und/oder unterschiedliche Vielfachheiten und/oder im Falle von Listen die Elemente anderes geordnet sind.
- Zwei Modellelemente sind unterschiedlich, wenn ihre Eigenschaften (Attribute/Referenzen) unterschiedlich sind, oder wenn ihre eindeutige Identifikation³ (im Falle von *medini base* Technologie eine eindeutige UUID) unterschiedlich ist. (Beispiel: Zwei Pakete (in der UML Sichtweise) sind unterschiedlich, wenn sie unterschiedliche Objekte enthalten. (containment-Relation))

Basierend auf diesem intuitiv verständlichen Differenzkonzept kann das unten beschriebene *medini* System Modellunterschiede erkennen und auswerten.

² Siehe „Meta Object Facility Specification Version 1.4“ (Document ptc/02-04-03) unter <http://www.omg.org/docs>

³ Es sei angemerkt, dass der voll qualifizierte Name von Modellelementen nicht als Identifikation taugt, da einerseits nicht alle Metamodelle das Konzept von namentlich identifizierbaren Elementen beinhalten und andererseits für zwei Elemente mit gleichem qualifizierten Namen in unterschiedlichen Modellen nicht unmittelbar klar ist, ob es sich um das gleiche Element oder um einen Konflikt durch parallele Arbeit handelt.

Grundvoraussetzungen

In unserem Ansatz zum Modellvergleich gehen wir davon aus, dass zunächst eine initiale Version des Modells erzeugt und diese dann an die beteiligten Modellierer weitergegeben wird. Diese Vorgehensweise ist wichtig für die spätere erfolgreiche Vereinigung der Resultate. Grund dafür ist die oben genannte eindeutige Identifikation: Elemente des initialen Modells können in den nachfolgend getrennt modifizierten Modellen somit auf Basis ihrer identischen ursprünglichen UUIDs als gleich erkannt werden. Alle am Prozess beteiligten Modelle werden in einem versionierten *medini* Repository abgelegt. Dieses Repository unterstützt zum einen die Teamarbeit in verschiedenen Entwicklungszweigen. Zum anderen bietet es die entsprechenden Schnittstellen, um Modelle auszulesen und deren Elemente zu traversieren.

Eine große Rolle beim Ablauf des Vergleichsalgorithmus spielt auch das Wissen um das den Modellen zugrundeliegende Metamodell. So kennt die *medini* Technologie zum Modellvergleich und zur Modellzusammenführung das entsprechende Metamodell und „weiß“, dass es eine Hierarchie der Modellelemente voraussetzen kann, die im Metamodell durch ausgezeichnete Kompositionen kenntlich gemacht (z.B. die *Containment*-Relation im UML-Metamodell). Dadurch kann eine Baumstruktur zur optimalen Darstellung der Quell- und Zielmodelle gewählt werden. Ohne Metamodellkenntnis ist lediglich die Anzeige der Modellelemente in einer einfachen Liste möglich.

Modellvergleich und Modellvereinigung mit *medini*

Auf Grundlage der gegebenen Definitionen und Grundvoraussetzungen kann man nun mit Hilfe von *medini* den oben genannten dreistufigen Modellvergleichs- und Modellvereinigungsprozess durchführen. Das Werkzeug unterstützt dabei durch folgende Funktionen:

- Laden der beiden ausgewählten Modellversionen aus einem *medini* Repository und Umsetzung in eine interne Repräsentation
- Berechnung der Differenzen auf Basis der Differenzdefinition, Anzeige der vollständigen Modelle in einer Baumstruktur sowie Hervorhebung aller Differenzen
- Navigation der Modelle und Differenzen in beiden Modellversionen auf Basis der internen Repräsentation
- Funktionen zum Kopieren und Löschen von Modellelementen innerhalb der grafischen Oberfläche sowie Funktionen zur Markierung der im Folgenden zu vereinigenden Elemente
- Vereinigung der Modelle in einem Zielmodell und Speicherung im Versions- Repository

Die Funktionen, welche die drei Prozessschritte realisieren, sind im Folgenden noch etwas detaillierter beschrieben.

Schritt 1: Der Modell-Ladealgorithmus

Der von ikv++ realisierte Modell-Ladealgorithmus benutzt zum Einen eine Basistechnologie der zur Modellspeicherung verwendeten *medini* Repositories, das sogenannte Reflection-API. Zum Anderen nutzt er das Metamodell-spezifische Wissen hinsichtlich der Modellstruktur. Er läuft rekursiv über die Elemente der beiden zu vergleichenden Modelle und erzeugt dabei zunächst eine interne Baum-Repräsentation, die dieser Metamodell-spezifischen Struktur entspricht. Das Durchlaufen der Modelle ist Metamodell-spezifisch (auf der Grundlage der erwähnten ausgezeichneten Kompositionen) realisiert. Das ebenfalls notwendige Laden der Eigenschaften der Modellelemente erfolgt Metamodell-unabhängig über das Reflection-API der MOF-Schnittstelle. Ist dieser Prozess beendet, können die Modellelemente der zu vergleichenden Modelle sowie ihre Eigenschaften in einer Baumdarstellung visualisiert werden. Nach dem Laden der Modelle werden in einem zweiten Durchgang die Differenzen ermittelt. Durch sein Wissen um das zugehörige Metamodell kann dieser sogenannte „Diff-Walker“ die Hierarchie der Elemente nutzen. Er beginnt mit dem Wurzel-Element und navigiert über die entsprechenden Metamodell-Relationen (wie beispielsweise die vorhandenen Containment-Relationen entsprechend der im Metamodell ausgezeichneten Komposition) von Element zu Element. Dabei kommt der Vorteil des Elementvergleichs basierend auf der eindeutigen Identifikationen per Element zum Tragen. Der Algorithmus kann anhand dieser Identifikationen herausfinden, welche Elemente in beiden Modellen vorkommen und welche in nur einem Modell vorhanden sind. Weiterhin kann nun bestimmt werden, ob die Eigenschaften zweier Elemente mit

gleicher Identifikation unterschiedlich sind. Das Erkennen der Differenzen folgt der oben genannten Definition von Differenzen.

Schritt 2: Festlegen von zu vereinigenden Elementen

In diesem Schritt müssen für alle gefundenen Differenzen Entscheidungen hinsichtlich ihrer Behandlung getroffen werden. Durch die Entscheidungen für das Auflösen der Differenzen entsteht dann im Schritt 3 das eigentliche Zielmodell des Vereinigungsprozesses. Dabei können diejenigen Elemente des Quell-Modells, die nicht Bestandteil des Zielmodells sind (UUID nicht vorhanden), übernommen werden. Umgekehrt können neue Elemente im Zielmodell beibehalten oder gelöscht werden. Für unterschiedliche Elemente mit identischer Identifikation (gleicher UUID) sind in gleicher Weise die einzelnen Attribute/Referenzen zu behandeln. Dadurch ist für jedes Element genau definierbar, ob und in welcher Form es im vereinigten Modell enthalten sein soll. Während dieses Konfliktauflösungsprozesses sorgen funktionale Restriktionen dafür, dass nur die Änderungen betrachtet werden, nicht aber die unveränderten Elemente. Weiterhin wird die Konsistenz des Zielmodells durchgehend gewährleistet. So können beispielsweise Assoziationen nur ins Zielmodell übernommen werden, wenn auch die entsprechend referenzierten Modellelemente Teil des Zielmodells sind. Gleiches gilt für aus Modelltransformationen resultierende „Traces“: Nur wenn ihre Referenzen gültig sind, werden sie ins Zielmodell übernommen.

Schritt 3: Automatische Modellvereinigung und Speicherung des Zielmodells

Im letzten Schritt wird nun automatisch auf der Grundlage der in Schritt 2 getroffenen Entscheidungen die tatsächliche Vereinigung der Modelle durchgeführt. Der Speicherungsalgorithmus durchläuft rekursiv die internen Repräsentationen der Modelle und führt die vorgemerkten Vereinigungsoperationen durch, also das Entfernen, Hinzufügen oder Überschreiben von Elementen. Das Ergebnis wird automatisch als neue Modellversion im versionierten Repository oder dem Dateisystem abgelegt.

Der Vergleich von Modellelementen basierend auf ihren UUID bildet eine solide Grundlage für Modellvereinigungen. Allerdings greift dieser Mechanismus nicht bei Modellen, die aus Transformationen entstanden sind. Die entsprechenden durch einen Transformator erzeugten Modellelemente erhalten dabei neue eindeutige Identifikationen. Dadurch können zwei aus Transformationen entstandene Modelle nicht mehr anhand dieser Identifikationen verglichen werden (sie sind vollständig verschieden), selbst wenn sie aus Modellen transformiert wurden, welche auf dem gleichen Ursprungsmodell basieren. Die Lösung dieses Problems ist eine interne Vereinheitlichungsfunktion. Mit ihrer Hilfe können zwei bezüglich ihrer UUID unterschiedliche Elemente im Quell- und Zielmodell vereinheitlicht werden - sie bekommen die gleiche Identifikation. Dadurch sind diese Elemente nun bezüglich ihrer Eigenschaften vergleichbar. Die in medini realisierten Modelltransformatoren nutzen die Vereinheitlichungsfunktion automatisch (basierend auf gespeicherten Trace-Beziehungen), so dass hier keine zusätzliche manuelle Arbeit entsteht.

Fazit

Die von der ikv++ technologies ag entwickelte Lösung eignet sich besonders gut zum Vergleich von Modellen, welche auf einem gemeinsamen Vorgängermodell basieren. Durch Kenntnis des zugrundeliegenden Metamodells können die beteiligten Modelle anschaulich (Baumstruktur) dargestellt werden, was die nicht automatisierbaren manuellen Entscheidungsprozesse stark vereinfacht. Generell bietet die beschriebene Basistechnologie zum Modellvergleich und zum Zusammenführen von Modellen eine Reihe von weiteren Anwendungsmöglichkeiten. So kann die Technologie parametrisiert benutzt werden, so dass z.B. Änderungen im Quellmodell einer Transformation immer die mit Trace-Beziehungen verknüpften Elemente des Zielmodells überschreiben, oder dass Änderungen am Zielmodell sich auch per Transformation auf die entsprechenden Elemente eines Quellmodells auswirken sollen. Weiterhin hat die ikv++ technologies ag auf der Grundlage dieser Technologie einen Import-Mechanismus für Modelle entwickelt, der auf einem Zusammenführen des importierten Modells in das importierende Modell basiert. Dieser Import-Mechanismus „merkt“ sich die Identifikationen aller importierten Elemente, so dass eine Änderung des importierten Modells automatisch in das importierende Modell übernommen werden können. Ein zukünftiger Schwerpunkt ist die Visualisierung von Differenzen für Modelle mit graphischer Notation zusätzlich zu der Metamodell-basierten Baumstruktur.