Entwicklung großer Systeme mit generischen Methoden – Eine Übersicht über den Sonderforschungsbereich 501 –

J. Avenhaus, R. Gotzhein, T. Härder, L. Litz, K. Madlener, J. Nehmer, M. Richter, N. Ritter, D. Rombach, B. Schürmann, G. Zimmermann

Universität Kaiserslautern, Postfach 30 49, 67653 Kaiserslautern

Zusammenfassung.
Am 01.05.1995 wurde an der Universität Kaiserslautern auf Initiative von Wissenschaftlern des Fachbereichs Informatik der Sonderforschungsbereich 501 "Entwicklung großer Systeme mit generischen Methoden" von der Deutschen Forschungsgemeinschaft (DFG) eingerichtet. Im Zentrum der Forschungsarbeiten des SFB 501 stehen Wiederverwendungskonzepte für Software und Softwareentwicklungsprozesse und deren Einbettung in Softwareentwicklungsumgebungen. Generizität wird als ein Schlüsselkonzept zur wiederverwendungsgerechten Aufbereitung von Software und Entwicklungswerkzeugen erforscht. Der vorliegende Artikel gibt eine Übersicht über den wissenschaftlichen Ansatz des SFB 501, seine Organisationsstruktur sowie die gegenwärtig geförderten Teilprojekte.

Schlüsselwörter: Softwaretechnologie, Softwareentwicklungsumgebungen, Prozeßmodelle, Produktmodelle, Generizität, Programmfamilien, Systemarchitektur, Domänenwissen

Abstract. Based on an initiative of a group of scientists at the department of Informatik of the University of Kaiserslautern the Sonderforschungsbereich 501 "Development of Large Systems by Generic Methods" has been founded by the Deutsche Forschungsgemeinschaft on January 1st 1995. The research interests of the SFB 501 center around reuse concepts for software and development processes and their integration into software development environments. Genericity is considered a key concept for a reuse-oriented design layout of software and development tools. This paper gives an overview of the scientific approach of the SFB 501, its organizational structure and its presently funded projects.

Key words: software technology, software development environments, process models, product models, genericity, program families, system architecture, domain knowledge

1 Einleitung

Am 1. Januar 1995 wurde von der Deutschen Forschungsgemeinschaft der Sonderforschungsbereich 501 an der Universität Kaiserslautern eingerichtet. Er hat sich das Thema *Entwicklung großer Systeme mit generischen Methoden* gestellt und greift damit ein Kernproblem der Informatik erneut auf: die systematische Konstruktion und Fertigung von Software nach ingenieurgemäßen Prinzipien.

30 Jahre nach Prägung des Begriffs *Software Engineering* durch F. L. Bauer [1] steckt die Disziplin noch immer in den Kinderschuhen. In einer kritischen Würdigung des Erreichten nach 25-jähriger Forschung auf diesem Gebiet stimmen führende Wissenschaftler und professionelle Softwareentwickler weitgehend darin überein, daß wesentliche Ziele der Forschung nicht erreicht wurden (Goos [7], Denert [5], Wendt [21], Endres [8, 9]). Der Einfluß auf die industrielle Softwarefertigung war deshalb bisher gering. Softwareentwicklungsprojekte sind nach wie vor mit einem hohen Risiko behaftet, da weder die Qualität der produzierten Software noch der Entwicklungsaufwand mit der notwendigen Genauigkeit planbar sind [15].

Der Sonderforschungsbereich 501 ist Teil weltweiter Anstrengungen, aus den Fehlern und Rückschlägen 25-jähriger Forschung zu lernen und der Disziplin des Software Engineering eine neue Richtung zu geben. Im Zentrum dieser Anstrengungen stehen Methoden, die eine systematische Wiederverwendung von Software und den zu ihrer Fertigung benutzten Prozessen zum Gegenstand haben [10, 11, 12, 13, 16].

Dieser Aufsatz ist wie folgt gegliedert: In Abschnitt 2 werden die Defizite der heute praktizierten Softwareentwicklungsmethodik herausgearbeitet. Sie bilden die Motivation für den wissenschaftlichen Ansatz des SFB, der in Abschnitt 3 dargestellt wird. Abschnitt 4 gibt eine Übersicht über die Gliederung des SFB 501 und die gegenwärtig geförderten Projekte. Abschnitt 5 schließt mit einem Ausblick auf die langfristigen Ziele.

2 Defizite heutiger Softwaretechnologie

Heutige Verfahren zur Softwareentwicklung folgen im allgemeinen dem Top-Down-Ansatz: ausgehend von einer vollständigen, möglichst formalen Aufgabenbeschreibung wird die Software in einem mehrere Phasen umfassenden Prozeß bis hin zum ablauffähigen Produkt entwikkelt, getestet, integriert und installiert. Wasserfallmodell, Spiralmodell und V-Modell unterscheiden sich zwar in der spezifischen Vorgehensweise, gründen aber einheitlich auf der Vorstellung, daß Softwareentwicklung als Prozeß der schrittweisen Verfeinerung/Konkretisierung einer initial nur in vagen Umrissen vorhandenen Lösungsstruktur charakterisiert werden kann. Softwareprodukte, die aus einem derartigen Prozeß entstehen, sind naturgemäß Unikate, da an keiner Stelle ein expliziter Rückgriff auf Software oder sonstige Erfahrungen aus früheren Entwicklungsprojekten eingeplant ist (er findet höchstens implizit in den Köpfen der Entwickler, ohne eine sichtbare Verankerung im Prozeßmodell, statt).

An der Effektivität dieses Ansatzes bestehen erhebliche Zweifel. Die Kritik konzentriert sich dabei auf die folgenden zwei Argumente:

a) Bei der enormen Komplexität heutiger Softwaresysteme ist es unrealistisch, von der Vollständigkeit einer Anforderungsdefinition vor Beginn des eigentlichen Entwicklungsprozesses auszugehen. Stand der Praxis ist vielmehr, daß Anforderungsdefinitionen unvollständige Skizzen einer Aufgabenbeschreibung sind, die erst im Projektverlauf allmählich präzisiert und vervollständigt werden. Ausdruck der Komplexität sind die folgenden typischen Eigenschaften großer Softwaresysteme:
   - Sie sind nebenläufig, verteilt und heterogen in Bezug auf die unterliegenden Hard- und Softwareplattformen.
   - Nichtfunktionale Eigenschaften wie Zeitbedingungen, Zuverlässigkeit, Erweiterbarkeit, Skalierbarkeit, Bedienungsfreundlichkeit etc. stellen einen wesentlichen Teil der Anforderungen dar.
   - Das Gesamtsystem kann ohne Abstraktionen, Hierarchisierung und Modularisierung nicht mehr von einer einzelnen Person verstanden werden.
   - Die Entwicklung der Software erfolgt durch ein größeres Team von Spezialisten, wobei jedes Teammitglied an einer abgegrenzten, überschaubaren Teilaufgabe arbeitet, die keine Einsicht in die Arbeitsweise des gesamten Systems gewährt.

b) Wiederverwendung wird als methodisches Element in heutigen Softwareentwicklungsprozessen praktisch ignoriert. Hierzu gehören
   - die Formalisierung und Speicherung der Konzept-, Struktur- und Vorgehensbeschreibungen,
   - das Auffinden dieser Sachverhalte (und ihrer Abhängigkeiten), deren Wiederverwendung angestrebt wird, wie z. B. Software, Entwicklungsschritte und sonstige Erfahrungen aus früheren Projekten,

– die Übernahme bewährter Konzepte und Vorgehensweisen und ihre Anpassung an aktuelle Anforderungen.

Ohne wiederverwendungsgerechte Aufbereitung der Software sowie der zu ihrer Herstellung benutzten Prozesse bleibt der Wiederverwendungsgrad naturgemäß gering.

Insgesamt hat man den Eindruck, daß die gegenwärtige Softwareentwicklungspraxis eher für die Entwicklung überschaubarer, vorzugsweise sequentieller Programme durch einzelne Entwickler geeignet ist und damit dem erreichten Komplexitätsgrad heutiger Softwaresysteme nicht gerecht wird. Durch Konzepte wie Prozeßmodellierung [17], Objektorientierung [4], Frameworks [11], Design Patterns [6] und Komponentenorientierung [20] wurden Entwicklungen eingeleitet, die eine wiederverwendungsgerechte Aufbereitung von Software zum Ziel haben. Die Einbettung dieser Konzepte in Prozeßmodelle ist aber noch weitgehend ungelöst.

3 Der wissenschaftliche Ansatz des SFB 501

Der im SFB 501 verfolgte wissenschaftliche Ansatz zielt auf die Formalisierung von Wiederverwendungskonzepten ab und basiert darauf, Softwareentwicklung als eine Labordisziplin zu begreifen, bei der durch entwicklungsbegleitendes Erfassen, Messen und Bewerten verschiedener Fakten Vorkenntnisse aus früheren Entwicklungsprojekten für zukünftige Entwicklungsprojekte bereitgestellt werden. Softwareentwicklungsprojekte sind damit kontrollierte Experimente [16]. Diesem Grundgedanken trägt die in Abb. 1 dargestellte Rahmenarchitektur einer Softwareentwicklungsumgebung Rechnung. Sie geht auf Basili zurück und wird als Software Experience Factory bezeichnet [3]. Wir unterscheiden grundsätzlich zwischen dem Softwareentwicklungs-Kernsystem (kurz SE-Kern) und der projektspezifischen Softwareentwicklungsumgebung für die Durchführung jeweils eines Entwicklungsprojektes.

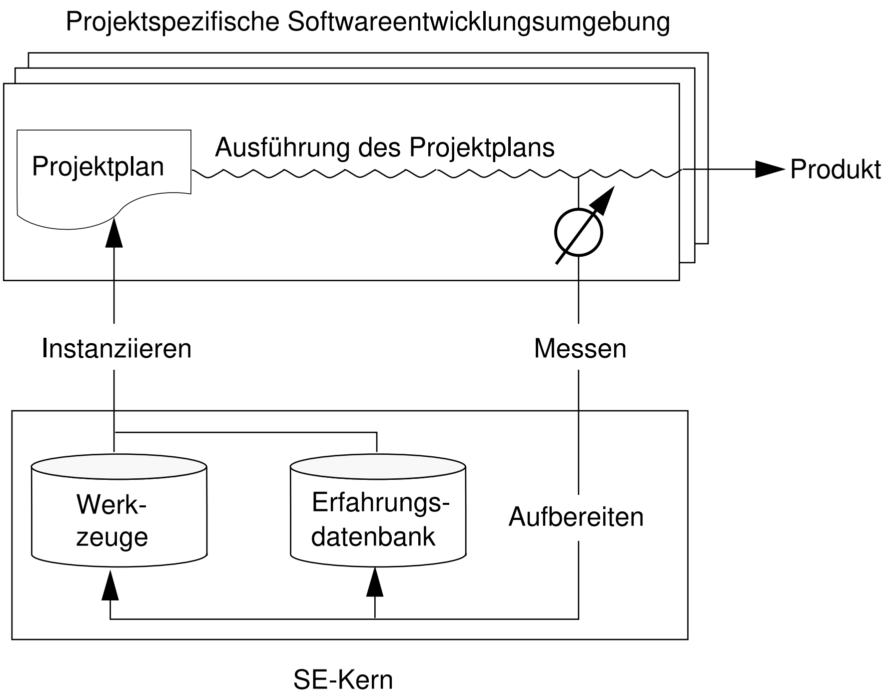

Abb. 1. Rahmenarchitektur einer domänenspezifischen Softwareentwicklungsumgebung

Der SE-Kern stellt eine Sammlung von Entwicklungswerkzeugen sowie ein domänenspezifisches Repository, das aus Mangel an präziseren Begriffen als *Erfahrungsdatenbank* (kurz Erfahrungs-DB) bezeichnet wird, bereit [14]. Die Erfahrungs-DB enthält in einer strukturierten Form nützliche Information aus früheren Projekten, die für die Wiederverwendung besonders aufbereitet wurde. Die formalisierte Darstellung dieser Information für Zwecke der automatischen Wiederverwendung in Generatoren ist ein besonderes Anliegen des SFB. Beispiele für Inhalte der Erfahrungs-DB sind Entwurfsmustersammlungen, Komponenten, Architekturbeschreibungen, dokumentierte Entwurfsentscheidungen, Prozeßmodelle sowie Erfahrungen mit Werkzeugen und sonstigen wiederverwendbaren Artefakten.

Die *domänenspezifische Auslegung* des SE-Kerns wird als essentiell angesehen. Nur durch die Spezialisierung auf eine bestimmte Anwendungsdomäne können Besonderheiten dieser Domäne in der Erfahrungs-DB gezielt berücksichtigt werden. Architekturzentrierte Prozeßmodelle, bei denen der Entwicklungsprozeß durch Vorkenntnisse über geeignete Architekturen einer Anwendungsdomäne gesteuert wird, erscheinen besonders aussichtsreich und stehen gegenwärtig im Vordergrund der Untersuchungen im SFB [2, 19]. Die Produkte aus verschiedenen Entwicklungsprojekten einer Anwendungsdomäne können als Varianten einer Systemfamilie aufgefaßt werden, die derselben Systemarchitektur genügen. Eine *komponentenbasierte Systementwicklung* erleichtert dabei die Konfiguration und Erweiterbarkeit solcher Produkte. Es besteht die berechtigte Hoffnung, daß sich die in einer Anwendungsdomäne bewährten Prinzipien für den Aufbau des SE-Kerns verallgemeinern und auf andere Anwendungsdomänen übertragen lassen.

Wird ein neues Entwicklungsprojekt gestartet, so muß vom SE-Kern eine *projektspezifische Softwareentwicklungsumgebung* bereitgestellt werden. Sie beinhaltet die Auswahl und Einstellung der benötigten Werkzeuge und Kontexte, die Erstellung eines Projektplanes inkl. Instrumentierung sowie die eigentliche Ausführungsbeschreibung des Projekts. Im Verlaufe der Projektdurchführung fallen aufgrund der begleitenden Messungen vielfältige Projektdaten an. Sie werden nach vorgegebenen Methoden analysiert und dienen der inkrementellen Verbesserung/ Ergänzung der in der Erfahrungs-DB gespeichten Pläne und Beschreibungen sowie der Werkzeugsammlung. Jedes Entwicklungsprojekt ist somit als ein rückgekoppeltes Experiment angelegt: neue Erkenntnisse aus der aktuellen Projektdurchführung fließen unmittelbar in die *Verbesserung der Entwicklungsmethodik und des Produktes* selbst zurück.

Eine Schlüsselrolle zur wiederverwendungsgerechten Gestaltung von Software sowie der Entwicklungsprozesse im SFB 501 spielt das Konzept der *Generizität*. Wir verstehen darunter die Trennung einer gegebenen Struktur in variante und invariante Teile sowie die Parametrisierung der varianten Teile, die nach einem vorgegebenen Generierungsschema die varianten Teile erzeugen. Die generische Auslegung einer Komponentenbibliothek im Rahmen einer domänenspezifischen Systemarchitektur ist ein Beispiel dafür [18]. Mit dem Konzept der Generizität wird in verschiedenen Teilprojekten des SFB intensiv experimentiert, wobei als erste Anwendungsdomäne die "Gebäudetechnik" ausgewählt wurde.

4 Die Organisation des SFB 501

Der SFB ist in die folgenden Projektbereiche gegliedert:

A: Softwareentwicklungslabor
B: Generische Modelle und Methoden der Systementwicklung
C: Beschreibungstechniken
D: Prototypanwendungen

Projektbereich A

Im Projektbereich A wird ein Softwareentwicklungslabor betrieben, das die in Abb. 2 gezeigte Rahmenarchitektur realisiert. Alle methodischen Ergebnisse des SFB, die Werkzeugreife erreicht haben, fließen im SE-Kern zusammen. Die Projekte dieses Projektbereichs stellen einerseits die für die Durchführung von Experimenten notwendige Infrastruktur bereit und unterstützen zudem gezielt Entwicklungsprojekte in ausgewählten Anwendungsdomänen. Gegenwärtig werden die folgenden Teilprojekte gefördert:

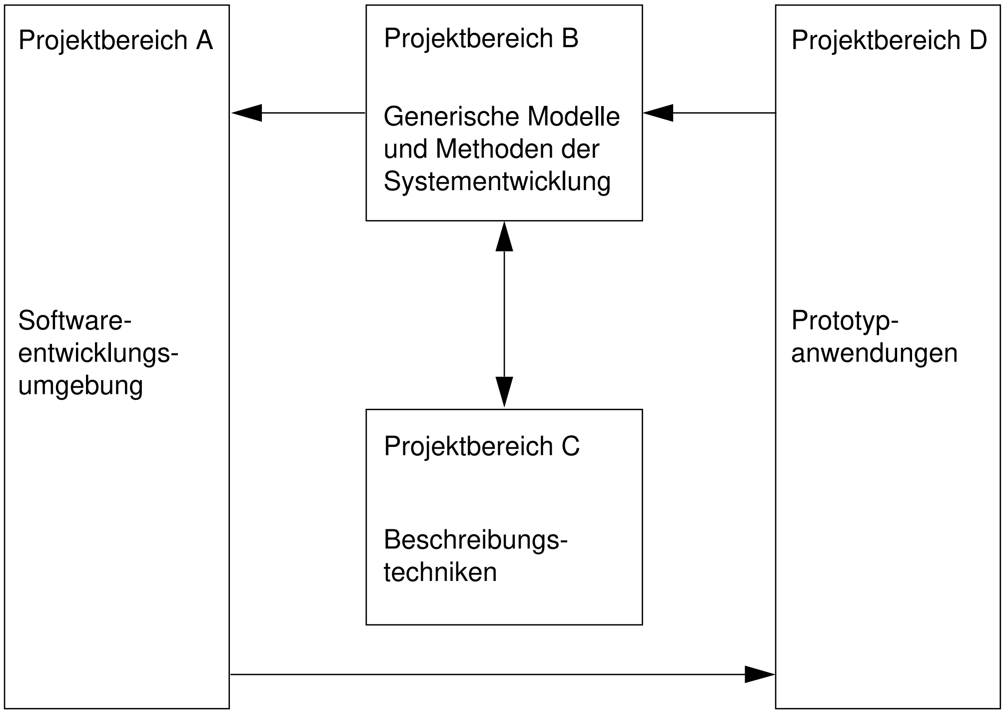

Abb. 2. Organisationsstruktur des SFB 501

Teilprojekt A1: SE-Labor

Softwareentwicklungsansätze (Techniken, Methoden und Werkzeuge) müssen experimentell erprobt werden. Experimente sind nicht nur für die Validierung wissenschaftlicher Hypothesen, sondern auch für die Gewinnung von Erfahrungswerten bezüglich des späteren Einsatzes unter realistischen Projektbedingungen unersetzlich. Ziele des Teilprojektes sind die Bereitstellung einer geeigneten Softwareentwicklungsumgebung für die experimentelle Erprobung der in den übrigen Teilprojekten entwickelten Ansätze sowie die Unterstützung der Teilprojekte bei Planung und Durchführung von Experimenten, bei Analyse und Aufbereitung der experimentell gewonnenen Daten sowie bei Ablage der Ergebnisse in einer Erfahrungs-DB. Die initial aus industriell käuflichen Komponenten aufgebaute Softwareentwicklungsumgebung wird gegenwärtig zur komfortableren Unterstützung von Experimenten um Funktionen für Instrumentierung, umfassendes Konfigurationsmanagement, objekt-relationale Datenbankverwaltung (siehe Teilprojekt A3) sowie Prozeßmanagement (siehe Teilprojekt A2) ergänzt.

Bislang wurden mehrere Experimente – teils in Form kontrollierter Experimente mit einzelnen Entwicklungsansätzen, teils in Form von Fallstudien in Entwicklungsprojekten – unterstützt. Erste Fallstudien in durchgängigen Softwareentwicklungen haben zu quantitativen Baselines über Zeit, Aufwand und Fehler geführt. Diese Baselines werden allen zukünftigen Fallstudien als Referenz für Verbesserung dienen. Mehrere Experimente mit alternativen Entwicklungsansätzen wurden bereits durchgeführt. Erfolgsversprechende Resultate wurden dabei insbesondere mit neuen Spezifikations- und Inspektionsansätzen erzielt. Alle Ergebnisse sind aufbereitet und in der Erfahrungs-DB verfügbar.

In Zukunft sollen verstärkt Variationen von Fallstudien mit dem Ziel durchgeführt werden, das in kontrollierten Experimenten entdeckte Potential neuer Entwicklungsansätze in realistischeren Projektkontexten nachzuprüfen. Dabei wird neben den von Mitarbeitern durchgeführten Fallstudien sowie den von Studenten im Rahmen von Praktika durchgeführten kontrollierten Experimenten zukünftig verstärkt auf industrielle Fallstudien gesetzt. Damit soll eine kontrollierte, durchgängige Erfahrungsgewinnung – ausgehend von kontrollierten Experimenten, über Fallstudien im Labor, bis hin zu Fallstudien in industriellen Pilotprojekten –, wie sie in anderen Ingenieurdisziplinen durchaus üblich ist, auch für Software Engineering realisiert werden.

Teilprojekt A2: Entwicklung einer flexiblen Modellierungs- und Ausführungsumgebung für Softwareentwicklungsprozesse

In jedem Softwareentwicklungsprozeß werden viele Entscheidungen von Entwicklern getroffen, die nicht vorab geplant werden können, sondern auf den Ergebnissen vorangegangener Schritte basieren. Diese Entscheidungen sind durch komplexe Wechselwirkungen miteinander verknüpft. Systematische Unterstützung bei der Durchführung von Softwareentwicklungsprojekten erfordert daher sowohl die integrierte Unterstützung technischer Prozesse als auch des globalen Managements vieler Einzelprozesse und deren Wechselbeziehungen. Dazu wird in diesem Teilprojekt ein Werkzeug entwickelt und experimentell erprobt.

Die Unterstützung kreativer Tätigkeiten erfordert Mechanismen zur Veränderung des Projektzustandes und des Projektplanes während der Abwicklung. Dabei entsteht zum einen das Problem, die Repräsentation des Projektplans zu modifizieren. Zum anderen müssen die Auswirkungen dieser Umplanung auf den real ablaufenden Entwicklungsprozeß kontrolliert werden. Das bedeutet, daß die Projektkoordination systemseitig durch das Verschicken von Benachrichtigungen nach Änderungen und die Verwaltung von Arbeitslisten unterstützt werden muß. Dies erlaubt es, die von einer Änderung (z. B. einer Anforderungsänderung) betroffenen Mitarbeiter zu informieren, so daß diese ihre alten Entscheidungen im Licht der neuen Eingaben nochmals überdenken können. Inhaltlich erzwingt dies, daß das Werkzeug kausale Abhängigkeiten erfaßt und verwaltet, dadurch Verfolgbarkeit herstellt und damit eine Basis für das automatische Benachrichtigen von betroffenen Mitarbeitern erhält.

Die in Teilprojekt B2 entwickelten Techniken und Methoden zur flexiblen Planung und Abwicklung von Softwareprozessen dienen dabei als methodische Grundlage. Die zu realisierende Prozeßunterstützungsumgebung soll ferner das zielorientierte Messen und Bewerten von Prozessen und Produkten eines Projekts gestatten.

Teilprojekt A3: Objekt-relationale Datenbanktechnologie zur Unterstützung des Softwareentwicklungsprozesses

Wiederverwendung ist in großen Softwareentwicklungsprojekten durch eine gemeinsame Datenbank, einem sogenannten Repository, zu unterstützen. Aus der Vielfalt der Entwicklungstätigkeiten ergeben sich weitreichende Anforderungen an das Repository. So müssen sowohl große Mengen einfach strukturierter Daten, wie z. B. in Experimenten gewonnene Meßdaten, als auch komplex strukturierte Entwurfsobjekte, wie z. B. Softwaremodule, und komplexe Dokumente, die auch Bilder und Texte umfassen können, integriert verwaltet und verarbeitet werden. Um die heterogenen Anforderungen bezüglich Daten- und Verarbeitungsmodell angemessen erfüllen zu können, muß das verwendete Datenbanksystem an das jeweilige Anwendungsprofil angepaßt werden können. Die Verwendung objekt-relationaler Datenbanktechnologie, die auf eine Integration objektorientierter Konzepte und relationaler Datenbanksysteme abzielt, scheint hier vielversprechend, da durch das Konzept der Erweiterbarkeit um benutzerdefinierte Datentypen, Funktionen und Zugriffspfadstrukturen eine Anpassung an die Anwendungsdomäne besonders unterstützt wird. Obwohl bereits kommerzielle Produkte verfügbar sind, die in die Klasse der objekt-relationalen Systeme eingeordnet werden, ist bei weitem noch nicht geklärt, inwieweit sich relationale und objektorientierte Konzepte mit dem Ziel einer geeigneten Modellierungsmächtigkeit, ausreichender Erweiterbarkeit, angepaßter Schnittstellen und angemessenem Leistungsverhalten integrieren lassen.

Folglich hat dieses Projekt die folgenden drei Ziele. Erstens soll unter Beachtung des entstehenden SQL3-Standards auf eine geeignete Integration relationaler und objektorientierter Konzepte hingearbeitet werden. Zweitens soll eine angemessene Client/Server-Architektur entwikkelt werden, die die reine Serverzentrierung gegenwärtiger objekt-relationaler Systeme ablöst, so daß ein flexibleres Verarbeitungskonzept für Entwurfsanwendungen, wie die Softwareentwicklung, geboten werden kann. Dies umfaßt neben der Bereitstellung von DBS-Komponenten zur Verwaltung eines Client-Caches auch die Erarbeitung von Konzepten und Mechanismen zur dynamischen Bestimmung des Ausführungsorts (Server oder Client) von Anwendungsfunktionen. Drittens sollen konkrete, für den Softwareentwicklungsprozeß spezifische Erweiterungen konzipiert und realisiert werden, die z. B. die Versionierung von Entwurfsdaten erlauben bzw. die Zusammenarbeit der Entwurfsdatenverwaltungskomponente mit der Softwareentwicklungsumgebung ermöglichen.

Projektbereich B

Im Projektbereich B wird in mehreren Projekten das Potential generischer Ansätze erforscht. Dazu werden generische Prozeßmodelle (Teilprojekte B1, B2) und generische Produktmodelle für Kommunikationssysteme (Teilprojekt B4), Betriebssysteme (Teilprojekt B5) und Anwendungssysteme (Teilprojekt B10) entwickelt und erprobt. In einigen Teilprojekten erstrecken sich die Untersuchungen auch auf Generierungstechniken bis hin zur Entwicklung von Generatoren. Gegenwärtig werden die folgenden Teilprojekte gefördert:

Teilprojekt B1: Generische Modellierung von Prozessen und Experimenten

Softwareentwicklungsprozesse bestimmen die Qualität der resultierenden Software. Es gibt eine Vielzahl von Prozessen sowie Entwicklungsansätzen (Techniken, Methoden und Werkzeuge), die im Rahmen dieser Prozesse eingesetzt werden können. Somit stellt sich folgende Planungsaufgabe: Welcher Prozeß und welche Entwicklungsansätze sind geeignet, vorgegebene Projektziele (z.B. hohe Zuverlässigkeit) unter gegebenen Randbedingungen (z.B. vorgegebene Projektzeit) zu erreichen? Zur Lösung dieser Planungsaufgabe sind Erfahrungswerte bezgl. des Beitrags aller in Frage kommenden Entwicklungsansätze sowie Prozesse auf die relevanten Projektziele unter den gegebenen Randbedingungen notwendig. Ziel dieses Teilprojekts ist die Entwicklung einer Experimentiermethodik für die Gewinnung derartiger (möglichst quantitativer) Erfahrungswerte.

Bislang wurden folgende Komponenten dieser Experimentiermethodik entwickelt: eine formale Sprache zur Formulierung von Softwareprozessen, eine Methodik zum Entwurf von Experimenten, eine Methode zur zielorientierten Ableitung und Definition von Meßpunkten, eine Menge von Methoden zur Datenanalyse sowie ein Repository-Schema zur Ablage experimentell gewonnener Erfahrungen. Diese Experimentiermethodik wird gegenwärtig im SE-Labor (Teilprojekt A1) eingesetzt. Laufende Arbeiten konzentrieren sich auf die Formalisierung sowie generische Auslegung der Experimentiermethodik zur leichteren Instanziierung neuer Experimente für die Applikationsdomäne "Gebäudetechnik" sowie die verbesserte Unterstützung von Softwareentwicklern bei der Ausführung instrumentierter Prozeßmodelle. Hierbei sollen insbesondere die Datenerfassung sowie die Synchronisation von Entwicklungsaufgaben vollständig automatisiert werden.

Wesentliche Forschungsaspekte betreffen:

- die generische Auslegung von Projektplänen und deren Instrumentierung
- die experimentübergreifende Querschnittsanalyse zur Identifikation wiederverwendbarer Erfahrungen
- die Spezifikation eines DB-Schemas für die Erfahrungsdatenbank
- die Entwicklung von Verfahren zur effizienten Wiederverwendung von Erfahrungen.

Die Forschungsergebnisse dieses Teilprojekts gehen in die im SE-Labor bereitgestellte Experimentierumgebung (siehe Teilprojekt A1) ein. Weiterhin wird das in Teilprojekt A2 entwikkelte Werkzeug zur integrierten Unterstützung von management- und technikorientierten Prozessen in Softwareentwicklungsprojekten maßgeblich durch diese Forschungsergebnisse gespeist.

Teilprojekt B2: Flexible Planung und Steuerung von Softwareentwicklungsprozessen

Ziel des Teilprojektes B2 ist die Entwicklung von Techniken und Methoden zur flexiblen Planung und Steuerung komplexer Softwareentwicklungsprojekte. Dazu wurden in der ersten Förderperiode Basistechniken zur Modellierung und Operationalisierung von Projektplänen sowie der verzahnten Planung und Ausführung entwickelt. Unter der verzahnten Planung und Ausführung verstehen wir dabei die Möglichkeit, den initialen, nicht vollständig spezifizierten Plan während der Projektabwicklung sukzessive zu ergänzen und dabei aus vorangegangenen Aktivitäten hervorgegangene oder nur kurzfristig erhältliche Information für die Planung zu nutzen. Große, zum Teil über Jahre laufende Softwareentwicklungsprojekte unterliegen darüber hinaus sich ändernden Rahmenbedingungen und Korrekturanforderungen, die eine Anpassung des Plans noch während der Projektlaufzeit erforderlich machen.

Mechanismen, die eine Umplanung während der Projektabwicklung ("on the fly") ermöglichen, müssen daher integraler Bestandteil vom prozeßsensitiven Softwareentwicklungsumgebungen sein. Aufbauend auf den bisher entwickelten Basistechniken werden daher Techniken und Methoden entwickelt, um Planänderungen "on the fly" zu unterstützen. Zum einen gehören dazu Techniken zur Ummodellierung der Projektpläne. Zum anderen müssen die Auswirkungen von Änderungen in abhängigen Teilen des Projektplans auf den Projektzustand berechnet und behandelt werden können. Dazu werden Techniken entwickelt, das Projekt nach einer Änderung automatisch in einen konsistenten Zustand zurückzuführen. Für Situationen, in denen eine automatische Reaktion auf Änderungen nicht möglich ist, wird ein Benachrichtigungsmechanismus entwickelt, der betroffene Mitarbeiter über Ursache und Art der Änderung informiert. Voraussetzung dafür ist die Kenntnis kausaler Abhängigkeiten zwischen den Prozessen, Entscheidungen sowie den im Projektverlauf erstellten Produkten. Um eine möglichst gezielte und genaue Benachrichtigung der Mitarbeiter zu ermöglichen, wird dabei Wissen über die Domäne des Projektes ausgenutzt. Teil der Forschungsaufgabe ist daher die Entwicklung von Methoden und Techniken zur automatischen Ableitung von Änderungsabhängigkeiten aus dem Projektplan sowie die Entwicklung einer Sprache zur Formulierung von domänenspezifischen Abhängigkeitsmustern. Weiterhin werden Techniken zur Abbildung der modellierten Abhängigkeitsmuster in konkrete Änderungsabhängigkeiten benötigt.

Die Forschungsergebnisse dieses Projektes gehen in die Entwicklung eines Werkzeugs zur integrierten Unterstützung von management- und technikorientierten Prozessen in Softwareentwicklungsprojekten (siehe Teilprojekt A2) ein.

Teilprojekt B4: Generische Kommunikationssysteme

Langfristiges Ziel des Teilprojekts ist die Bereitstellung von Verfahren und Techniken zur Entwicklung anwendungsangepaßter, maßgeschneiderter Kommunikationssysteme unter besonderer Berücksichtigung von Echtzeitanforderungen. Die hierzu erforderliche generische Auslegung von Produkten wird durch musterbasierte Ansätze zur Steigerung der Wiederverwendung von Lösungskomponenten und Entwicklungs-Know-how erreicht.

Grundlagen sind ein allgemeiner, durchgängiger Entwicklungsprozeß, der zur Steigerung des Wiederverwendungspotentials verfeinert worden ist, sowie mehrere Mustersammlungen. In der Anforderungsphase kommen Requirement Patterns zur Anwendung, die aus einer Mustersammlung selektiert, an die spezielle Problemstellung adaptiert und schließlich durch einen Kompositionsschritt integriert werden. Im Rahmen des Systementwurfs erfolgt anschließend die Festlegung einer Systemarchitektur und die Zuordnung bzw. Verfeinerung von Anforderungen zu Architekturkomponenten. Ein Teilergebnis ist die Definition von Kommunikationsdiensten, die den Ausgangspunkt für die musterbasierte Entwicklung von Kommunikationsprotokollen bilden. Unterstützt wird diese Entwicklung durch eine umfangreiche Protokollmustersammlung, die sich in mehreren Fallstudien (u.a. ST2+, IPv6, RTP) bewährt hat. Zur Formulierung von Protocol Patterns bzw. zu deren Instanziierung wird SDL (Specification and Description Language), eine international genormte, objektorientierte, graphische FDT (Formal Description Technique), verwendet. Für SDL existieren kommerzielle, rechnergestützte Werkzeugumgebungen, die u.a. die graphische Erstellung von Spezifikationen, deren Analyse und Simulation sowie die automatische Code-Erzeugung abdecken und im SFB 501 eingesetzt werden. Protocol Design Patterns und Entwicklungs-Know-how werden in der Erfahrungsdatenbank gesammelt und sind damit auch in Folgeprojekten nutzbar.

Zukünftige Arbeiten dienen u.a. dem Ausbau des SDL-spezifischen Teils der Erfahrungsdatenbank sowie der kontinuierlichen, systematischen Verbesserung des Entwicklungsprozesses und der Mustersammlungen unter Einsatz des Quality Improvement Paradigms (QIP) und des Goal/Question/Metric-Modells (GQM).

Teilprojekt B5: GeneSys - Generische Systemsoftware

Betriebssysteme bilden eine unabdingbare Grundlage für fast jede Art von Softwarelösung. Das Ziel des Teilprojektes B5 besteht darin, maßgeschneiderte Laufzeitplattformen für eingebettete Systeme zur Verfügung zu stellen, die optimal an die Gegebenheiten der jeweiligen Anwendung angepaßt sind. Der verwendete Ansatz basiert auf dem Konzept generischer Softwarekomponenten und dem Einsatz von Generatortechniken.

Generische Komponenten sind Softwaremodule, die es erlauben, ihre Eigenschaften in gewissem Rahmen zu modifizieren, ohne notwendigerweise manuelle Code-Änderungen vornehmen zu müssen. Die Anpassung kann dabei sowohl funktionale als auch nichtfunktionale Aspekte wie Skalierbarkeit, Fehlertoleranz, Zeit- und Speicherkomplexität betreffen. Der Variationsspielraum generischer Komponenten wird durch sogenannte generische Parameter beschrieben, "Platzhalter" für spezifische, variabel gelassene Eigenschaften der Komponente. Sie stellen auch Ansatzpunkte für Generatoren dar, die es erlauben, Komponenten-Code aus einer abstrakteren Beschreibung zu erzeugen. Aus solchen generischen Komponenten kann durch einen komplexen Prozeß, der die wiederholte Anwendung der Teilschritte Auswahl, Konfiguration und Kombination von Komponenten umfaßt, eine auf eine spezifische Anwendung zugeschnittene Laufzeitplattform konstruiert werden.

Die gegenwärtigen Forschungsarbeiten im Teilprojekt B5 konzentrieren sich zum einen darauf, die einzelnen Prozesse für Softwareentwicklung auf Basis generischer Komponenten detailliert zu untersuchen und zu beschreiben sowie geeignete Unterstützungswerkzeuge zur Verfügung zu stellen. Zum anderen bilden Architekturaspekte bei komponentenbasierter Softwareentwicklung, beispielsweise architekturelle Kompatibilität als Voraussetzung für Komposition, einen weiteren Forschungsschungsschwerpunkt im Teilprojekt B5.

Teilprojekt B10: Anwendungsentwicklung mit vorkonfektionierten Softwaresystemen

Das Teilprojekt B10 verfolgt einen architekturbasierten Ansatz zur Softwareentwicklung, der in hohem Maße auf die Wiederverwendung von Artefakten ausgerichtet ist, die in früheren Projekten erstellt wurden. Es wird dabei angestrebt, bei der Realisierung von Softwareprojekten aus einem bestimmten Anwendungsfeld die Architektur – d.h. die Struktur bzw. Strukturen – erprobter Systeme wiederzuverwenden. Diese wiederverwendete Architektur stellt dann einen Rahmen dar, in dem die weiteren Entwicklungstätigkeiten ablaufen, beispielsweise die Zerlegung des Systems in seine Modulstruktur. Durch Wiederverwendung einer vollständigen Systemarchitektur anstelle einzelner Subsysteme oder Komponenten können architekturelle

Fehlentscheidungen vermieden werden, die ansonsten gravierende Auswirkungen auf das Gesamtsystem und den gesamten Entwicklungsprozeß haben könnten.

Charakteristisch für einen Laboransatz zur Softwareentwicklung, wie ihn der SFB verfolgt, ist die aufeinanderfolgende Realisierung von Softwareprojekten mit sich nur gering unterscheidenden Anforderungen aus demselben Anwendungsfeld, also eine sogenannte Produktfamilie. Die Wiederverwendung von Architektur, wie sie in diesem Projekt untersucht wird, stellt eine geeignete Grundlage für solche Entwicklungsprozesse dar. Neben der Wiederverwendung auf struktureller Ebene werden dadurch auch günstige Voraussetzungen für die systematische Wiederverwendung auf der Ebene von Softwareprodukten geschaffen. Dies wird mit Hilfe von White-Box Frameworks umgesetzt, die in einem offenen Rahmen zum einen vorgefertigte Softwareelemente enthalten, zum anderen aber an definierten Stellen Lücken lassen, die in einem Entwicklungsprojekt auf anwendungsspezifische Weise zu füllen sind.

Die intensive Wiederverwendung in allen Phasen des Entwicklungsprozesses hat Konsequenzen für die Gestaltung praktisch aller Entwicklungsschritte. So kommen beispielsweise bereits bei der Anforderungsbeschreibung anwendungsfeldspezifische Kriterien und bekannte Strukturen zum Einsatz. Diese Ausrichtung der einzelnen Aktivitäten im Entwicklungsprozeß auf Architekturaspekte und auf Wiederverwendung stellt einen der wichtigsten Forschungsschwerpunkte in B10 dar.

Projektbereich C

Im Projektbereich C werden die Anforderungen an Beschreibungstechniken, die sich aus dem Projektbereich B sowie SFB-übergreifenden Experimenten ergeben, erfaßt und geeignete Lösungen durch Erweiterung/Modifikation existierender Beschreibungstechniken entwickelt. Die eingesetzten Techniken müssen insbesondere Durchgängigkeit und Verfolgbarkeit von Entwicklungsprozessen garantieren, die sich über mehrere Beschreibungsebenen erstrecken. Der Projektbereich leistet auch Hilfestellung bei der Formalisierung von Schlüsselkonzepten wie Generizität und Komposition, die in verschiedener Ausprägung im Projektbereich B benutzt werden. Gegenwärtig wird ein Teilprojekt im Projektbereich C gefördert:

Teilprojekt C1: Formale Beschreibungstechniken

Das Teilprojekt C1 untersucht formale Beschreibungstechniken im Kontext der Anwendungsdomäne und unterstützt die übrigen Teilprojekte bei der Verwendung derartiger Techniken. Formale Beschreibungstechniken werden sowohl zur Beschreibung einzelner (Teil-)Produkte als auch des Entwicklungsprozesses eingesetzt. Die Untersuchungen befassen sich vor allem mit der methodischen Verwendung dieser Techniken.

Für zu erstellende Systeme werden Beschreibungen im Bereich von Anforderungen bis zu Entwürfen betrachtet. Dabei werden zunächst funktionale und temporale Eigenschaften, später auch weitere nichtfunktionale Eigenschaften berücksichtigt. Um einzelne Entwicklungsschritte formal begründen und Beschreibungen analysieren zu können, wird Domänenwissen – z.B. in Form physikalischer Modelle – und Entwurfswissen – z. B. Modelle generischer Architekturen – formalisiert und einbezogen.

Für dieses breite Spektrum von Beschreibungen werden existierende Sprachen verwendet und integriert. Diese werden nötigenfalls in ihrer Beschreibungskraft auf die Bedürfnisse der Anwendungsdomäne hin erweitert oder eingeschränkt. Zentrale Aufgaben sind die Einbeziehung von Techniken wie Parametrisierung und Objektorientierung in die Beschreibungen sowie die Entwicklung und Untersuchung geeigneter architekturbedingter Kompositionsoperatoren, die über die übliche sequentielle oder parallele Komposition von Systemen hinausgehen.

Projektbereich D

Der Projektbereich D ist den Anwendungen gewidmet. Der SFB hat sich als erstes Anwendungsfeld die Gebäudeautomation aus der Domäne Steuerungs- und Überwachungssysteme ausgewählt. Die hier angesiedelten Projekte werden von Experten aus der Automatisierungstechnik begleitet und verfolgen das Ziel, Anwendungswissen mit dem spezifischen Wissen im SFB 501 bei der Systementwicklung zu koppeln. Der Projektbereich spielt bei SFB-übergreifenden Experimenten auch die Rolle des Kunden, der Entwicklungsaufträge in Form einer Problembeschreibung vorbereitet. In dieser Rolle kommt dem Teilprojekt auch die Aufgabe zu, Simulations- und Testumgebungen für das bearbeitete Anwendungsfeld bereitzustellen. Gegenwärtig werden die folgenden Teilprojekte gefördert:

Teilprojekt D1: Anwendungssystem Gebäude

Methoden zur Entwicklung komplexer Systeme müssen an entsprechenden Anwendungen erprobt werden. Das Anwendungsfeld des SFB ist die Gebäudeautomation als Ausschnitt reaktiver Systeme, da dieses ein weites Spektrum von recht einfachen bis zu äußerst komplexen Steuerungen mit verschiedensten funktionalen und nichtfunktionalen Eigenschaften bietet.
Das Teilprojekt D1 bearbeitet zwei Aufgaben. Als Infrastrukturprojekt unterstützt D1 den SFB bei Fallstudien und, während des Softwareentwicklungsprozesses, bei Experimenten von der Anwendungsseite her. Hierzu stellt D1 Gebäudesimulatoren und ein reales, instrumentiertes Testfeld zur Verfügung.

Darüber hinaus untersucht D1, wie sich existierende Techniken und Methoden domänenspezifisch anpassen lassen, um eine hohe Softwarequalität bei erheblich verringerten Entwicklungskosten zu erreichen. Der Schwerpunkt dieser Forschungsarbeiten liegt in den Analyse- und Entwurfsphasen, die zur Validierung der erzeugten Produkte von Prototyping und Simulation begleitet werden.

In der Analysephase werden vorgegebene Gebäude- und Installationsstrukturen zunächst auf eine generische Modellarchitektur abgebildet. Das hierbei notwendige Expertenwissen wird von D1 und D2 in die Erfahrungsdatenbank eingebracht. Das Ergebnis sind ausführbare SDL-Modelle, die mittels Generatoren direkt zum Prototyping geeignet sind. Hierzu notwendige Gebäudesimulatoren werden auf ähnliche Weise mit Hilfe von Design Patterns erzeugt.

In der anschließenden Entwurfsphase werden die getesteten SDL-Modelle auf eine reale, verteilte Umgebung kostenminimal abgebildet, wobei funktionale und nichtfunktionale Anforderungen eingehalten werden müssen. Das Ergebnis sind erneut ausführbare Modelle für ein genaueres Prototyping sowie Anforderungen an die Hardwareschnittstelle, die unter anderem zu einer speziell zugeschnittenen Realisierung der System- und Kommunikationssoftware durch B5 und B4 führen.

Teilprojekt D2: Modellbasierte Entwicklung wiederverwendbarer Regelungsalgorithmen

Software zur gezielten Beeinflussung von Prozessen der realen Welt verkörpert Regelungs- und Steuerungsalgorithmen. Mittels Sensoren gewonnene Prozeßinformation wird zu Befehlen an Aktoren verarbeitet. Je nach Größe und Komplexität des betrachteten reaktiven Systems können Hunderte bis Tausende von Sensor- bzw. Aktorsignalen auftreten. Die Forderung nach Wiederverwendbarkeit komplexer Softwarelösungen überträgt sich so auf die zugrundeliegenden Regelungs- und Steuerungsalgorithmen auf unterschiedlichen Hierarchieebenen.

Gerade die Applikation im Bereich der Gebäudeautomation zur Regelung bzw. Steuerung von Heizung, Klima und Lüftung weist eine Vielzahl miteinander verkoppelter Teilprozesse unterschiedlicher Eigendynamik auf, was in die Entwicklung der Algorithmen einfließen muß. Zu betrachten sind hierbei Elemente der Wärme- und Kälteerzeugung (Heizkessel mit konventionellen Brennern, Solarkollektoren, Kälteaggregate u.a.) sowie der Verteilung (Kollektorheizung, Fußbodenheizung, Wärmetauscher u.a.). Zudem kommt es nach Architektur und bauphysikalischen Gegebenheiten zu weiteren thermischen Verkopplungen.

Es soll zunächst eine umfassende Bibliothek mathematischer Teilprozeßmodelle, geeignet zum Regelungsentwurf, inklusive eines Regelwerkes zur generischen Erzeugung mathematischer Gesamtmodelle angelegt werden (Teil A). Hierauf basierend werden dann die eigentlichen Regelungsalgorithmen entworfen. Ziel ist es, eine Sammlung validierter Regelungs- und Steuerungsalgorithmen zu erhalten, aus denen spezifisch für die jeweiligen Anwendungen ein Gesamtregelungs- und -steuerungsalgorithmus erzeugt werden kann (Teil B). Wiederverwendet werden in beiden Teilen die Algorithmen, ihre Realisierung in Software sowie die im Regelwerk niedergelegten Erfahrungen.

Wesentlicher Bestandteil ist daher die Entwicklung des Regelwerkes zur Auswahl, Anpassung und Verknüpfung der Algorithmen nach den konkreten architektonischen, bauphysikalischen und automatisierungstechnischen Daten. Hierbei soll mit möglichst wenigen Algorithmen ein möglichst großes Anwendungsspektrum abgedeckt werden. Auch ist die Zahl der zur Anpassung benötigten Parameter zu minimieren. Dies erfordert eine hohe Robustheit der Regelung und die adaptive Anpassung an größere Änderungen der Strecke.

Querschnittsprojekte

Neben diesen geförderten Teilprojekten führt der SFB auf Eigeninitiative Querschnittsprojekte durch, an denen Mitarbeiter aus mehreren oder allen Teilprojekten beteiligt sind. Es handelt sich hier meist um experimentelle Entwicklungsprojekte mit dem Ziel, spezielle Erfahrungen zu sammeln. Diese Form der Zusammenarbeit zwischen den Teilprojekten hat sich gut bewährt, um bei den SFB-Mitarbeitern ein übergreifendes Verständnis der SFB-Thematik zu erzeugen.

5 Langfristige Ziele

Bei einer angenommenen Laufzeit von 12 Jahren hat sich der SFB nach Ablauf von jeweils dreijährigen Förderungsperioden die folgenden Ziele gesetzt:

1. Periode
Bereitstellung einer initialen domänenspezifischen Softwareentwicklungsumgebung für das Anwendungsfeld Gebäudeautomation.

2. Periode
Durchführung von Fallstudien und Experimenten im Anwendungsfeld Gebäudeautomation mit dem primären Ziel, eine Erfahrungs-DB aufzubauen und den Effekt ihrer Wiederverwendung zu erproben.

3. Periode
Übertragung der Ergebnisse auf ein bis zwei weitere Anwendungsfelder; verstärkte Einbeziehung der Industrie.

4. Periode
Transferprojekte mit dem Ziel, die entwickelte Technologie bei ein bis zwei industriellen Kooperationspartnern zu installieren.

Als wesentliches Ergebnis des SFB 501 bei erfolgreichem Verlauf unserer Forschungen erhoffen wir uns das Vorliegen eines methodischen Rahmens inkl. darin eingebetteter Werkzeuge, die einem Softwareunternehmen die Instanziierung einer domänenspezifischen Softwareentwicklungsumgebung erlauben. Anwendungen dienen ausschließlich der Erprobung unserer softwaretechnologischen Methoden; sie selbst sind nicht Gegenstand der Forschung.

Weitere Informationen stehen unter http://www.sfb501.uni-kl.de zur Verfügung.

Literatur

1. Bauer, F. L.: Software-Engineering – wie es begann, Informatik-Spektrum 16(5), 259-260 (1993)
2. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, Addison-Wesley, Reading, MA (1998)
3. 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)
4. Coad, P., North, D., Mayfield, M.: Object Models – Strategies, Patterns & Applications, Prentice Hall, Englewood Cliffs, NJ (1995)
5. Denert, E.: Software-Engineering in Wissenschaft und Wirtschaft: Wie breit ist die Kluft?, Informatik-Spektrum 16(5), 295-229 (1993)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, MA (1994)
7. Goos, G.: Programmiertechnik zwischen Wissenschaft und industrieller Praxis, Informatik-Spektrum 17(1), 11-20 (1994)
8. Endres, A.: Lessons Learned in an Industrial Software Lab, IEEE Software, 58-61 (Sept. 1993)
9. Endres, A.: Software und Software-Entwicklung im Wandel: ein historischer Vergleich, Informatik-Spektrum 16(5), 261-265 (1993)
10. Frakes, W., Terry, C.: Software Reuse: Metrics and Models, ACM Computing Surveys 28(2), 415-435 (1996)
11. Johnson, R., Foote, B.: Designing Reusable Classes, Journal on Object-Oriented Programming 1(2), SIGS Publications, (1988)
12. Krueger, C. W.: Software Reuse, ACM Computing Surveys 24(2), 131-183 (1992)
13. Mili, H., Mili, F., Mili, A.: Reusing Software: Issues and Research Directions, IEEE Transactions on Software Engineering Vol. 21(6), 528-562 (1995)
14. Nagl, M.: Software-Entwicklungsumgebungen: Einordnung und zukünftige Entwicklungslinien, Informatik-Spektrum 16(5), 273-280 (1995)
15. Rombach, H. D.: Software-Qualität und Qualitätssicherung, Informatik-Spektrum 16(5), 267-272 (1993)
16. 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, Springer, Berlin (1993)
17. Rombach, H. D., Verlage, M.: Directions in Software Process Research, in: Advances in Computers 41, M. V. Zelkowitz (Ed.), Academic Press, 1-63 (1995)
18. Rossak, W., Kirova, V., Jololian, L., Lawson, H., Zemel, T.: A Generic Model for Software Architectures, IEEE Software, 84-92 (1997)
19. Shaw, M., Garlan, D.: Software Architecture-Perspectives on an Emerging Discipline, Prentice Hall, Upper Saddle River, NJ (1996)
20. Szyperski, C.: Component Software – Beyond Object-Oriented Programming, Addison-Wesley, Harlow, U.K. (1997)
21. Wendt, S.: Defizite im Software Engineering, Informatik-Spektrum 16(1), 34-38 (1993)