Drei Ansätze zur Formalisierung informeller Beschreibungen von Software-Entwicklungsprozessen+

Martin Verlage[1], Christian Bunse[2], und Peter Giese[2*]

[1] Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern

[2] Fraunhofer-Einrichtung für Experimentelles Software Engineering, Technopark II, Sauerwiesen 6, 67661 Kaiserslautern

email: {verlage, bunse, giese} @informatik.uni-kl.de

Kurzfassung: Informelle Dokumente sind typisch für die Beschreibung von Software-Entwicklungsprozessen. Software-Projekte müssen jedoch formaler beschrieben werden, um bessere Kommunikation und Kontrolle zu erzielen. Die resultierenden Prozeßmodelle können auf Vollständigkeit, Konsistenz und Eindeutigkeit überprüft werden. Die Formalisierung einer informellen Beschreibung unter Benutzung der Prozeßmodellierungssprache MVP-L wird gezeigt. Das Beispiel, ein Cleanroom-Prozeß, ist einem industriellen Handbuch entnommen. Drei Ansätze werden zur Formalisierung dieses Beispiels angewendet (phasen-, ebenen- und konzept-basierte Modellierung). Konzept-basierte Modellierung scheint ein geeigneter Kandidat zu sein.

1. Einleitung

Software-Projekte bestehen aus einer Vielzahl unterschiedlicher Prozesse. Eine Teilmenge davon, die *Entwicklungsprozesse*, beschäftigen sich mit der Evolution des Produkts. Aus folgenden Gründen ist die Dokumentation dieser Prozesse notwendig:

- Stelle explizites Know-how der Software-Entwicklung für Kommunikation und Verbesserungen zur Verfügung [14].
- Demonstriere die Fähigkeiten der Organisation den Kunden, so wie es von Qualitätsstandards, etwa ISO 9000-3 [10] und Reifemodellen, etwa dem Capability Maturity Model [7] verlangt wird.

Es entstehen üblicherweise Projekthandbücher, Qualitätssicherungspläne, Standards, usw., die strukturiert in natürlicher Sprache verfaßt sind. Diese informellen Prozeßbeschreibungen haben eine Reihe von Nachteilen im Vergleich zu Prozeßmodellen (d.h. formalen Prozeßdefinitionen), die dem Rechner zugänglich sind:

- Informelle Prozeßbeschreibungen sind oftmals lang - sie können mehrere hundert Seiten umfassen - und unzureichend strukturiert. Die Extraktion von Informationen ist aufwendig.
  Prozeßmodelle können von Systemen verwaltet werden. Browser erleichtern das Lesen (z.B. suchen, filtern, sortieren). Zusätzliche Analysen können durchgeführt werden, um weitere Informationen abzuleiten. [13]
- Informelle Prozeßbeschreibungen sind oftmals zu generisch, was den Nutzen in einer konkreten Situation mindert. Sie enthalten akkumuliertes Wissen, das aus mehreren Projekten stammt.
  Prozeßmodelle können Parameter besitzen, mit denen sie auf einzelne Projekte zugeschnitten werden können [4].
- Bei manchen Handbüchern vermißt man rollenspezifische Beschreibungen bzw. eine dementsprechende Strukturierung. Für Projektmitglieder ist es schwer, Informationen, die für ihre Aufgaben spezifisch sind, zu extrahieren.
  Prozeßmodelle können an Rollen angepaßt werden. Software-Entwickler werden durch ihre eigenen Prozeßmodelle angeleitet, ohne Überladung durch unnütze Informationen [14].
- Informelle Beschreibungen auf Papier repräsentieren statische Information. Das Verhalten der Prozesse ist schwierig zu verstehen.
  Prozeßmodelle können durch Simulatoren oder Prozeßmaschinen (engl. *process engines*) interpretiert werden. Die dynamisch präsentierte Information erhöht das Verständnis [13].

Die Mächtigkeit der Unterstützung hängt unmittelbar vom Grad der Formalisierung ab. Je formaler die Modelle sind, um so mehr Unterstützung kann durch den Rechner angeboten werden.

+ Verbesserte und übersetzte Version des Papiers mit W. Petsch "Three Approaches for Formalizing Informal Process Desciptions" präsentiert auf dem 5th International Workshop on Experience with the Management of Software Projects, 27.-29.09.95, Karlsruhe.

* Unterstützt durch ESPRIT Projekt No. 9090 PERFECT (“Process Enhancement for Reduction of Software Defects”).

Hauptvorteil formaler Prozeßmodelle gegenüber informellen Beschreibungen ist aber, daß Prozeßmodelle *vollständiger*, *konsistenter* und *eindeutiger* sind [1]. Im Kontext einer Prozeßmodellierungssprache werden bessere Ergebnisse bezüglich Vollständigkeit erreicht durch die vorgeschriebene Verwendung der Konstrukte; Konsistenz kann durch Regeln definiert werden, die die Gültigkeit eines Satzes der Sprache definieren; Eindeutigkeit basiert auf der Wohldefiniertheit der Sprachkonstrukte. Dieses Papier konzentriert sich auf diese drei wesentlichen Punkte, da sie Voraussetzungen für die oben diskutierten Vorteile formaler Prozeßmodelle sind.

Prozeßhandbücher und andere informelle Dokumente existieren in fast allen Software-entwickelnden Organisationen. Trotz ihrer oben genannten Nachteile repräsentieren sie wertvolles Wissen. Dieses Wissen sollte nicht unbeachtet bleiben, sondern es kann Ausgangspunkt für formalere Beschreibungen sein. Welches Vorgehen ist aber für die Formalisierung zu wählen? Dieses Papier versucht eine Antwort darauf zu geben, indem über Erfahrungen bei der Formalisierung eines Cleanroom-Prozesses berichtet wird. Drei potentielle Ansätze (phasen-, ebenen- und konzept-basierte Modellierung) wurden betrachtet. Phasen-basiertes Modellieren orientiert sich an der zeitlichen Abfolge des Prozesses. Ebenen-basiertes Modellieren verfeinert sukzessive abstrakte Prozesse bis hin zu atomaren Prozessen. Konzept-basierte Modellierung orientiert sich an der eingesetzten Zielsprache, indem zuerst einfache und dann komplexere Konzepte verwendet werden.

Jeder Ansatz zur Modellierung wird beeinflußt von der eingesetzten Modellierungssprache. Der Formalismus sollte im Hinblick auf die Ziele der Modellierung ausgewählt werden, da Sprachen aufgrund ihrer Charakteristika für jede Aufgabe ihre spezielle Eignung besitzen [12]. Manche Sprachen sind geeigneter für analytische Aufgaben (z.B. Petri-Netz-Formalismen), andere wiederum sind für die Automatisierung geeignet (z.B. Prozedurale Modellierungssprachen) oder für die Begutachtung von Prozeßmodellen (z.B. rein grafische Notationen) [13]. Für diese Studie wurde die Prozeßmodellierungssprache MVP-L verwendet.

Abschnitt 2 behandelt den Kontext der Studie. Abschnitt 3 stellt die drei ausgewählten Modellierungsansätze vor und beschreibt deren Anwendung auf das Cleanroom-Beispiel. Abschnitt 4 vergleicht die drei Ansätze und Abschnitt 5 berichtet über zusätzliche Untersuchungen. Abschnitt 6 schließt diese Arbeit ab.

2. Kontext der Studie

MVP-L ist die Prozeßmodellierungssprache des Multi-View Process Modeling (MVP) Projekts [3].

```
process_model E14_Write_Specifications () is
-- The Write Specification process is where the Specification team
writes the first five volumes of the six volume Specifications. Those
five volumes are: the Mission Volume, the User's Reference Manual
Volume, the Functional Specification Volume, the Functional Spe-
cification Verification Volume, and the Usage Profile Volume.
process_interface
   ..<deleted>..
   exports
      effort : Process_effort := 0;
   product_flow
      consume
         index1: Index;
         externRe: External_References_Container;
         prorep: Project_Reports_Container;
         worpap: Working_Papers_Container;
   ..<deleted>..
      produce
      consume_produce
         mv: Mission_Volume;
         fs: Functional_Spec;
   ..<deleted>..
   entry_exit_criteria
      local_entry_criteria
         (index1.status  = 'complete' and
         worpap.status  = 'complete' and
         prorep.status  = 'complete') and ..<deleted>.. ;
      local_exit_criteria
         (mv.status = 'complete' and
         fs.status = 'complete' ) and ..<deleted>..;
end process_interface
```

Abb. 1: Prozeßschnittstelle in MVP-L (unvollständig)

Im MVP-Projekt wird Unterstützung für Modellierung, Planung, Abwicklung, Erfassung, Aufzeichnung und Verpacken (*packaging*) von Prozeßinformationen entwickelt. Die wesentlichen Merkmale von MVP-L sind: natürliche Repräsentation, verschiedene elementare Konzepte, Benutzung formaler Parameter für Prozeßtypen und die Instrumentierung der Software-Entwicklungsprozesse für die Datensammlung. MVP-L wurde für die Beschreibung von Modellen definiert, die von Menschen benutzt und von Maschinen interpretiert werden sollen.

Die Basiskonzepte von MVP-L sind Produkte, Prozesse, Ressourcen (sowohl Menschen als auch Werkzeuge) und Attribute, die zur genaueren Charakterisierung der Instanzen der ersten drei Konzepte eingesetzt werden können. Attributwerte korrespondieren mit Meßwerten, die während des laufenden Projekts erfaßt werden [6]. Zusammen mit den Repräsentationen der Produkte, Prozesse und Ressourcen werden Attribute benutzt, um die Entwickler bei der Arbeit anzuleiten und um das Management über den Projektzustand zu informieren. Zusätzlich zu einer Anleitung auf Basis quantitativer Kenngrößen erlauben die Attribute außerdem die Beobachtung von Produkten, Prozessen und Ressourcen. Dies erlaubt die Identifikation von Problemen und zeigt Ansatzpunkte für die Verbesserung der Produkte und Prozesse auf. Messungen bilden die Grundlage für die Evaluierung neu eingesetzter Methoden und Techniken.

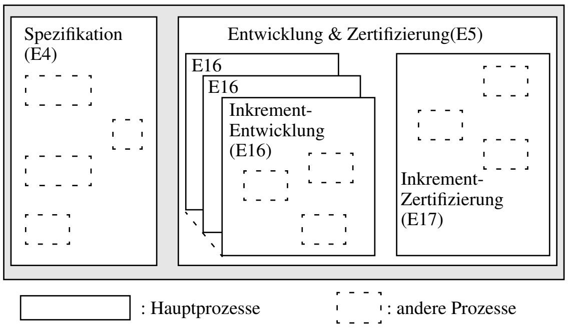

Abb. 2: Wesentliche Entwicklungsprozesse von Cleanroom

In MVP-L werden die Beziehungen zwischen Objekten explizit modelliert. Prozesse und Produkte sind mittels Produktfluß-Klauseln verbunden; Ressourcen, verantwortlich für die Abwicklung, werden Prozeßinstanzen zugewiesen. MVP-L ist eine regel-basierte Sprache: Kontrollfluß zwischen Prozessen wird durch Vor- und Nachbedingungen ausgedrückt (*entry criteria* und *exit criteria*). Ein Auszug aus einem Prozeßmodell in MVP-L ist in Abbildung 1 wiedergegeben. MVP-L wurde für die in diesem Papier beschrieben Studie eingesetzt, da sie auch für andere Forschungsarbeiten die von den Autoren favorisierte Technik ist.

Cleanroom-Prozesse werden zur Zeit intensiv untersucht [5]. Sie sollen Software mit einer vorhersagbaren Qualität entwickeln. Die wesentlichen Charakteristika eines Cleanroom-Prozesses sind die Aufteilung der Spezifikation in Inkremente, die Spezifikation, Entwicklung und Zertifizierung durch verschiedene Teams und die iterative Entwicklung. Das Verhalten von Cleanroom-Prozessen ist aufgrund ihrer verschachtelten Struktur nicht trivial. Deswegen wurden sie zur Modellierung ausgewählt.

Ausgangspunkt der Studie war eine informelle Beschreibung eines Cleanroom-Prozesses, die für eine spezielle Organisation (IBM) entwickelt wurde [8]. Das Dokument enthält 26 Hauptprozesse auf etwa 250 Seiten. Die wesentlichen Entwicklungsprozesse sind in Abbildung 2 skizziert. Zur Beschreibung aller Prozesse wird ein einheitliches Schema verwendet. Jede Dokumentation eines Prozesses enthält eine kurze Beschreibung (ca. 5-10 Zeilen), eine Liste der verwendeten Produkte, eine Liste der notwendigen Ressourcen, ein grafische Darstellung des Prozesses, die Angabe des vorherigen Prozesses, eine Vorbedingung, nachfolgende Prozesse, Ereignisse, die diesen Prozeß starten oder bei seiner Terminierung generiert werden, eine Liste der Aktionen, die auszuführen sind, eine Liste von Bedingungen, die bei Terminierung des Prozesses erfüllt sein müssen und Verweise auf andere Abschnitte.

3. Ansätze der Formalisierung

Es werden die angewendeten drei Modellierungsansätze vorgestellt. Die Evaluierung der Ansätze stellt kein Experiment dar, sondern beruht auf Beobachtungen während der Formalisierung.

3.1 Phasen-basierte Modellierung

Durch das Wasserfallmodell ist der Begriff *Phase* bekannt als ein zeitlicher Abschnitt, in dem Aktivitäten mit einem gemeinsamen Ziel stattfinden (z.B. *Entwurfsphase*). Die phasen-basierte Modellierung orientiert sich an den Prozessen einer Phase und kann deshalb als eine Divide & Conquer-Strategie angesehen werden. Durch die Konzentration auf eine Phase werden Entscheidungen bezüglich Alternativen nur im Kontext der gerade modellierten Phase getroffen. Dies kann dazu führen, daß die Schnittstellen der Prozesse ungeschickt für nachfolgende Phasen gestaltet werden.

In der Studie wurden die Schnittstellen zwischen den wesentlichen Entwicklungsprozessen in MVP-L formuliert. Anschließend wurde eine phasenweise Formalisierung vorgenommen. Diese orientierte sich an der Aufteilung von Cleanroom-Prozessen in Spezifikation, Inkrement-Entwicklung, Integration und Zertifizierung. Die Transformation begann mit den frühen Phasen. Die Modellierungsaufgabe wurde mit fortschreitender Dauer komplizierter. Durch die inhärente Rückkopplungsstrukturen von Cleanroom-Prozessen war man bei der Modellierung der späten Phasen gezwungen, Änderungen an den Modellen der frühen Phasen vorzunehmen. Dies wurde u.a. hervorgerufen durch die Art der Kontrollflußdefinition in MVP-L: Vorbedingungen müssen die relevanten Projektzustände für eine Prozeßaktivierung berücksichtigen. Bei der ersten Spezifikation der Vorbedingungen waren die Zustände nachfolgender Iterationen noch nicht bekannt. Die Bedingungen und Attribute mußten laufend erweitert werden. Da dieses Problem mehr und mehr Schwierigkeiten verursachte, wurde diese Modellierung schließlich abgebrochen.

3.2 Ebenen-basierte Modellierung

Der zweite Ansatz, ebenen-basierte Modellierung, orientiert sich an den Abstraktionsebenen einer Prozeßbeschreibung. In einer Top-down-Vorgehensweise werden die Prozesse einer Ebene transformiert und dann die Prozesse der nächstfeineren Ebene. Die Aggregation als eine Form der Abstraktion wird benutzt, um die Komplexität bei der Modellierung zu reduzieren. Die Modellierung wird fortgesetzt, bis die elementaren (d.h. nicht weiter verfeinerten) Prozesse erreicht sind. Die wesentliche Annahme ist, daß die Verfeinerung eines Prozesses nicht die Prozeßmodelle der nächsten Ebene berühren. Prozesse werden demnach als Black-boxes mit einer Schnittstelle aufgefaßt.

1. Modellierung elementarer Produkte

2. Aufbau komplexer Produkte

3. Modellierung elementarer Prozesse

4. Aufbau komplexer Prozesse

5. Hinzufügen von Attributen zu Produkten und Prozessen

6. Spezifikation des Kontrollflusses

7. Zuweisung von Ressourcen an Prozesse

Abb. 3: Sieben Schritte der konzept-basierten Modellierung

In der originalen Beschreibung des Cleanroom-Prozesses existieren mehrere Abstraktionsebenen (siehe auch Abb. 2). Bei der Anwendung der ebenen-basierten Modellierung fand eine Konzentration auf die Beziehungen zwischen Prozessen gleicher Ebene statt. Man tendiert dazu, sequentielle Modelle zu bauen. Im Falle des speziellen Cleanroom-Beispiels überlappen sich jedoch die abstrakten Prozesse. Dies wurde erst bei der Modellierung der fein-granularen Prozesse festgestellt. Die informelle Beschreibung auf den abstrakten Ebenen versteckt viel von den Beziehungen zwischen Prozessen auf den unteren Ebenen. Bottom-up-Modifikationen der Schnittstellen auf höheren Abstraktionsebenen waren die Folge. Darauf folgte eine Top-down-Änderung anderer Prozeßmodelle. Da dieses Problem mehr und mehr Schwierigkeiten verursachte, wurde diese Modellierung nicht zu Ende geführt.

3.3 Konzept-basierte Modellierung

Die beiden ersten Ansätze zur Formalisierung orientieren sich an der Struktur des Dokuments. Dagegen orientiert sich die konzept-basierte Modellierung an den Konzepten der eingesetzten Modellierungssprache (z.B. Produkthierarchie, Produktfluß, Prozeßhierarchie, Kontrollfluß). Das Vorgehen bei der konzept-basierten Modellierung wird also stark von der Sprache vorgegeben.

Im Fall von MVP-L besteht die konzept-basierte Modellierung aus sieben Schritten (siehe Abb. 3). In Schritt 1 werden die elementaren Produkte modelliert worauf in Schritt 2 die komplette Produkthierarchie in einer Bottom-up-Weise vervollständigt wird. In den Schritten 3 und 4 wird die Prozeßhierarchie auf ähnliche Art erstellt, indem man wiederum bei den elementaren Einheiten anfängt und mit den komplexeren fortfährt. Die letzten beiden Schritte beinhalten außerdem die Spezifikation der 'benutzt'-Beziehungen zwischen Prozessen und Produkten, den Produktfluß. In Schritt 5 werden die Attribute zur Aufnahme beobachtbarer und meßbarer Eigenschaften modelliert. Unter Benutzung dieser Attribute wird in Schritt 6 der Kontrollfluß spezifiziert. Zwischen diesen beiden Schritten kann iteriert werden, da die Definition des Kontrollflusses durch Vor- und Nachbedingungen eine schwierige Aufgabe darstellt. Schließlich werden in Schritt 7 die zur Abwicklung der Prozesse notwendigen Ressourcen spezifiziert.

Die Formalisierung des Beispiels nach der konzept-basierten Modellierung war relativ geradlinig. Signifikante Modifikationen von Ergebnissen vorhergehender Modellierungsschritte mußten nicht durchgeführt werden, außer der Beseitigung kleinerer Fehler. Es ist offensichtlich, daß hier Erfahrungen aus den vorangegangenen fehlgeschlagenen Modellierungsversuchen Eingang gefunden haben. Nichtsdestotrotz sind die Autoren davon überzeugt, daß die Schlußfolgerungen aus dieser Studie (siehe nächsten Abschnitt) gültig sind.

4. Diskussion

Während der Formalisierung des originalen Cleanroom-Beispiels wurden einige wenige Lücken, Inkonsistenzen und Mehrdeutigkeiten aufgedeckt. Eine Auswahl wird hier vorgestellt. Dies geschieht nicht, um die Arbeit der Autoren zu schmälern, sondern um die Vorzüge der Verwendung einer formalen Prozeßbeschreibungssprache aufzuzeigen. Es bleibt eine unbeantwortete Frage, ob ein vollständiges, konsistentes und eindeutiges Dokument unter ausschließlicher Verwendung natürlicher Sprache entstehen könnte. Für einige Schwächen wird skizziert, wie sie in MVP-L aufgelöst wurden. Diese Auflösungen beruhen auf subjektiven Urteilen. Selbstverständlich sollten bei Modellierungen realer Prozesse die betroffenen Personen befragt werden.

Im IBM-Dokument wurden keine auffälligen Inkonsistenzen gefunden. Aber folgende Konsistenzregeln von MVP-L wurden von der Prozeßhierarchie verletzt: Der Prozeß *Increase Understanding of Problem & Solution Domain as Required* (als *E21* im Dokument abgekürzt) wird als Subprozeß zweier Prozesse auf der nächsthöheren Ebene beschrieben. MVP-L verlangt eine Baumstruktur der Prozeßhierarchie. Ansonsten können u.a. Probleme bezüglich der Prozeßzustände auftreten. Die Inkonsistenz wurde dadurch behoben, daß zwei Varianten des Prozesses E21 mit entsprechenden Verantwortlichkeiten modelliert wurden. Eine andere sprachbedingte Inkonsistenz betrifft die unterschiedliche Gestaltung der Produkt- und Prozeßhierarchie. Dies macht das Erkennen und Verwalten des Produktflusses schwierig. Es ist problematisch zu erkennen, wie Produktinformationen behandelt werden. In beiden Fällen weicht die MVP-L-Beschreibung vom Original ab, um die Inkonsistenzen aufzulösen.

Einige Mehrdeutigkeiten wurden aufgedeckt, die nicht durch natürliche Sprache verursacht, sondern durch unpräzise Darstellungen hervorgerufen werden. Zum Beispiel ist es komplett dem Leser überlassen, die Quelle eines Dokuments herauszufinden, da nur modifizierende Zugriffe spezifiziert sind; oft stehen mehrere mögliche produzierende Prozesse zur Auswahl. Ein anderes Problem entsteht bei der Spezifikation paralleler Prozesse mit schreibendem Zugriff auf ein Dokument (z.B. *E12* und *E13*). Es fehlt Information, wie ein Konflikt vermieden oder aufgelöst wird. Desweiteren ist nichts darüber erwähnt, welche Informationen in einem Produkt modifiziert, ergänzt oder entfernt werden wenn Prozeßiterationen stattfinden.

Im Beispiel-Dokument fehlen Informationen über Produkte. Zum Beispiel sollen Produktmetriken Eigenschaften der Produkte genauer charakterisieren, aber es wird fast nichts über Maßdefinitionen und Datensammlung geschrieben. Außerdem wäre es für den Leser effektiver und effizienter, wenn die Beziehungen zwischen Prozessen und Produkten explizit dokumentiert würden. Wir benutzten zur Ableitung dieser Information das Modellierungswerkzeug MoST [2], welches die Handhabung der Prozeßmodelle einfacher machte als das Lesen des strikt kontrollfluß-orientierten Dokuments.

Alle drei vorgestellten Ansätze sind inkrementell (Phase nach Phase, Ebene nach Ebene oder Konzept nach Konzept). Phasen-basierte und ebenen-basierte Modellierung produzieren dabei Zwischenresultate, die Teile des Prozesses beschreiben, aber im Sinn der verwendeten Prozeßmodellierungssprache vollständig sind. Diese Teile (etwa die *frühe Phasen* oder die *Top-level-Prozesse*) können während der Formalisierung begutachtet werden. Ein vollständiges Prozeßmodell (i.S. der Sprache) existiert bei konzept-basierter Modellierung erst am Ende. Der Grund, warum konzept-basierte Modellierung im Kontext der Studie vorzuziehen ist, liegt in der Sprach-Orientierung des Ansatzes. Konzept-basierte Modellierung konzentriert sich zuerst auf Konzepte mit geringer Komplexität und benutzt die damit erstellten Teile als Bausteine (so z.B. von Schritt 1 zu Schritt 2, oder von Schritt 3 zu Schritt 4). Die in MVP-L komplexeste Aufgabe, die Spezifikation des Kontrollflusses, wird zuletzt durchgeführt. Diese Vorgehensweise garantiert Stabilität während der Transformation. Bei den anderen Modellierungsansätzen wechseln sich die Modellierung komplexer und weniger komplexer Teile ab.

Im Vergleich zu ebenen- und phasen-basierter Modellierung erforderte die konzept-basierte Modellierung weniger Rückschritte und erlaubte eine bessere Handhabung der Prozeßmodelle. Welcher Modellierungsansatz vorzuziehen ist, wenn keine expliziten Modelle vorliegen bleibt unbeantwortet. Eine vollkommen andere Vorgehensweise könnte notwendig sein.

5. Verwandte Arbeiten

Die in diesem Papier beschriebenen Studie ist nur ein Beispiel für den Einsatz von MVP-L. Viele andere Modellierungen sind jedoch mit der in diesem Papier präsentierten Studie nicht vergleichbar, da MVP-L meist verwendet wird, um initiale Prozeßmodelle zu erstellen. Etliche Modellierungen hatten z.B. das Ziel, die Einführung von Meßprogrammen in Software-entwickelnde Organisationen zu unterstützen. Dieser Abschnitt beschäftigt sich dagegen nur mit Modellierungsbeispielen, in denen informelle Beschreibungen tranformiert wurden.

In einer industriellen Studie bei TRW (USA) wurde zuerst die Beschreibung eines Wiederverwendungsprozesses formalisiert und dann das Modell analysiert und verbessert. Strukturelle Schwächen und Mehrdeutigkeiten in der Terminologie wurden entfernt. Schließlich wurde der gesamte Prozeß in natürlicher Sprache re-dokumentiert, um ihn Projektangehörigen zugänglich zu machen, die MVP-L nicht verstehen. Die Ausgangsbeschreibung war im wesentlichen konsistent, aber durch Mehrdeutigkeiten und fehlende Informationen wurden Probleme verursacht. Vergleichbar mit der Cleanroom-Studie, war der Produktfluß zwischen feinen Prozessen auf abstrakten Ebenen nicht mehr dokumentiert. Die generische Beschreibung beinhaltete mehrere Varianten des Wiederverwendungsprozesses. Es war schwer zu entscheiden, welche Variante in einem Teil des Prozesses mit einer Variante in einem anderen Teil in Beziehung steht [11].

Ein weiteres Beispiel ist die Formalisierung des IEEE-Standard 1074 [9] mittels der konzept-basierten Modellierung. Er enthält eine Liste von Prozessen aus denen für eine Organisation eigene Lebenszyklusmodelle aufgebaut werden. Das Dokument beschreibt sechs Prozeßhauptgruppen auf 110 Seiten. Vier dieser Gruppen wurden in MVP-L formuliert, da sie produktorientierte Entwicklungsprozesse beschreiben. Prozesse zur Auswahl des Lebenszyklusmodells und des Managements technischer Prozesse wurden nicht berücksichtigt, da MVP-L hierfür nicht entwickelt wurde. Vergleichbar zum IBM-Dokument enthält der IEEE-Standard strukturierte aber informelle Beschreibungen. Die MVP-L-Beschreibung umfaßt 85 Prozesse, 57 Produkte und 11 Attributtypen auf 6.200 Zeilen, einschließlich Kommentaren. Schritt 7 der konzept-basierten Modellierung wurde nicht durchgeführt, da der Standard keine Beschreibung von Ressourcen enthält. Ähnlich zur TRW-Studie gab es Probleme mit der inhärenten Generezität der IEEE-Dokumentation. Der Standard beschreibt nicht durchgängig einen Prozeß, sondern Bausteine. Die Prozesse und ihre Beziehungen müssen an ein konkretes Projekt angepaßt werden. Die Information ist oft mehrdeutig. Es wird zuwenig über die Beziehungen zwischen den Prozessen ausgesagt. Zum Beispiel ist nicht klar, welche Alternativen bei Prozeßverfeinerungen in verschiedenen Prozessen zueinander gehören. Das Dokument ist weitgehend konsistent und vollständig, was auch nicht verwunderlich ist, wenn man den Aufwand für Begutachtung durch die Kommissionsmitglieder berücksichtigt.

Trotz der Erfahrungen aus diesen Beispielen ist es notwendig, daß Experimente durchgeführt werden, um wissenschaftliche Resultate zu produzieren. Explizite Modelle von Software-Entwicklungen haben sich als hilfreich für die Kommunikation und notwendig für die Analyse erwiesen. Simulation kann ein tieferes Verständnis der dynamischen Eigenschaften der Prozesse bringen.

6. Abschluß

Software-Projekte müssen formaler beschrieben werden, um bessere Kommunikation und Kontrolle zu erzielen. Die resultierenden Prozeßmodelle können auf Vollständigkeit, Konsistenz und Eindeutigkeit überprüft werden. Die Formalisierung einer informellen Beschreibung unter Benutzung der Prozeßmodellierungssprache MVP-L wurde gezeigt. Das Beispiel, ein Cleanroom-Prozeß, wurde einem industriellen Handbuch entnommen. Drei Ansätze wurden zur Formalisierung dieses Beispiels angewendet. Konzept-basierte Modellierung erwies sich als geeigneter Kandidat. Weitere Beispiele haben die Eignung dieses Ansatzes bestätigt.

Die in dieser Arbeit beschrieben Resultate stehen im Kontext sowohl des IBM-Dokuments als auch von MVP-L. Ein anderer Aufbau des Dokuments hätte unter Umständen andere Ergebnisse hervorgerufen. Falls eine andere Sprache verwendet wird, kann die konzept-basierte Modellierung variieren. Wird z.B. eine kontrollfluß-orientierte Sprache (etwa Appl/A [15]) verwendet, so würde vielleicht die phasen-basierte Modellierung gute Resultate produzieren. Würde man eine Sprache verwenden, die das Verhalten der Prozesse beschreibt (z.B. irgendein Petri-Netz-basierter Ansatz, etwa SLANG [1]) könnte ein komplett neuer Ansatz notwendig sein. Vielleicht wäre aber auch ein solcher Ansatz ungeeignet, da die Art von Dokumentation wie sie hier betrachtet wurde, Verhalten von Prozessen unvollständig und mehrdeutig beschreibt [13].

Abschließend kann davon ausgegangen werden, daß die Formalisierung ein geeignetes Mittel ist, um Schwächen - im Rahmen der Modellierungssprache - in informellen Dokumenten zu entdecken. Dabei sollte zuerst eine Prozeßmodellierungssprache abhängig von den Zielen gewählt werden. Dann muß der Ansatz zur Transformation bestimmt werden. Dieser sollte, wie die hier vorgelegte Studie zeigt, an den Konzepten der Sprache orientiert sein.

Die Autoren bedanken sich bei Wolfram Petsch für seine Mühen bei der Modellierung. Chris Lott und Jürgen Münch kommentierten das Papier sorgfältig. Die anonymen Gutachter des MSP'95-Workshops gaben wichtige Hinweise zu einer früheren Version des Papiers.

Literatur

[1] Sergio Bandinelli, Alfonso Fuggetta, Luigi Lavazza, Maurizio Loi, and Gian Pietro Picco. Modeling and improving an industrial software process. *IEEE Transactions on Software Engineering*, 21(5):440–454, May 1995.

[2] Ulrike Becker and 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, May 1995.

[3] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach, and Martin Verlage. MVP–L language report version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany, 1995.

[4] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Communications of the ACM*, 35(9):75–90, September 1992.

[5] M. Dyer. *The Cleanroom Approach to Quality Software Development*. John Wiley and Sons, Inc., 1992.

[6] P. Giese, B. Hoisl, C. M. Lott, and H. D. Rombach. Data collection in a process-sensitive software engineering environment. In *Proceedings of the Ninth International Software Process Workshop*, pages 47–49, October 1994.

[7] Watts S. Humphrey. *Managing the Software Process*. Addison Wesley, Reading, Massachusetts, 1989.

[8] IBM. Software technology for adaptable, reliable systems (STARS) program. Technical Report Task IR70E, IBM, 1991.

[9] IEEE Standards Board. IEEE standard for developing software life cycle processes. In *IEEE Standards Collection Software Engineering*, chapter Std 1074. IEEE, 1991.

[10] International Organization for Standardization. *ISO 9000: Quality management and quality assurance standards; Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software*. Geneva, Switzerland, 1991.

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

[12] H. Dieter Rombach and Martin Verlage. How to assess a software process modeling formalism from a project member's point of view. In *Proceedings of the Second International Conference on the Software Process*, pages 147–158, February 1993.

[13] H. Dieter Rombach and Martin Verlage. Directions in software process research. In Marvin V. Zelkowitz, editor, *Advances in Computers, vol. 41*, pages 1–63. Academic Press, 1995.

[14] Kristine D. Saracelli and Kurt F. Bandat. Process automation in software application development. *IBM Systems Journal*, 32(3):376–396, 1993.

[15] Stanley M. Sutton, Jr. *APPL/A: A Prototype Language for Software–Process Programming*. PhD thesis, University of Colorado, Boulder, CO, August 1990.