Tailoring großer Prozeßmodelle auf der Basis von MVP-L*

Jürgen Münch, Markus Schmitz, Martin Verlage

Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany
muench@informatik.uni-kl.de, verlage@iese.fhg.de

Zusammenfassung. *Projekthandbücher müssen an projektspezifische Ziele und Charakteristika angepaßt werden. Die bisher verfügbaren Techniken sind weder effektiv noch effizient. Dieses wird anhand zweier Projekthandbücher (IEEE Standard 1074 und V-Modell) dargestellt. Zum einen sind die angepaßten Projekthandbücher oft fehlerhaft, zum anderen sind die manuellen Änderungen nur mit einigem Aufwand durchzuführen. In diesem Papier werden Anforderungen an Tailoring-Mechanismen für formale Projekthandbücher, sogenannte Prozeßmodelle, aufgestellt. Für Ansätze zur Wiederverwendung von Software-Produkten wird gezeigt, daß sie nur bedingt für die Wiederverwendung von Prozeßmodellen geeignet sind. Es wird ein Ansatz vorgestellt, der ein effektives und effizientes Tailoring von Prozeßmodellen gestattet. Dabei sind die Prozeßmodelle in der Sprache MVP-L ausgedrückt. In einem Ausblick wird auf mögliche Erweiterungen des Ansatzes eingegangen.*

1 Einleitung

Viele der Probleme in Software-Projekten sind auf eine Mißachtung der Ziele (z. B.: Aufwandsvorgaben, Qualitätsanforderungen) und Charakteristika (z. B.: Domäne, Erfahrung der Entwickler) bei der Erstellung von Prozeßmodellen zurückzuführen [1]. Jedes Projekt ist durch das Zusammenkommen spezieller Ziele und Charakteristika einzigartig. Soll Erfahrung aus vorangegangenen Projekten in Form von Prozeßmodellen in die Projektplanung eingehen, so müssen diese Modelle angepaßt werden. Besonders gravierend wird dieser Aspekt bei der Anwendung von Standards zur SW-Entwicklung (z. B.: V-Modell). Solche Standards beschreiben in der Regel das Vorgehen für eine große Klasse von Software-Projekten, die oftmals verschiedene Ziele verfolgen und in unterschiedlichen Entwicklungskontexten durchgeführt werden.

Als erster Lösungsansatz einer Anpassung an spezifische Ziele und Charakteristika wurde in der Entwicklung „perfekter" Prozeßmodelle gesehen, die für jede Kombination von Projektzielen und Kontextcharakteristika geeignet sind. Heutige Ansätze verfolgen eine andere Richtung: Es werden Techniken zur Wiederverwendung und Anpassung von Prozeßmodellen entwickelt, mit denen Prozeßvarianten für konkrete Projektziele und

* Die Arbeit wurde durch die Deutsche Forschungsgemeinschaft im Rahmen des Teilprojekts B1 „Experimentell gestützte Modellierung von SE-Prozessen" des Sonderforschungsbereichs 501 („Entwicklung großer Systeme mit generischen Methoden") sowie durch die Fraunhofer-Einrichtung für Experimentelles Software Engineering (IESE) unterstützt.

-charakteristika erzeugt werden können (z. B.: Prozeßmodellgeneratoren [1]). Die erfolgreiche Realisierung solcher Techniken ist erheblich wahrscheinlicher als die Realisierung „perfekter" Prozeßmodelle. Als Input verwenden solche Techniken Projektziele und -charakteristika sowie evtl. generische Prozeßmodelle, um ein projektspezifisch angepaßtes Prozeßmodell zu produzieren.

Das Anpassen von Prozeßmodellen an Projektziele- und charakteristika wird als Tailoring bezeichnet [8]. Prozeßmodelle müssen in geeigneter Form für die Wiederverwendung zur Verfügung stehen und für verschiedene Kontexte parametrisiert sein. Voraussetzung ist, daß die Einflußfaktoren, d. h. die Parameter in Form von Zielen und Charakteristika, bekannt sind. Dieses Papier beschäftigt sich mit dem Tailoring vor Projektstart[1], bei dem ein Life-Cycle-Rahmen sowie ausgewählte Methoden für ein Projekt vorgegeben werden. Hiervon wird das Tailoring während der Projektausführung unterschieden[2], bei dem notwendige Anpassungen und Veränderungen der Prozesse dynamisch durchgeführt werden (z. B. bei Umplanungen).

Neuere Standards wie beispielsweise MIL-STD-498 oder das Vorgehensmodell erkennen die Notwendigkeit zum Tailoring an und beinhalten entsprechende Richtlinien. Diese Richtlinien sind informell und stark vereinfacht. Die genannten Standards beschreiben ein allgemeines Vorgehen zur Erstellung von Software und beschränken die Anpassung auf das Entfernen unnötiger Teilschritte und Produkte. Solche Anpassungen müssen momentan manuell erfolgen und sind sehr aufwendig und fehleranfällig.

Ziel des hier vorgestellten Ansatzes ist die effektive und effiziente Bereitstellung kontext-orientierter Prozeßmodelle. Dabei werden Anpassungen auf der Basis formaler Prozeßmodelle einer Prozeßmodellierungssprache durchgeführt. Hierbei können neben der Aufwandsreduktion und der Gewährleistung der Konsistenz bei der Anpassung folgende Vorteile gesehen werden [14]: 1) Formale Prozeßmodelle erleichtern die Erstellung und Modifikation konsistenter Software-Engineering-Standards. Vorteilhaft sind insbesondere die werkzeuggestützten Möglichkeiten der Modellierung und Konsistenzprüfung sowie die Möglichkeit der separaten Modellierung und Integration einzelner Sichten. 2) Formale Prozeßmodelle sind ein geeignetes Mittel zur Ablage von Software-Entwicklungswissen. Formale Prozeßmodelle können als initiale Wiederverwendungskandidaten einer Erfahrungsdatenbank verwendet werden [5]. Vorteilhaft ist, daß sie die relevanten Aspekte von Aktivitäten der realen Welt erfassen, verglichen mit informellen Beschreibungen einfach wartbar sind und daß sie in einer Erfahrungsdatenbank bezüglich verschiedener Zugriffsstrukturen abgelegt werden können (z. B.: Typhierarchien, domänenspezifische Prozeßmodell-Cluster). 3) Formale Prozeßmodelle erlauben die Anwendung hochentwickelter Analysemethoden (z. B.: Risikoanalysen vor Projektausführung). 4) Formale Prozeßmodelle sind die Basis für prozeßsensitive Software-Engineering-Umgebungen.

Der Rest des Papiers ist folgendermaßen aufgebaut: Kapitel 2 beschreibt Erfahrungen, die bei der Formalisierung von SE-Standards bezüglich des Tailorings gemacht wurden. Daraus werden Anforderungen an Tailoring-Mechanismen abgeleitet. In Kapitel 3 werden gängige Mechanismen der Wiederverwendung von Produkten bezüglich ihrer Eignung für die Wiederverwendung von Prozessen untersucht. Kapitel 4 beschreibt einen werkzeuggestützten Ansatz zum Tailoring von Prozeßmodellen auf der Basis einer Pro-

1. Im V-Modell als Ausschreibungsrelevantes Tailoring bezeichnet.
2. Im V-Modell als Technisches Tailoring bezeichnet.

zeßmodellierungssprache und Kapitel 5 gibt einen Ausblick auf Erweiterungen des Ansatzes.

2 Empirische Ableitung von Anforderungen

Die Anforderungen an Tailoring-Mechanismen wurden aus der Formulierung eines internationaler Standard (*IEEE Standard 1074 Developing Life Cycle Processes* [7]) und eines nationalen Standards (*Vorgehensmodell, kurz: V-Modell* [2]) gewonnen [11,13,14]. Beim V-Modell wurde für zwei konkrete Projekte das Tailoring durchgeführt. Als Prozeßmodellierungssprache wurde hierbei MVP-L [3] verwendet. MVP-L ist eine Sprache für die Prozeßmodellierung im Großen, d. h., es wird stärker auf die Beziehungen zwischen Objekten (Prozessen, Produkten, Ressourcen) eingegangen als auf deren Implementierung. Die Konzepte von MVP-L sind ausreichend für die Modellierung von realen Entwicklungsprozessen [9].

Bei der Formalisierung der Standards wurden Erfahrungen bezüglich folgender Aspekte gemacht: 1) Beschreibung des Standards, 2) Eignung der Modellierungssprache MVP-L für die Darstellung generischer Prozesse, 3) Tailoring-Mechanismen des Standards. Auf letzteres soll hier eingegangen werden, bezüglich der ersten beiden Punkte sei auf [13,14] verwiesen.

Das V-Modell beinhaltet ein Grundmodell und Tailoring-Richtlinien zur Erzeugung von Varianten. Das Tailoring erfolgt durch die Überprüfung von Bedingungen und entsprechendes Streichen von Objekten aus dem Grundmodell. Das V-Modell unterscheidet Ausschreibungsrelevantes Tailoring vor Projektstart und Technisches Tailoring, das zu vordefinierten festen Zeitpunkten im Life-Cycle während der Ausführung stattfinden kann. Das Tailoring der formalen Modelle hat gezeigt, daß das Entfernen von Produkten und Prozessen umfangreiche Änderungen des Grundmodells nach sich zieht (z. B.: Das Streichen von Produkten impliziert das Löschen eines Prozesses). Eine wesentliche Schwierigkeit beim Tailoring bestand darin, diese Folgeänderungen zu identifizieren, da sie nicht explizit in den Streichbedingungen angegeben sind. Den erforderlichen Anpassungsaufwand verdeutlicht das Beispiel eines Pflegeobjekts: Das notwendige Entfernen von 4 Produkten führt zu Folgeänderungen an der Produkthierarchie, dem Produktfluß und den diese Produkte konsumierenden oder produzierenden Prozessen. Letztlich sind 68 Änderungen an 26 Modellen (Produkten und Prozessen) durchzuführen. Mit dem hier vorgestellten Ansatz konnten diese Änderungen vollständig automatisiert werden.

Ein weiteres Problem beim Tailoring des V-Modells ist, daß alle Anpassungen als Streichungen formuliert sind, obwohl es sich in einigen Fällen eher um Ersetzungen handelt. Beispielsweise können die externen Vorgaben die Systemanforderungen ersetzen. Dies wird durch Streichen der Systemanforderungen realisiert, obwohl die Prozesse diese Anforderungen weiterhin als Eingabe benötigen. Ein intelligenter Ersetzungsmechanismus wäre hier hilfreich.

Der IEEE Standard beansprucht, alle Projekte mit einem Modell abzudecken und klassifiziert daher Elemente als obligatorisch („mandatory”) oder optional („if applicable”). Drei Klassen von optionalen Elementen existieren: optionale Prozesse, optionale Produkte und optionaler Produktfluß. Als Problem beim Tailoring hat sich erwiesen, daß Alternativen in verschiedenen Teilen des Standards nicht in Beziehung zueinander ste-

hen (z. B.: Der Prozeß „System Transition” ist bei der Installation nur sinnvoll, falls ein älteres System bereits existiert). Dies muß in den Anforderungen dokumentiert sein.

Zusammenfassend kann man sagen, daß die Standards bisher nur sehr einfache Unterstützung für das Tailoring bieten. Die besseren Möglichkeiten bietet das V-Modell. Es beschränkt sich jedoch auf Streichungen. Durch die Formalisierung der Standards mit MVP-L konnten Inkonsistenzen und Mehrdeutigkeiten gefunden und beseitigt werden. Beim Tailoring des formalisierten V-Modells war die Möglichkeit von Konsistenzchecks hilfreich. So konnten viele übersehene Folgeänderungen entdeckt werden. Letztlich erfordert manuelles Tailoring einen hohen Aufwand. Formalisierte Prozeßmodelle haben den Vorteil, daß sie semi-automatisch angepaßt werden können, wobei die Konsistenz der resultierenden Prozeßvarianten gewährleistet ist.

Basierend auf den oben beschriebenen Erfahrungen wurden folgende Anforderungen abgeleitet, die eine werkzeuggestützte Methodik zum Tailoring großer Prozeßmodelle erfüllen sollte:

1. Die Methodik soll aufgrund von Projektcharakteristika und -zielen sowie mit Hilfe von generischen Prozeßmodellen angepaßte Projektpläne erzeugen können.
2. Der Aufwand zur Anpassung eines Projektplans soll gegenüber der manuellen Anpassung erheblich reduziert werden.
3. Die Konsistenz der angepaßten Prozeßmodelle ist zu gewährleisten.
4. Die Anpassungen sind auf geeignete Art zu dokumentieren, so daß eine Analyse von Prozeßvarianten möglich ist.

3 Eignung existierender Wiederverwendungsansätze

Zwischen der Programmierung von Software und der Modellierung von Prozessen existieren starke Parallelen [10]. Deshalb werden hier wesentliche Konzepte zur Wiederverwendung aus der Produktwelt auf ihre Eignung für die Wiederverwendung von Prozeßmodellen untersucht.

	Wiederverwendungsansätze					
Prinzip	Komposition				Generierung	
Wiederverwendete Objekte	Bauteile				Muster	
Mechanismus	Bedingte Anweisung	(optionale) Parameter	Information Hiding	Vererbung	Generatoren	Transformationssysteme
Vorteile	Tailoring-Bedingung erfaßt	Anpassung erfolgt bei Instantiierung von Objekten	---	Integration der Objekte gewährleistet	Integration der Objekte gewährleistet, hoher Automatisierungsgrad, hohe Geschwindigkeit	Tailoring-Regeln sind explizit und änderbar
Nachteile	Integration der Objekte gefährdet	Tailoring-Bedingung nicht erfaßt	Prozeßschnittstellen variieren bei Anpassung	Tailoring-Bedingung nicht erfaßt	Tailoring-Regeln schwierig explizit darstellbar / änderbar	Automatisierung und Integration der Objekte eingeschränkt
Abgedeckte Anforderungen	1, 2	2	---	2, 3	1-4	1-4

Tabelle 1: Eignung existierender Produkt-Wiederverwendungsansätze

Hinsichtlich Wiederverwendung von Softwareprodukten werden zwei prinzipielle Ansätze unterschieden [4]:

1. Der *kompositionsbasierte Ansatz* versucht, durch Zusammenfügen von Bauteilen Gesamtsysteme zu erstellen.
2. Der *generierungsbasierte Ansatz* versucht, durch Einsatz von typischen Architekturen und Verwendungsmustern ein System zu erzeugen.

Typische Vertreter beider Ansätze wurden hinsichtlich ihrer Eignung für das Tailoring untersucht. Wesentliche Vor- und Nachteile sowie die Abdeckung der in Kapitel 2 beschriebenen Anforderungen durch die jeweiligen Mechanismen sind in Tabelle 1 zusammengefaßt. Als Tailoring-Bedingung wird hierbei diejenige Bedingung verstanden, die angibt, welche Anpassungsalternative ausgewählt werden soll. Eine Tailoring-Regel setzt sich aus einer Tailoring-Bedingung und entsprechenden Anpassungsoperationen zusammen. Außerdem wird in der Tabelle beschrieben, inwieweit die Integration der Objekte der angepaßten Prozeßmodellvariante gewährleistet werden kann, wenn der jeweilige Mechanismus zur Wiederverwendung verwendet wird.

Die kompositionsbasierten Ansätze erweisen sich als weniger geeignet für die Wiederverwendung von Prozeßmodellen, da nicht gleichzeitig die Bedingungen für Anpassungen in Form von Tailoring-Regeln dargestellt und die Integration der erzeugten Prozeßmodellvarianten gewährleistet werden kann.[1] Dies ist bei generierungsbasierten Ansätzen möglich.

4 Transformationsbasiertes Tailoring mit ProTail

Aufgrund einer detaillierten Analyse existierender Wiederverwendungsansätze, deren Ergebnisse im vorhergehenden Kapitel angedeutet sind, wird hier ein transformationsbasierter Tailoring-Ansatz verfolgt. Die Anpassung der Prozeßmodelle erfolgt hierbei durch eine Folge von Transformationsschritten, die mit der sequentiellen Anwendung von Tailoring-Regeln (hier in Form von Transformationsregeln) verglichen werden können. Der transformationsbasierte Tailoring-Ansatz bietet die Möglichkeit, Tailoring-Regeln explizit zu erfassen, so daß sie einfach an neue Erkenntnisse angepaßt werden können. Zur Unterstützung des gewählten Ansatzes wurde der Werkzeug-Prototyp *ProTail* (*Pro*cess *Tail*oring Tool) entwickelt (siehe Abb. 1).

Die in Kapitel 2 genannten Anforderungen wurden folgendermaßen konkretisiert:

Anforderung 1: ProTail erhält als Eingabe Ziele und Charakteristika des vorliegenden Projekts. Diese ändern sich von Projekt zu Projekt. Wesentlich stabiler sind das Grundmodell und die Transformationsregeln, die das Tailoring beschreiben. Beide sind jedoch konfigurierbar, so daß eine Anpassung bzw. Ersetzung für spezifische Organisationen und Domänen erfolgen kann. Das Grundmodell wird durch Anwendung der Transformationsregeln durch die Komponente „Transformer” zu einem angepaßten Projektplan transformiert.

1.Information Hiding ist für das Tailoring von Prozeßmodellen ein ungeeignetes Prinzip, da sich bei der Anpassung die Schnittstellen des Prozesses (z. B.: Produktfluß-, Kontrollflußschnittstelle) ändern.

Anforderung 2: Benutzer-Eingriffe werden nur erforderlich, falls eine Transformationsregel nicht automatisch durchgeführt werden kann.

Anforderung 3: Die Konsistenz der angepaßten Prozeßvariante wird angestrebt, indem Benutzer-Eingriffe minimiert, automatische Konsistenz-Prüfungen angeboten und Folgeänderungen automatisch berücksichtigt werden.

Anforderung 4: Alle Modifikationen des Grundmodells werden dokumentiert (einschließlich eventueller Folgeänderungen). Hierfür ist die Komponente „Documenter" zuständig.

Eine weitere Komponente „Backtracker" ist vorgesehen, mit der innerhalb eines Tailoring-Vorgangs einzelne Transformationen zurückgenommen werden können.

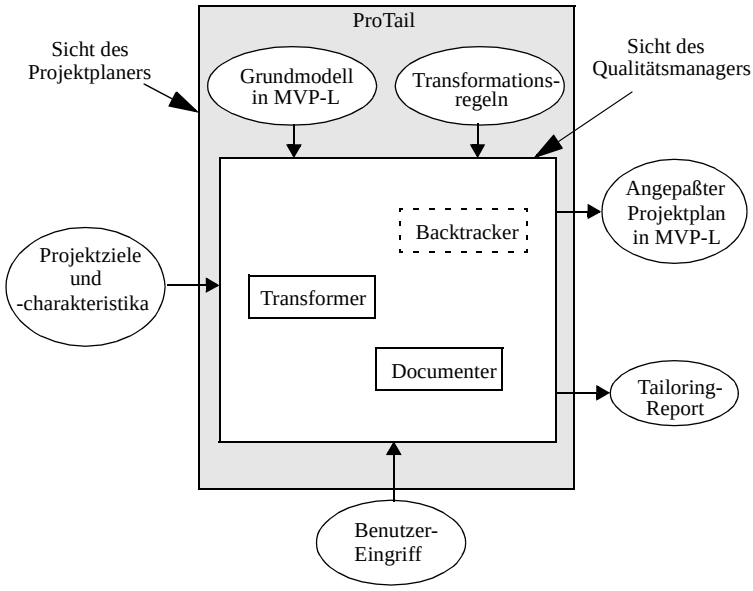

Abb. 1: Architektur von ProTail

ProTail unterstützt vorwiegend den Projektplaner und den Qualitätsmanager. Der *Projektplaner* ist an der effizienten Durchführung des Tailoring interessiert. Er betrachtet das Grundmodell und die Transformationsregeln als gegeben, charakterisiert das Projekt und läßt ProTail die Regeln automatisch anwenden. Der Projektplaner greift nur ein, wenn er vom System dazu aufgefordert wird. Nach einem Durchlauf liegt ein fertig angepaßter Projektplan vor. Abb. 2 zeigt die Benutzungsschnittstelle des Projektplaners.

Der *Qualitäts-Manager* ist für die Ablage, Verwaltung und Bereitstellung von Erfahrungen zuständig. Er erfaßt deshalb einen umfassenden Projektplan als Grundmodell und spezifiziert Tailoring-Regeln als Transformationsregeln in ProTail. Hierbei kann er folgendermaßen vorgehen:

1. Der Qualitätsmanager analysiert verschiedene Projektpläne einer Domäne (z. B.: Projektpläne zur Entwicklung von Realzeitsystemen) und identifiziert relevante Variationsparameter (z. B.: Kritikalität der Anwendung), die Einfluß auf den Software-Entwicklungsprozeß haben.

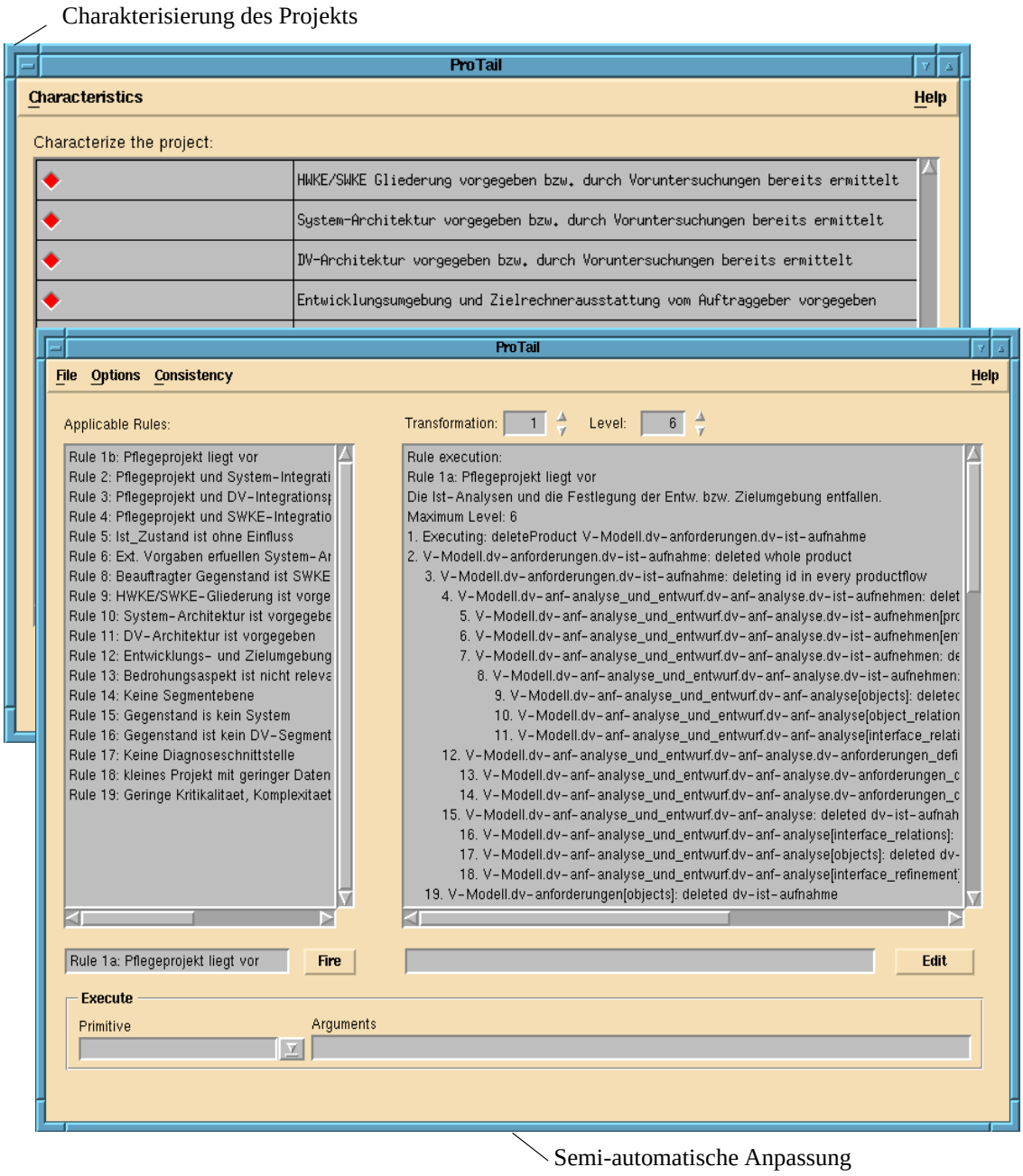

Abb. 2: Sicht des Projektplaners

2. Es wird ein (evtl. mehrstufiges) Charakterisierungsschema für die Variationsparameter der Domäne entworfen.
3. Aus den lokalisierten Gemeinsamkeiten und Unterschieden der untersuchten Projektpläne sowie dem Charakterisierungsschema, das die Projektpläne klassifiziert, wird ein Grundmodell entwickelt und in MVP-L modelliert. Dieses Grundmodell entspricht einem generischen Projektplan.
4. Anhand des Grundmodells und des Charakterisierungsschemas werden Tailoring-Regeln spezifiziert und als Transformationsregeln in ProTail implementiert.

5. Zur Evaluierung kann der Qualitätsmanager die Regeln semi-automatisch ausführen. Die Dokumentation vergleicht er mit seinen eigenen Erwartungen und verbessert in einem iterativen Prozeß Grundmodell und Transformationsregeln.

Abb. 3 gibt eine Übersicht der ersten 4 Schritte. Dabei wird zwischen der Projektorganisation, in der individuelle Software-Entwicklungsprojekte durchgeführt werden, und der Experience Factory, in der Erfahrungen aller Projekte gespeichert und aktiv aufbereitet werden, unterschieden. Dies entspricht dem wiederverwendungsorientierten Ansatz zur Software-Entwicklung nach Basili und Rombach [5].

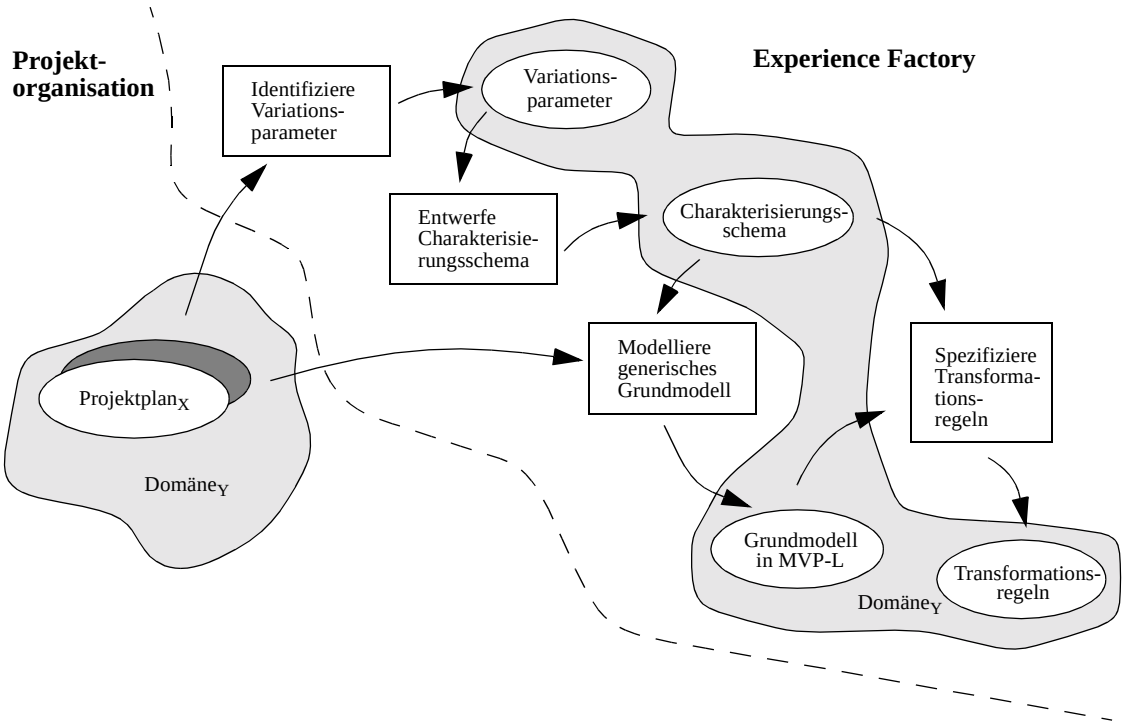

Abb. 3: Generische Auslegung von Prozeßmodellen

Bei der Entwicklung und Anwendung von ProTail wurden folgende erste Erfahrungen gemacht:

1. Die Durchführung von Anpassungen inklusive Erfassung aller Folgeänderungen ist sehr komplex und aufwendig zu implementieren. Der Grund hierfür liegt in der starken Vernetzung von Informationen in Prozeßmodellen. Sehr hilfreich ist in diesem Zusammenhang eine redundanzfreie interne Darstellung von Prozeßmodellen.
2. Als problematisch erweisen sich Tailoring-Regeln, deren Aktionen sich gegenseitig beeinflussen (z. B.: „Lösche Produkt x" und „Ersetze Produkt x durch y"). ProTail löst dieses Problem momentan, indem sich Regeln gegenseitig ausschalten können. Die Auswirkungen dieses Vorgehens bzw. andere Konflikt-Lösungsstrategien müssen weiter untersucht werden.

3. Der zur Anpassung eines Prozeßmodells nötige Aufwand kann drastisch reduziert werden. Erste Erfahrungen mit dem V-Modell zeigten eine mehr als 95%-ige Aufwandsreduzierung.

5 Ausblick

Derzeitige Tailoring-Ansätze beschränken sich auf das Erstellen von angepaßten Life-Cycle-Rahmen und evtl. die lose Einbindung von Methoden. Wünschenswert wären Mechanismen, mit denen einzelne Methoden, die in verschiedenen Phasen des Life-Cycles angewendet werden, geeignet in den angepaßten Life-Cycle-Rahmen integriert werden können. Derzeit erfolgt die Charakterisierung der Projektziele und des Kontexts anhand einfacher Schemata. Relevante Einflußfaktoren müssen für einzelne Domänen identifiziert und in geeigneten Schemata repräsentiert werden. Die derzeitige Situation ist durch einen Mangel an adäquaten Stilen für die Repräsentation von Prozeßmodellen gekennzeichnet. ProTail nimmt die Anpassungen aufgrund textueller Repräsentation von MVP-L-Modellen vor. Werkzeuge, mit denen Anpassungen in einer graphischen Repräsentation vorgenommen werden können, sind derzeit in Entwicklung. Angepaßte Prozeßmodelle können als unmittelbarer Input für eine Software-Entwicklungsumgebung verwendet werden. Im Rahmen des Sonderforschungsbereichs 501 wird derzeit die Prozeßunterstützungsumgebung MILOS [6,12] entwickelt, die durch Möglichkeiten zur dynamischen Umplanung Tailoring während der Projektdurchführung unterstützt. Durch Kombination von ProTail und MILOS kann eine Umgebung erreicht werden, in der die Wiederverwendung von Prozeßmodellen während der gesamten Lebensdauer eines Projekts von der ersten Planung bis zur Terminierung unterstützt wird.

Referenzen

[1] Barry Boehm, Frank Belz, „Experiences with the Spiral Model as a Process Model Generator", IEEE 1990.

[2] Adolf-Peter Bröhl, Wolfgang Dröschel, „Das V-Modell", Oldenbourg, 1995.

[3] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach, Martin Verlage, „MVP-L Language Report Version 2", Technischer Bericht Nr. 265/95, Universität Kaiserslautern, 1995.

[4] Ted J. Biggerstaff, Alan J. Perlis, „Software Reusability", ACM Press, Frontier Series, 1989.

[5] Victor R. Basili, H. Dieter Rombach, „Support for Comprehensive Reuse", IEEE Software Engineering Journal, 6(5): 303-316, September 1991.

[6] Barbara Dellen, Frank Maurer, Jürgen Münch, Martin Verlage: „Enriching Software Process Support by Knowledge-based Techniques", to be published in the special issue of Int. Journal of Software Engineering and Knowledge Engineering, 1997.

[7] Institute of Electrical and Electronics Engineers, „IEEE Standard for Developing Software Life Cycle Processes", IEEE Std. 1074-1991, 1992.

[8] Peter H. Feiler, Watts S. Humphrey, „Software Process Development and Enactment: Concepts and Definitions", SEI Technical Report CMU/SEI-92-TR-04, 1992.

[9] C. D. Klinger, M. Neviaser, A. Marmor-Squires, C. M. Lott, D. Rombach, „A Case Study in Process Representation using MVP-L", in Proceedings of the Seventh Annual Conference on Computer Assurance (COMPASS 92): 137-146, 1992.

[10] Leon Osterweil, „Software Processes are Software Too", Proceedings of the Ninth International Conference on Software Engineering, Monterey, CA, 1987.

[11] Martin Verlage, Christian Bunse, Peter Giese, Wolfram Petsch: „Three Approaches for Formalizing Informal Process Descriptions", in GI/GMA/IFIP/IFAC 5th International Workshop on Experience with the Management of Software Projects (MSP-95), Karlsruhe, Germany, September 27-29, 1995.

[12] Martin Verlage, Barbara Dellen, Frank Maurer, Jürgen Münch: „A Synthesis of Two Process Support Approaches", Proceedings of the Eighth International Conference on Software Engineering and Knowledge Engineering (SEKE'96), Lake Tahoe, Nevada, USA, June 1996.

[13] Martin Verlage: "Erfahrungen bei der Formalisierung von Projekthandbüchern", in Tagungsband Informatik '96 (GI/OCG-Jahrestagung), Klagenfurt, Österreich, September 25-27, 1996.

[14] Martin Verlage, Jürgen Münch: „Formalizing Software Engineering Standards", to be published in Proceedings of the Third International Symposium and Forum on Software Engineering Standards (ISESS'97), California, USA, June 1-6, 1997.

[15] H. Dieter Rombach, Martin Verlage, „Directions in Software Process Research", Advances in Computers, Volume 41, Marvin V. Zelkowitz (Ed.), Pages 1-63, Academic Press, Boston, MA, 1995.