Informationsgewinnung aus Modellhistorien

Positionspapier zum VVUM'07
Sven Wenzel
Praktische Informatik, Universität Siegen
wenzel@informatik.uni-siegen.de

1 Motivation

Die moderne Softwareentwicklung ist ohne die Hilfe von Konfigurationsmanagement (SKM) kaum noch vorstellbar. Zum einen werden die Softwareprojekte immer umfangreicher und erfordern die Zusammenarbeit von Entwicklern, die oftmals von verschiedenen Standorten aus aggieren. Zum anderen werden die Anforderungen der Kunden immer individueller, sodass mehrere Varianten einer Software parallel entwickelt werden.

Eine Vielzahl frei verfügbarer Versionsmanagementsysteme wie CVS oder Subversion, aber auch kommerzielle Produkte wie Microsoft Visual SourceSafe, bieten Unterstützung bei Versionierung, zentraler Speicherung und Variantenbildung von Softwaredokumenten. Textuelle Dokumente (z.B. Quellcode) können dabei mit Hilfe integrierter Werkzeuge problemlos verglichen und gemischt werden. Für grafische Dokumente (z.B. Analyse- oder Entwurfsmodelle) werden zusätzliche Werkzeuge benötigt, da die physikalische Repräsentation der Modelle zur Differenzberechnung ungeeignet ist [1, 2]. Differenzwerkzeuge wie *SiDiff* [3, 4] können diese Dokumente auf der Modellebene vergleichen. Sie erlauben damit nicht nur ein Versionsmanagement für Modelldokumente, sondern unterstützen insbesondere auch den aktuellen Trend modellgetriebener Softwareentwicklung.

In der modellgetriebenen Softwareentwicklung (MDSD), die in diesem Papier unterstellt wird, wird Software durchgängig mit Hilfe von Modellen beschrieben. Die Modelle sind eine abstrakte Darstellung der Software. Sie sind auch für Nicht-Entwickler verständlich und werden schrittweise in immer konkretere Modelle und letztlich ausführbare Modelle oder Programmcode transformiert.

Dieser Ansatz ergänzt sich gut mit dem stetigen Zuwachs von Einsatzbereichen der Software und individuellen Kundenanforderungen. Hierzu kann ein großer Teil der abstrakten Modelle wiederverwendet werden. Die konkreteren Modelle variieren dann in parallelen Entwicklungszweigen. So entstehen im Laufe der Zeit verschiedene Modelle, die weitgehend unabhängig voneinander weiterentwickelt werden, obwohl sie einen gemeinsamen Ursprung haben.

2 Modelllandschaften

Der gemeinsame Ursprung der verschiedenen Modelle wird oft vernachlässigt oder gar nicht dokumentiert. Damit sind die Zusammenhänge, die zwischen den einzelnen Entwicklungszweigen der Modelle bestehen, oft nicht bekannt. Spätestens wenn die Entwicklerteams wechseln, verschwindet damit auch das implizite Wissen über die Modellzusammenhänge.

Neue Entwickler oder Außenstehende stehen damit meist vor einer gewachsenen Modelllandschaft, die zunächst nicht vollständig überblickt werden kann. Aus dieser Wissenslücke über die einzelnen Modelle und ihre Zusammenhänge resultieren u.a. folgende Fragestellungen und Probleme:

- Welche Modellbestandteile sind von zentraler Bedeutung? Welche spielen nur eine untergeordnete Rolle?
- Was wird in den einzelnen Modellen überhaupt modelliert? Wie sind die Modelle genau zu verstehen?
- Enthalten die verschiedenen Entwicklungszweige Dubletten, die zu weiteren Wartungsproblemen führen?
- Wie können Erweiterungen eines Entwicklungszweiges auf andere Entwicklungszweige oder das Hauptprodukt übertragen werden?
- Sind Fehlerkorrekturen, die in einem Modell eines Entwicklungszweiges durchgeführt werden, auch auf andere Modelle oder Zweige zu übertragen?

Die Situation ist damit sehr ähnlich zu der des Reverse-Engineering, wo man zunächst vor einem relativ unbekannten Softwaregebilde steht und ähnliche Fragen zu beantworten sind.

3 Einbeziehung der Evolution

Eine wesentliche Ursache für die o.g. Probleme ist, dass die Historie einzelner Modelle nicht oder nur teilweise nachverfolgbar ist. Ein Blick in die Versionsmanagementsysteme zeigt, dass die meisten Modelle eine Vorgeschichte haben.

Eine Nachverfolgung der Modellevolution, wie sie derzeit in [5] erarbeitet wird, gibt erste Informationen über Modellzusammenhänge, indem paarweise Differenzen zwischen Vorgänger- und Nachfolgermodellen berechnet werden.

So kann z.B. für einzelne Elemente zweier Modelle in unterschiedlichen Entwicklungszweigen ein gemeinsamer Vorgänger gefunden werden, während ein direkter Vergleich der Modelle keinen Zusammenhang erkannt hätte. Damit wäre also ein Drei-Wege-Vergleich über mehrere Versionen hinweg möglich.

4 Weitere Informationsgewinnung

Die Berechnung der evolutionären Zusammenhänge zwischen den Modellen und Entwicklungszweigen führt zu aufbereiteten Modellhistorien. Die verschiedenen Modelle sind also nicht nur zeitlich geordnet[1], sondern sie erlauben auch die Verfolgung der einzelnen Modellelemente.

Die Modellhistorien können nun hinsichtlich verschiedenster Informationen untersucht werden. Um die in Abschnitt 2 genannten Fragen zu beantworten, lassen sich die aus dem Reverse-Engineering bekannten Verfahren auch auf die Modellwelt übertragen.

In [6] wurden polymetrische Sichten vorgestellt, um einen Überblick über ein Softwaresystem zu bieten. So können auf Basis grafisch aufbereiteter Metriken einflussreiche Systemkomponenten ermittelt werden. Die Metrikberechnung lässt sich auch auf Modelle übertragen und erlaubt unter Hinzunahme der Historieninformation z.B. die Ermittlung änderungsintensiver Modelleelemente. Geeignete polymetrische Sichten für Modelldifferenzen werden derzeit in [7] entwickelt.

Ein besseres Modellverständnis kann durch Erkennung und Annotation von Mustern erreicht werden. In [8, 9] werden verschiedene Ansätze zur Mustererkennung verglichen und ein Ansatz zur Erkennung von unvollständigen Musterinstanzen in Modellen vorgestellt. Die hier gewonnene Information wird in der Regel herangezogen, um die Modelle sowie das Zusammenspiel ihrer Elemente besser zu verstehen.

Die Suche von Dubletten wird im Reverse-Engineering bereits länger untersucht. [10] stellt einige der Ansätze vor, die sich in der Regel auf die Suche nach dupliziertem Quellcode konzentrieren. Die grundlegenden Konzepte sollten sich jedoch auch auf Modelldokumente übertragen lassen, um duplizierte Modellartefakte zu identifizieren.

Der Zusammenhang zwischen verschiedenen Modellen, Modellartefakten und Entwicklungszweigen wird in der sog. Kopplungserkennung untersucht. Dabei werden Verwaltungsinformationen der Versionsmanagementsysteme analysiert, ob z.B. mehrere Fragmente einer Software immer gemeinsam geändert werden. Hieraus lässt sich dann ein Zusammenhang zwischen diesen Fragmenten ableiten [11]. Dieses Verfahren lässt sich ebenfalls auf die Modellwelt übertragen, wie aktuell in [5] untersucht wird.

5 Ausblick

Wie bereits aufgezeigt, lassen sich viele der aus dem Reverse-Engineering bekannten Verfahren auf Modellhistorien übertragen. Ihre Anwendung ermöglicht einen hohen Informationsgewinn aus vorliegenden Modellhistoren. Diese Informationen können einerseits genutzt werden, um eine vorliegende Historie besser zu verstehen, sie bieten aber auch viele Hinweise, die eine Weiterentwicklung der Modelllandschaft unterstützen.

Ein wesentliches Ziel ist es, die gewonnenen Informationen so aufzubereiten, dass die effiziente (Weiter-)Entwicklung der verschiedenen Modelle durch geeignete Werkzeuge untersützt werden kann. Die Werkzeuge könnten dem Entwickler beim Bearbeiten eines Modells den Kontext zu anderen Modellen aufzeigen und damit eine Vielzahl neuer Möglichkeiten anbieten, wie z.B. die Mischung von parallelen Entwicklungszweigen.

[1] Die zeitliche Ordnung ist schon durch das Versionsmanagement gegeben.

Literatur

[1] Kelter, U.: Dokumentdifferenzen; Lehrmodul, Universität Siegen, `http://pi.informatik.uni-siegen.de/kelter/lehre/lm/dif` ; 2006

[2] Ohst, D.: Versionierungskonzepte mit Unterstützung für Differenz- und Mischwerkzeuge; Dissertation, Universität Siegen; 2004

[3] Kelter, U.; Wehren, J.; Niere, J.: A Generic Difference Algorithm for UML Models; Software Engineering 2005; 2005

[4] Fachgruppe Praktische Informatik, Universität Siegen: SiDiff – Siegener Differenzwerkzeuge; `www.sidiff.org`; 2007

[5] Hutter, H.: Nachverfolgbarkeit von Modellelementen in Versionshistorien (vorl. Arbeitstitel); Diplomarbeit (in Arbeit), Universität Siegen; 2007

[6] Lanza, M.: Object-Oriented Reverse Engineering - Coarse-grained, Fine-grained, and Evolutionary Software Visualization; Dissertation, Universität Bern, Schweiz; 2003

[7] Falk, J.: Entwicklung von Differenzmetriken und deren Visualisierung (vorl. Arbeitstitel); Diplomarbeit (in Arbeit), Universität Siegen; 2007

[8] Wenzel, S.: An Approach to the Automatic Recognition of Design Forms; Diplomarbeit, Universität Dortmund; 2005

[9] Wenzel, S.: Automatic Detection of Incomplete Instances of Structural Patterns in UML Class Diagrams; Nordic Journal of Computing, 12(4); 2005.

[10] Van Rysselberghe, F.; Demeyer, S.: Evaluating Clone Detection Techniques; International Workshop on Evolution of Large Scale Industrial Applications (ELISA); 2003

[11] Zimmermann, T.; Weißgerber, P.; Diehl, S.; Zeller, A.: Mining Version Histories to Guide Software Changes; IEEE Transactions on Software Engineering, 31(6); 2005.