Anpassung von Vorgehensmodellen im Rahmen ingenieurmäßiger Softwarequalitätssicherung*

Jürgen Münch

Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern
muench@informatik.uni-kl.de

Zusammenfassung. *Die Einbettung der Software-Qualitätssicherung in ein ingenieurmäßiges Software-Entwicklungsmodell verspricht eine Überwindung des heutigen Zustands, der weitgehend durch die Überprüfung unklar formulierter Qualitätsmerkmale oder syntaktischer Dokumentationsvorschriften gekennzeichnet ist. Software-Entwicklung nach ingenieurmäßigen Grundsätzen erfordert a) die genaue Definition von Qualitätszielen für das zu entwickelnde Software-System, b) die Auswahl und geeignete Anpassung von Vorgehensmodellen, Techniken, Methoden und Werkzeugen zur Erfüllung der vorgegebenen Qualitätsziele und c) die Gewährleistung der Einhaltung vorgegebener Qualitätsziele durch geeignete Maßnahmen. Ingenieurmäßige Softwarequalitätssicherung unterstützt dieses Vorgehen durch aufeinander abgestimmte analytische und konstruktive Maßnahmen. Einen Schwerpunkt der konstruktiven Maßnahmen stellt die Auswahl und Anpassung geeigneter Vorgehensmodelle, Techniken, Methoden und Werkzeuge dar, die die Erfüllung vorgegebener Qualitätsziele gewährleisten.*

Dieser Beitrag beschäftigt sich mit der Anpassung von Vorgehensmodellen. Basierend auf den Erfordernissen ingenieurmäßiger Softwarequalitätssicherung werden wünschenswerte Möglichkeiten der Anpassung von Vorgehensmodellen sowie dafür notwendige Voraussetzungen beschrieben. Anhand des ESA Software Engineering Standards PSS-05 wird gezeigt, welche Möglichkeiten der Anpassung in der Praxis heute vorzufinden sind. Abschließend wird eine Übersicht existierender Forschungsansätze gegeben, die sich mit der Charakterisierung, Auswahl und Anpassung von Vorgehensmodellen beschäftigen.

1 Einleitung

Die hohe Qualität eines zu entwickelnden Software-Produkts oder einer Software-Dienstleistung und die damit verbundene Kundenzufriedenheit ist ein wesentliches Ziel der Software-Entwicklung. Der Begriff Qualität subsumiert diejenigen relevanten Merkmale eines Softwareobjekts, anhand derer sich der Erfüllungsgrad von Anforderungen an das Softwareobjekt bestimmen läßt. Beispiele für Qualitätsmerkmale sind Zuverlässigkeit, Lesbarkeit, Robustheit oder Kosten. Der Begriff Qualität ist stets in Abhängigkeit von dem zu betrachtenden Softwareobjekt (z. B.: Entwurfsdokument, Testprozeß), dem Blickwinkel (z. B.: Projektmanager, Kunde) und den relevanten Qualitätsmerkmalen (z. B.: Robustheit, Effizienz) zu sehen. Insofern variiert die relative Bedeutung einzelner Qualitätsmerkmale. “Kosten” könnte ein wichtiges Qualitätsmerkmal für den Testprozeß aus Sicht des Projektmanagers sein; Robustheit könnte ein wichtiges Qualitätsmerkmal des installierten Software-Systems aus Sicht des Benutzers sein. Anforderungen an Qualitätsmerkmale, sogenannte Qualitätsziele, bestehen aus einem oder mehreren Qualitätsmerkmalen, zugehörigen Metriken sowie quantitativen Kennziffern, die als Sollvorgaben dienen. Metriken sind Abbildungen, die meßbare Größen auf quantitative Ausprägungen von Qualitätsmerkmalen abbilden. Metriken und Soll-Werte sollten unter Berücksichtigung aller Dimensionen (Qualitätsmerkmal, Zweck, Softwareobjekt, Blickwinkel, Kontext) eines Qualitätsziels definiert sein. Verschiedene Qualitätsziele können vonein-

* Die Arbeit wurde durch die Deutsche Forschungsgemeinschaft im Rahmen des Teilprojekts B1 „Generische Modellierung von Prozessen und Experimenten” des Sonderforschungsbereichs 501 („Entwicklung großer Systeme mit generischen Methoden”) unterstützt.

ander unabhängig sein, sich gegenseitig ergänzen oder konfliktär sein (z. B. hohe Wartbarkeit könnte einem geringem Erstellungsaufwand widersprechen). Zielkonflikte können durch Priorisierung gelöst werden. Produktbezogene Qualitätsziele sind in der Regel vorrangig. Prozeßbezogene Qualitätsziele werden definiert, um zu einer hohen Produktqualität beizutragen. Dahinter steht die Erkenntnis, daß im Falle von planbaren, kosteneffizienten Entwicklungsprozessen hohe Produktqualität unmittelbar mit qualitativ hochwertigen Entwicklungsprozessen verbunden ist. Es gibt andere Möglichkeiten, qualitativ hochwertige Software zu erstellen. Dies gelingt vereinzelt mit hochqualifizierten Experten und ohne einen systematischen, explizit definierten Entwicklungsprozeß. Allerdings ist dies in der Regel mit erhöhten Kosten und erheblichen Nachteilen im Falle von Personalfluktuation verbunden. Eine andere Möglichkeit ist die Entwicklung nach dem “Open Source”-Ansatz, der jedoch nur für sehr wenige Anwendungsdomänen geeignet ist und systematische Planung nahezu ausschließt.

Qualitätssicherung und -management unterstützt die Erreichung von Qualitätszielen. Im Gegensatz zu industriellen Produktionsprozessen ist man in der Software-Entwicklung noch weit davon entfernt, die Erreichung vorgegebener Qualitätsziele garantieren zu können. Wesentliche Herausforderungen liegen hierbei zum einen in der schwierig zu kontrollierenden Variabilität von Entwicklungsprozessen (bedingt durch viele kreative Tätigkeiten) und zum anderen in ihrer kontext-abhängigen Gültigkeit (d. h., verschiedene Prozesse können für unterschiedliche Entwicklungskontexte gelten oder müssen geeignet an Entwicklungskontexte angepaßt werden).

Es gibt verschiedene Ansätze, die Softwarequalitätssicherung dahingehend zu verbessern, daß die Erreichung vorgegebener Qualitätsziele garantiert werden kann. So ist in einigen Software-Entwicklungsstandards (z. B.: V-Modell [9], ESA-Standard PSS-05 [10]) die Unterstützung der Qualitätssicherungsaufgaben durch ein eigens hierfür vorhandenes Vorgehensmodell realisiert. Andere Ansätze beinhalten die Vorgabe statischer und vermeintlich allgemeingültiger Sätze von relevanten Qualitätsmerkmalen. Sie sind vor dem Hintergrund der oben beschriebenen Vielschichtigkeit des Begriffes “Qualität” fragwürdig, und es ist kein signifikanter Beitrag zur Verbesserung der Softwarequalitätssicherung zu erwarten. Wiederum andere Ansätze sind durch die Einführung sogenannter Reifegrade einer Unternehmung gekennzeichnet, die jeweils durch eine Menge von Maßnahmen (z. B.: Einführung eines Konfigurationsmanagements, explizite Prozeßbeschreibung) erreicht werden können. Hierbei wird in der Regel auf eine systematische Untersuchung der Wirkungen eingeführter Maßnahmen verzichtet, so daß eventuell vorhandene Verbesserungen in Bezug auf die Erreichung von Qualitätszielen nicht nachweisbar sind.

Voraussetzung für eine entscheidende Verbesserung der Softwarequalitätssicherung ist ihre Integration in ein ingenieurmäßiges Software-Entwicklungsmodell [17]. Software-Entwicklung nach ingenieurmäßigen Grundsätzen erfordert a) die genaue Definition von Qualitätszielen für das zu entwickelnde Software-System, b) die Auswahl und geeignete Anpassung von Vorgehensmodellen, Techniken, Methoden und Werkzeugen zur Erfüllung der vorgegebenen Qualitätsziele und c) die Gewährleistung der Einhaltung vorgegebener Qualitätsziele durch geeignete Maßnahmen. Eine ausführliche Diskussion der ingenieurmäßigen Software-Entwicklung sowie sich daraus ergebender Konsequenzen für die Qualitätssicherung (insbesondere Qualitätsplanung, -lenkung und -prüfung) kann [17] entnommen werden. Eine wichtige Konsequenz ist, daß konstruktive und analytische Maßnahmen der Softwarequalitätssicherung sorgfältig aufeinander abgestimmt sein müssen. Qualitätssicherung betrifft insofern mehrere Aufgabenfelder, die integriert werden müssen: Im Bereich der analytischen Maßnahmen ist das Messen und Bewerten hervorzuheben, insbesondere die Identifikation und Definition relevanter Meßziele, die Datenerfassung und -validierung sowie die Analyse der erfaßten Daten. Einen Schwerpunkt der konstruktiven Maßnahmen der Qualitätssicherung stellt die Auswahl und Anpassung geeigneter Vorgehensmodelle, Techniken, Methoden und Werkzeuge dar, die die Erfüllung vorgegebener Qualitätsziele zu einem bestimmten Grad gewährleisten.

Für eine solche Auswahl und Anpassung ist es besonders wichtig, die Auswirkungen unterschiedlicher Vorgehensmodelle, Techniken, Methoden und Werkzeuge im Hinblick auf die Erreichung von Qualtitätszielen zu kennen. Erkenntnisse dieser Art können mit Hilfe von Experimenten gewonnen werden. Es gibt verschiedene Typen von Experimenten mit unterschiedlichem Grad der Kontrolle der Einflußfaktoren: Er ist gering bei Fallstudien und hoch bei formal definierten Experimenten. Dies ist jeweils mit den entsprechenden Vor- und Nachteilen bezüglich der statistischen Signifikanz und der Übertragbarkeit der gewonnenen Erkenntnisse verbunden.

Vorgehensmodelle enthalten Erfahrungen aus vergangenen Projekten in unterschiedlichem Maße. Sie lassen sich in zwei Klassen aufteilen:

- Einerseits gibt es *theoretisch ermittelte Vorgehensmodelle*, die in der Regel von Standardisierungsorganisationen beschlossen werden. Erfahrungen aus vergangenen Projekten fließen entweder nur sehr allgemein ein oder solche Vorgehensmodelle sind sehr speziell. Im ersten Fall sind Angaben über die Auswirkung ihrer Anwendung auf bestimmte Qualitätsmerkmale kaum möglich. Solche Modelle können als Orientierungshilfen für initiale Projektdurchführungen verwendet werden. Im zweiten Fall sind solche Modelle für die meisten Projekte nicht anwendbar.
- Andererseits gibt es *empirisch ermittelte Vorgehensmodelle*. Sie werden oftmals innerhalb einer Entwicklungsorganisation im Zusammenhang mit Meß- und Prozeßverbesserungsprogrammen in einzelnen Fallstudien ermittelt. Vergleiche solcher Modelle können zu abstrakteren Beschreibungen organisationsspezifischer Vorgehensmodelle führen. Gemeinsamkeiten in den Vorgehensmodellen deuten auf Kernbestandteile des Entwicklungsprozesses hin, Unterschiede können Variationsparameter bestimmen, und die in einzelnen Projekten ermittelten Meßdaten können Hinweise für Anpassungen der Modelle an unterschiedliche Projektcharakteristika geben.

2 Möglichkeiten zur Beschreibung anpaßbarer Vorgehensmodelle

Die Bereitstellung effektiver und effizienter Vorgehenspläne, die sowohl für ein Projekt maßgeschneidert sind als auch auf Erfahrungen aus vorangegangenen Projekten beruhen, ist wesentliche Voraussetzung zur Entwicklung qualitativ hochwertiger Software. Basierend auf den Erfordernissen ingenieurmäßiger Softwarequalitätssicherung werden in diesem Abschnitt wünschenswerte Möglichkeiten der Anpassung von Vorgehensmodellen sowie dafür notwendige Voraussetzungen beschrieben.

Die *Projektplanung* verwendet Modelle (d. h., sie instantiiert und verbindet sie), um eine Repräsentation des Projekts unter Berücksichtigung der Qualitätsziele und der Projektcharakteristika zu erstellen. Wesentlicher Bestandteil der Projektplanung ist die Erstellung eines Vorgehensplans (d. h., der Instanz eines Vorgehensmodells für ein bestimmtes Projekt) sowie einer Menge ergänzender Pläne (z. B.: Kosten- und Zeitpläne, Personalplan, Konfigurationsplan, Qualitätssicherungsplan, Risikomanagementplan zur Absicherung). Bei der *Vorgehensplanung* werden Prozesse und Produkte identifiziert, der Produktfluß bestimmt, Attribute definiert, Ein- und Ausgangsbedingungen formuliert sowie Techniken, Methoden und Werkzeuge integriert. Ein Vorgehensplan bestimmt den Projektverlauf maßgeblich und ist ein unverzichtbares Instrument für Projektmanager (zur Projektkontrolle), für Entwickler/Entwicklungsteams (zur Anleitung und Koordination) und für Qualitätsmanager (zur Verbesserung).

Die Anpassung von Vorgehensmodellen im Rahmen der Vorgehensplanung kann als Bestandteil eines Wiederverwendungsprozesses gesehen werden. Gängige Wiederverwendungsprozesse (siehe z. B.: [2]) bestehen in der Regel aus den Aktivitäten: Identifikation einer Menge von Wiederverwendungskandidaten, Evaluierung dieser Kandidaten, Anpassung gemäß Wiederverwendungsanforderungen und Integration in das umgebende System. Ein monolithisches Vorgehensmodell im Sinne einer vollständigen Beschreibung aller Prozesse eines Projekts eignet sich nur bedingt als Wiederverwendungsobjekt, da es aus einer Vielzahl von Bestandteilen zusammengesetzt ist, die bei der Wiederverwendung unterschiedlich behandelt werden können. Solche Bestandteile sind u. a. Lebenszyklusmodelle (z. B.: Wasserfall-Modell, Iterative Enhancement-Modell), Phasenmodelle, Teilprozesse, Methoden und Techniken. Es existiert keine einheitliche definitorische Abgrenzung dieser Bestandteile. Sie lassen sich jedoch alle durch sogenannte Prozeßmodelle beschreiben. Daher wird im folgenden von Prozeßmodellen gesprochen, die bei der Vorgehensplanung kombiniert und instantiiert werden. Prozeßmodelle beschreiben Typen von Prozessen. Prozesse repräsentieren konkrete Aktivitäten (z. B.: Komponenten-Entwurf) in einer Entwicklungsorganisation.

Reale Software-Entwicklungsprozesse variieren in Abhängigkeit von gegebenen Projektzielen und -charakteristika. Die Variabilität von Software-Entwicklungsprozessen bezieht sich zum einen auf die Beschaffenheit eines einzelnen Prozeßmodells (im Sinne eines Elementarobjekts) als auch auf die Kombination verschiedener Prozeßmodelle. Ansätze zur Vorgehensplanung sollten daher in der Lage sein, ausgehend von Zielen und Charakteristika eines Projekts und beruhend auf Erfahrungen aus vorangegangenen, ähnlichen Projekten zielführend geeignete Prozeßmodelle zu identifizieren, zu kombinieren, entsprechend dem aktuellen Projektkontext anzupassen und in einen Vorgehensplan zu integrieren. Existierende Planungsansätze eignen sich hierfür nur partiell. Angemerkt sei, daß die Vorgehensplanung ein dynamischer Prozeß

ist. Dies bedeutet insbesondere, daß Anpassungen von Prozeßmodellen zu verschiedenen Zeitpunkten stattfinden können (vor, während und nach der Projektabwicklung).

Ausgehend von einem Modell zur umfassenden Wiederverwendung von Software-Artefakten [2] werden im folgenden wünschenswerte Eigenschaften von Vorgehensmodellen (bzw. Prozeßmodellen) aufgeführt. Eine weitgehende Erfüllung dieser Eigenschaften verspricht, daß die Anpassung von Vorgehensmodellen zur Erreichung von Qualitätszielen beiträgt und somit als Maßnahme der ingenieurmäßigen Qualitätssicherung angesehen werden kann. Die Eigenschaften können als Anregung zur Gestaltung zukünftiger Vorgehensmodelle verstanden werden.

1. *Beschreibung des Gültigkeitsbereich*s. Ein einzelnes Prozeßmodell kann nicht als geeignet für alle Arten von Projekten angesehen werden. Daher sollte die Beschreibung eines Prozeßmodells jeweils um die Beschreibung seines Gültigkeitsbereichs erweitert werden (z. B.: mittels eines Charakterisierungsvektors), der das Prozeßmodell sowie den Kontext, in dem es eingesetzt werden kann, charakterisiert. Die Beschreibung des Gültigkeitsbereichs sollte variable Charakterisierungsschemata zulassen, d. h. nicht von der Annahme ausgehen, daß eine starre Menge von Attributen zur Charakterisierung ausreicht.

2. *Flexible Beschreibung von Prozeßmodellen.* Prozeßmodelle sollten so beschrieben sein, daß es möglich ist, sie an verschiedene Projektkontexte anzupassen. Hierbei sollte beschrieben sein, welche Teile eines Prozeßmodells variieren können und welche fix sind. Des weiteren sollte genau beschrieben sein, wie die Anpassung vorgenommen werden kann. Es reicht beispielsweise nicht, anzugeben, daß ein bestimmter Prozeß gestrichen werden kann. Da ein solcher Prozeß über den Kontroll- und Produktfluß mit anderen Prozessen verbunden ist sowie möglicherweise in einer Verfeinerungshierarchie eingebunden ist, muß genau angegeben werden, wie diese Beziehungen bei einem Wegfall des Prozesses angepaßt werden. Es gibt verschiedenste Arten der Anpassung von Prozessen. Eine Auswahl ist in Tabelle 1 wiedergegeben. Anleitungen zum Anpassen von Prozeßmodellen können durch Kommunalitätsanalyse (d.h.: systematischer Vergleich) empirisch basierter Prozeßmodelle einer Organisation oder Domäne gewonnen werden.

<table>
<tr><th>Prozeßaspekte</th><th>Produktaspekte</th><th>Ressourcenaspekte</th><th>Qualitätsaspekte</th></tr>
<tr><td>Prozesse streichen, hinzufügen, zusammenfassen, aufteilen, ersetzen</td><td>Produkte streichen, hinzufügen, zusammenfassen, aufteilen, ersetzen</td><td>Werkzeugunterstützung hinzufügen, entfernen</td><td>Prozeß-Sollvorgaben (z. B.: Aufwand, Kalenderzeit) ändern</td></tr>
<tr><td>Prozeßaggregation ändern</td><td>Produktaggregation ändern</td><td>Alternative Werkzeuge wählen</td><td>Produkt-Sollvorgaben (z. B.: Zuverlässigkeit, Komplexität) ändern</td></tr>
<tr><td>Kontrollfluß ändern</td><td>Alternative Beschreibungstechniken/Repräsentationen wählen</td><td rowspan="3">Personalzuordnung ändern (z. B. bezüglich Anzahl, Qualifikation, Rolle)</td><td rowspan="3">Modelle zur Vorhersage von Qualitätsmerkmalen ändern</td></tr>
<tr><td>Produktfluß ändern</td><td rowspan="2">Änderung der Inhalte von Produkten</td></tr>
<tr><td>Alternative Entwicklungs-Methoden/ -Techniken wählen</td></tr>
</table>

Tabelle 1: Arten der Anpassung von Vorgehensmodellen (Auswahl)

3. *Entscheidungsunterstützung für Anpassungen.* Mögliche Anpassungen sollten mit Bedingungen versehen sein, unter denen die Anpassungen vorgenommen werden können. Solche Bedingungen können beispielsweis an Unterschiede zwischen der Projektcharakterisierung und dem Gültigkeitsbereich des Prozeßmodells geknüpft sein. Man kann pragmatisch-orientierte Bedingungen von empirisch-basierten Bedingungen unterscheiden. Erstere ergeben sich aus unmittelbaren Projektzwängen (z. B.: ”Falls (Kritikalität = hoch) dann integriere Prozeß Bedrohungsanalyse”). Letztere sollten auf Erfahrungen der Organisation, insbesondere auf Ergebnissen aus Fallstudien und kontrollierten Experimenten beruhen (z. B.: ”Falls (Schnittstellenfehler = kritisch) dann ersetze Ad-hoc-Verifikation durch Kode-Lesen mittels schrittweiser Abstraktion”).

4. *Angaben über Auswirkungen*. Die Auswirkungen der Anwendung bestimmter Prozeßmodelle und möglicher Anpassungen im Hinblick auf die Erreichung von Qualitätszielen sollte beschrieben sein, sofern sie bekannt sind. Dies ist um so eher möglich, je mehr Erfahrungen mit einem Prozeßmodell innerhalb einer Domäne oder Organisation vorhanden ist. Beispielsweise könnte sich eine solche Angabe aus der Beschreibung der Beziehung zwischen der Komponentenkomplexität und den zu erwartenden Aufwänden für verschiedene Testmethoden zusammensetzen.

5. *Anpassungskosten*. Die Einführung neuer Prozesse bzw. die Änderung existierender Prozesse in einer Organisation ist mit Kosten verbunden (z. B. für Reorganisation, Umschulung der Entwickler, Anpassung des Produktmanagements). Die zu erwartenden Kosten sollten daher angegeben werden, falls Erfahrungswerte vorliegen. Es ist wichtig, eine Abwägung zwischen den Kosten für die Einführung/Änderung von Prozessen und dem Nutzen, der hiervon erwartet wird, durchzuführen. Dies entspricht in Analogie zum Wiederverwendungsprozeß der Evaluierungsphase.

6. *Kombinationen*. Anpassungen von Prozeßmodellen sollten nicht isoliert betrachtet werden. Beliebige Kombinationen von Anpassungen sind in der Regel auszuschließen. Mögliche Kombinationen sollten beschrieben sein (z. B. durch Abhängigkeitsgraphen). Ein Beispiel hierfür sind technische Abhängigkeiten. Technische Entwicklungsprozesse sind nicht beliebig kombinierbar, sondern es bestehen Abhängigkeiten zwischen ihnen. So ist z. B. die strukturierte Programmierung im Cleanroom-Ansatz Voraussetzung für die Anwendung der funktionalen Verifikationstechnik. Auch bei der Verfeinerung von Prozessen sind Abhängigkeiten zu berücksichtigen. Diese können z. B. durch definierte Verfeinerungsoperatoren beschrieben werden.

7. *Klassifikation und Ablage*. Die Ablage von Prozeßmodellen (z. B. in Repositories, Handbüchern, Hypertext-Dokumenten) sollte so organisiert sein, daß geeignete Suchmechanismen darauf aufsetzen können. Beispielsweise könnten Spezialisierungs- und Verfeinerungsbeziehungen in einer geeigneten Typhierarchie dargestellt werden, auf der der Auswahlprozeß aufsetzen kann.

8. *Werkzeuggestützte Konfigurierung von Prozeßmodellen*. Die Anpassung von Prozeßmodellen an Projektziele und -charakteristika ist eine komplexe Aufgabe. Die Änderung eines Prozeßmodells ist in der Regel mit einer Vielzahl an Folgeänderungen verknüpft, die notwendig sind, um die Konsistenz des angepaßten Prozeßmodells zu gewährleisten. Ein effektiver und effizienter Anpassungsvorgang sollte daher durch geeignete Beschreibungsformen (z. B.: parametrisierte formale Prozeßmodelle) und Anpassungsmechanismen (z. B.: Generierung, Transformation) unterstützt werden. Ein Beispiel wäre die Verwendung einer musterorientierten Beschreibung von Prozessen (im Sinne von Komponenten) verbunden mit einem Konfigurierungswerkzeug zur Adaption dieser Komponenten. Ein prototypisches Werkzeug zur Adjustierung von Prozeßmodellen ist in [15] zu finden.

9. *Veränderung der Anpassungsmöglichkeiten.* Die Möglichkeiten zur Anpassung von Prozeßmodellen sollten systematisch veränderbar sein, d. h., Erfahrungen einer Entwicklungsorganisation sollten in das Prozeßmodell einfließen können. Eine Änderung könnte beispielsweise die Berücksichtigung eines neuen Einflußfaktors sein, über dessen Auswirkungen man bisher noch keine Erfahrungen hatte.

10. *Verifikation von Vorgehensplänen.* Die erstellten Vorgehenspläne sollten auf ihre interne Konsistenz geprüft werden, d. h., es sollte sichergestellt sein, daß die Integration der ausgewählten Prozeßmodelle korrekt erfolgt ist. Für Vorgehenspläne, die in einer formalen Prozeßmodellierungssprache vorliegen, kann dies beispielsweise durch die Anwendung von Konsistenzregeln erfolgen.

3 Anpassung von Vorgehensmodellen in der Praxis

Kürzere Innovationszyklen sowie beschleunigte Veränderungen der Unternehmensumwelt zwingen leistungsfähige Organisationen einerseits, schnell und wirksam auf veränderte Rahmenbedingungen mittels veränderter Entwicklungsprozesse zu reagieren. Andererseits erfordert die zunehmende Bedeutung des Faktors “Software-Zuverlässigkeit” sowie die steigende Komplexität von Software-Systemen bei der Vorgehensplanung eine erhöhte Nutzung von explizit dokumentiertem Erfahrungswissen.

Demgegenüber steht die derzeitige Praxis bei der Vorgehensplanung. Zum einen fehlt systematische, zielführende Planung auf der Basis expliziter Prozeßmodelle. Zum anderen werden Projekte oftmals als Einzelprojekte geplant, ohne explizit dokumentierte Erfahrungen bezüglich des Vorgehens aus vergangenen,

ähnlichen Projekten zu nutzen. Die Erstellung von Zeit-/Kosten- und Personalplänen sowie eine grobe Aufgabenverteilung (z. B.: mittels Work breakdown structures) ist neben der Anwendung von Standards für Vorgehensmodelle bis heute oftmals die einzige Art der Planung von Software-Entwicklungsprojekten in der Praxis.

Betrachtet man allgemein die Anwendung von Software-Prozeßmodellierung in der Praxis, so lassen sich anlehnend an Gruhn [13] folgende Phasen identifizieren: 1) *Enthusiasmus* (1987-1990). Die Ideen und Konzepte der Darstellung von Entwicklungsaktivitäten in Form von Software-Prozeßmodellen (PM) wurden als geeignet angesehen und es gab kaum Zweifel, daß reale Prozesse formale Prozeßmodellierung und -abwicklung dringend benötigen. 2) *Frustration* (1990-1993). Es zeigte sich zwar, daß Prozeßmodellierungssprachen mächtig genug waren, um den Kern realer Software-Entwicklungsprozesse zu repräsentieren. Die geringe Reife industrieller Prozesse und die Fokussierung auf die Realisierung von Prozeßmaschinen (d. h.: Abwicklungsunterstützung) versprachen jedoch keine unmittelbare Rendite. Die Probleme bei der Abwicklungsunterstützung führten zu einer Verschiebung des Fokus: Die explizite Darstellung von Software-Entwicklungsprozessen an sich (in Form von Prozeßmodellen) sowie deren Einsatz bei der Projektplanung, -kontrolle, Entwickleranleitung und Prozeßverbesserung wurde als wesentlicher Nutzen der Software-Prozeßmodellierung betrachtet. 3) *Konsolidierung* (1992-1996). Die Schwäche bei der Unterstützung der Abwicklung von Software-Prozessen konnte durch folgende Einschränkungen teilweise behoben werden: Zum einen wurden Werkzeuge zur Abwicklung von Softwareprozessen erfolgreich für Geschäftsprozesse eingesetzt. Hierbei vereinfachte der geringe Anteil kreativer Tätigkeiten in Geschäftsprozessen die Unterstützung der Abwicklung. Zum anderen wurde die Abwicklung von Software-Prozessen auf spezielle Einzelprozesse (z. B.: Problem Tracking) begrenzt. 4) *Derzeitiger Stand (1996-).* Stand der Praxis bezüglich des Einsatzes von expliziten Prozeßmodellen ist, daß in einigen Organisationen Standards vorgegeben werden (internationale, nationale, organisationsspezifische) beziehungsweise ausgewählte Prozesse deskriptiv modelliert werden. Ersteres ist u. a. mit dem Problem der "nicht-passenden" Prozesse verbunden, letzteres führt häufig nicht zur Wiederverwendung von Prozeßmodellen in zukünftigen Projekten. Vereinzelte Ansätze zur Wiederverwendung von Prozeßmodellen in der Praxis existieren (z. B. [8]). Derzeit liegt der Fokus im Bereich "Wiederverwendung" jedoch auf Produkten.

Wie bereits oben erläutert, ist für die Verwendung von Standards im Rahmen ingenieurmäßiger Softwarequalitätssicherung besonders wichtig, daß sie an besondere Charakteristika und Ziele eines Projekts angepaßt werden können. Heutige Standards berücksichtigen diesen Aspekt jedoch nur sehr rudimentär. Im folgenden werden die Anpassungsmöglichkeiten des ESA Software Engineering Standards PSS-05 beschrieben, um zu illustrieren, wie Anpassungen von Vorgehensmodellen derzeit typischerweise in der Praxis vorgenommen werden. Anschließend erfolgt ein Abgleich mit den im vorangegangen Kapitel aufgelisteten wünschenswerten Eigenschaften für die Beschreibung von anpaßbaren Vorgehensmodellen.

Grundlage für die Untersuchung der Anpassungsmöglichkeiten des ESA-Standards PSS-05 ist zum einen die textuelle Beschreibung des Standards [10] sowie eine Beschreibung der vorzunehmenden Anpassungen für kleine Projekte [12]. In diesen Dokumenten sind folgende Anpassungsmöglichkeiten vorgesehen, zu denen hier jeweils ein Beispiel angegeben wird:

- *Zusammenfassen von Prozessen*. Beispiel: Für kleine Projekte (Entwicklungsaufwand < 2 Personenjahre, Größe des Entwicklerteams < 6 Personen, LoC < 10 K) wird die Zusammenlegung der Software-Requirements- (SR-) und der Architectural-Design-Phase vorgeschlagen. Begründet wird dies mit dem Wegfall eines zeitaufwendigen Reviews am Ende der SR-Phase, an dem laut Standard der Benutzer beteiligt ist. Die Anpassung erfolgt auf der obersten Ebene der Prozeßhierarchie. Eine genaue Angabe, wie die Zusammenlegung geschehen soll, erfolgt nicht. Es ist jedoch offensichtlich, daß auch Prozesse auf untergeordneten Abstraktionsebenen entfernt bzw. geändert werden müssen.

- *Hinzufügen von Prozessen*. Beispiel: Der Aktivität "Specification of the architectural design" kann die Unter-Aktivität "Inspection of the architectural design" hinzugefügt werden. Unter welchen Umständen die Methode bevorzugt angewendet werden sollte, wird nicht angegeben.

- *Entfernen von Prozessen*. Das Entfernen von Prozessen erfolgt als Folgeänderung, d. h., es ist Konsequenz anderer Anpassungsmaßnahmen. Beispiel: Die Aktivität "Update PHD" in der OM-Phase entfällt, wenn das Produkt "Project History Documents (PHD)" gestrichen wird.

- *Ändern des Kontrollflusses.* Beispiel: Es ist erlaubt, den Kontrollfluß der sechs Phasen entsprechend vorgegeber Lebenszyklus-Modelle anzupassen. Für die Anwendung einiger Lebenszyklus-Modelle werden Bedingungen angegeben (z. B.: Budgetrestriktionen, die nur eine partielle finanzielle Unterstützung für einen bestimmten Zeitraum erlauben, gelten als Merkmal für die Eignung des “Incremental delivery approach”).
- *Ändern des Produktflusses.* Beispiel: Im Falle, daß das zu entwickelnde System mit externen Hard- oder Softwarekomponenten kommunizieren muß, geht das Produkt “Interface Control Documents” als zusätzlicher Input in die Aktivität “Specification of the architectural design” ein.
- *Alternative Techniken/Methoden wählen.* Beispiel: Zur Konstruktion des “logical models” werden verschiedene Methoden angeboten (z. B. “functional decomposition”, “structured analysis”). Es gibt keine Empfehlungen, wann welche Methode zu bevorzugen ist.
- *Zusammenfassen von Produkten.* Beispiel: Für kleine Softwareprojekte wird empfohlen, die Dokumente SRD und ADD zusammenzufassen.
- *Aufteilen von Produkten.* Beispiel: Eine Aufteilung des Produkts “detailed design ducument” in mehrere Bände ist im Falle von sehr großen Software-Projekten erlaubt. Der Zweck ist wohl die Komplexitätsbeherrschung mittels Modularisierung.
- *Entfernen von Produkten.* Beispiel: Bei kleinen Projekten kann auf das Produkt “Project History Documents (PHD)” verzichtet werden.
- *Alternative Beschreibungstechniken/Repräsentationen.* Beispiel: Zur Beschreibung der Entwickler-Anforderungen (“software requirements”) können verschiedene Beschreibungstechniken angewendet werden (z. B. informell, Z, VDM).
- *Änderung der Inhalte von Produkten.* Beispiel: Für kleine Projekte wird empfohlen, einen Großteil der Detailed-Design-Informationen aus dem Dokument DDD in den Sourcecode (SDD) zu integrieren. Aus dem Standard geht nicht hervor, welchem Zweck diese Maßnahme dient.
- *Werkzeugunterstützung hinzufügen.* Beispiel: Beim “Unit testing” wird vorgeschlagen, daß bei großen Systemen Testwerkzeuge unterstützend eingesetzt werden sollen. Es ist nicht spezifiziert, was unter einem “großen System” zu verstehen ist.
- *Werkzeugunterstützung entfernen.* Beispiel: Bei sehr kleinen Projekten kann auf CASE-Werkzeuge zur Erstellung des “physical models” verzichtet werden. Zu beachten ist, daß die Werkzeuge als Ressource nicht gänzlich aus dem Vorgehensmodell entfernt werden dürfen, da sie noch an anderer Stelle benötigt werden.

Weitere Anpassungsmöglichkeiten sind im Standard nicht vorgesehen. Überprüft man den Standard hinsichtlich der wünschenswerten Eigenschaften aus Kapitel 2, so läßt sich feststellen, daß der Standard viele dieser Eigenschaften nur rudimentär erfüllt. Im einzelnen läßt sich zu den Eigenschaften folgendes feststellen: *Zu 1)* Mit Ausnahme eines starren Charakterisierungsschemas für kleine Projekte gibt es keine Angaben zum Gültigkeitsbereich des Vorgehensmodells. Bei einigen Methoden und Techniken wird auf Referenzen verwiesen, die solche Angaben möglicherweise enthalten. *Zu 2)* Es wird im Standard zwischen variablen und fixen Anteilen unterschieden: Es gibt zwingend vorgeschriebene Teile (“mandatory practices”), empfohlene Teile (“recommended practices”) und Teile, die ohne Angabe von Gründen entfallen können (“guideline practices”). Es ist unzureichend oder gar nicht spezifiziert, wie Anpassungen vorgenommen werden können. Folgeänderungen werden nicht beschrieben. *Zu 3)* Bedingungen für Anpassungen werden selten und dann nur sehr allgemein angegeben (siehe obige Beispiele). *Zu 4)* Angaben über Auswirkungen sind nicht im Standard enthalten. *Zu 5)* Es gibt sehr wenige Hinweise bezüglich der Kosten. So wird z. B. darauf hingewiesen, daß die Entwicklung von HW-Simulatoren für eingebettete Systeme mit erheblichem Mehraufwand verbunden ist. *Zu 6)* Abhängigkeiten zwischen Anpassungen werden nicht explizit beschrieben, sind jedoch vorhanden (siehe obiges Beispiel zum Zusammenfassen von Prozessen). *Zu 7)* Der Standard ist geordnet nach Phasen und Aktivitäten mit separaten Anhängen für Techniken und Methoden. Die Struktur eignet sich kaum für eine zielorientierte Vorgehensplanung, ist jedoch zur Verschaffung einer Übersicht über die Gesamtstruktur des Vorgehensmodells gut geeignet. *Zu 8)* Zur Überführung des Standards in einen Vorgehensplan ist die Beschreibung nicht geeignet. *Zu 9)* Über erlaubte Veränderungen des Standards basierend auf Erfahrungen ist nichts vermerkt. *Zu 10)* Auf einen Vorgehensplan wird nicht eingegangen; es werden keine Instanzen von Prozessen betrachtet.

Zusammenfassend läßt sich festellen, daß der ESA-Standard PSS-05 erste Ansätze zur Anpassung an Projektziele und -charakteristika enthält. Zur Erstellung eines Vorgehensplans unter Berücksichtigung der Erreichung von Qualitätszielen bietet er jedoch kaum Unterstützung.

4 Forschungsansätze zur Anpassung von Vorgehensmodellen

Ein historisch erster Lösungsansatz zur Erstellung eines Vorgehensplans wurde in der Entwicklung *"perfekter" Prozeßmodelle* gesehen, die für jede Kombination von Projektzielen und Kontextcharakteristika geeignet sind [3]. Heutige Ansätze verfolgen eine andere Richtung: Es werden Techniken zur Charakterisierung, Selektion (d. h. Auswahl) und Anpassung sowie Notationen zur Beschreibung von Prozeßmodellen entwickelt, mit denen Prozeßvarianten für konkrete Entwicklungskontexte erzeugt werden können.

Wesentliche Ansätze zur *Charakterisierung* und *Auswahl* von Prozeßmodellen stammen von Alexander und Davis [1], Boehm und Belz [3], Budlong et al. [6], Pietro-Diaz und Freeman [19] sowie Basili und Rombach [2]. Bei den ersten drei Ansätzen sind die Wiederverwendungsobjekte ausschließlich Prozeßmodelle, der Ansatz von Pietro-Diaz und Freeman kann auf Prozeßmodelle angewendet werden und der Ansatz von Basili und Rombach ist für jegliche Form von Software Engineering-Erfahrungen ausgelegt. Die Klassifikation der Prozeßmodelle erfolgt in [1], [3] sowie [6] nach einem starren Klassifikationsschema, in [19] facetten-orientiert und in [2] modell-orientiert. Die Ansätze verwenden als Selektionsmechanismen u. a. Vergleiche von Kontextvektoren mittels Matrizenberechnung [1], Entscheidungstabellen [3] oder Suchmechanismen mit Generalisierungs- und Erweiterungsmöglichkeiten [19].

Ansätze zur Anpassung sowie Notationen zur Beschreibung können entsprechend gängiger *Software-Wiederverwendungsansätze* klassifiziert werden, da nach Osterweil [16] zwischen der Programmierung von Software und der Modellierung von Prozessen Parallelen bestehen. Es lassen sich prinzipiell zwei Gruppen von Ansätzen unterscheiden. Eine Gruppe versucht, durch das Zusammenfügen von Prozeßbausteinen ein Gesamtmodell zu erstellen (z. B.: [6]). Diese Bausteine werden dabei idealerweise unverändert aus früheren Entwicklungen übernommen. Solche Ansätze lassen sich unter dem Stichwort "Komposition" zusammenfassen. Die zweite Gruppe von Ansätzen versucht, aus einer Projektcharakterisierung halb- bzw. vollautomatisch ein ausführbares Prozeßmodell zu generieren (z. B.: Process Model Generator [3]). Dabei werden nicht Bausteine, sondern vielmehr typische Strukturen oder Verwendungsmuster wiederverwendet.

Die Ansätze, die mittels Komposition Wiederverwendung unterstützen, fallen in zwei Kategorien: Die Verwaltung von Bibliotheken von Komponenten sowie die Prinzipien zur Konstruktion von Komponenten. Ersteres untersucht, wie große Mengen von Komponenten abgelegt und verwaltet werden können und ist schwerpunktmäßig ein technisches Problem. Letzteres untersucht, wie sich Komponenten so formulieren lassen, daß sie mit minimalen Anpassungen zu größeren Komponenten zusammengefügt werden können. Diese Konstruktionsprinzipien drücken sich in Prozeßmodellierungssprachen durch Konstrukte zur Darstellung von Generezität aus. Existierende Prozeßmodellierungssprachen bieten in geringem Umfang solche Konstrukte (z. B.: Parameter, Optionalitätsangaben, bedingte Anweisungen).

Unter den generierungsbasierten Ansätzen finden sich verschiedene Schwerpunkte. Allgemein akzeptierte Konzepte sind hier allerdings nicht deutlich sichtbar. Wesentliche Schwerpunkte sind die architekturbasierte Wiederverwendung, Generatoren sowie Transformationssysteme. Die Anwendung von generierungsbasierten Ansätzen auf Prozeßmodelle ist in der Literatur bisher nur vereinzelt vorzufinden (ein Beispiel für Lebenszyklusmodelle findet sich in [3]).

Im Bereich der *Künstlichen Intelligenz* bedeutet Planung die Konstruktion einer Folge von Aktionen, um, ausgehend von einer Startsituation, eine festgelegte Menge von Zielen zu erreichen. Ein Planungsproblem besteht (vereinfacht) aus einem Startzustand und einer Menge von Zielen. Eine Planungsaufgabe besteht darin, einen Lösungsplan zu finden. Der Lösungsplan setzt sich aus einer Folge von Aktionen zusammen, die den Startzustand in einen Endzustand transformieren, in dem die Ziele erfüllt sind. Aktionen transformieren Zustände in neue Zustände. Normalerweise wird ein Zustand durch eine Menge logischer Formeln dargestellt. Aktionen werden durch sogenannte Operatoren beschrieben. Planungssysteme (sog. "planner") sind Programme, die Planungsaufgaben bearbeiten. Man unterscheidet Generatives Planen von Fallbasiertem Planen. Ziel der Generativen Planung ist es, die Lösung eines gegebenen Planungsproblems durch eine systematische Suche im Raum der möglichen Operatoren zu generieren. Ziel des Fallbasierten Planens ist es, die Effizienz der Plangenerierung zu verbessern, indem die Ähnlichkeit zwischen einem neuen Problem und vorangegangenen Problemlösungs-Situationen beurteilt wird und ähnliche Problemlösungen

wiederverwendet werden. Derzeitige KI-Planungssysteme sind gekennzeichnet durch einen hohen Automatisierungsgrad, geringe manuellen Eingriffsmöglichkeiten bei der Planerstellung und fehlenden Möglichkeiten, empirische Modelle zur Auswahl und Anpassung von Planbausteinen zu verwenden.

Ansätze des *Domain Engineering (Product Line Approach)* berücksichtigen die Wiederverwendung hauptsächlich bei der Abwicklung eines Projekts. Die Wiederverwendung von Prozeßmodellen hat erst in einige wenige Methoden (z. B.: Organizational domain modeling, kurz ODM) Einzug gehalten. Schwerpunkt dieser Ansätze ist die Entwicklung eines Domänen-Modells (zur Unterstützung der Requirements-Phase), einer Domänen-Architektur (zur Unterstützung der Design-Phase) und einer Asset-Base (zur Unterstützung der Implementierung und Wartung). Eine Asset-Base für die Vorgehensplanung ist in bisherigen Ansätzen nicht vorgesehen.

Klassische Projektplanungsmethoden lassen sich nur teilweise für die Vorgehensplanung einsetzen (z.B: Work breakdown structures) und sind in der Regel für Kosten- und Zeit sowie Personalplanung geeignet (z.B.: Critical path method). Eine hybride Planungstechnik ist *Structured planning*, die auf WBS und Structured Analysis aufbaut. Sie bietet eine übersichtliche Möglichkeit, Prozesse zu verfeinern und dies geeignet darzustellen.

Existierende *Prozeßmodellierungssprachen* verfügen oftmals über Konstrukte, mit denen Informationen zur Anpassung eines Projektmodells an verschiedene Projektkontexte im formalen Prozeßmodell beschrieben werden können. Eine ausführliche Beschreibung der Möglichkeiten, die die Prozeßmodellierungssprache MVP-L bietet, findet sich in [20]. Zu einigen Prozeßmodellierungssprachen existieren Werkzeugumgebungen, die verschiedene Funktionen unterstützen (z. B.: Modellierung, Abwicklung, Risikoanalyse). Wiederum einige dieser Umgebungen (z. B.: FUNSOFT net approach) beinhalten *Process Model Repositories*, die allerdings hauptsächlich die Verwaltung von Prozeßmodellen während der Abwicklung übernehmen.

5 Ausblick

Die aufgestellten Anforderungen an die Beschreibung von anpaßbaren Vorgehensmodellen im Rahmen der ingenieursmäßigen Softwarequalitätssicherung können als Idealziel angesehen werden, dem man sich am ehesten in lernenden Organisationen nähern dürfte, die organisationsspezifische Prozeßmodelle anwenden und kontinuierlich verbessern. Ein höherer Grad der Strukturierung und Formalisierung von Prozeßbeschreibungen kann zur Erreichung der Anforderungen beitragen. Derzeit erfolgt die Charakterisierung der Qualitätsziele 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. Werkzeuge, mit denen Anpassungen an formalen Prozeßmodellen vorgenommen werden können, sind derzeit in Entwicklung (z. B. das Werkzeug ProTail [15]). Angepaßte Prozeßmodelle können als unmittelbarer Input für eine Software-Entwicklungsumgebung verwendet werden. Im Rahmen des Sonderforschungsbereichs 501 wird derzeit die Prozeßunterstützungsumgebung MILOS [[7],[22]] entwikkelt, die durch Möglichkeiten zur dynamischen Umplanung Anpassungen 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.

Danksagung. Der Autor dankt Björn Schmidt für die sorgfältige Modellierung des ESA-Standards PSS-05 in der Prozeßmodellierungssprache MVP-L sowie die hierbei durchgeführte vollständige Ermittlung der vorgesehenen Anpassungsmöglichkeiten.

Referenzen

[1] Linda C. Alexander, Alan M. Davis, “Criteria for Selecting Software Process Models”, Proceesings of the 15th Annual International Compute Software and Applications Conference, IEEE Computer Society Press, Tokyo, pp. 521-528, 11.-13. September 1991.

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

[3] Barry Boehm, Frank Belz, “Experiences with the Spiral Model as a Process Model Generator”, Proceedings of the 5th International Software Process Workshop, IEEE Computer Society Press, Washington DC, pp. 43-45, 1990.

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

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

[6] Faye C. Budlong, Paul A. Szulewski, Ralph J. Ganska, "Process Tailoring for Software Project Plans", Software Technology Support Center (STSC), Hill AFB, Utah (USA), January 1996.

[7] Barbara Dellen, Frank Maurer, Jürgen Münch, Martin Verlage, „Enriching Software Process Support by Knowledge-based Techniques", special issue of Int. Journal of Software Engineering and Knowledge Engineering, 1997.

[8] M. Dyer, T. Kraly, F. Luppino, R. Waldron, "Integrating an enterprise's engineering processes". Information and Software Technology, 35 (6/7): 355-363, June 1993.

[9] Entwicklungsstandard für IT-Systeme des Bundes ("V-Modell´97"), Allgemeine Umdrucke 250-252, Bundesministerium der Verteidigung, 1997.

[10] ESA Software Engineering Standards, PSS-05-{0-7}, Issue 2, ESABoard for Software Standardisation and Control (BSSC), European Space Agency, February 1991.

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

[12] Guide to applying the ESA Software engineering standards to small projects. BSSC-96-2, ESABoard for Software Standardisation and Control (BSSC), European Space Agency.

[13] Volker Gruhn, Juri Urbaincyk, "Software Process Modeling and Enactment: An Experience Report related to Problem Tracking in an Industrial Project", ICSE 1998.

[14] 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.

[15] Jürgen Münch, Markus Schmitz, Martin Verlage, "Tailoring großer Prozeßmodelle auf der Basis von MVP-L", In Sergio Montenegro, Ralf Kneuper, Günther Müller-Luschnat (Eds): Proceedings of the 4. Workshop der Fachgruppe 5.1.1 (GI): Vorgehensmodelle - Einführung, betrieblicher Einsatz, Werkzeug-Unterstützung und Migration, Berlin, Germany, March 17-18, 1997.

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

[17] Dieter Rombach, "Software-Qualität und -Qualitätssicherung", Informatik-Spektrum, 16: 267-272, 1993.

[18] 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.

[19] Ruben Prieto-Diaz, Peter Freeman, "Classifying Software for Reusability", IEEE Software, IEEE Computer Society, Vol. 4, No. 1, pp.6-16 January 1987.

[20] Markus Schmitz, "Transformationsbasiertes Zuschneidern von Prozeßmodellen", Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, 1997.

[21] 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.

[22] 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.

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

[24] Martin Verlage, Jürgen Münch, "Formalizing Software Engineering Standards", In Proc. of the Third International Symposium and Forum on Software Engineering Standards (ISESS'97), California, USA, June 1-6, 1997. Also available as Technical Report SFB-501-TR-10-96.