Das Werkzeug MoST zur Entwicklung von Prozeßmodellen

Ulrike Becker[1] und Martin Verlage[1,2]

[1] Fraunhofer-Einrichtung für
Experimentelles Software Engineering (IESE)
Sauerwiesen 6
67661 Kaiserslautern
Tel. 06301 - 707 135
email: becker@iese.fhg.de

[2] Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern
Tel. 0631 - 205 3337
email: verlage@iese.fhg.de

Software-Entwicklungsaktivitäten werden durch Prozeßmodelle explizit erfaßt. Das *Modeling Support Tool* (MoST) verwaltet, analysiert und prüft Prozeßmodelle. Es wurde für die Prozeßmodellierungssprache MVP-L (multi-view process modeling language) entwickelt und soll Prozeßingenieure sowohl bei der deskriptiven Modellierung von Prozessen im Rahmen von Verbesserungsprogrammen unterstützen als auch die Entwicklung präskriptiver Prozeßmodelle (z.B. Projekthandbücher) erleichtern. Wesentliches Ziel bei der Entwicklung war die einfache Handhabung und inkrementelle Entwicklung von Prozeßmodellen. Dieses Papier stellt das Werkzeug MoST vor und berichtet über Erfahrungen, die beim Einsatz des Werkzeugs gemacht wurden.

1. Motivation

Die Entwicklung von Software folgt keinem statischen, wiederholbaren Prozeß. Unterschiede zwischen Organisationen und Projekten sowie die Einführung neuer Technologie erfordert Varianten und Versionen von Prozessen. Die Steigerung der Effizienz und Effektivität bei der Software-Entwicklung ist jedoch in starkem Maße davon abhängig, ob es gelingt, Erfahrungen aus durchgeführten Projekten erneut verwenden zu können. Es besteht ein Spannungsfeld zwischen der Notwendigkeit zur Anpassung und dem Wunsch nach Stabilität.

Als geeignete Träger von Wissen über Software-Entwicklungsprozesse haben sich Prozeßmodelle erwiesen. Prozeßmodelle sind explizite (formale) Beschreibungen von Software-Entwicklungsprozessen. Diese Modelle dienen der Kommunikation, Analyse, Verbesserung, Anleitung und automatischen Abwicklung. Verschiedene Beschreibungsformalismen, sogenannte Prozeßmodellierungssprachen, wurden in den letzten Jahren vor allem in universitären Umfeldern entwickelt, meist im Zusammenhang mit der Erforschung von Software-Entwicklungsumgebungen [17].

Die Modellierungen von Software-Entwicklungsprozessen soll von Spezialisten, sog. Prozeßingenieuren, durchgeführt werden. Entwickler oder organisatorische Rollen (z.B. Projektmanager) sollten ohne eine eingehende Schulung nicht eigenständig modellieren. Es hat sich jedoch gezeigt, daß Prozeßingenieure bei der Entwicklung von Prozeßmodel-

len nicht optimal unterstützt werden. Im Kontext einer konkreten Prozeßmodellierungssprache (multi-view process modeling language, oder kurz MVP-L) wurde eine erste Version eines Modellierungswerkzeugs speziell für den Prozeßingenieur entwickelt.

Dieser Beitrag berichtet über den Entwurf des Werkzeugs MoST (Modeling Support Tool) [11] für die Prozeßmodellierungssprache MVP-L und stellt die Verwendung von MoST anhand von zwei Szenarien dar. Bisherige Einsätze von MoST gaben wertvolle Rückmeldungen für zukünftige Entwicklungen.

Diese Arbeit ist wie folgt aufgebaut: Der zweite Abschnitt stellt kurz die Prozeßmodellierungssprache MVP-L vor. Abschnitt 3 diskutiert das Werkzeug MoST. Den Umgang mit MoST bei der deskriptiven und präskriptiven Modellierung illustriert Abschnitt 4. Über Erfahrungen berichtet Abschnitt 5, bevor Abschnitt 6 ausgehend vom aktuellen Stand zukünftige Arbeiten vorstellt.

2. Die Prozeßmodellierung in MVP-L

MVP-L ist eine text-basierte Prozeßmodellierungssprache. MVP-L unterscheidet *Prozesse* (Aktivitäten), *Produkte* und *Ressourcen*. Sie sind Objekte (Instanzen) von Grundtypen (Modelle genannt). Alle Prozesse, Produkte und Ressourcen können Attribute besitzen und in Subobjekte verfeinert werden. Die Prozesse als komplexeste Objekte sind darüberhinaus noch strukturiert in Produktschnittstelle, abwickelnde Ressourcen (Personen oder Werkzeuge), Kontextinformationen und Vor- und Nachbedingungen. Die Beziehungen *produziert, modifiziert* und *konsumiert* zwischen Prozessen und Produkten werden unter dem Begriff *Produktfluß* zusammengefaßt. In einem *Projektplan* kann man die Instanzierungen der Modelle in einem konkreten Projektkontext beschreiben und die Objekte mit Parametern versorgen.

Abbildung 1 gibt die wesentlichen Konzepte von MVP-L in OMT-Notation wieder. MVP-L verfügt bisher über keine Konzepte zur Formulierung von Typhierarchien. Die bisherigen Erfahrungen ließen Vererbungen zwischen Prozeßmodellen nicht für wichtig erscheinen. Oftmals existiert von einer Klasse nur ein Exemplar.

MVP-L ist eine Sprache für *process modeling in-the-large* [17], d.h. es wird stärker auf die Beziehungen zwischen Prozessen, Produkten und Ressourcen eingegangen als auf deren Semantik oder Implementierung. Die wesentlichen Beziehungen zwischen Objekten sind die Anordnung von Prozessen und Produkten in Aggregationshierarchien, die Zuordnung von Ressourcen zu Prozessen, die Produktflußbeziehungen und die Kontrollflußbeziehungen über die Vor- und Nachbedingungen (entry und exit criteria genannt). Die Spezifikation des Kontrollflusses über Bedingungen machen MVP-L zu einer regelbasierten Sprache. Die Konzepte von MVP–L sind für die Modellierung von realen Entwicklungsprozessen geeignet [14]. Zusammen mit den für MVP-L definierten Regeln erlauben sie eine adäquate Beschreibung von Software-Prozessen. Eine Abgrenzung zu anderen Prozeßmodellierungssprachen findet sich in [17]. MVP-L wird neben der initialen Charakterisierung von Prozessen vor allem für die Instrumentierung von Prozessen mit Verfahren des Qualitätsmanagements eingesetzt (siehe z.B. [9]).

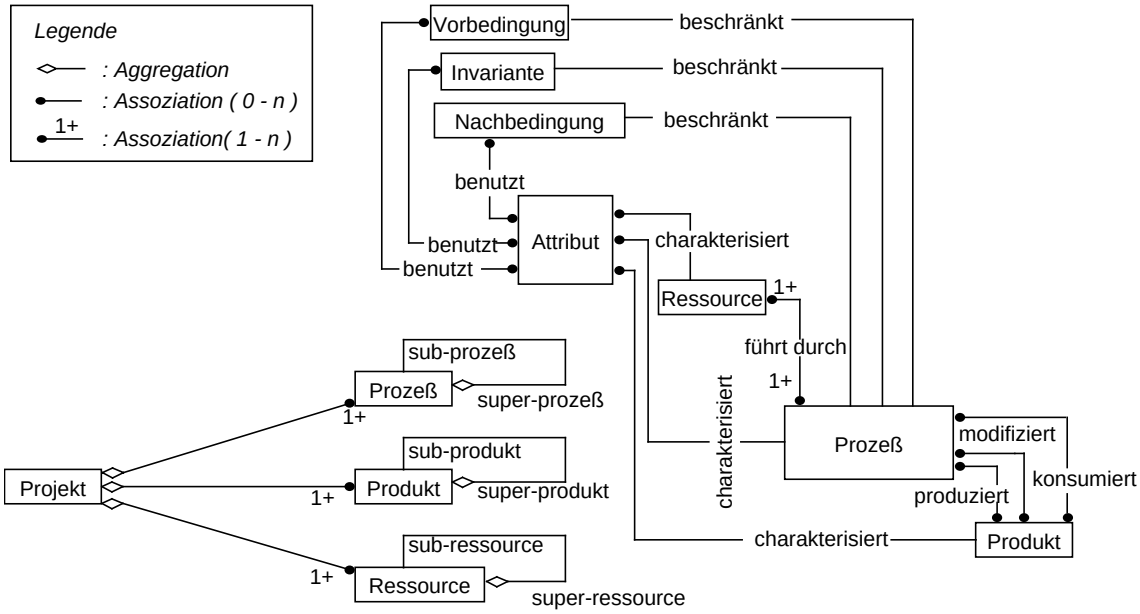


Abbildung 1: Wesentliche Konzepte der Sprache MVP–L

Zur Veranschaulichung ist in Abbildung 2 ein Ausschnitt aus einem MVP-L-Prozeßmodell wiedergegeben. Die grau unterlegten Stellen werden durch Kommentare erläutert. Diese textuelle Darstellung von MVP-L ist nur für den Prozeßingenieur vorgesehen. Soll ein Prozeßmodell von Entwicklern oder Managern begutachtet werden, so muß eine Transformation in eine adäquate Darstellungsform (etwa grafischer Produktfluß) durchgeführt werden.

3. Das Werkzeug MoST

MVP-L wurde mehrfach erfolgreich in Industrieprojekten eingesetzt. Bei den ersten Einsätzen wurde jedoch sowohl von den Prozeßingenieuren als auch von den Entwicklern auf Kundenseite eine mangelnde Werkzeugunterstützung beklagt. Bei der Modellierung finden häufig Reviews statt, die Änderungen der Prozeßmodelle erfordern. Manuell sind diese Änderungen aufwendig durchzuführen. Zudem werden dabei Inkonsistenzen in das Prozeßmodell eingebracht, die ohne Werkzeugunterstützung nur sehr schwer zu erkennen sind.

Die Dienste

Die von MoST bereitgestellten Funktionen lassen sich in drei Gruppen unterteilen: Verwaltungsfunktionen, Funktionen zur strukturellen Analyse und Funktionen, die die erstellten Modelle mit Hilfe der für MVP-L definierten Regeln auf eventuell vorhandene Inkonsistenzen prüfen.

MoST unterstützt den Prozeßingenieur beim Erstellen neuer und bei der Modifikation bereits existierender Prozeßmodelle. Der modulare Aufbau der Grundtypen erlaubt eine inkrementelle Erstellung bzw. Erweiterung oder Verfeinerung. Teile von Prozeßmodellen

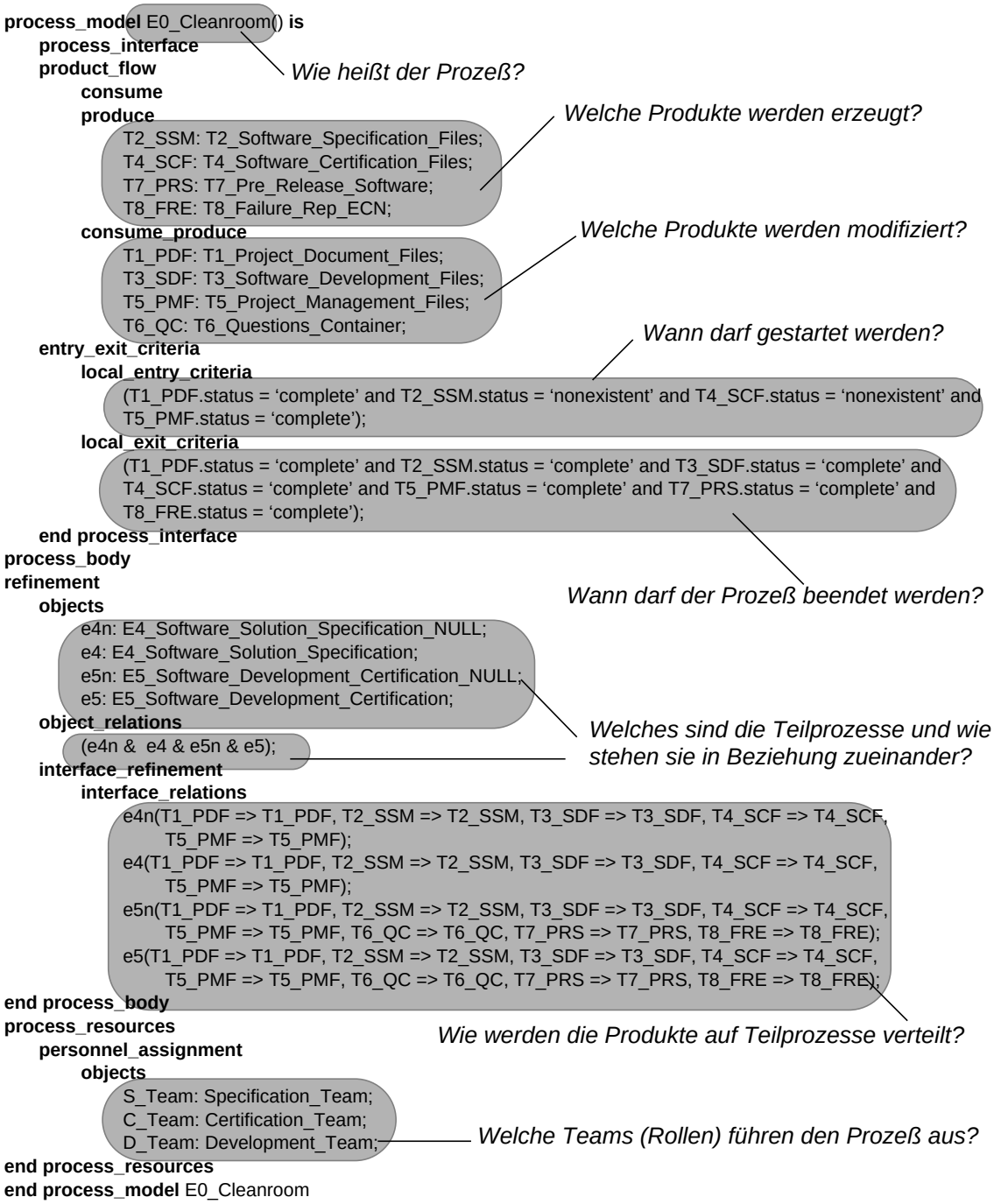


Abbildung 2: MVP-L-Prozeßmodell (unvollständig) eines Cleanroom-Prozesses

können unabhängig voneinander entfernt, modifiziert oder neu eingefügt werden. Dabei können auch bereits existierende Fragmente aus anderen Prozeßmodellen importiert werden. Die Erstellung und Modifikation der einzelnen Modelle wird durch Editierfunktionen wie z.B. Text- oder Zeilensuche unterstützt. Im Zusammenhang mit syntaktischer und semantischer Analyse der Modelle ist es möglich, fehlerhafte Textzeilen direkt anzuspringen. MoST ist in der Lage, syntaktisch inkorrekte Modelle als Text zu speichern und zu laden, damit Modifikationen nicht verloren gehen. Modelle können auch einzeln, unabhängig von ihrer zugehörigen Projektbeschreibung, gespeichert werden.

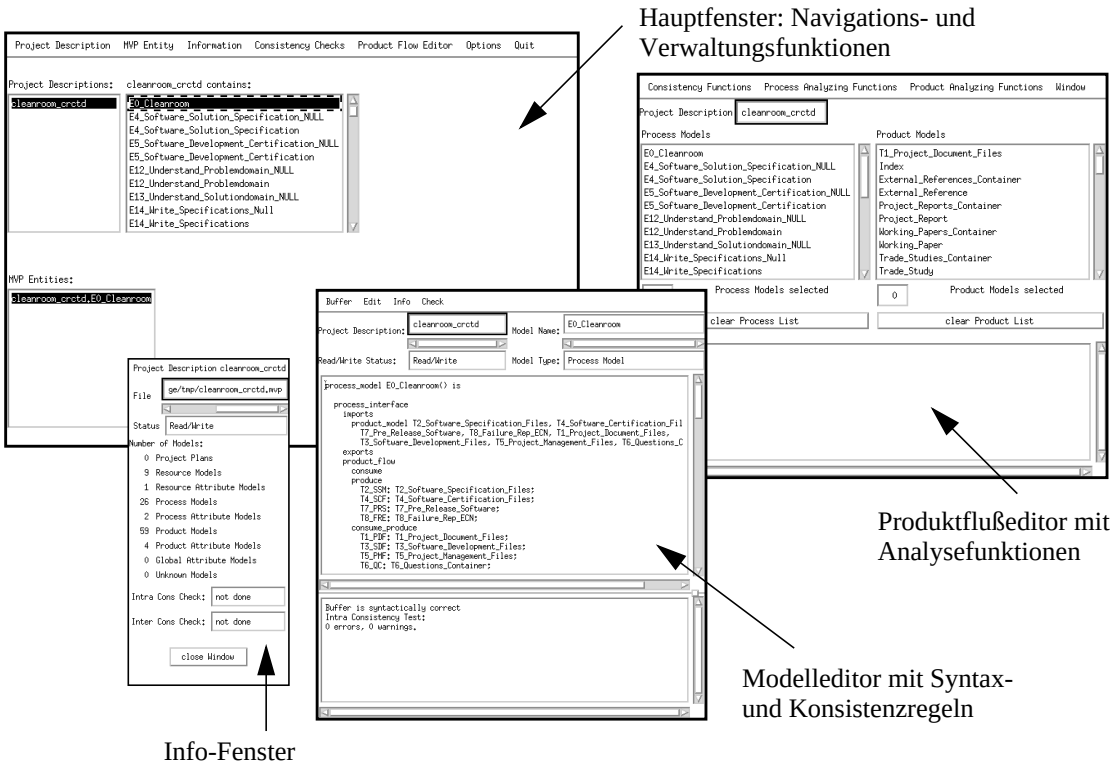


Abbildung 3: Die Benutzerschnittstelle von MoST

MoST unterstützt den Prozeßingenieur mittels einer Reihe von Analysefunktionen, die speziell auf die Bedürfnisse bei der Erstellung oder Modifikation von Prozeßmodellen zugeschnitten sind. Es werden vor allem Funktionen zur Produktflußanalyse angeboten. Hierzu gehört die Möglichkeit, eine Liste der elementaren (nicht verfeinerten) Produkte zu generieren oder anzeigen zu lassen, welche elementaren Prozesse auf verfeinerte Produkte zugreifen (was ein Zeichen für eine mögliche Unterspezifikation der Prozeßhierarchie ist). Insgesamt stehen ca. 20 Analysefunktionen zur Verfügung.

Konsistenzregeln prüfen, ob die Prozeßmodelle den Regeln von MVP-L entsprechen. Dies sind z.B. Regeln bezüglich der Typsicherheit, einer vollständigen Verfeinerung, der Zykelfreiheit der Modelle oder der Vollständigkeit des Produktflusses. Die Verletzung einer Konsistenzregel deutet auf Fehler in der Modellierung, einen falsch verstandenen Prozeß oder eine Inkonsistenz im Prozeß selbst hin. Die Konsistenzregeln können zu einem beliebigen Zeitpunkt einzeln überprüft werden. Desweiteren können die Regeln einzeln auf Teile der Prozeßmodelle angewendet werden. Hierzu wird unterschieden zwischen Regeln, die Beziehungen zwischen Teilen innerhalb eines Prozeßmodells überprüfen (sog. *inter-Modell*-Regeln), und Regeln, die Beziehungen innerhalb eines Teils überprüfen (sog. *intra-Modell*-Regeln). Es gibt 15 intra-Modell-Regeln und 17 inter-Modell-Regeln. Der Prozeßingenieur kann einen einzelnen Baustein bearbeiten, ihn auf interne Konsistenz überprüfen und später in einen größeren Kontext einbauen. Eine inkrementelle Arbeitsweise ist also möglich.

Die Konsistenzregeln müssen nicht restriktiv eingesetzt werden. Viele Dienste von MoST sind auch auf inkonsistente Prozeßmodelle anwendbar. Eine verletzte Konsistenzregel ist

jedoch immer ein Hinweis auf eine potentielle Schwäche des Prozeßmodells. Es existieren weitere MVP-L-Werkzeuge (z.B. Simulator, prozeß-sensitive Software-Entwicklungsumgebung, Tailoring und Transformator für Workflow-Engine) die zu einem gewissen Grad konsistente Prozeßmodelle verlangen. Hier müssen die entsprechenden Regeln eingehalten werden.

MoST besitzt mit den Analysefunktionen und Konsistenzregeln umfangreiches Wissen über Software-Entwicklungsprozesse, die auf Erfahrungen aus realen Projekten beruhen. Bei der (Weiter-) Entwicklung von MoST wurde und wird darauf geachtet, daß Informationen aus Projekten zurückfließen, um eine geeignete Unterstützung des Prozeßmodellierers zu gewährleisten.

Die Architektur

Die Architektur von MoST reflektiert die Entwicklungsgeschichte. Ausgehend von der Sprachdefinition von MVP-L wurde zunächst ein konzeptuelles Modell erstellt, das in industriellen Anwendungen verbessert wurde (siehe auch Abbildung 1). Darauf aufbauend begann die Realisierung des Werkzeugs: Zuerst wurde eine interne Repräsentation sowie ein Front-End, das textuelle Darstellungen in die interne Repräsentation konvertiert, realisiert. Der Produktflußeditor beinhaltet die wesentlichen Analysefunktionen und eine weitere Komponente beinhaltet die Konsistenzchecks. Durch die grafische Benutzerschnittstelle (siehe auch Abbildung 3), die alle Komponenten integriert, wird dem Prozeßingenieur eine übersichtliche und einheitliche Oberfläche geboten.

4. Zwei Szenarien für den Einsatz von MoST

MoST kann für zwei unterschiedliche Aufgabentypen eingesetzt werden: a) Prozeßmodelle werden für die präskriptive Verwendung in Projekten erstellt (z.B. Projekthandbücher), b) Prozeßmodelle werden deskriptiv erstellt, um Verbesserungen der Software-Entwicklung zu ermöglichen [15].

Projektpläne besitzen im allgemeinen ein Prozeßmodell, das die Grundlage für viele Planungsschritte (z.B. Aufwandsschätzung, Ressourceneinteilung, Meilensteinplanung) ist. Prozeßmodelle finden sich in Standards wieder, die während des Projekts beachtet werden müssen (z.B. Das Vorgehensmodell [8]). Organisationsspezifische Handbücher beschreiben einzelne Schritte der Software-Entwicklung unter Umständen detailliert; auch sie stellen Prozeßmodelle dar. Alle die gerade genannten Dokumente müssen erstellt und gegebenenfalls modifiziert werden. Insbesondere beim *Tailoring* (Anpassen) bedarf es zum Teil umfangreicher Änderungen. Hier hilft MoST dem Prozeßingenieur durch Konsistenzprüfungen ein qualitativ hochwertiges Dokument zu erstellen. Das kürzlich entwickelte MVP-L-Werkzeug ProTail [15] erlaubt das automatische Tailoring von Prozeßmodellen, jedoch sind auch beim Einsatz von ProTail Änderungen an den Prozeßmodellen notwendig, die manuell mithilfe von MoST durchgeführt werden können.

Deskriptive Prozeßmodelle werden vor allem für Verbesserungsprogramme benötigt. Aufgrund von Analysen sollen Schwachstellen entdeckt und behoben werden. Diese Analysen beruhen auf Meßdaten, die während der Durchführung von Projekten gesammelt

werden. Die Meßdaten können jedoch nur dann zuverlässig gesammelt und interpretiert werden, wenn ein Prozeßmodell den Gegenstand der Messung, also den Prozeß oder das Produkt beschreibt. Prozeßmodelle werden für die Erstellung von Meßplänen (Wer sammelt wann wo welche Daten welcher Qualität?) benötigt. Die Prozeßmodellierung ist in diesem Szenario ein wesentliches Element des Qualitätsmanagements und der Qualitätssicherung.

5. Erfahrungen beim Einsatz von MoST

MoST wurde für die präskriptive Modellierung dreier Standards bzw. Projekthandbücher (IEEE Std-1074 [13], Das Vorgehensmodell und ein unternehmenseigener Standard [12]) eingesetzt. Sie wurden mit Hilfe des Werkzeugs von natürlicher Sprache nach MVP-L übersetzt. Wie dabei vorgegangen wurde ist ausführlich in [18] beschrieben und wird deshalb nur kurz dargestellt: Das Vorgehen orientiert sich an den Konzepten von MVP-L. Zuerst werden die Produkte modelliert, da sie diejenigen Objekte mit den wenigsten Beziehungen zu anderen Typen von Objekten beinhalten. Die Erstellung verläuft 'bottom-up', d.h. man beginnt mit elementaren Produkten. Danach werden die Prozesse in MVP-L beschrieben, wobei man auch hier bei den elementaren Prozessen beginnt. Erst dann wird der Kontrollfluß anhand von Attributen und die Ressourcenzuordnung beschrieben. Die drei Modellierungen wurden relativ problemlos durchgeführt [20]. MoST hat sich insbesondere bei der inkrementellen Erstellung und Überprüfung der Modelle als hilfreich herausgestellt. Die Konsistenzregeln von MVP-L machten an einigen Stellen eingeschränkte Modifikationen der Prozesse notwendig (z.B. Einfügen von Prozessen, die externe Produkte zur Verfügung stellen). Ob es sich in diesen Fällen um Inkonsistenzen im Sinne von unerwünschten Fällen handelt, kann nicht generell beantwortet werden. Es sei hier auf die Diskussion über die restriktive Anwendung von Konsistenzregeln im Abschnitt *Das Werkzeug MoST* verwiesen. Aber auch bei der Anpassung von Prozeßmodellen (Tailoring) hat MoST wertvolle Unterstützung geleistet: Das Löschen von Informationen aus dem Prozeßmodell zieht viele nachträgliche Änderungen mit sich, wodurch Konsistenzregeln verletzt werden (z.B. erfordert eine Anpassung im V-Modell bis zu 50 Änderungen, siehe auch [15]).

MoST wurde für die deskriptive Modellierung bei einer Reihe von Industrieprojekten eingesetzt. Es wurde jeweils ein Prozeßmodell für die Erstellung der Meßpläne entwickelt. Die Modellierungen müssen iterativ durchgeführt werden, da durch die Rücksprache mit den Entwicklern und Managern Modifikationen notwendig werden. Es hat sich auch hier als sinnvoll herausgestellt, zuerst die Produkte und die Produkthierarchie zu modellieren und erst dann die komplexeren Konzepte (Prozesse, Ressourcen) zu bearbeiten. Die Kommunikation zwischen dem Prozeßingenieur und dem Projektteam erfolgt auf unterschiedlichste Art (email, Fax, Telefon, Gruppendiskussionen und Einzelgespräche). Der Prozeßingenieur setzt die gesammelten Informationen mittels MoST in ein MVP-L-Prozeßmodell um. Durch die Konsistenzprüfungen und Analysefragen ergeben sich Qualitätsaussagen in Bezug auf das Prozeßmodell, die eine eventuelle Rückfrage beim Projektteam notwendig machen. Von den Konsistenzregeln hat sich insbesondere die Regel bezüglich der *Balanciertheit des Produktflusses* als problematisch erwiesen.[1] Die Verletzung dieser Regel weist darauf hin, daß die Produktstruktur beziehungsweise die Prozeßstruktur noch nicht genügend durchdacht ist. Projektmitglieder neigen dazu, in Prozeßverfeinerungen *unscharfe* Subprozesse anzugeben. Ein Symptom dessen ist, daß

diese Prozesse ganze Produktaggregate als Parameter erhalten statt dedizierte Produkte. Bisher konnten alle unbalancierten Produktflüsse aufgelöst werden, was zu einer wesentlich verbesserten Prozeßstruktur führte. Es sei noch einmal darauf hingewiesen, daß eine Verletzung einer Konsistenzregel nicht unbedingt eine Schwäche des Prozeßmodells bedeutet. Die Konsistenzregeln sind jedoch sinnvoll, um eventuelle Probleme aufzudecken.

6. Zusammenfassung und Ausblick

Das Werkzeug MoST zur Modellierung von Software-Entwicklungsprozessen hat sich als eine gute Unterstützung für Prozeßingenieure in Verbesserungsprogrammen erwiesen. Die Unterstützung einer inkrementellen Vorgehensweise bei der Modellierung und die Definition der Konsistenz aufgrund von Erfahrungen in realen Projekten haben sich dabei als wertvollste Eigenschaften von MoST herausgestellt. Die fehlende Unterstützung einer grafischen Darstellung des Prozeßmodells führte zur Erstellung eines grafischen Editors, der zur Zeit in einer prototypischen Version entwickelt wird.

MoST dient als ein Prototyp für ein internes Projekt des Fraunhofer IESE. Es soll eine Modellierungsumgebung neu entwickelt werden, in die Erfahrungen aus MoST einfließen sollen. Wichtiges Kennzeichen dieser SPEARMINT genannten Umgebung ist die gleichzeitige Darstellung mehrerer, größtenteils grafischer Sichten. Dadurch soll der Modellierer die Prozeßmodelle unter verschiedenen Gesichtspunkten betrachten und besser erfassen können.

7. Literatur

[8] Adolf-Peter Bröhl und Wolfgang Dröschel. *Das V-Modell*. Oldenbourg, 1995.

[9] Alfred Bröckers, Christiane Differding und Günter Threin. The role of software process modeling in planning industrial measurement programs. In *Proceedings of the Third International Software Metrics Symposium*, Berlin, März 1996. IEEE Computer Society Press.

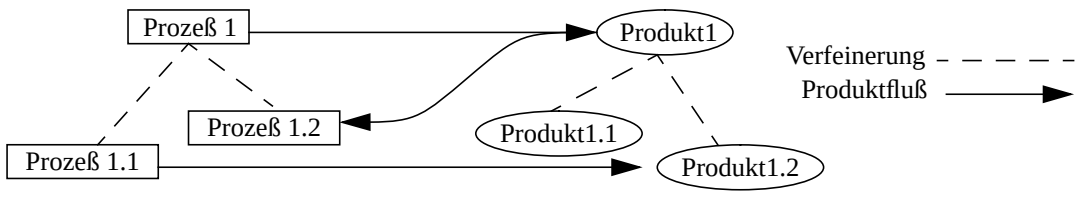
[10] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach und Martin Verlage. MVP–L language report version 2. Technischer Bericht 265/95, Fachbereich Informatik, Universität Kaiserslautern, 67653 Kaiserslautern, 1995.

[11] Ulrike Becker und Martin Verlage. MVP–L's modeling support tool MoST. Technical Report STTI-95-03-E, Software-Technology-Transfer-Initiative Kaiserslautern, University of Kaiserslautern, 67653 Kaiserslautern, Germany, Mai 1995.

[12] IBM. Cleanroom Software Development Processes. Report Task IR70E. 1989.

[13] The Institute of Electrical and Electronics Engineers. Standard for Developing Software Life Cycle Processes. IEEE Computer Society Proess. IEEE-Standard 1074-1991, 1991.

1. Unter Balanciertheit des Produktflusses ist zu verstehen, daß eine Verfeinerung eines Produkts sich in der Prozeßhierarchie widerspiegelt. Prozesse einer Prozeßverfeinerung dürfen nicht auf unterschiedliche Verfeinerungsebenen eines Produktes zugreifen (siehe auch nachfolgende Skizze, hier ist der Produktfluß unbalanciert, da die Subprozesse von Prozeß1 sowohl auf Produkt1 als auch Produkt1.2 zugreifen).



[14] C. D. Klingler, M. Neviaser, A. Marmor-Squires, C. M. Lott und H. D. Rombach. A case study in process representation using MVP–L. In *Proceedings of the Seventh Annual Conference on Computer Assurance (COMPASS 92)*, Seiten 137–146, Juni 1992.

[15] Jürgen Münch, Markus Schmitz und Martin Verlage. Tailoring Large Process Models. Tagungsband des 4. Workshop der Fachgruppe 5.1.1 (GI): Vorgehensmodelle - Einführung, betrieblicher Einsatz, Werkzeug-Unterstützung und Migration, Berlin, 17. und 18. März 1997.

[16] H. Dieter Rombach, Alfred Bröckers, Christopher M. Lott und Martin Verlage. Entwicklungsumgebungen zur Unterstützung qualitätsorientierter Projektpläne. *Softwaretechnik-Trends: Mitteilungen der GI-Fachgruppen 'Software-Engineering' und 'Requirements-Engineering'*, 13(3):1–8, August 1993.

[17] H. Dieter Rombach und Martin Verlage. Directions in software process research. In Marvin V. Zelkowitz, Hrsg., *Advances in Computers, vol. 41*, Seiten 1–63. Academic Press, 1995.

[18] Martin Verlage, Christian Bunse, Peter Giese und Wolfram Petsch. Three approaches for formalizing informal process descriptions. In Peter F. Elzer, Hrsg., *Proceedings of the 5th IFAC/IFIP Workshop on Experience with the Management of Software Projects*, 1995.

[19] Martin Verlage und Jürgen Münch. Formalizing Software Engineering Standards. Erscheint in Proceedings of the Third International Symposium and Forum on Software Engineering Standards (ISESS'97), Walnut Creek, California, USA, 1. - 6. Juni 1997.

[20] Martin Verlage. Erfahrungen bei der Formalisierung von Projekthandbüchern. In *Tagungsband der GI/OCG Jahrestagung Informatik '96*, Seiten 383–402. Oldenbourg Verlag, München – Wien, September 1996.