Systematische Integration von Prozeß- und Produktmanagement

Stefan Vorwieger

Universität Kaiserslautern, Fachbereich Informatik
SFB 501 "Entwicklung großer Systeme mit generischen Methoden"
Postfach 3049
D-67653 Kaiserslautern
vorwiege@informatik.uni-kl.de

Zusammenfassung. Versions- und Konfigurationsmanagement sind zentrale Instrumente zur intellektuellen Beherrschung komplexer Softwareentwicklungen. In stark wiederverwendungsorientierten Softwareentwicklungsansätzen —wie vom SFB 501 bereitgestellt— muß der Begriff der Konfiguration von traditionell produktorientierten Artefakten auf Prozesse und sonstige Entwicklungserfahrungen erweitert werden. In dieser Veröffentlichung wird ein derartig erweitertes Konfigurationsmodell vorgestellt. Darüberhinaus wird eine Ergänzung traditioneller Projektplanungsinformationen diskutiert, die die Ableitung maßgeschneiderter Versions- und Konfigurationsmanagementmechanismen vor Projektbeginn ermöglichen.

1 SFB 501: "Entwicklung großer Systeme mit generischen Methoden"

Am 1. Januar 1995 wurde von der Deutschen Forschungsgemeinschaft der Sonderforschungsbereich 501 im Fachbereich Informatik der Universität Kaiserslautern eingerichtet. Sein Thema ist die systematische Konstruktion und Fertigung von Software nach ingenieursmäßigen Prinzipien. Im Zentrum stehen Methoden, die eine systematische Wiederverwendung von Software, den zu ihrer Fertigung benutzten Prozessen sowie sonstiger Erfahrungen zum Gegenstand haben [3, 6, 7].
Heutige Verfahren zur Softwareentwicklung weisen im allgemeinen die folgenden Defizite auf:

- In der Planung wird von einer initial vollständigen Problemspezifikation ausgegangen. Diese ist jedoch üblicherweise initial unvollständig und laufenden Änderungen während der Projektdurchführung unterworfen.
- Eine Wiederverwendung von Software-Entwicklungsartefakten findet unsystematisch und meist nur auf Kode-Ebene beschränkt statt.

Die Ideen des SFB 501 beruhen zum einen darauf, den Umgang mit Unvollständigkeiten und Änderungen durch die Entwicklung generischer Beschreibungstechniken, die fixe von variablen Teilen unterscheiden (s. z.B. [8]), zu erleichtern. Des weiteren wird angestrebt, Wiederverwendung auf alle SE-Erfahrungen[1] auszuweiten und Methoden

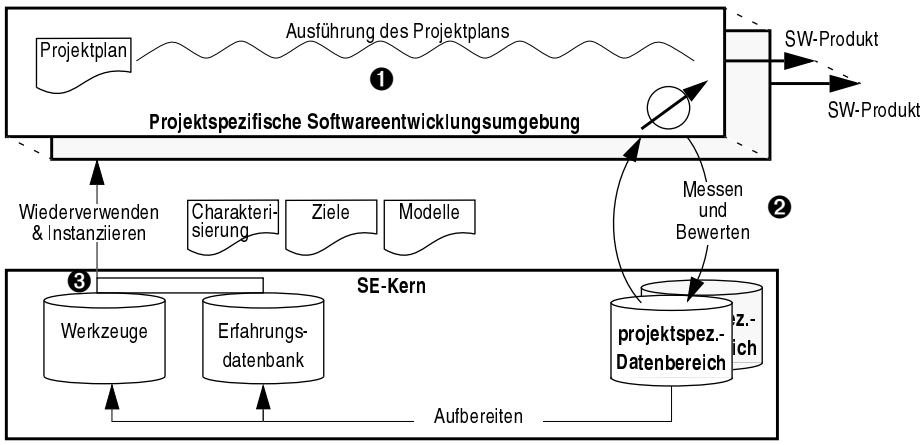

Abb. 1: Rahmenarchitektur einer domänenspezifischen Softwareentwicklungsumgebung

für die kontinuierliche Erfassung und Wiederverwendung von SE-Erfahrungen anzubieten. Ziel ist es, die Entwicklung von Software-Systemen in einer überschaubaren Domäne mit weniger Aufwand, höherer Qualität und in kürzerer Zeit zu ermöglichen. Die empirische Bestätigung dieser Annahmen ist ebenfalls Teil der Aufgaben des SFB 501.

Zum Erreichen dieses Ziels wird ein SE-Modell [1] (s. Abb. 1) zugrunde gelegt, das die Rahmenarchitektur einer domänenspezifischen Softwareentwicklungsumgebung beschreibt und auf den folgenden drei Prinzipien aufbaut: ❶ Es unterstützt die Planung und Ausführung eines Projekts auf der Basis expliziter Projektpläne. ❷ Es erlaubt die meßbasierte Projektverfolgung und -kontrolle und ❸ Es ermöglicht projektübergreifendes Lernen durch Erfahrungsaufbereitung und -wiederverwendung.

Die folgenden Kapitel beschreiben meine Forschungsarbeit zum Thema “Integriertes Produkt- und Prozeßmanagement” und deren Einbettung in das SE-Modell des SFB 501. Eingeschränkt auf den projektspezifischen Datenbereich des SE-Kerns (s. Abb. 1) werden im zweiten Kapitel die Anforderungen an ein erweitertes Konfigurationsmodell aufgestellt. Kapitel drei leitet davon die Zielsetzung der eigenen Forschung ab: die Erweiterung des Konfigurationsbegriffs sowie die Integration des Produktmanagements in die Projektplanung. Kapitel vier und fünf beschreiben einen konzeptionellen Lösungsansatz zur Realisierung dieser Ziele.

2 Anforderungen an ein umfassendes Produktmanagement

Die Verwaltung von Artefakten der Software-Entwicklung ist eine zentrale Komponente bei der Entwicklung großer Software-Systeme [2]. Versions- und Konfigurati-

1. Als SE-Erfahrungen werden im SFB 501 alle wiederverwendbaren Artefakte verstanden. Dies sind neben produktorientierten Entwicklungsartefakten insbesondere Meßdaten, die projektbegleitend erfaßt werden, qualitative Erfahrungen (wie z. B. Lessons learned), und aus diesen Erfahrungen abgeleitete Modelle wie Prozeß-, Produkt- oder Qualitätsmodelle.

onsmanagement[2] sind Verwaltungsprozesse, welche die Projektkontrolle (z.B. durch Koordination von Teams) als auch die Arbeit der Entwickler (z.B. durch Verwaltung von Arbeitsbereichen) unterstützen.

Die Verwaltung von Produkten beschränkt sich in Umgebungen, die nicht explizit als wiederverwendungsorientierte Entwicklungsumgebung ausgelegt sind, auf die Verwaltung von produktorientierten Artefakten wie Source-Code, Entwürfe oder Anforderungsbeschreibungen (s. ⓐ in Abb. 2). In wiederverwendungsorientierten Umgebungen werden jedoch neue Anforderungen an die Verwaltung von Produkten gestellt:

- *Verwaltung aller SE-Erfahrungen*: Das Produktmanagement muß **alle** Informationen, die während der Projektplanung und -durchführung anfallen, berücksichtigen:
  - *Meßdaten und qualitative Erfahrungen*: Zum Zeitpunkt der Projektdurchführung sind zum einen *Meßdaten* zu berücksichtigen, die neben der projektinternen Kontrolle zum Zwecke des Lernens über Projekte hinweg zielgerichtet erfaßt werden. Zum anderen werden *informell erfaßte Erfahrungen* (im weiteren *qualitative Erfahrungen* genannt) abgelegt, in denen Auffälligkeiten oder Probleme, deren Kontext, Ursache und Symptome sowie Lösungsempfehlungen dokumentiert sind (ⓑ). Beispielsweise ist die folgende Anfrage der Post-Mortem-Analyse zur Aufdeckung von Schwachstellen sinnvoll: *Liefere alle Aufwandsdaten zu derjenigen Überarbeitung (Prozeßmaß) des Teilsystems GUI-1, in der Version 3.4 der Komponente "Window-2" erstellt wurde*. Es wird ein Prozeßmaß erfragt, dessen zugehörige Prozeßinstanz nur aus dem Bezug zu einer Version des bearbeiteten Produkts abgeleitet werden kann.
  - *Modelle und Projektplan*: Gleichermaßen wie zu Projektbeginn eine initial nur unvollständige Aufgabenbeschreibung möglich ist, ist folglich auch die Planung eines Projekts initial unvollständig und kann erst im Verlauf der Durchführung vervollständigt und verfeinert werden. Die Dokumentation des Prozesses der "Umplanung" ist für das Verständnis einer Projektentwicklung

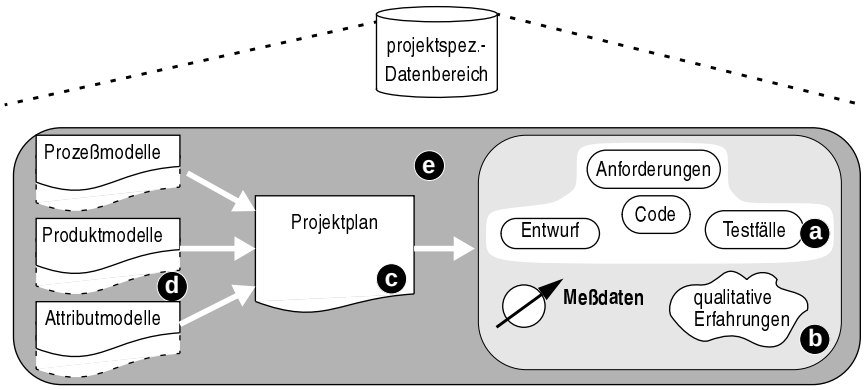

Abb. 2: Inhalte des projektspezifischen Datenbereichs

2. Im weiteren wird der Begriff *Produktmanagement* bzw. *Verwaltung von Produkten* im Sinne einer Disziplin mit Schwerpunkt auf Konfigurations- und Versionsmanagement verwendet.

sowie für das Lernen über Projekte hinweg maßgeblich. Für die Post-Mortem-Analyse, die u.a. Modelle für nachfolgende Projekte erstellt, ist jedoch eine reine Dokumentation der Projektplanungsevolution (Projekttrace) nicht ausreichend. Unterschiedliche Modelle und Projektpläne, die zwischenzeitlich Gültigkeit hatten, müssen reproduzierbar und deren Unterschiede entsprechend dokumentiert sein[3].

- *Konsistente Planung von Entwicklungsprozessen und Produktmanagement*: Für die Planung von Projekten werden im SFB 501 unterschiedliche Artefakte erstellt: *Modelle*, welche Abstraktionen von Produkten und Prozessen beschreiben sowie deren Instanziierung und Integration zu einem *Projektplan*, welcher die Basis für die Abwicklung eines Projekts darstellt. Modelle beschreiben den prinzipiellen Ablauf der Entwicklung sowie Abhängigkeiten zwischen Prozessen, Produkten, Ressourcen und Maßen. Sie sind ein geeignetes Mittel zur Darstellung von SE-Erfahrungen und unterstützen die Wiederverwendung durch eine an das konkrete Projekt maßgeschneiderte Instanziierung zum Projektplan.
  Die Planung von Mechanismen des Produktmanagements orientiert sich derzeit an der zu erstellenden Produktstruktur und läßt Entwicklungsprozesse weitgehend unbeachtet. Das Produktmanagement sollte jedoch seine Mechanismen an die Art des durchzuführenden Entwicklungsprozesses anpassen. So darf bspw. einer Testerin die zu testende Code-Komponente nicht schreibbar verfügbar gemacht werden, obwohl Codierern in der Implementierungsphase die Bearbeitung erlaubt ist.

3 Zielsetzungen

Aus den Anforderungen lassen sich u.a. folgende Ziele für ein umfassendes Produktmanagement ableiten:

- *Erweiterung des Konfigurationsbegriffs:* Für die *Verwaltung der Meßdaten und qualitativer Erfahrung* werden Mechanismen etabliert, mit denen Meßdaten Prozessen und Produkten zugeordnet, diesbezüglich abgelegt und wiedergefunden werden können. Meßdaten werden mit Versionen und Konfigurationen von Produkten und Prozessen in Verbindung gebracht.
  Die *Verwaltung von Modellen und dem Projektplan* erlaubt eine Dokumentation der Modell- und Projektplanänderungen, die Anforderungen der Post-Mortem-Analyse genügen. Des weiteren wird eine Differenz-Reportfunktion für Modelle und Projektpläne zur Verfügung gestellt, die geänderte und unveränderte Modelle sowie Änderungen im Projektplan angibt.
- *Integration des Produktmanagements in die Projektplanung*: Um das Prinzip des empirischen Lernens (d.h., kontrollierte Wiederverwendung über Modelle) auch auf das Produktmanagement zu erweitern, wird die Projektplanung um Konzepte des Produktmanagements erweitert (Ⓒ u. Ⓓ in Abb. 2). Die Erweiterung betrifft hauptsächlich Produkt-, Prozeß- und Attributmodelle und fügt weitere Modelle

3. Die Evolution der Modelle und des Projektplans während der Durchführung der Entwicklung wird im folgenden *Projektevolution* genannt.

hinzu. Sie dient als Basis für eine Formalisierung der Produktmanagement-Planung und stellt —durch gemeinsame Instanziierung zu einem Projektplan— die Konsistenz zwischen Produkt- und Prozeßplanung sicher. Eine Plattform zur Unterstützung des geplanten Produktmanagements wird zur Verfügung gestellt ("generiert").

4 Erweiterung des Konfigurationsbegriffs

Maße als Modell von Meßdaten lassen sich auf Modell-Ebene als Attribute von Prozeßmodellen und Produktmodellen beschreiben. Für die Erfassung während der Ausführung muß jedoch ein Mechanismus etabliert werden, der bspw. die Bewahrung von Maßen gelöschter Produkte, die Beziehung zu Prozessen oder die mehrfache Erfassung von Erfahrungen zu demselben Objekt[4] erlaubt. Daher werden Maße als eigenständige Produkte verwaltet, denen zum Erfassungszeitpunkt (s. *Trigger* in Abb. 3) ein Objekt zugeordnet werden kann (s. *Objekt-Beziehung* in Abb. 3). Für Meßdaten geschieht diese Zuordnung zu Objekten im Projektplan automatisch gemäß der Beziehungen im Prozeßmodell. Zu qualitativen Erfahrungen werden zum Erfassungszeitpunkt alle Objekte, die in der Kontextbeschreibung maßgeblich sind, eingetragen. Referenzierbar sind konkrete (versionierte) Artefakte, Elemente aus dem Projektplan und Modelle im allgemeinen.

Zur *Definition* eines Meßdatums gehört neben dem Datentyp (kompatibel zum Typ des Attributs im Prozeßmodell) auch die Angabe von Defaultwerten. Die *Definition* einer qualitativer Erfahrung enthält die Felder, die zu deren Dokumentation benötigt werden (z.B. Problem, Ursache, Lösungsvorschlag). Ebenso wie Entwicklungsprodukten können auch Maßen und qualitativer Erfahrung weitere Attribute zugewiesen werden, wie z.B. das Datum der Erfassung oder der Erfasser.

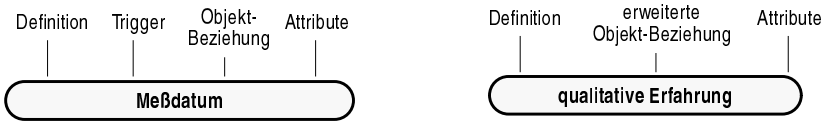

Abb. 3: Elemente zur Verwaltung von Meßdaten und qualitativer Erfahrung

Die explizite Dokumentation von Prozeß-, Produkt- und Qualitätsmodellen sowie deren Integration zu einem Projektplan erlaubt deren Verwaltung als Produkte, d.h. auf ihnen ist Versions- und Konfigurationskontrolle möglich. Eine einfache, syntaktische Versionierung ist jedoch nicht ausreichend. Die gezielte Annotation der Projektevolution ist essentiell für eine Post-Mortem-Analyse, da sie die Ursachenbeschreibung für eine Änderung zur Bewertung einzelner Evolutionsstufen benötigt.

Im folgenden werden zwei Arten von Änderungen unterschieden: *Planänderungen ohne Änderung der Modelle* sowie *Änderung der Modelle mit anschließender Ände-*

4. Unter Objekt werden in diesem Abschnitt Produkte, Prozesse sowie jegliche Art von Modellen verstanden.

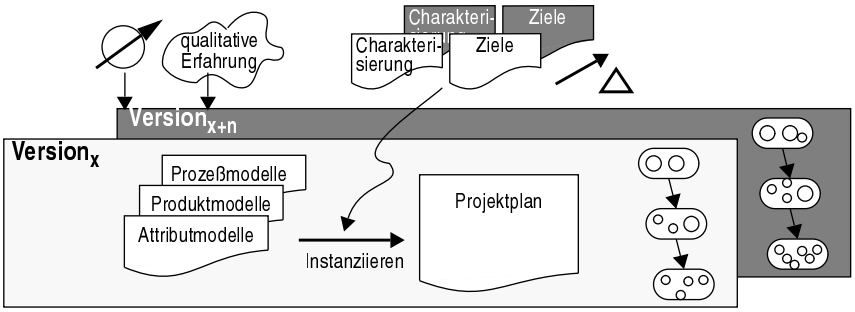

Abb. 4: Projektevolution: Verwaltung von Planungs- und Ausführungsartefakten als Konfiguration

rung des Plans. *Planänderungen ohne Änderung der Modelle* werden hauptsächlich durch Änderung der Zielsetzung (z.B. Änderung von Milestones) oder des Kontextes (d.h. Änderungen im Parametersatz für die Instanziierung) eingeleitet. Diese werden zu der Beschreibung der Änderung an den Projektplan angehängt und mitversioniert (s. Abb. 4). Die *Änderung eines der Modelle* wird häufig durch projektbegleitende Analyse von Meßdaten oder durch das Erkennen einer ungünstigen Situation, die in Form qualitativer Erfahrung notiert wird, initiiert. Diese werden an die neue Version des Modells angehängt. In beiden Fällen von Änderungen wird eine neue Version der Konfiguration bestehend aus allen Modellbeschreibungen und dem Projektplan erzeugt.

5 Integration des Produktmanagements in die Projektplanung

Im weiteren werden die Konzepte der Prozeßmodellierung[5] anhand der im SFB 501 zur Projektplanung eingesetzten Sprache MVP-L[6] erläutert und gezeigt, welche Erweiterungen zur Integration des Produktmanagements sinnvoll sind.

Bei der Prozeßmodellierung steht die Modellierung des Entwicklungsprozesses im Mittelpunkt (s. Abb. 5); weitere Elemente wie Produkte (z.B. Source Code), Ressourcen (z.B. ausführende Personen), Attribute (z.B. meßbare Eigenschaften) werden ebenfalls modelliert. Produkte können von Prozessen produziert und konsumiert werden. Ressourcen können Prozessen zugeordnet werden. Attribute können für Produkte, Prozesse und Ressourcen definiert werden. Attribute können auch zur Formulierung von Entry- und Exit-Bedingungen genutzt werden [4].

Für die Erweiterungen[7] mit Mechanismen des Produktmanagements können *Ressourcen* auch Produkten zugeordnet werden[8]. Dies erscheint sinnvoll, da in einem Prozeß auf unterschiedliche Produkte von unterschiedlichen Werkzeugen und Entwicklern

5. Die grafische Darstellung sowie die Konzepte sind der Prozeßmodellierungssprache MVP-L entlehnt [5].
6. Es wird davon ausgegangen, daß MVP-L alle wichtigen Konzepte zur präskriptiven Modellierung von Entwicklungsprozessen besitzt.
7. Abb. 6 zeigt die Erweiterungen durch dunklere Färbung an.
8. In MVP-L können Ressourcen nur Prozessen zugeordnet werden.

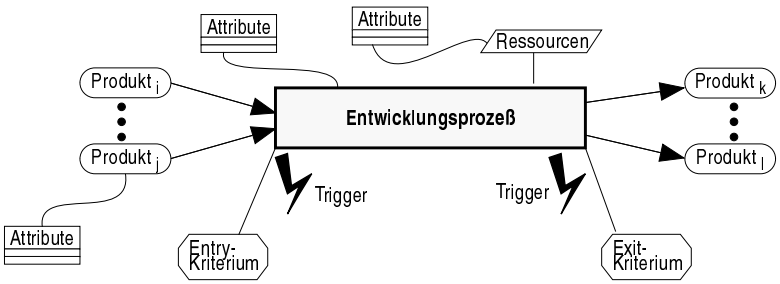

Abb. 5: Elemente eines Entwicklungsprozeßmodells

zugegriffen werden kann. Die Konsistenz zu Ressourcen des Prozesses und den daran beteiligten Produkten muß gewährleistet sein: Die Produktressourcen müssen eine Teilmenge der Prozeßressourcen sein.

Im *Versionenmodell* wird beschrieben, wie Versionsnummern gestaltet sind und welche Inkrementfunktionen existieren, um Versionsnummern zu manipulieren. Diese Funktionen können ebenfalls durch Trigger im Prozeßmodell aufgerufen werden, um automatische Versionierung zu unterstützen. Gleiches gilt für Versionierung von Konfigurationen als ein Mittel zur Fixierung von konsistenten Zuständen.

Das *Attributmodell* für Produkte wird um Werte erweitert, die für das Produktmanagement erfaßt werden müssen, bspw. beim Erzeugen einer neuen Version. Neben der üblichen Angabe des Typs eines Attributs muß festgelegt werden, ob das Attribut für eine Version oder das gesamte Produkt Gültigkeit hat. Versionierte Attribute müssen beim Erzeugen einer neuen Version erneut bestimmt werden, z. B.: Datum der Versionserzeugung (versioniert), Kurzbeschreibung des Produkts (nicht versioniert). Des weiteren ist es wünschenswert, manche versionierte Attribute ändern zu können, ohne die Versionsnummer zu erhöhen (Beispiel: der Status eines Produkts kann sich nach einem erfolgreichen Test ohne Fehler von ‘testable’ auf ‘tested-ok’ ändern). Das Attributmodell ist ebenso anwendbar auf versionierte Konfigurationen.

Das *Typ-Modell* für Produkte dient hauptsächlich zur Einstellung eines Produktmanagement-Werkzeugs und soll daher hier nicht weiter ausgeführt werden. Im *Typ-Modell* einer Konfiguration werden die Elemente einer Konfiguration definiert. Neben einer aufzählenden Liste von Produkten kann auch eine Liste von Prozessen angege-

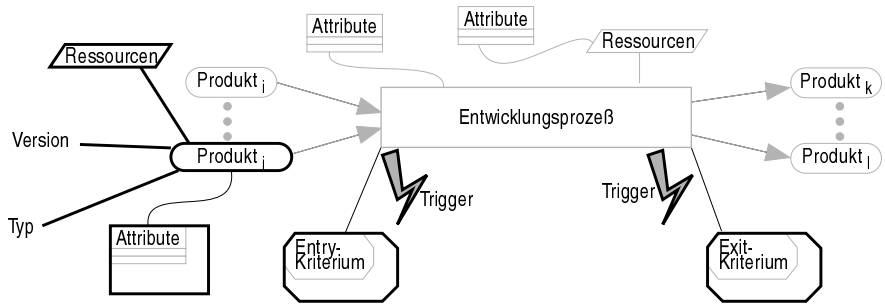

Abb. 6: Erweiterung eines Prozeßmodells um Produktmanagement-Konzepte

ben werden; die Konfigurationselemente ergeben sich dann aus allen Produkten, die in den Schnittstellen der aufgelisteten Prozesse aufgeführt sind. Des weiteren können zur Bestimmung der Versionen der Elemente Attributwerte hinzugezogen werden.

Die *Trigger*-Möglichkeiten im Prozeßmodell werden erweitert: Hier können Funktionen aus dem Versionenmodell von Produkten und Konfigurationen aufgerufen werden. So können Versionen von Produkten und Konfigurationen automatisch beim Ein- oder Austritt aus einem Prozeß aktiviert werden.

Entry- und Exit-Kriterien im Prozeßmodell können zur Auswertung ihrer Bedingungen zusätzlich die Attribute von Konfigurationen oder Versionsnummern heranziehen. So ist es bspw. leicht möglich, nach dem Testen eines Teilsystems diejenigen Komponenten direkt dem nachfolgenden Systemintegrationsprozeß zu übergeben, bei denen sich durch eine Überarbeitung keine Änderungen ergeben haben (d.h. die Versionsnummer hat sich nicht geändert).[9]

Danksagung

Ich danke Prof. Dr. Dieter Rombach für sein kritisches Lesen früherer Fassungen und dem damit verbundenen konstruktivem Feedback. Ihm obliegt die Leitung des Teilprojekts A1 (SE-Labor) des SFB 501, in dem die Arbeiten entstanden sind.

Literatur

1. Basili, V.R., Caldera, G., McGarry, F., Pajerski, R., Page, G., Walegora, S.: The Software Engineering Laboratory — An Operational Software Factory. Proc 14th Int. Conf. on Software Engineering, 370-381 (1992)
2. Bersoff, E. H., Henderson, V. D., Siegel, S. G. *Software Configuration Management: An Investment in Product Integrity*, Prentice Hall, 1980.
3. Frakes, W., Terry, C.: Software Reuse: Metrics and Models. ACM Computing Surveys 28(2), 415-435 (1996)
4. 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
5. Bröckers A., Differding, C., Hoisl, B., Kollnischko, F., Lott C. M., Münch J., Verlage M. and Vorwieger S. A graphical representation schema for the software process modelling language MVP-L. *Technical Report 270/95, Department of Computer Science, University of Kaiserslautern*, Germany, 1995
6. Mili, H, Mili, F., Mili, A.: Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering Vol. 21(6), 528-562 (1995)
7. Rombach, H.D., Basili, V.R., Selby, R.W. (eds): Experimental Software Engineering Issues: Critical Assessment and Future Directions. Dagstuhl Workshop, Sept. 1992, Lecture Notes in Computer Science 706, Berlin: Springer 1993
8. Rossak, W., Kirova, V., Jololian, L., Lawson, H., Zemel, T.: A Generic Model for Software Architectures. IEEE Software, 84-92 (1997)

9. Die Elemente haben sich im bisherigen praktischen Einsatz als ausreichend erwiesen. Jedoch erhebt das Modell keinen Anspruch auf Vollständigkeit.