1 Eine Prozessplattform zur erfahrungsbasierten Softwareentwicklung

Jürgen Münch, Dieter Rombach

Zusammenfassung

Der Sonderforschungsbereich 501 mit dem Titel „Entwicklung großer Softwaresysteme mit generischen Methoden“ befindet sich gegenwärtig in seiner dritten dreijährigen Förderperiode. Ziel ist die Schaffung einer Software-Entwicklungmethodik, die auf systematische Wiederverwendung setzt. Lösungsansätze konzentrieren sich auf die Spezialisierung von Anwendungsdomänen, änderungsinvariante Softwarearchitekturen als Repräsentation kondensierten Domänenwissens, Softwareentwicklungsprozesse als Instanzierungen festgelegter Softwarearchitekturen, Generizität als Schlüsselansatz zur wiederverwendungsgerechten Gestaltung von Software sowie Entwicklungserfahrung als entscheidendem Motor für zielorientierte Verbesserung.

Im Rahmen des SFB 501 ist die hybride Softwaremodellierungsmethodik MILOS entwickelt worden, die den constraint-orientierten Ansatz MVP-L mit dem planungs-orientierten Ansatz Co-Mo-Kit verheiratet hat. MILOS ermöglicht die benutzer-akzeptierte Modellierung von Prozessaspekten sowohl algorithmischer als auch kreativer Natur, die Einbindung von Messdatenerfassung zum Zwecke der Prozesssteuerung als auch der Erfahrungsgewinnung, sowie die Anleitung von Entwicklungsteams. Die Prozessplattform unterstützt alle Phasen der Softwareprojektabwicklung – Planung, Durchführung und Erfahrungsmanagement. Über eine Serie von Baselining-Software-Entwicklungsprojekten sowie kontrollierten Experimenten ist die Prozessplattform evaluiert worden. Die Nutzung der dabei gewonnenen Erfahrungen hat zu einer messbaren Verbesserung des Entwicklungsstandes geführt. Erste industrielle Anwendungen werden gegenwärtig durchgeführt.

In diesem Artikel wird eine kurze Übersicht über den SFB 501 gegeben. Im Vordergrund steht die Präsentation einer hybriden Modellierungstechnik für Softwareentwicklungsprozesse, sowie die dafür entwickelte Prozessplattform zur Unterstützung von Modellierung, Ausführung und Erfahrungsmanagement. Abschließend wird an einem ausgewählten Beispiel das Verbesserungs- und Weiterentwicklungspotential von Softwareentwicklungsprozessen auf der Basis empirischer Erfahrungsdaten demonstriert.

Referenz: Jürgen Münch, Dieter Rombach. Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen, chapter Eine Prozessplattform zur erfahrungsbasierten Softwareentwicklung, pages 93-106. Wiley-VCH, 2003.

1.1 Motivation

Motivation für den Sonderforschungsbereich 501 [1] sind die offensichtlichen software-technischen Defizite in der industriellen Praxis, große Softwaresysteme mit ausreichender und nachweisbarer Qualität mit vertretbaren Kosten erstellen zu können. Software-Systeme in der industriellen Praxis sind vielfach gekennzeichnet durch eine sehr hohe Software-Komplexität, die mit bisherigen Herangehensweisen nicht mehr kontrollierbar ist. Qualität und Kosten nehmen mit zunehmender Komplexität vielfach exponentiell ab- bzw. zu. Diese Situation ist insbesondere für eingebettete Software (z. B. Steuerung technischer Systeme im Automobilbereich) untragbar. Als Hauptursachen werden unzureichende Software-Entwicklungstechniken (insbesondere für Familien ähnlicher Systemvarianten) sowie unzureichende Erfahrungen bei der effizienten Integration derartiger Techniken in ingenieurmäßige Produktmodelle und Vorgehensweisen angesehen. Von einem ingenieurmäßigem Einsatz solcher Techniken wird insbesondere eine Annäherung des Kostenanstiegs an lineares Verhalten bei zunehmender Software-Komplexität erwartet (siehe Abbildung 1).

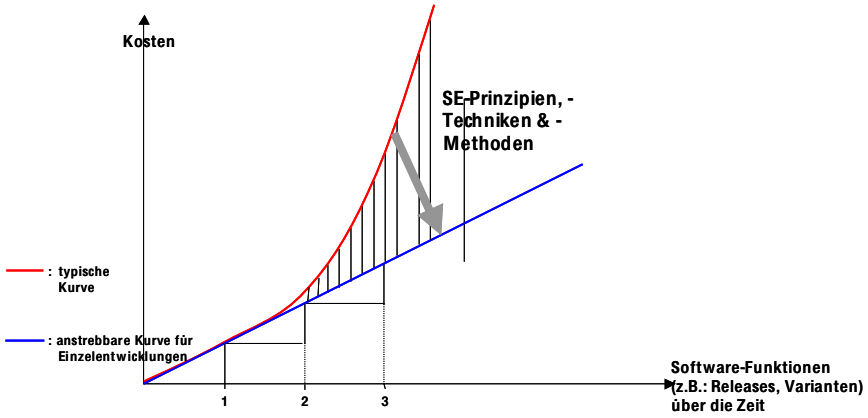

Abbildung 1: Kostenreduktion mittels SE-Prinzipen, -Techniken und -Methoden

Ziele des SFB 501 sind deshalb, (a) eine Menge von Entwicklungstechniken und Werkzeugen zur wiederverwendungs-orientierten Beschreibung aller Aspekte großer Systeme bereitzustellen, (b) eine Methodik und unterstützende Werkzeuge zur Entwicklung von Produktmodellen, Vorgehensweisen und sonstigen Erfahrungen zur effizienten, zielorientierten Benutzung derartiger Techniken in einem gegebenen Applikationskontext zu entwickeln. Verfahren zur Entwicklung großer Softwaresysteme sollen im SFB 501 auf „generischen Methoden“ basieren. Unter dem Begriff „generische Methoden“ fassen wir alle Beschreibungstechniken und Generierungstechniken mit den dazugehörigen Werkzeugen zusammen, durch die eine Wiederverwendung von existierenden Softwareprodukten, Entwicklungsschritten und sonstigen Erfahrungen bei der Entwicklung eines neuen Systems methodisch unterstützt wird. Der derzeit geringe Wiederverwendungsgrad bei der Software-Entwicklung soll erhöht und Kosten- und Qualitätsproblemen entgegengewirkt werden. Im industriellen Kontext wird der Einsatz generischer Methoden im Rahmen von sogenannten Produktlinien-Ansätzen adressiert.

Wiederverwendung von Softwareprodukten (d.h. Kode und Dokumentation), Entwicklungsschritten und sonstigen Entwicklungserfahrungen durch generische Methoden in dem hier verstan-

denen Sinn bedeutet einen systematischen Rückgriff auf relevante Informationen und Erfahrungen aus vergangenen Projekten. Hier liegt ein entscheidendes Defizit heute verfügbarer Software-Technik. Neue Systementwicklungen können von vergangenen Projekten ähnlichen Typs nur sehr begrenz profitieren, da als Resultate dieser vergangenen Produkte lediglich das Endprodukt – im Allgemeinen in Form von Kode und unterstützender Dokumentation – zur Verfügung steht. Dagegen existieren gewöhnlich weder Aufzeichnungen über die durchgeführten Entwicklungsschritte von der ursprünglichen Anforderungsbeschreibung bis zum Endprodukt, noch Dokumentationen über Entwicklungsaufwand und Zeit, Fehler oder berücksichtigte beziehungsweise verworfene Entwurfsentscheidungen. Die sinnvolle Aufbereitung von Software-Wissen für zukünftige Anwendungen und eine umfassende Nutzung des Wiederverwendungspotentials versprechen eine Annäherung der Kostenkurve an konstantes Verhalten (siehe Abbildung 2).

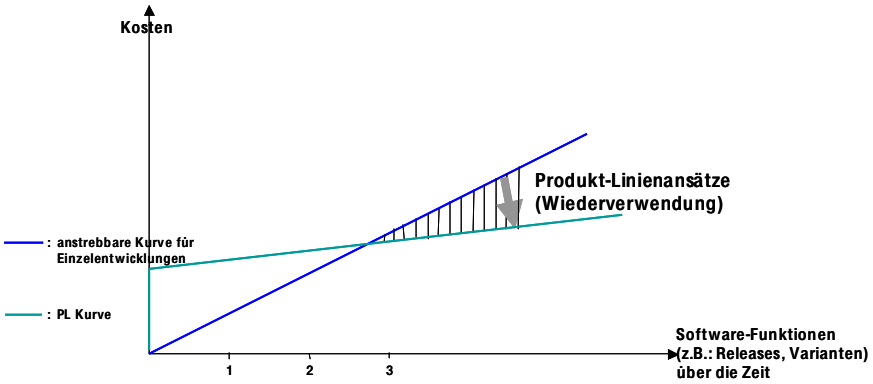

Abbildung 2: Kostenreduktion mittels Wiederverwendung

Der im SFB 501 verwendete Wissenschaftsansatz ist - in Anerkennung der Spezifika von Softwareentwicklung – experimentell. Dies bedeutet, dass Entwicklungstechniken und unterstützende Werkzeuge sowohl isoliert in kontrollierten Experimenten als auch in prototypischen Applikationsentwicklungen (sog. Baselining-Entwicklungen) und industriellen Anwendungen empirisch untersucht werden müssen. Diese empirischen Daten motivieren die Weiterentwicklung von Techniken und bilden die Basis für die Optimierung von Vorgehensmodellen und sonstigen Erfahrungswerten. Die praktische Umsetzung dieser Idee basiert im SFB 501 darauf, den Software-Entwicklungsprozess in allen relevanten Aspekten zu modellieren, projektspezifisch zu instanzieren und als integralen Bestandteile in einem Software-Engineering-Labor zu speichern [9]. Wiederverwendung in diesem Zusammenhang bedeutet dann primär Wiederholung gespeicherter Entwicklungsprozesse bis zum Auftreten von Wiedersprüchen aufgrund modifizierter Randbedingungen im aktuellen Projekt. Generatoren können als parametrisierte Entwicklungsprozesse aufgefasst werden. Die Nutzung von Erfahrungswissen verspricht darüber hinaus eine bessere Planbarkeit von Software-Entwicklungsprozessen. So können beispielweise mess-basierte Modelle zur Aufwandsvorhersage herangezogen werden, sofern sie für die aktuellen Projektrandbedingungen gültig sind (siehe Abbildung 3). Hiermit verbunden ist eine Reduktion der Risiken bei der Kostenplanung (z. B. für Budget- und Angebotsplanung) sowie verbesserte Möglichkeiten der Projektverfolgung und –lenkung.

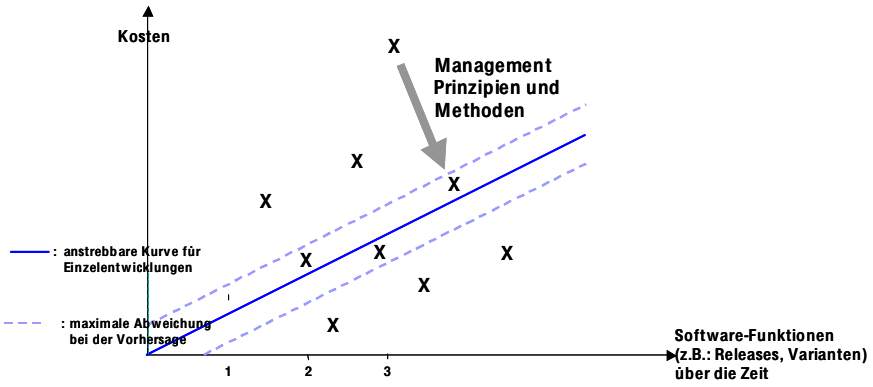

Abbildung 3: Messbasierte Vorhersagemodelle

Als Anwendungsfeld zur experimentellen Erprobung wurden Steuerungs- und Überwachungssysteme gewählt, und zwar zunächst das Anwendungsfeld Gebäudeautomation. Die entwickelten Modellvorstellungen und Methoden wurden durch dieses Anwendungsfeld primär gesteuert. Eine derzeitige Anwendung in anderen Anwendungsfeldern und -gebieten (z. B. Automobil) soll ihre Übertragbarkeit nachweisen.

1.2 Projektstruktur des SFB 501

Die Projektstruktur des SFB 501 (siehe Abbildung 4) berücksichtigt den experimentellen Ansatz.

Im Projektbereich A wird ein Softwareentwicklungslabor betrieben, in das alle methodischen Ergebnisse des SFB, die Werkzeugreife erreicht haben, integriert sind. Es dient dazu, kontrollierte Experimente mit den einzelnen Techniken sowie Baselining-Entwicklungen und Industrieerprobungen zum Zwecke der empirischen Erfahrungsgewinnung zu unterstützen. Hierzu wird die in Abschnitt 1.3 beschriebene Prozessplattform zur erfahrungsbasierten Software-Entwicklung genutzt. Von zentraler Bedeutung ist die effiziente Organisation und Pflege der empirisch gewonnenen Modelle zum Zwecke der Wiederverwendung.

Projekte:

- A1: SE-Labor (Prof. Rombach)
- A2: Entwicklung einer flexiblen Modellierungs- und Ausführungsumgebung für Software-Entwicklungsprozesse (Prof. Richter, Prof. Rombach)
- A3: Unterstützung des SW-Entwicklungsprozesses durch objekt-relationale Datenbank-Technologie (Prof. Härder)
- A4: Experimentieren mit Techniken für generische Software-Produktlinien (Dr. Münch)

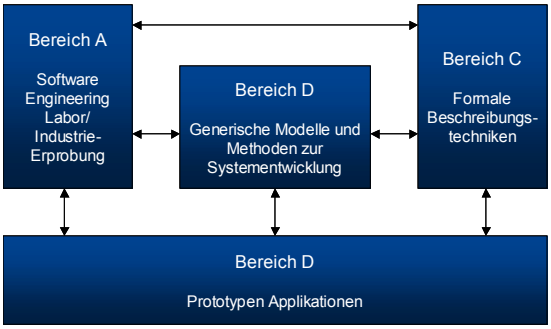

Abbildung 4: Grundstruktur des SFB 501

Im Projektbereich B werden Techniken und Werkzeuge zur generischen Beschreibung und Entwicklung von Basissoftware (z.B. Betriebssysteme, Kommunikationssysteme) sowie Anwendungssoftware entwickelt und Experimente für die Erprobung im Rahmen des Projektbereichs A vorbereitet.
Projekte:

- B1: Generische Modellierung von Prozessen und Experimenten (Prof. Rombach)
- B2: Flexible Planung und Steuerung von Software-Entwicklungsprozessen (Prof. Richter)
- B4: Generische Kommunikationssysteme (Prof. Gotzhein)
- B5: GeneSys – Generische Systemsoftware (Prof. Nehmer)
- B10: Anwendungsentwicklung mit vorkonfektionierten Software-Systemen (Prof. Nehmer)

Im Projektbereich C werden Beschreibungstechniken untersucht und gegebenenfalls entwickelt, die für die unterschiedlichen Phasen im Software-Entwicklungsprozess benötigt werden. Ein besonderes Anliegen ist die Verfolgbarkeit aller Entwicklungsschritte, die eine Durchgängigkeit zwischen verschiedenen formalen und informellen Beschreibungstechniken verlangt.
Projekte:

- C1: Formale Beschreibungstechniken (Prof. Avenhaus, Prof. Madlener)

Im Projektbereich D ist das Expertenwissen auf ausgewählten Anwendungsfeldern konzentriert. Neben eigenständigen Forschungsaufgaben aus den Anwendungsfeldern stellen die Teilprojekte die Infrastruktur für Zielsysteme bereit. Dazu gehören Hard- und Softwareplattformen für die Zielsysteme sowie Testumgebungen. Der Projektbereich spielt auch die Rolle des Kunden bei Entwicklungsprojekten gegenüber den anderen Projektbereichen.
Projekte:

- D1: Anwendungssystem Gebäude (Prof. Zimmermann, PD Dr. Schürmann)
- D2: Modellbasierte Entwicklung wiederverwendbarer Regelungsalgorithmen (Prof. Litz)

1.3 Prozessplattform

Software-Entwicklungsprozesse haben die Entwicklung komplexer Software-Systeme zum Ziel, wobei zahlreiche kreative (stochastisch unbestimmte) Aktivitäten durchgeführt werden. Große Software-Entwicklungsprozesse sind einmalig und gekennzeichnet durch 1) iterative Verfeinerung des Plans, 2) Risiken, die zu bedingten Teilen des Plans führen, 3) unvorhergesehene Ereignisse, die zu Änderungen der beabsichtigten Prozessabwicklung führen, 4) komplexe Abhängigkeiten zwischen Prozessen, die bei Änderungen zu berücksichtigen sind. Software-Entwicklungsprojekte beinhalten eine Vielzahl von Einzelprozessen. Wir unterscheiden technisch-orientierte Prozesse zur Erstellung von Software-Produkten und management-orientierte Prozesse zur Planung und Koordination verteilter Team- und Einzelaktivitäten. Die Prozessplattform des SFB 501 zur erfahrungsbasierten Software-Entwicklung unterstützt beide Arten von Prozessen während aller Phasen der Software-Projektabwicklung – Planung, Durchführung und Erfahrungsmanagement. Sie integriert dabei folgende Aspekte: Die Vorabanalyse von Projektplänen, die Koordination verteilter Aktivitäten auf der Basis messbarer Projektpläne, die Verfolgbarkeit von Entwurfsentscheidungen, die dynamische Umplanung sowie die Verzahnung von Planung und Ausführung (im Sinne einer sukzessiven Verfeinerung der Prozesse während der Abwicklung).

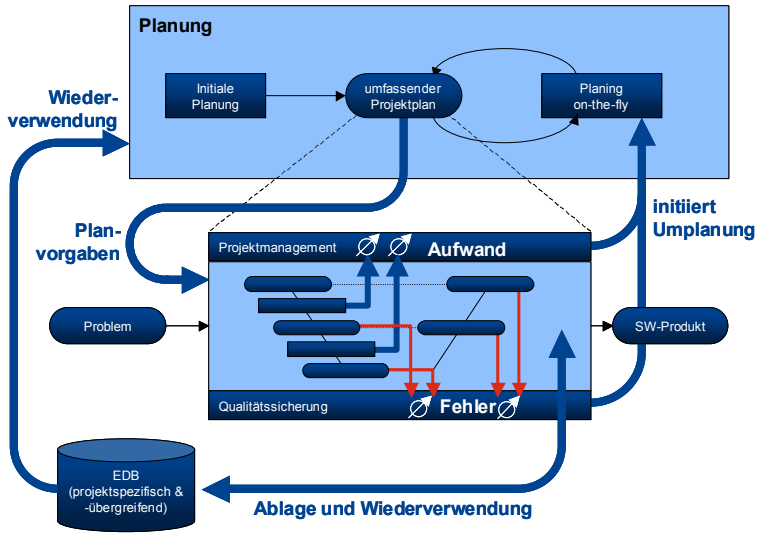

Abbildung 5: Erfahrungsbasierte Projektplanung

Die Prozessplattform unterscheidet *Planung* in eine Vorplanung, d.h., die Festlegung allen vor Projektbeginn bekannten Prozesswissens, und eine projektbegleitende Umplanung, d.h., Ergänzung bzw. Änderung des initialen Prozesswissens aufgrund des jeweiligen Projektzustandes (siehe Abbildung 5). Bei der *Durchführung* von Entwicklungsprojekten sind die Schwerpunkte zum einen auf Maßnahmen zur effizienten Erfassung von Messdaten sowie deren Verwendung zur Projektsteuerung, zum anderen auf die frühzeitige Notifikation einzelner Projektmitarbeiter über eventuell not-

wendige Abstimmungen oder Umplanungen gelegt. Beim *Erfahrungsmanagement* werden nach dem Modell der „Experience Factory“ von Basili Erfahrungsdaten aller Art (z.B.: Produkte, Prozessmodelle, Messdaten, qualitative Erfahrungen) erfasst, nach ihrem generischen Wiederverwendungspotential in zukünftigen Projekten untersucht, und kontext-sensitiv zur Wiederverwendung in einer Erfahrungsdatenbank (EDB) abgelegt. Bei der Planung wird entsprechend einer Projektcharakterisierung auf diese kontext-sensitive Erfahrung zugegriffen.

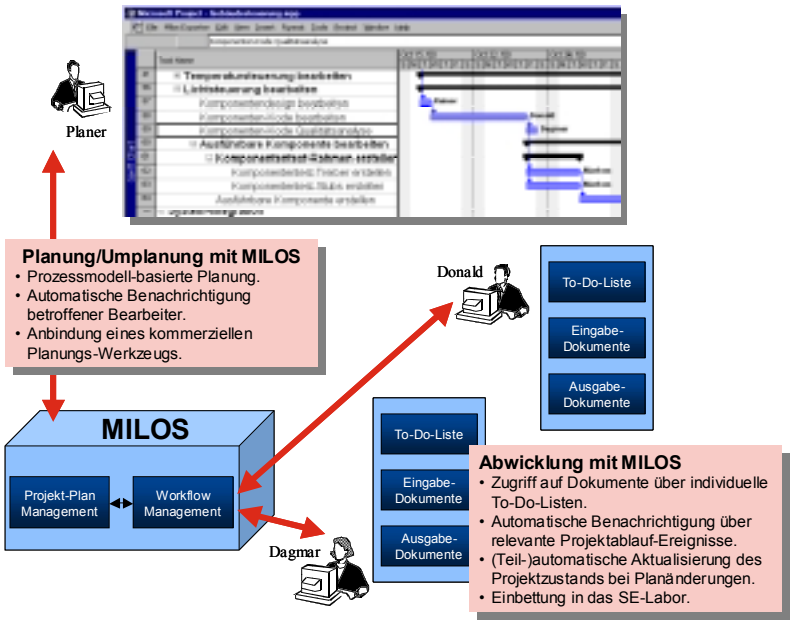

Abbildung 6: Integrierte Planung und Abwicklung mit MILOS

Für die Prozessplattform wurde im SFB 501 die hybride Softwaremodellierungsmethodik MILOS entwickelt, die den Constraint-orientierten Ansatz MVP-L mit dem planungs-orientierten Ansatz CoMo-Kit verheiratet, wobei die Stärken der jeweiligen Ansätze kombiniert wurden. MILOS unterstützt insbesondere die verzahnte Planung und Durchführung (siehe Abbildung 6).

Der Constraint-orientierte Ansatz MVP-L [3] unterstützt die Prozessmodellierung im Großen, d.h., es wird stärker auf die Beziehungen zwischen Team- bzw. Einzelprozessen eingegangen als auf deren Implementierung. Er wurde im Rahmen des Teilprojekts B1 des SFB 501 in Bezug auf die Analyse von Projektrisiken, die sichten-basierte Modellierung, sowie die messbasierte Abwicklung von Projektplänen weiterentwickelt. Die Stärken des MVP-Ansatzes liegen in der Bereitstellung mächtiger Modellierungskonzepte, dem Vorhandensein statischer und dynamischer Analysemethoden sowie der messgesteuerten Projektabwicklung.

Der planungs-orientierte CoMo-Kit-Ansatz ist ausgerichtet auf die Unterstützung der dynamischen Planung und Abwicklung technischer Einzelprozesse, die durch eine inhärente Kreativität gekennzeichnet sind und daher eine hohe Planungsflexibilität erfordern. Er wurde im Rahmen des Teilprojekts B2 des SFB 501 in Bezug auf die Flexibilität der Prozessmaschine weiterentwickelt. Darüber hinaus verfügt der Ansatz über Mechanismen zur Verwaltung von Abhängigkeiten zwischen Produkten und Prozessen, die die Verfolgbarkeit des Entwicklungsprozesses erhöhen. Die

Stärken des CoMo-Kit-Ansatzes liegen in der Unterstützung kreativer Prozesse durch die Bereitstellung flexibler Umplanungsmechanismen sowie dem Management von Abhängigkeiten zwischen Prozesselementen.

MILOS [4,6,12] ermöglicht die benutzer-akzeptierte Modellierung von Prozessaspekten sowohl algorithmischer als auch kreativer Natur, die Einbindung von Messdatenerfassung zum Zwecke der Prozesssteuerung als auch der Erfahrungsgewinnung, sowie die Anleitung von Entwicklungsteams. Die konzeptionelle Integration des MVP- und des CoMo-Kit-Ansatzes erforderte die Zusammenführung von Techniken der Prozessmodellierung, der Abhängigkeitsverwaltung mit Truth-Maintenance-Systemen, des Workflow-Managements und der Aktionsplanung. Zur Operationalisierung der Prozessmodelle werden in MILOS Techniken des Workflow-Managements (Verwaltung von Arbeitslisten, Weiterleiten von Informationen zwischen Bearbeitern) und der Aktionsplanung (Operatoren, Constraints) benutzt. Die Unterstützung globaler Managementprozesse erfolgt durch Mechanismen zur Vorabanalyse von Projektplänen und zur Abwicklung auf der Basis messbarer Projektpläne, die dem MVP-Ansatz entstammen. MILOS verwendet modifizierte Mechanismen des MVP-Ansatzes zur messbasierten Steuerung von Projekten. Während MVP die grundlegenden Modellierungskonzepte zur Verfügung stellt, bilden KI-Planungstechniken die technische Basis der Operationalisierungskonzepte. Um die Einbeziehung solcher KI-Planungstechniken in den Prozessmodellierungsansatz zu ermöglichen, wurde das Prozessverfeinerungskonzept in MILOS um das Methodenkonzept erweitert, welches eine Trennung zwischen einer expliziten Zielbeschreibung und möglicher Lösungsalternativen zur Erreichung des Ziels vorsieht.

Die Prozessplattform des SFB 510 integriert die folgenden, im SFB 501 entwickelten Werkzeuge, deren Zusammenwirken in Abbildung 7 dargestellt wird:

- *die Erfahrungsdatenbank (SFB-EDB) zur kontextorientierten Ablage und zum Retrieval von Erfahrungen [5,13].* Alle Arten von Erfahrungen werden in der Erfahrungsdatenbank des SFB 501 kontextorientiert (d.h., versehen mit Angaben über ihren Gültigkeitsbereich) und mit einer Validitätsaussage (d.h., versehen mit Angaben über den Grad ihrer Erprobung) abgelegt. Anpassungen der Struktur, der Objekthierarchie oder der Querbeziehungen innerhalb der Erfahrungsdatenbank können schritthaltend mit der Gewinnung neuer Kenntnisse vorgenommen werden.
- *das Werkzeug ProTail für die Anpassung von wiederverwendeten Prozessmodellen während der Vorplanung [10].* Das Werkzeugs ProTail unterstützt die effektive und effiziente Bereitstellung kontext-orientierter Projektpläne während der Vorplanung. ProTail erhält als Eingabe Ziele und Charakteristika des vorliegenden Projekts. Bei der Anpassung werden Prozessmodelle aus der Erfahrungsdatenbank, die für verschiedene Kontexte parametrisiert sind, in geeigneter Form wiederverwendet. Mit ProTail erstellte maßgeschneiderte Projektpläne können als Input für die MILOS-Prozessmaschine verwendet werden. Durch Kombination von ProTail und MILOS steht eine Umgebung zur Verfügung, in der die Wiederverwendung von Prozessmodellen während der gesamten Lebensdauer eines Projekts von der ersten Planung bis zur Terminierung unterstützt wird.
- *die MILOS-Prozessmaschine zur Ausführung und Umplanung während der Durchführung von Software-Entwicklungsprojekten [7,11].* Herausragende Eigenschaften der MILOS-Prozessmaschine sind Möglichkeiten zur dynamische Instanzierung von Prozessalternativen, zur dynamischen Verfeinerung von Produkten und Prozessen, und zum Zurücksetzen des Projektzustands. Hierfür werden komplexe Abhängigkeiten zwischen den Prozessen verwaltet.

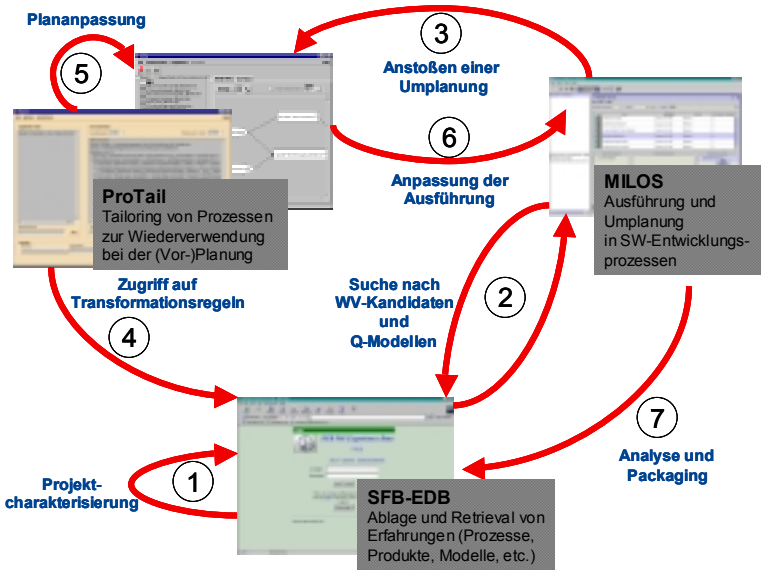

Abbildung 7: Integrierte Werkzeugumgebung

1.4 Gewinnung von Erfahrungen

Ein effizienter Einsatz von Methoden und Werkzeugen in der Software-Entwicklung setzt die genaue Kenntnis der Bedingungen und der Auswirkungen ihrer Anwendung voraus: Man muss möglichst genau wissen, unter welchen Umständen die angewandten Methoden/Werkzeuge wie gut funktionieren. Daher werden im SFB Technologien vor ihrem Einsatz systematisch geprüft und evaluiert. Erst wenn durch Experimente und darauf folgende quantitative und qualitative Analysen Stärken und Schwächen der jeweiligen Technologien herausgefunden wurden, ist eine abgesicherte Aussage darüber möglich, für welche Aufgabenstellung und unter welchen Randbedingungen sich die Technologien eignen. Der SFB 501 setzt diese Grundeinsicht durch konsequente Verwirklichung des experimentellen Ansatzes praktisch um. Hierzu werden Experimente verschiedener Art durchgeführt, mit denen das Ziel verfolgt wird, die Wirkung einzelner Methoden/Werkzeuge zu verstehen. Im Detail verläuft die Erprobung in drei Schritten:

1. *Durchführung kontrollierter Technologie-Experimente:* Ziel ist das Verständnis von Ursache-Wirkungsbeziehungen einzelner Methoden/Werkzeuge. Dabei werden fertigentwickelte Methoden/Werkzeuge auf ihre Verbesserung gegenüber vergleichbaren Alternativen unter kontrollierten Bedingungen überprüft.
2. *Durchführung von Baselining-Entwicklungen*: Ziel ist das Verständnis der Wirkung von experimentell erfolgreichen SFB-Methoden/Werkzeuge in durchgängigen Anwendungsentwicklungen. Der positive Beitrag einer Technik oder eines Werkzeugs (z. B.: Reduktion des Entwick-

lungsaufwands, Verbesserung der Qualität) unterliegt in praktischen Entwicklungensprojekten sehr vielen Einflüssen (z. B.: Erfahrung der Entwickler, Einhaltung des Prozesses, Robustheit der Entwicklungsplattform). Darüber hinaus können sich unterschiedliche Techniken/Werkzeuge gegenseitig beeinflussen. Deshalb ist es nicht ausreichend, neu entwickelte Techniken und Werkzeuge isoliert empirisch zu untersuchen, sondern sie müssen in geeigneter Kombination (definiert über ein Prozessmodell) erprobt werden. Solche notwendigen Scale-Up-Experimente, sogenannte Baselining-Entwicklungen, definieren Baselines in Bezug auf den praktischen Effekt einer neuen Kombination einzelner Methoden/Werkzeuge. Die Resultate von Baselining-Entwicklungen sowie die hierbei gemachten Erfahrungen sollen wiederum Änderungen oder Neuentwicklungen von Methoden/Werkzeugen motivieren, die zu Verbesserungen führen (siehe Abbildung 8).

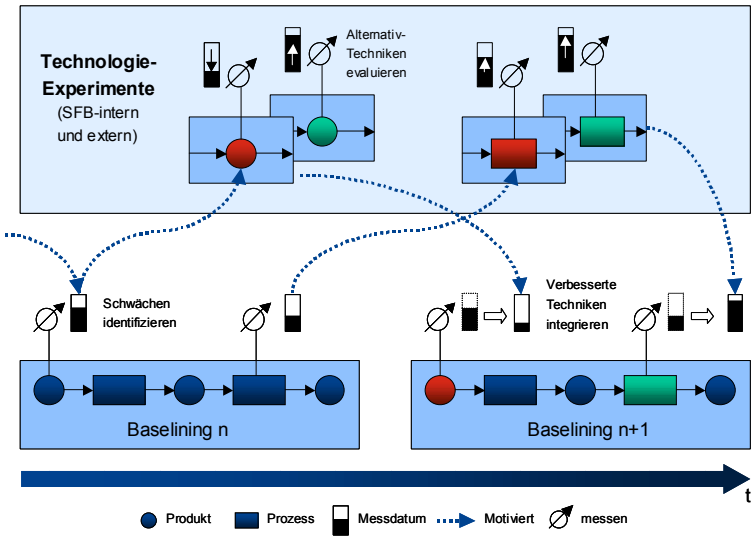

Abbildung 8: Technologie-Experimente und Baselining-Entwicklungen

3. *Durchführung industrieller Anwendungen:* Mit den gewonnenen Erkenntnissen aus kontrollierten Experimenten und Baselining-Entwicklungen wird schrittweise eine Erprobung und Verbesserung von Methoden/Werkzeugen unter industriellen Bedingungen in dafür ausgewählten Pilotprojekten vorgenommen. Ziel ist das Verständnis der Wirkung einzelner Methoden/Werkzeuge in einem industriellen Kontext. Hierdurch werden praxisrelevante Problemstellungen berücksichtigt und Lösungen erarbeitet und erprobt, die in einem rein universitären Umfeld nicht machbar sind. Die in den industriellen Technologie-Erprobungen gesammelten Erfahrungen bilden den Grundstock für die maßgeschneiderte industrielle Nutzung von SFB-Ergebnissen in der Breite.

Die der Prozessplattform zugrundeliegende Methodik zur empirisch-basierten Erfahrungsgewinnung und Modellierung geht auf den QIP-Ansatz von Basili und Rombach zurück [2], der die quantitative Formulierung von Mess- und Bewertungszielen, die Instrumentierung von Vorgehensmodellen oder Methoden/Werkzeugen durch Metriken sowie die experimentbegleitende Datenerfassung regelt. Je nach Mess- und Bewertungsziel werden geeignete experimentelle Entwürte empfohlen. Zur wiederverwendbaren Ablage werden geeignete Analyseprozeduren sowie Datenbankschemata angeboten.

1.5 Beispiel

Im Folgenden wird an einem ausgewählten Beispiel das Verbesserungs- und Weiterentwicklungspotential von Softwareentwicklungsprozessen auf der Basis empirischer Erfahrungsdaten demonstriert. Hierbei wird eine Baseline-Entwicklung skizziert, deren Ergebnisse die Entwicklung eines verbesserten Analyseprozesses zur Folge hatte.

Ziel der Baselining-Entwicklung war die Entwicklung einer kompletten Baseline von Modellen und Erfahrungen für einen ausgewählten Satz von Methoden, Techniken und Werkzeugen mit der Prozessplattform des SFB 501. Der Satz umfasste die für die OO-Entwicklung gängigen Beschreibungstechniken für die Erstellung der Anforderungen (UML-Use-Case-Diagramme, Szenarienbeschreibungen, UML-Klassendiagramme, UML-Sequenzdiagramme), die zugehörigen Analyse-Entwicklungsmethoden aus BOE und dem Unified Process, Kodierungstechniken- und Werkzeuge (C++, UNIX-Werkzeuge), sowie Test-Techniken für die Komponenten- und System-Validation (Black-Box-Testen). Die Aufgabenstellung lautete, ausgehend von einer Problembeschreibung ein Gebäudeautomationssystem mit Licht- und Temperaturregelung zu erstellen. Die Techniken, Methoden und Werkzeuge wurden nach einem Wasserfall-ähnlichen SFB-Entwicklungsprozess eingesetzt. Als Baseline dokumentiert wurden das tatsächlich durchgeführte Prozessmodell, Aufwands-/Zeit-/Fehlermodelle (entsprechend des Prozessmodells), Aufwands-/Zeit-/Fehlermodelle bezüglich einzelner Techniken, Methoden und Werkzeuge. Gemäß der Intention von Baselining-Entwicklungen im SFB wurde der gesamten Anwendungsentwicklungsprozesses von der OO-Analyse bis zum Systemtest durchgeführt. Für die Entwicklung wurden folgende Werkzeuge verwendet: Zur Modellierung und (Teil-)Generierung der UML-Diagramme wurde Software through Pictures (StP) von Aonix in der UML-Version eingesetzt. Zur Übersetzung von Kode und zur Systemintegration wurden UNIX-übliche Werkzeuge verwendet. Die Planung (insbesondere gemachte Sollvorgaben) basierte auf Schätzungen für Aufwand, Fehler und Kalenderzeit aus früheren Baseline-Entwicklungen, die der Erfahrungsdatenbank des SFB entnommen wurden. Die Baselining-Entwicklung wurde von 17 Software-Entwicklern (Informatikstudenten mit Software-Engineering-Kenntnissen) an der Universität Kaiserslautern durchgeführt. Fünf Mitarbeiter/-innen der SFB-Teilprojekte A1 und B1 und der Arbeitsgruppe Software Engineerung betreuten die parallelen Teams unter Leitung von Prof. Rombach.

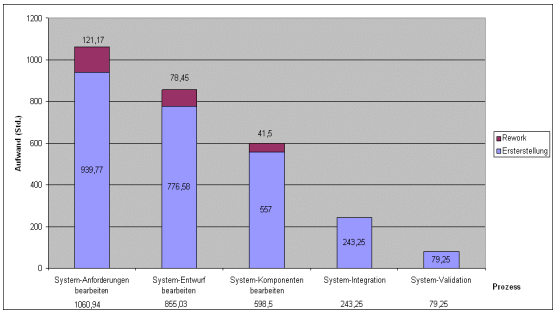

Abbildung 9: Aufwandsmodell

Wesentliche Resultate der Baselining-Entwicklung sind ein Prozessmodell für die Entwicklung eingebetteter Software sowie eine Menge von Qualitätsmodellen. Das Aufwandsmodell zeigt beispielsweise eine (37%, 30%, 21%, 9%, 3%)-Verteilung auf die Phasen Analyse, Entwurf, Kodierung (mit integrierter Komponenten-Validation), Integration und System-Validation (siehe Abbildung 9). Der Aufwand in den frühen Prozessschritten (Analyse und Entwurf) war verhältnismäßig hoch (67%). Dies ist im Wesentlichen auf zwei Analyseschritte und die umfangreiche Modellierung verschiedener UML-Sichten zurückzuführen. Das Fehlermodell zeigt, dass die meisten der besonders teuren Anforderungsfehler (96%) bereits frühzeitig bei der Anforderungsinspektion entdeckt wurden. Das Fehlerklassenmodell zeigt, dass in sämtlichen Fehlerklassen eine Abnahme der Fehleranzahl im Verlauf der Entwicklung festzustellen ist (mit einer Ausnahme beim Entwurf). Dies spricht für eine frühzeitige Erkennung unterschiedlichster Fehlerarten. Analysiert man die Ursachen für Rework-Aufwand, so fällt eine hohe Zahl an Unvollständigkeiten und Inkonsistenzen im Analysedokument auf (siehe Abbildung 10), die auf eine noch unzureichende Verfolgbarkeit zwischen den verschiedenen UML-Modellen schließen lässt.

Fehlerklasse§	Problemlösungs-dokumentation§
Unvollständigkeit§	85§
Inkonsistenz§	71§
Falsche ¶ Information§	53§
Überspezifikation§	19§
Vorwärts-¶ referenzen§	28§

Abbildung 10: Fehlerverteilung bei der Analyse

Eine Schlussfolgerung aus der Baselining-Entwicklung ist, dass das Sichern der Konsistenz zwischen UML-Modellen während der Analyse einen hohen Aufwand verursacht. Eine wesentliche offene Frage und ein damit verbundenes Verbesserungspotential ist: Wie lassen sich Unvollständigkeiten und Inkonsistenzen bei der Analyse reduzieren bzw. frühzeitiger entdecken? Der Einsatz von Werkzeugen zur Verbesserung der Verfolgbarkeit, explizite Verfolgbarkeitsinformation zur Verbesserung der Verfolgbarkeit sowie eine Änderung der Entwurfsinspektionen können mögliche Antworten sein.

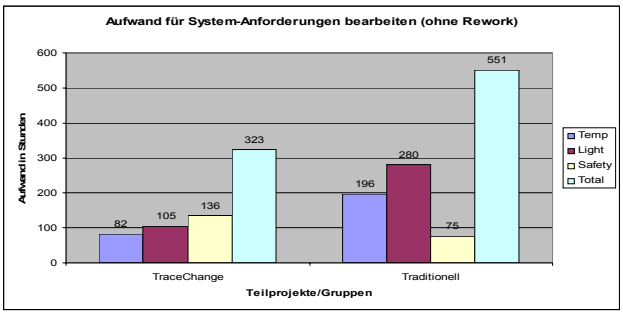

Abbildung 11: Änderungen mit unterschiedlich dokumentierten Anforderungen

Im Anschluss an die Baseline-Entwicklung wurde in einem kontrollierten Experiment der Einsatz expliziter Verfolgbarkeitsinformationen zur Verbesserung der Verfolgbarkeit mit der bisherigen Vorgehensweise verglichen. Zur Integration der Verfolgbarkeitsinformation wurde der im SFB entwickelte TraceChange-Ansatz [8] für die Beschreibung der Anforderungen verwendet. Das Experiment beschränkte sich auf die Analysephase. Mit TraceChange zeigte sich eine Aufwandsreduktion um ca. 33%, wobei der Unterschied bei der Analyse komplexer Systemteile (z.B. Lichtsteuerung) besonders hoch war (siehe Abbildung 11). Auch hinsichtlich der Unvollständigkeiten und Inkonsistenzen im Analysedokument konnte ein positiver Effekt festgestellt werden (siehe Abbildung 12).

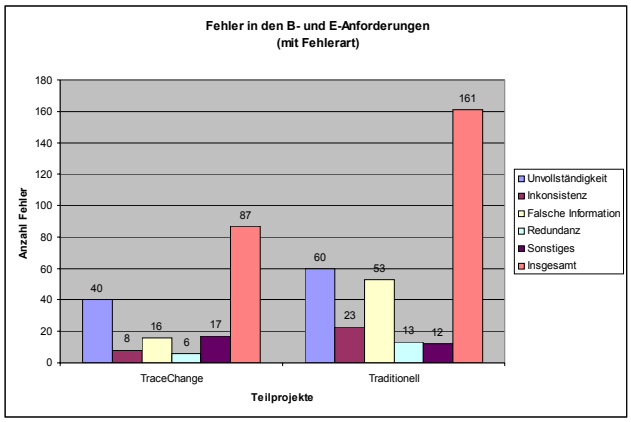

Abbildung 12: Fehlerverteilung

Das Beispiel zeigt, wie Erfahrungen im SFB 501 gewonnen werden und zu einer Verbesserung des Software-Entwicklungsprozesses genutzt werden können. Die Analyse der quantitativen und qualitativen Erfahrungen führten zur Identifikation von Verbesserungspotentialen und lieferte damit Hinweise auf weitere Forschung. Die Durchführung von Baselining-Entwicklungen im SFB und Anwendungsentwicklungen in industriellen Umgebungen mit professionellen, erfahrenen Entwicklern kann zu einer weiteren Steigerung der externen Validität (d. h., Übertragbarkeit) der Resultate führen. Derzeit erfolgt eine industrielle Erprobung von TraceChange im Kontext des SFB-Transferprojekts A4.

Literatur

[1] J. Avenhaus, R. Gotzhein, T. Härder, L. Litz, K. Madlener, J. Nehmer, M. Richter, N. Ritter, H. D. Rombach, B. Schürmann und G. Zimmermann: Entwicklung großer Systeme mit generischen Methoden - Eine Übersicht über den Sonderforschungsbereich 501. Informatik, Forschung und Entwicklung, 13(4):227–234, Dezember 1998.

[2] Victor R. Basili: The experimental paradigm in software engineering. In D. Rombach et al. (Herausgeber), Experimental Software Engineering Issues: Critical assessments and future directions, LNCS Nr. 706, S. 3-12, Springer-Verlag, September 1992.

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

[4] Barbara Dellen, Frank Maurer, Jürgen Münch, Martin Verlage: Enriching Software Process Support by Knowledge-based Techniques; International Journal of Software Engineering and Knowledge Engineering, 1997.

[5] Raimund L. Feldmann, Jürgen Münch und Stefan Vorwieger: Towards Goal-Oriented Organizational Learning: Representing and Maintaining Knowledge in an Experience Base. In Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering (SEKE'98), San Francisco, USA, 18.-20. Juni 1998.

[6] Sigrid Goldmann, Jürgen Münch und Harald Holz: MILOS: A Model of Interleaved Planning, Scheduling, and Enactment. In Web-Proceedings of the 2nd Workshop on Software Engineering over the Internet, Los Angeles, CA, USA, 17. Mai 1999.

[7] Sigrid Goldmann, Jürgen Münch und Harald Holz: Distributed Process Planning Support with MILOS. In Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99), Kaiserslautern, 17.-19. Juni 1999. Erweiterte Fassung in International Journal of Software Engineering and Knowledge Engineering, Oktober 2000.

[8] Antje von Knethen: Trace Model for System Requirements Changes on Embedded Systems. International Workshop on Principles of Software Evolution (IWPSE 2001), 10.-11. September, 2001.

[9] Oliver Laitenberger, Jürgen Münch: Ein Prozessmodell zur experimentellen Erprobung von Software-Entwicklungsprozessen; SFB 501, Technischer Bericht 04/1996, Fachbereicht Informatik, Universität Kaiserslautern, 1996.

[10] Jürgen Münch: Anpassung von Vorgehensmodellen im Rahmen ingenieurmäßiger Softwarequalitätssicherung. Im Tagungsband des 6. Workshops der Fachgruppe 5.1.1 (GI), Kaiserslautern, 19.-20. April, 1999.

[11] M.M. Richter, B. Kötting, S. Goldmann: Flexibles Workflow Management in Software Development Processes; Proceedings of the "Forschungsforum Wissensmanagement und schnelle Produktentwicklung", Stuttgart, Germany, 1999.

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

[13] S. Vorwieger, H. Damczyk, M. Fechtig, B. Schadt, O. Swienty: Struktur und Werkzeuge des experimentspezifischen Datenbereichs der SFB 501 Erfahrungsdatenbank; SFB Report 11/99, FB Informatik, Universität Kaiserslautern, 1999.