9 Migration von Altsystemen zu dienstorientierten Architekturen

Stefan Krieghoff, Wilhelm Hasselbring, Ralf Reussner

Während die Bedeutung von Multi-Schichten-Architekturen für Unternehmensinformationssysteme allgemein akzeptiert ist und deren Vorteile ausführlich publiziert wurden [Has00], ist die systematische Migration monolithischer Legacy-Systeme hin zu solchen Multi-Schichten-Architekturen nur in einem wesentlich geringeren Maße bekannt. In diesem Kapitel werden die Möglichkeiten einer »sanften« Migration diskutiert.

Wir berichten über Erfahrungen, die wir mit der Migration kommunaler Informationssysteme hin zu einer Multi-Schichten-Architektur gemacht haben. Die Erfahrungen werden durch die Beschreibung des zugrunde liegenden Musters verallgemeinert, so dass sie für ähnliche Aufgaben der Migration von Architekturen wiederverwendet werden können. Das abgeleitete *Dublo*-Muster basiert auf der teilweisen Duplikation der Geschäftslogik zwischen Legacy-System und neuer Mittelschicht. Auch wenn dadurch zu einem gewissen Grad das Prinzip »Separation of Concerns« verletzt wird, erhält man im Gegenzug ein hohes Maß an Flexibilität und die Möglichkeit einer sanften Migration.

Zumindest aus Anwendersicht sind viele Software-Einsatzbereiche durch Produkte mit vollständigem Funktionsumfang und ausgezeichneter Qualität abgedeckt. Entwicklung, Stabilisierung und Pflege dieser Software-Produkte stellen aus Sicht der Software-Hersteller erhebliche Investitionen dar. Aufgrund der Schnelllebigkeit im Bereich des Software Engineering veralten die bei solchen Entwicklungen verwendeten Software-Technologien schneller als das Produkt selbst. Somit fällt ein deutlich wachsender Anteil aller Software-Projekte in den Bereich der Migration und oft auch in den der Architekturmigration, ein im Gegensatz zu Verfahren zur Neuentwicklung recht neues, wenig bearbeitetes Forschungsgebiet.

In einer Kooperation zwischen der Forschung und einem kommunalen IT-Dienstleister wurde eine zukunftsfähige Software-Architektur erarbeitet, die aktuellen Standards entspricht und die die speziellen Randbedingungen kommunaler Verwaltungen berücksichtigt. Die selbst entwickelten Produkte des Softwarehauses sind bei ca. 200 kommunalen Kunden im Einsatz, haben lange und kosten-

intensive Phasen der qualitativen Stabilisierung und Funktionsanreicherung (was wir mit IQTF für Increasing Quality Times Functionality abkürzen) hinter sich und erfordern ständige Pflege, häufig durch Änderungen gesetzgeberischer Verwaltungsvorgaben.

Bei der Einführung einer neuen Software-Architektur in das Produktportfolio darf der Aufwand nicht unterschätzt werden. Insbesondere ist ein Konzept für eine Architekturmigration notwendig. Daraus entstand das Dublo-Architekturmuster (Dublo = Dual Business Logic) [HRJ+04].

Bei der Umsetzung des Dublo-Architekturmusters mit den vorhandenen, erfahrenen Software-Entwicklern entstanden neue Probleme und Verzögerungen, deren Ursachen häufig im nichttechnischen Bereich lagen und die man bei Beachtung bestimmter organisatorischer, psychologischer und betriebswirtschaftlicher Aspekte hätte verhindern können. Aus den gemachten Erfahrungen werden diese Aspekte isoliert und aufgezählt. Ein derartiger interdisziplinärer Ansatz – verwendet bereits bei Definition und Auswahl einer zukünftigen Software-Architektur und des Migrationspfades dorthin – kann helfen, die Erfolgsquote solcher Umstellungsprojekte anzuheben.

9.1 Projektkontext

Die KDO (Zweckverband Kommunale Datenverarbeitung Oldenburg) ist ein erfolgreicher IT-Dienstleister, der domänenspezifische Software-Lösungen für kommunale Verwaltungen anbietet (`http://www.kdo.de`). Bisher basieren die Client-Server-Lösungen der KDO hauptsächlich auf Informix-Datenbanken mit Informix 4GL [IBM] und dem 4Js-Laufzeitsystem (serverseitig) inklusive dem 4Js-Windows-Frontend für Clients [Fou]. Die Lösungen der KDO können wahlweise dezentral oder zentral mit der KDO als ASP betrieben werden.

Die KDO entschied sich, von der (monolithischen) Zwei-Schichten-Architektur hin zu einer standardbasierten und zukunftsorientierten Multi-Schichten-Architektur zu migrieren, die den Einsatz moderner Software-Engineering-Methoden ermöglicht. Bei der Veränderung von traditioneller hin zu komponentenbasierter Software-Entwicklung galt es, besonders auf die sich ergebenden Risiken und neuen Herausforderungen zu achten [RR03, Vit03].

Um den Übergang zu neuen Technologien zu bewältigen, entschied sich die KDO für eine Zusammenarbeit mit OFFIS, dem Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -systeme. OFFIS ist ein An-Institut des Department für Informatik der Universität Oldenburg (`http://www.offis.de`). Die hier vorgestellte Arbeit ist das Ergebnis gemeinsamer Anstrengungen der Software-Engineering-Gruppe der Universität, der Abteilung Betriebliches Informations- und Wissensmanagement von OFFIS und der KDO. Die Kooperation wird begleitet von einer Schulung der KDO-Mitarbeiter in objektorientierter Modellierung und Enterprise- Java-Technologien, die von OFFIS

durchgeführt wird. Dadurch werden neue Software-Engineering-Methoden aus der universitären Forschung durch ein assoziiertes Technologie-Transfer-Institut in die industrielle Praxis übertragen.[1]

9.2 Technische Aspekte sanfter Migration

Es ist offensichtlich, dass man nicht einfach vorhandenen Quellcode vernachlässigen und in einem Schritt durch neuen ersetzen kann:

- Legacy-Systeme stellen wichtige Investitionen dar, die nicht einfach außer Betrieb genommen werden können.
- Der Betrieb muss während des Übergangs weitergehen. Ein Unternehmen kann nicht – nur zur Einführung einer neuen Software-Architektur – mehrere Monate oder auch Jahre lang seinen Betrieb einstellen oder aufhören, seine Produkte und Dienstleistungen zu verkaufen.
- Legacy-Quellcode ist häufig der einzige Ort, an dem die Geschäftslogik dokumentiert ist; die entsprechenden Entwickler können bereits das Unternehmen verlassen haben. Eine vollständige Reimplementierung der Geschäftslogik ist deswegen meist nicht mit vertretbarem Aufwand durchführbar.
- Die Entwicklung eines neuen Systems erfordert einen signifikanten Zeitraum, einschließlich einer ausreichend langen Phase der Stabilisierung des neuen Systems durch den praktischen Einsatz. Während dieses Gesamtzeitraums muss das alte Legacy-System i.d.R. ebenfalls weitergepflegt werden, wodurch zeitgleich Kosten sowohl für das Alt- als auch das Neusystem entstehen.

Folglich sind sanfte Migrationspfade und die Integration von Altsystemen essenziell für die Praxis der Integration von Informationssystemen [BS95, Has00].

In diesem Abschnitt präsentieren wir ein Architekturmuster, das die Integration monolithischer Legacy-Systeme in moderne Multi-Schichten-Architekturen beschreibt. Dieses Muster, genannt *Dublo* für DUal Business LOgic [HRJ+04], implementiert Geschäftslogik an zwei Stellen: im Legacy-Code und in der neuen Mittelschicht. Während diese (teilweise) Duplikation einer klaren Trennung der Aspekte scheinbar widerspricht, so erlaubt sie doch, wie wir zeigen werden, einen sanften Migrationspfad.

Das Muster verallgemeinert unsere speziellen Erfahrungen und Entwurfsdiskussionen in einem Migrationsprojekt für kommunale Informationssysteme. Wir diskutieren die Legacy-Architektur vor Projektstart und die im Projektverlauf aufgetretenen Probleme wie Technologieauswahl und Migrationsalternativen. Obgleich jedes Projekt seine Eigenheiten aufweist, verallgemeinern wir vom vorge-

[1]Die Autoren möchten den ehemaligen OFFIS-Mitarbeitern Holger Jaekel, Goran Martinic, Jürgen Schlegelmilch und Thorsten Teschke sowie den KDO-Mitarbeitern Marc Langnickel und Detlef Meyer für die sehr gute Mitarbeit in diesem Projekt danken.

stellten Projekt durch Diskussion seiner wiederkehrenden Aspekte. Darüber hinaus ist die monolithische Systemarchitektur, von der wir ausgingen, sicher repräsentativ für viele existierende Systeme, die zu migrieren sind.

9.2.1 Multi-Schichten-Architekturen für betriebliche Informationssysteme

Der Hauptvorteil der Verwendung von Multi-Schichten-Architekturen für Informationssysteme ist die klare Trennung der Aspekte (Separation of Concerns) zwischen Nutzungsschnittstellen-Logik (in der Präsentationsschicht), Geschäftslogik (in der Mittelschicht) und dem Management der Datenpersistenz (in der Datenschicht). Diese Trennung der Aspekte liefert viele nützliche Eigenschaften wie etwa die unabhängige Austausch- und Anpassbarkeit von Komponenten, die Möglichkeit, betriebliche Erfordernisse in der Geschäftslogik zu verfolgen, und die Transparenz von Aspekten des Datenbankmanagements. Deswegen haben sich Multi-Schichten-Architekturen in modernen Unternehmensinformationssystemen allgemein durchgesetzt. Da jedoch Altsysteme meist nicht derart strukturiert, sondern in der Regel monolithische Ansammlungen von Legacy-Code sind, stehen viele Unternehmen vor dem Problem, die Altsysteme in die neuen Multi-Schichten-Architekturen migrieren zu müssen.

9.2.2 Architektur und Migrationsprozess

Vorhandene 4GL-Architektur

Die vorhandene Architektur ist in Abbildung 9.1 dargestellt. Informix- bzw. 4Js-4GL ist die zur Implementierung der serverseitigen Geschäftslogik verwendete Sprache. 4Js ist eine Entwicklungs- und Laufzeitumgebung, die es erlaubt, grafische User-Interfaces auf Basis des Tcl/Tk-Toolkits [Ous94] zu generieren. Hierbei wird die Geschäftslogik fest mit dem Datenbankmanagementsystem verbunden; deswegen ist die Serverseite als eine Schicht zu betrachten.

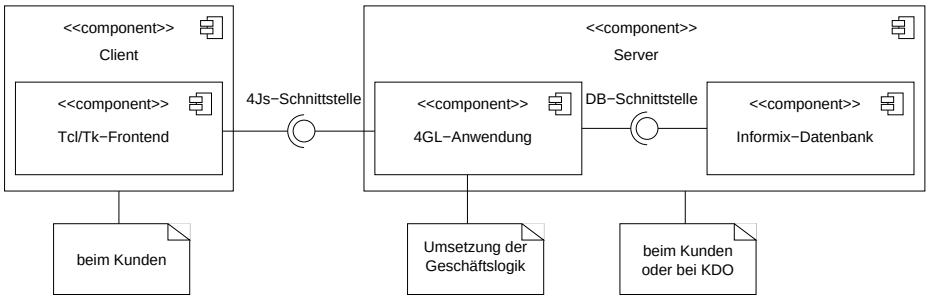

Abbildung 9.1: Vorhandene Informix Zwei-Schichten-Architektur

Technologieauswahl

Die erste Aufgabe war die Auswahl der für die Realisierung zu verwendenden Komponententechnologie, deren Fähigkeiten und Eigenschaften von hoher Wichtigkeit sind. Im vorgestellten Kontext wurde J2EE [Micb] gewählt.

Als konkrete Werkzeuge wurden ausgewählt der BEA Weblogic-Applikationsserver [BEA], Together [Togb] für die Modellierung und JBuilder [JBu] als Entwicklungsumgebung. Zusätzlich zu kommerziellen Werkzeugen und System-Software sind verschiedene Open-Source-Systeme im Einsatz, z.B. Linux als Server-Betriebssystem.

Standardisierte und etablierte Muster sind für J2EE-Systeme dokumentiert [ACM03, BCJ+02, Bie02, Bro02, Mar02, Sun, VSW02]. Dieses wird als wichtige Vorbedingung erachtet, damit die geeigneten Architekturen für unsere Domäne entworfen werden können. Für Details zum Auswahlprozess sei auf [HRJ+04] verwiesen.

Man beachte, dass das Architekturmuster, das in Abschnitt 9.2.3 vorgestellt wird, unabhängig ist von der eingesetzten Middleware-Technologie.

Der Prozess der Architekturauswahl

Der J2EE-Standard empfiehlt eine Multi-Schichten-Architektur. Ein typisches Beispiel mit 4 Schichten wird in Abbildung 9.2 veranschaulicht. In diesem Beispiel enthält die Clientschicht Java-Client-Anwendungen, die auf die Mittelschicht per Remote Method Invocation (RMI) zugreifen. Auf diese Mittelschicht, die aus einem Applikationsserver mit einem Container für Enterprise Java Beans (EJB) [Micb] besteht, kann auch aus Web-Containern heraus zugegriffen werden, wobei der Web-Container Web-Browser-Anfragen via HTTP bedient; zur Vereinfachung der Darstellung wird auf diese Alternative verzichtet. Dieses Kapitel konzentriert sich auf die serverseitige Komponententechnologie. Der EJB-Container verwal-

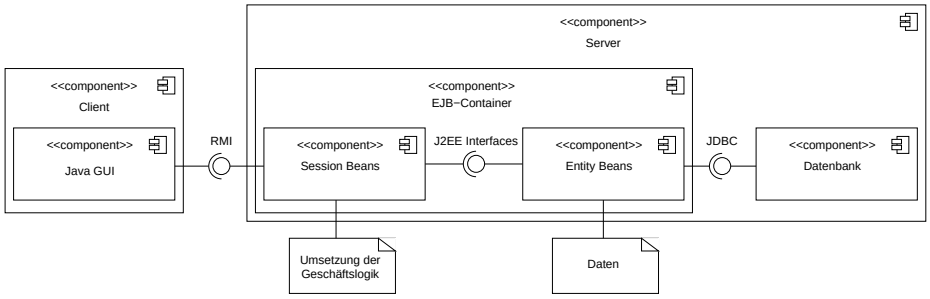

Abbildung 9.2: Die 4-Schichten-J2EE-Zielarchitektur

tet Session und Entity Beans. Entity Beans repräsentieren (passive) Datenobjekte,

die persistent in Datenbanken gespeichert werden. Session Beans repräsentieren (kleine) Geschäftsprozesse, die auf die Entity Beans zugreifen, aber selbst keine persistenten Daten enthalten. Diese zwei Arten von EJBs begründen zwei logische Schichten in unserer Zielarchitektur: Geschäftsprozesse und Geschäftsobjekte. Weitere Klassifikationen von EJBs sind für unsere Diskussion nicht relevant.

Es entsteht nun die Frage, *wie* von der 2-Schichten-Legacy-Architektur in Abbildung 9.1 zur 4-Schichten-Architektur in Abbildung 9.2 migriert werden kann. Abbildung 9.3 veranschaulicht einen unserer ersten Ansätze für eine Migrationsarchitektur am Beispiel einer Kfz-Zulassungs-Anwendung, die – wie zuvor beschrieben – eine Informix 4GL/4Js-Anwendung ist. Während der Migrationsphase können sowohl die alten 4GL/4Js-Informix-Clients als auch die neu entwickelten Java-Clients koexistieren, da sie gleichzeitig auf einem Endgerät lauffähig sind. Die Kommunikation zwischen Java-Clients und Applikationsserver ist

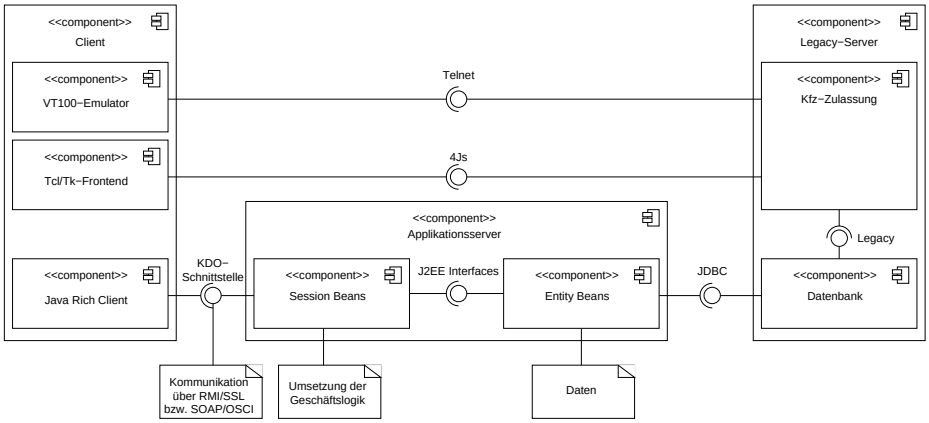

Abbildung 9.3: Erster Ansatz für eine Migrationsarchitektur

durch spezielle KDO-Interfaces gekapselt, um unterschiedliche Formen der technischen Kommunikation zu unterstützen: RMI mit SSL für schnelle und sichere Kommunikation und Web-Services über SOAP mit OSCI (Online Services Computer Interface) [OSC] für eine zertifiziert sichere Kommunikation über unsichere Kanäle wie das Internet. Eine ausführliche Diskussion dieser Sicherheitsbetrachtungen liegt außerhalb dieses Kapitels und ist nicht relevant für unsere Architekturdiskussion. Es ist jedoch wichtig zu erwähnen, dass Sicherheitsbetrachtungen auf der Architekturebene häufig von Bedeutung sind und dass sie in unserer speziellen Architektur gelöst wurden, soweit dies in der Anwendungsdomäne kommunaler Informationssysteme erforderlich ist [KBS].

Die Kommunikation zwischen Applikationsserver und dem Server des Legacy-Systems ist der kritische Punkt in dieser Architektur. Der erste Ansatz sah einen Zugriff des Applikationsservers auf die Datenbank des Legacy-Systems per JDBC

vor. Die gewachsenen Datenbankstrukturen in Legacy-Systemen offenbaren aber nicht alle die für eine korrekte Benutzung erforderliche Semantik der gespeicherten Datenbankobjekte. Außerdem passen diese Strukturen in der Regel nicht zu den neu definierten Geschäftsobjekten der Mittelschicht.

So bestand unser zweiter Ansatz darin, die neuen Datenobjekte, die in neuer Technologie implementiert sind, in einer zusätzlichen Datenbank zu speichern. Dieser Ansatz hat den offensichtlichen und hochkritischen Nachteil, Konsistenzmechanismen zu benötigen, die die Daten zwischen alter und neuer Datenbank replizieren [GHOS96, Has97].

Die genannten Betrachtungen führten uns zu der schließlich ausgewählten Migrationsarchitektur, die in Abbildung 9.4 gezeigt ist. Für Legacy-Informationssysteme ist es offensichtlich, dass auf ihre internen Datenbanken niemals direkt zugegriffen werden sollte. Es ist empfehlenswert, auf sie über ein API (Application Programming Interface) zuzugreifen, das die Geschäftslogik »kennt«.

Wir haben uns hier für die Verwendung von SOAP entschlossen, als ein Kommunikationsprotokoll, das spezielle Adapter sowohl auf der EJB- als auch auf der 4GL-Seite erfordert. Die Web-Service-Technologie basiert auf SOAP als Kommunikationsprotokoll, der Web Service Definition Language (WSDL) zur Schnittstellenbeschreibung und dem UDDI-Protokoll (Universal Description, Discovery and Integration), um dynamisch Web-Services im Internet zu finden und zu benutzen [Lin03]. Andere Kommunikationsmechanismen wie JCA [SSN02] wären in diesem Zusammenhang ebenfalls möglich. Dieser bevorzugte Ansatz, auf Legacy-

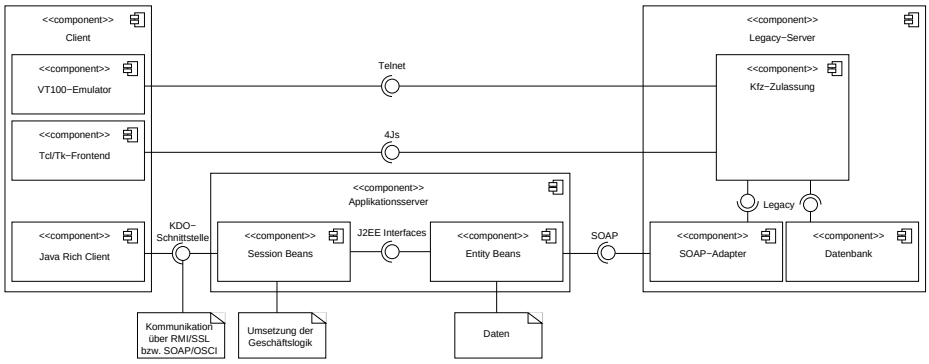

Abbildung 9.4: Endgültiger Ansatz für eine Migrationsarchitektur

Informationssysteme über ein API zuzugreifen, ist aus vielen anderen Projekten, bei denen Legacy-Systeme zu integrieren waren, gut bekannt [NHW+02, WF98].

Die Vorteile einer Multi-Schichten-Architektur sind ausführlich in der Literatur vorgestellt [Bro00, FRF+02]. Multi-Schichtenarchitekturen entstanden aus den Client-Server-Architekturen.

9.2.3 Das Dublo-Muster: DUal Business LOgic

Die Vorstellung des Dublo-Musters geschieht entsprechend [BMR+96]: Zuerst erfolgt eine Definition des Problems und des Kontextes, gefolgt von einer Beschreibung der Lösungsstruktur und vervollständigt durch eine Diskussion der Einschränkungen.

Problem und Kontext

Legacy-Systeme unterscheiden sich häufig von aktuellen Software-Architekturen dadurch, dass sie unterschiedliche Schichten nicht unterscheiden. Daher ist die Präsentationslogik oft vermischt mit der Geschäftslogik und dem Datenzugriffscode. Mit dem Aufkommen von Multi-Schichten-Architekturen wurde die Trennung dieser unterschiedlichen Aspekte allgemein üblich. Einer der Hauptgründe, warum Legacy-Systeme diese verschiedenen Schichten nicht unterscheiden, liegt jedoch darin, dass das System oder die Sprache, die zur Implementierung der Geschäftslogik verwendet wird, gleichermaßen mächtige Datenzugriffsfunktionen und die Möglichkeit der Implementierung von Nutzungsschnittstellen (entweder textbasierte Interfaces oder durch Werkzeuge generierte grafische Interfaces) zur Verfügung stellt. Bemerkenswerterweise weichen weder COBOL noch andere Sprachen der vierten Generation hiervon ab (abgesehen von Unterschieden im Entwurf von Nutzungsschnittstellen). Wir haben solch eine Beispielarchitektur in Abschnitt 9.2.2 vorgestellt.

Diese Entwicklungssysteme erlauben eine Integration der Präsentations-, Geschäfts- und Datenlogik in einer Schicht. Wie in Abschnitt 9.2.2 besprochen, besteht nun die Notwendigkeit, diese Systeme hin zu Multi-Schichten-Architekturen zu migrieren. Dieser Migrationsprozess hängt sehr stark davon ab, ob und auf welche Weise das Legacy-System in das neue System integriert wird. Es existieren unterschiedliche Lösungen für Migration und Integration:

- Austausch des alten monolithischen Systems durch eine Multi-Schichten-Architektur in Form eines »Big-Bang«: Zweifelsfrei ist diese »Strategie« nur anwendbar auf sehr kleine Systeme, die auf wohldokumentierte Art und Weise in einer gut bekannten Domäne arbeiten. Sie hat dann den Vorteil, dass sie einfach zu handhaben ist und keinen redundanten Code erfordert.
- Gleichzeitiger Austausch der Clients und der Geschäftslogik, direkter Zugriff der neu eingeführten Mittelschicht auf die Datenbank: diese Strategie erhält die alte Datenbank und ersetzt die alte Kombination aus Präsentations-, Geschäfts- und Datenzugriffsebene durch getrennte Präsentations- und Geschäftslogikebenen mit dem Vorteil, dass diese Strategie sofort eine Drei-Schichten-Architektur mit den gut getrennten Aspekten Präsentation, Geschäftslogik und Datenzugriff liefert. Nachteilig ist, dass dieser Ansatz eine komplette Ersetzung der Geschäfts- umd Präsentationslogik erforderlich macht. Da die Geschäftslogik in der Mehrzahl der Fälle einen wesentlichen

Kostenfaktor darstellt und den Kern des Unternehmensinformationssystems bildet, ist eine vollständige Ersetzung in einem Schritt nur sehr schwer zu erreichen.

- Geschäftslogik im Legacy-Code belassen, neue Geschäftslogik in der neuen Mittelschicht realisieren, Datenbankzugriff über Adapter zum Legacy-Code: Diese Strategie erlaubt bis zu einem gewissen Maße die Wiederverwendung des vorhandenen Legacy-Code. Dieser Ansatz trennt die Aspekte in Multi-Schichten-Architekturen weniger als der vorherige Ansatz, da Geschäftslogik an zwei Stellen vorgehalten wird: im alten Legacy-System und in der neuen Geschäftslogikschicht. Da die Entwicklung der neuen Präsentations- und Geschäftslogik von der Funktion des alten Systems getrennt wird, ist eine sanfte Migration möglich. Dies stellt das Dublo-Muster dar, das im folgenden Abschnitt beschrieben wird.

An dieser Stelle sei darauf hingewiesen, dass solche Verdopplungen von Fachlogik in der Praxis auch in anderen Kontexten vorkommen. Eine typische Situation sind Web-Shop-Lösungen als Frontend, die im Hintergrund auf Mainframe-Backend-Systeme zugreifen.

Lösung

Struktur Die Dublo-Lösungstruktur ist in Abbildung 9.5 dargestellt. Die Grundidee besteht in der Entwicklung der Geschäftslogik in der neuen Geschäftslogikschicht, der Erstellung eines Legacy-Adapters für den Zugriff der neuen Geschäftslogik auf die existierende Legacy-Geschäftslogik und der Benutzung dieses Adapters für den Datenzugriff. Folglich wird auf die Datenbank nur durch den vorhandenen Legacy-Code zugegriffen. Der vorhandene Code dient als funktionale Zugriffsebene für die Datenbank. Auf Funktionalität, die in der neuen Geschäftslogikschicht entwickelt wird, erfolgt der Zugriff durch eine ebenfalls neue Präsentationsschicht.

Migration Bei dem Dublo-Muster können alte Geschäftslogik und vorhandene Nutzungsschnittstellen so lange wiederverwendet werden, wie sie Funktionalität bereitstellen, die in dem neuen Anwendungskontext sinnvoll ist. Die alte Logik kann Schritt für Schritt durch eine neue Geschäftslogikschicht ersetzt werden. In vielen Fällen ist die Ersetzung der alten Nutzungsschnittstellen durch die neue integrierte Client-Technologie mindestens genauso wichtig wie die Realisierung der neuen Geschäftslogik. Das Dublo-Muster unterstützt die schnelle Aktualisierung der Nutzungsschnittstellen durch Kapselung der Legacy-Geschäftslogik mit Hilfe eines Adapters. Infolgedessen kann die neue Geschäftslogikschicht die Anfragen der neuen Präsentationsschicht einfach zum Legacy-System durchleiten, ohne selbst neue Geschäftslogik zu implementieren. Diese Durchleitung der neuen Geschäftslogik zum alten Legacy-Code entkoppelt die Entwicklung der neuen

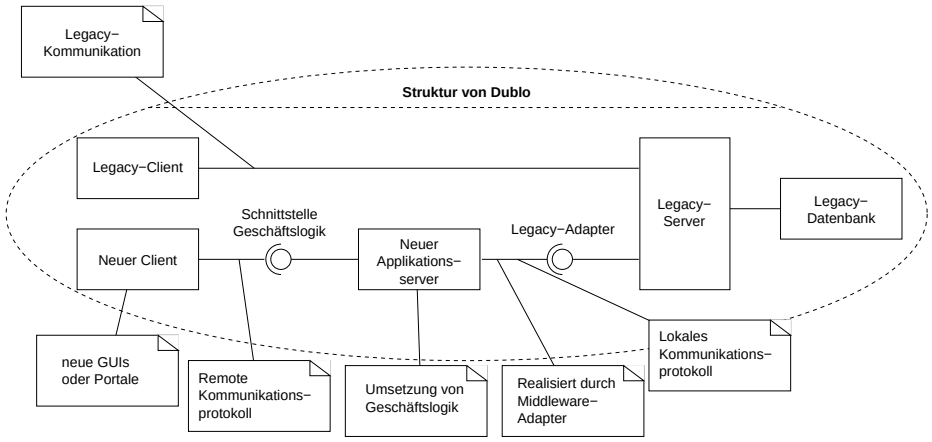

Abbildung 9.5: Strukturelle Sicht auf das Dublo-Muster

Präsentationsschicht von der Portierung des Legacy-Codes in die neue Geschäftslogikschicht.

Vorteile Die Anwendung des Dublo-Musters ist vernünftig, wenn die folgenden Vorteile, die das Muster bietet, von Bedeutung sind:

- *Sanfte Migration:* Inkrementeller Austausch alter Geschäftslogik- und Client-Software durch neue Geschäftslogik in der Mittelschicht. Insbesondere, wenn die Migration nicht in kurzer Zeit durchgeführt werden kann, wird dieser Aspekt essenziell.
- *Konsistenz der Datenbank:* Da keine zusätzliche Datenbank eingeführt wird, entstehen keine Konsistenz- oder Abgleichsprobleme zwischen neuer und alter Datenbank.
- *Datenbankunabhängigkeit:* Ein neues DBMS kann eingeführt werden, ohne die Mittel-Schicht zu verändern. Dies ist jedoch bei Legacy-Systemen häufig nicht möglich.
- *Wiederverwendung* der existierenden Geschäftslogik, indem auf sie durch einen Adapter zugegriffen wird.

Die Bedeutung des letztgenannten Vorteils verdient einige Beachtung: Die Durchleitung des alten Legacy-Systems durch die neue Geschäftslogikschicht hat zur Folge, dass die Unterscheidung, ob Geschäftslogik bereits in der neuen Mittelschicht oder noch im alten Legacy-Code implemetiert ist, für die modernisierten Clients transparent ist. Somit kann der Übergang von alten Nutzungsschnittstellen (implementiert im alten Legacy-Code und verflochten mit Datenzugriffs- und

Geschäftslogikcode) zu einer neuen Präsentationsschicht in drei Schritten erfolgen:

1. Erhaltung wenig benutzter, alter Nutzungsschnittstellen. Betrachtet man typische Benutzungsprofile, so wird nur ein Bruchteil aller Eingabemasken häufig benutzt. Weil viele Geschäftsprozesse zwar regelmäßig, aber selten verwendet werden (z.B. Inventuren, jährliche oder vierteljährliche kaufmännische Berichte), werden die entsprechenden Eingabemasken ebenfalls selten benutzt. Im Dublo-Muster müssen diese (vorhandenen) Eingabemasken nicht geändert werden, solange die zugrunde liegende Legacy-Geschäftslogik gültig ist.
2. Austausch alter Eingabemasken durch neue Eingabemasken in der Präsentationsschicht, ohne die Geschäftslogik zu reimplementieren. Die neuen Eingabemasken können auf einen Geschäftslogik-Proxy in der Geschäftslogikschicht zugreifen, der die Aufrufe einfach an den vorhandenen Legacy-Code weiterreicht.
3. Austausch des Legacy-Codes durch neue Geschäftslogik.

Man beachte, dass die letzten beiden Schritte nicht gleichzeitig stattfinden müssen. Dies ist ein Ergebnis der oben erwähnten Entkopplung der Entwicklung von Präsentations- und Geschäftslogikcode, erreicht durch die Erhaltung des Legacy-Codes. Diese Entkopplung liefert viel Flexibilität im Migrationsprozess.

Die Existenz von Geschäftslogik an zwei Stellen (dem alten Legacy-System und der neuen Geschäftslogikschicht) verhindert die klare Trennung der Aspekte, die von Multi-Schichten-Architekturen versprochen werden. Alternativ und im Gegensatz zum Dublo-Muster könnte man auf die Legacy-Datenbank direkt zugreifen. Während dies noch den Vorteil hat, dass man keine Datenbank duplizieren muss (und somit Konsistenzprobleme vermeidet), wird im Folgenden erläutert, warum wir für eine funktionale Zugriffsschicht und die Anwesenheit des Legacy-Codes argumentieren:

- Austauschmöglichkeit des alten DBMS (bei gleichzeitigem Erhalt der Legacy-Schicht). Auch wenn auf die Datenbank direkt zugegriffen wird, ist es sinnvoll, einen Adapter zu verwenden, da das Hinzufügen neuen Codes zu einer Legacy-Datenbank in der Regel problematisch sind.
- Wiederverwendung existierender Geschäftslogik, die im Legacy-System implementiert ist. Wie bereits begründet, ist der vollständige abrupte Übergang zu einem neuen System in jedem großen System unmöglich, womit die Erhaltung von Teilen des alten Codes und die Verfolgung eines durch ein Muster empfohlenen Migrationspfads vorteilhaft ist.
- Wenn das Legacy-System ein DBMS verwendet, das den direkten Zugriff erlaubt, bleibt die Möglichkeit des Direktzugriffs auf die Datenbank ohne Verwendung von Legacy-Code noch als Option. Dies ist der Fall, weil

das Dublo-Muster einen Adapter zwischen der neuen Geschäftslogikschicht und der Legacy-Schicht verwendet.

Anmerkungen und Einschränkungen Die Anwendung des Dublo-Musters wird stark vereinfacht durch die Existenz einer funktionalen Zugriffsebene auf Datenbanken im Legacy-Code. Wenn es diese funktionale Zugriffsschicht nicht gibt, wird der Aufwand, diese hinzuzufügen, durch den Vorteil der Wiederverwendung von Legacy-Code gerechtfertigt.

Im Allgemeinen ist der Grad der Wiederverwendung anwendungsspezifisch und hängt von vielen unterschiedlichen Aspekten ab. Es ist jedoch ein Vorteil dieses Musters, dass es für unterschiedliche Grade der Wiederverwendung von Legacy-Code anwendbar ist.

Die Anwendung des Dublo-Musters wird – verglichen mit einem abrupten Übergang der gesamten Geschäftslogik – typischerweise zu etwas redundantem Code und Zusatzaufwand führen:

- Der Adapter zwischen Legacy- und neuer Geschäftslogikschicht.
- Die funktionale Zugriffsschicht für den Datenbankzugriff innerhalb des Legacy-Systems (falls dort nicht vorhanden).
- Durch das Hinzufügen neuer Schichten wird typischerweise die Performanz eines Systems reduziert. Die bisherige Erfahrung zeigt, dass der Overhead in unserem Kontext vertretbar ist. Die erhaltene Flexibilität ist wichtiger.
- Mögliche Verdopplung von Legacy-Code. Da neue Systemanforderungen (in der Regel nicht funktionale Anforderungen) erheblichen Einfluss auf den Entwurf der Geschäftslogik haben können, kann es passieren, dass alter Legacy-Code nicht wiederverwendet werden kann und in der neuen Geschäftslogikschicht sofort reimplementiert werden muss.

Natürlich ist der letzte Aspekt wohlbekannt aus vielen Legacy-Integrations-Projekten. Infolgedessen reduziert dies die Anwendbarkeit des Dublo-Musters.

9.3 Dienstorientierte Zielarchitektur

In dienstorientierten Architekturen (beziehungsweise serviceorientierten Architekturen [RHS05]) werden Dienste über einen *Enterprise Service Bus* lose gekoppelt. Hinter Dienstschnittstellen stehen dann Dienstimplementierungen (Komponenten), die über den Bus aufgerufen werden können. Auf technologischer Ebene werden die Schnittstellen beispielsweise mit CORBA in IDL spezifiziert und mit Web-Services in WSDL. Dienstorientierte Architekturen sind zurzeit sehr populär im Kontext betrieblicher Informationssysteme, da bei entsprechender Unterstützung aus dem Management eine ganzheitliche Sicht auf die Integration von

Software-Systemen erreicht werden kann, in der insbesondere auch die Geschäftsprozesse durch »Orchestrierung« der Dienstaufrufe unterstützt werden können. Für eine managementorientierte Diskussion dienstorientierter Architekturen sei auf Kapitel 12 verwiesen.

Es gibt mittlerweile viele verschiedene Definitionen für Dienst (Service) im Sinne einer dienstorientierten Architektur. Im Kern sollte man sich darauf konzentrieren, dass ein Dienst einen über eine wohldefinierte Schnittstellenbeschreibung dargestellten, wiederverwendbaren fachlichen Anwendungsbaustein beschreibt, der jeweils eine klar umrissene fachliche Aufgabe wahrnimmt. Dienste können von anderen Komponenten über einen Verzeichnisdienst gefunden und aufgerufen werden. Die Dienste sind lose miteinander gekoppelt.

Speziell für die Integration von Legacy-Systemen bestehen durch dienstorientierte Architekturen die folgenden Erwartungen:

- Bei Entwicklung und Wartung der Informationssysteme können durch Wiederverwendung bestehender fachlicher Komponenten sowie durch die Integration über die einheitliche Infrastruktur (Enterprise Service Bus) Kosten eingespart werden.
- Bewährte Legacy-Systeme können weiter betrieben werden, ohne dass die modernen Systeme von technologischen Altlasten abhängig gemacht werden müssen. Dies ermöglicht einen Investitionsschutz und eine evolutionäre Weiterentwicklung der Anwendungslandschaft, da einzelne Komponenten durch die lose Kopplung leicht abgelöst werden können.
- Legacy-Systeme können fachlich so partitioniert und in erneuerbare Bestandteile zerlegt werden, dass eine inkrementelle Ablösung auch von monolithischer Software möglich wird.

Web-Services sind eine aktuelle Technik zur Realisierung dienstorientierter Architekturen. Bei der Integration von Legacy-Systemen besteht eine wesentliche Herausforderung in der Identifikation der Teile der alten Geschäftslogik, die die Dienste in der neuen Geschäftslogik werden können. Die alternativen Ansätze zur Identifikation von (Web-)Diensten können wie folgt kategorisiert werden [WBF97]:

1. *Datenorientiert*: Identifikation von Diensten aufgrund der Legacy-Datenstrukturen.
2. *Funktionsorientiert*: Identifikation von Diensten aufgrund der im Legacy-System vorhandenen (Geschäfts-)Funktionen.
3. *Objektorientiert*: Spezifikation eines neuen objektorientierten Modells für Daten und Funktionen des Legacy-Systems.

In unserem Kontext sind daten- und objektorientierte Ansätze [DK99] nicht wirklich geeignet: Die Legacy-Datenstrukturen haben im Laufe der Zeit eine unübersichtliche Struktur bekommen. Wir wählten stattdessen den funktionsorientier-

ten Ansatz [SES02], wobei Use Cases im Legacy-System aufgrund der User-Interaktion mit dem System identifiziert werden. Für Einzelheiten verweisen wir auf [TJK+04].

9.4 Nichttechnische Aspekte sanfter Migration

Das Software Engineering als Ingenieurdisziplin beschäftigt sich nicht nur mit technischen, sondern auch mit betriebswirtschaftlichen und psychologischen Fragestellungen. Ein hoher Anteil großer Software-Projekte überzieht das Budget und die Termine in erheblichem Maße, wobei nichttechnische Gründe einen wesentlichen Anteil am Scheitern dieser Projekte haben [Gla02a, MBP+04]. Im Folgenden werden wir aus der Erfahrung heraus nichttechnische Aspekte aus diesen Bereichen ansprechen, deren Beachtung – je nach gesamtem Projektkontext – erheblichen Einfluss auf den Projekterfolg haben kann.

9.4.1 Betriebswirtschaftliche Aspekte sanfter Migration

Im Einsatz befindliche große Software-Systeme haben häufig – nach Fertigstellung der Erstversion – lange und kostenintensive Phasen der qualitativen Stabilisierung und Funktionsanreicherung (IQTF = Increasing Quality Times Functionality) hinter sich. Die Gesamtzeit seit Beginn der Entwicklung kann dadurch bis zu 10 Jahre in Anspruch nehmen. Die hier genannten Zeiträume stammen aus einem Entwicklungskontext mit den in Tabelle 9.1 angegebenen Kennzahlen. Die angeführten Zeiträume sind stark abhängig von verschiedensten Faktoren wie z.B. Umfang der Software, Komplexitätsgrad der Software, Dynamik der Anforderungen, Anzahl der eingesetzten Entwickler und deshalb nur als Beispiele zu betrachten.

Das Produkt aus Qualität und Funktionalität QTF hat dabei über die Entwicklungsdauer etwa den in Abbildung 9.6 gezeigten Verlauf.

Anzahl	Maß
890.000	Lines of Code (4GL)
45	Personenjahre Entwicklungs- und Pflegeaufwand
9	Jahre Entwicklungsdauer
700	Dialogmasken
bis zu 6	beteiligte Mitabeiter
19.000	SQL-Statements

Tabelle 9.1: Kennzahlen

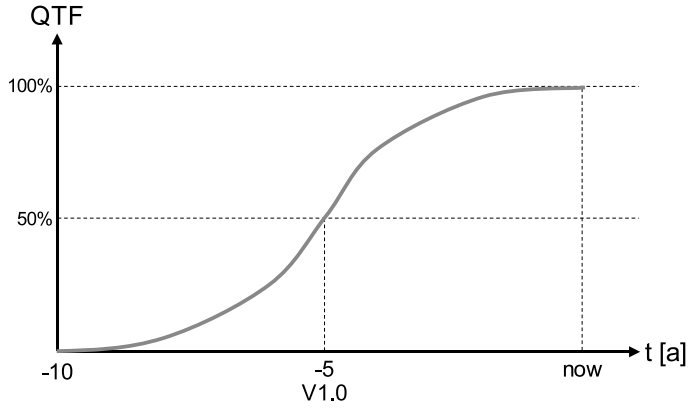

Abbildung 9.6: Produktlebenszyklus

Da erfahrungsgemäß in den wenigsten Projekten während der Entwicklungszeit auf eine völlig neue Technologie umgestellt wird, basieren solche bewährten Systeme nach beispielsweise 10 Jahren Entwicklungsdauer auf veralteten Technologien. Der Software-Hersteller muss auf diese Tatsache reagieren und sich zwischen Neuentwicklung und Migration entscheiden.

Big-Bang-Neuentwicklung

Eine vollständige Neuentwicklung nur zur Verwendung zeitgemäßer Technologien führt erst nach einer neuen IQTF-Phase zu einem Folgeprodukt mit ähnlicher Qualität und ähnlichem Funktionsumfang: Die Vorarbeiten für die Neuentwicklung einer Folgeversion eines großen Software-Systems (Technologieauswahl, Architekturauswahl) erfordern i.d.R. etwa ein Jahr, die Entwicklung der neuen Erstversion mindestens 2-3 Jahre, gefolgt von der o.g. IQTF-Phase, die erfahrungsgemäß ebenfalls mindestens 3-4 Jahre erfordert.

Normalerweise wird man außerdem für die Neuentwicklung die fachlich erfahrenen Software-Entwickler des Erstprodukts einsetzen – diese müssen zunächst in den neuen Technologien geschult werden und dann Erfahrungen sammeln, was ebenfalls etwa ein Jahr dauert. Dadurch wird weniger an der Altversion gearbeitet, die QTF nimmt ab. Der Software-Hersteller kann also frühestens nach etwa 4 Jahren wieder mit Neuigkeiten am Markt erscheinen. Diese Zeitdauer bis zum nächsten Release kann kritisch für das Überleben eines Unternehmens werden und muss – durch geeignete langfristige Planung – unbedingt verkürzt werden, z.B. durch frühzeitigen Beginn der Arbeiten am Folgeprodukt. Es werden jetzt die Software-Entwickler des Altprodukts noch früher abgezogen, so dass die QTF der Altversion spürbar abnimmt (siehe Abbildung 9.7).

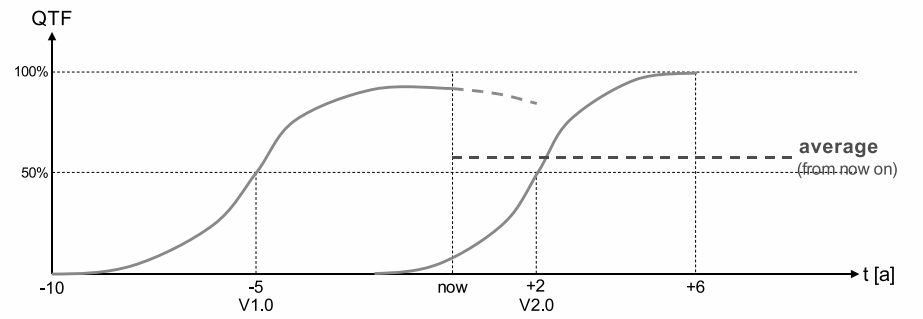

Abbildung 9.7: Produktlebenszyklen optimiert

Trotz Optimierung ergibt sich noch ein Zeitraum von etwa 2-3 Jahren, während dessen die Pflege des Altprodukts spürbar reduziert werden muss und noch kein Folgeprodukt zur Verfügung steht. Dies kann zu einem empfindlichen Verlust an Marktanteilen führen, auch bei den Bestandskunden, was kein Software-Hersteller in Kauf nehmen kann. Anwender müssen sich mit der Einführung des Folgeprodukts vollständig umstellen und zusätzlich auf Teile des gewohnten Funktionsumfangs verzichten.

Sanfte Migration

Eine sanfte Migration setzt auf der hohen QTF des Erstprodukts auf. Es werden in kleinen Schritten von 1–10% der jeweiligen Gesamtfunktionalität immer wieder Teile des Erstprodukts durch neu entwickelte Teile ersetzt. Man kann dies als ein fortlaufendes Refactoring [Fow00] im Rahmen eines großen Plans, der Architekturmigration, betrachten. Dieses Vorgehen hat – wie beim Dublo-Muster bereits erläutert – den Nachteil, dass bestimmte Zusatzaufwände entstehen und die Entwicklung mit mehreren Technologien gleichzeitig erfolgen muss. Dies wird aber mehr als aufgewogen dadurch, dass man permanent ein stabiles und funktionsreiches Produkt am Markt halten kann: Die QTF des vorhandenen Produkts wird dabei potenziell bei jedem Schritt nur gering reduziert und kann anschließend direkt wieder aufgebaut werden (siehe Abbildung 9.8).

Der Anwender muss bei dieser Vorgehensweise evtl. mehrere Benutzungsschnittstellen gleichzeitig bedienen, die sich ihm zwar auf dem gleichen Endgerät präsentieren, aber doch unterschiedlich bedient werden. Dieser Nachteil wird i.d.R. durch die Vorteile und Verbesserungen des ersetzten Produktteils wieder hinreichend aufgehoben. Außerdem haben die Anwender bei der sanften Migration Zeit, sich langsam an die Neuerungen zu gewöhnen. Hinsichtlich des Vertriebs gilt: Bestandskunden können gehalten werden, da diese erkennen, dass das Produkt in die Zukunft migriert wird, und Neukunden können gut geworben

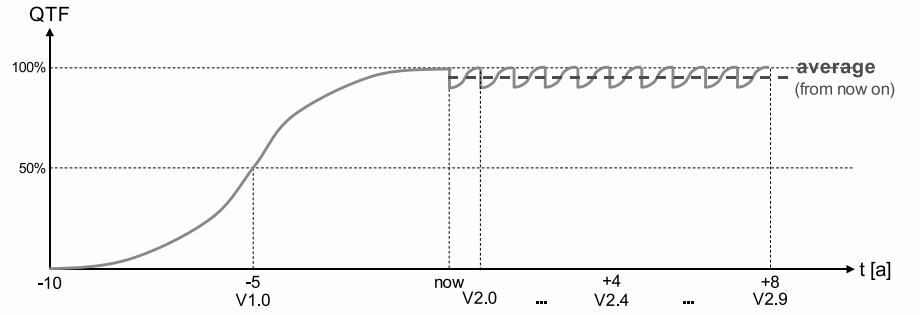

Abbildung 9.8: Produktlebenszyklus bei sanfter Migration

werden, da die Zukunftsfähigkeit des Produkts darstellbar und das vorhandene Produkt vertrieblich einsetzbar ist.

Aus der internen Sicht des Software-Herstellers können anfangs zunächst die Entwickler eingesetzt werden, die den neuen Technologien offen gegenüberstehen. Entsprechend dem Migrationsfortschritt können dann immer mehr Mitarbeiter auf die neuen Technologien wechseln – die Akzeptanz bei den Anwendern und die Vertriebserfolge werden irgendwann alle Mitarbeiter mit ins Boot bringen. Zusätzlich kann bei der sanften Migration der Personaleinsatz im Bereich Entwicklung konstant gehalten werden. Weitere Hinweise zur Aufwandsabschätzung von Migrationsprojekten finden sich beispielsweise in [SPL03, Kapitel 17] und zu Einsparpotenzialen in [Sch04].

9.4.2 Organisatorische Aspekte sanfter Migration

Der Software-Entwicklungsbereich unterliegt deutlichen Wandlungen – neue Technologien und Paradigmen tauchen schnell am Markt auf. Der Software-Hersteller muss deshalb insbesondere im Bereich der Entwicklung versuchen, ein innovationsfreundliches Klima zu schaffen. Besuch geeigneter Schulungen, Studium von Literatur und Fachzeitschriften sind als Zukunftsmotoren und nicht als Kostenfaktor anzusehen.

Es empfiehlt sich, viele Software-Entwickler am Technologiefindungsprozess zu beteiligen oder zumindest ein offenes Arbeitsklima herzustellen, in dem jederzeit Entscheidungen hinterfragt oder Kritik geübt werden kann.

Außerdem sollte frühzeitig bedacht werden, ob für die wichtigen Anfangsentscheidungen (Technologie- und Werkzeugauswahl, Definition des Migrationsweges) wegen zu geringer eigener Erfahrung nicht externe Hilfe in Anspruch zu nehmen ist.

Es sollte ein Mehrphasenmodell für die Schulungsblöcke der Software-Entwickler vorgedacht und kommuniziert werden (z.B. im 1. Jahr die ersten 10%, im 2. Jahr weitere 20% usw.), in dem sich jeder wiederfindet.

Die mit den neuen Technologien arbeitenden Software-Entwickler sollten dies in einem Umfang von mindestens 50%, besser aber 75% und mehr tun und sind entsprechend einzuplanen.

Die Software-Entwickler, die bereits an der sanften Migration arbeiten, sollten möglichst in gemeinsamen Räumen untergebracht werden, da dadurch der Sog der Alt-Technologien reduziert werden kann.

9.4.3 Psychologische Aspekte sanfter Migration

Software-Entwickler sind hochqualifizierte Mitarbeiter. Den höchsten Produktivitätszuwachs erreicht man durch Überzeugung in Bezug auf Ziel und Weg. Dies gilt im Allgemeinen für die Software-Entwicklung und auch für die sanfte Migration, denn zumindest ein Teil der Software-Entwickler sieht es häufig lieber, noch einmal neu anfangen zu können (Big Bang), da das entstehende Produkt dann »sauberer« gestaltet werden könnte.

Software-Entwickler sind möglichst entsprechend ihren persönlichen Zielen und Neigungen einzusetzen, die man bei geeigneten Mitarbeitergesprächen erfahren kann.

Erfahrene Software-Entwickler besitzen bei Verwendung der aktuellen (alten) Technologie eine hohe Produktivität. Sie sollen sich bei der sanften Migration in Technologien einarbeiten und mit Technologien beschäftigen, bei denen sie anfangs das Gefühl haben, nur eine sehr viel geringere Produktivität zu besitzen, da man eben nicht mehr direkt mit der Kodierung beginnt. Auch hier ist immer wieder Überzeugungsarbeit erforderlich.

Neue Technologien benötigen i.d.R. auch einen neuen Entwicklungsprozess, der nach Problemerkennung erst die Anforderungsanalyse, dann die Anforderungsmodellierung und schließlich die Lösungsmodellierung erfordert, bevor mit der Kodierung begonnen werden kann. Es empfiehlt sich, die Vorteile dieser Arbeitsweisen immer wieder zu kommunizieren.

Literatur

[ABB+01] Atkinson, C.; Bayer, J.; Bunse, C.; Kamsties, E.; Laitenberger, O.; Laqua, R.; Muthig, D.; Paech, B.; Wüst, J.; Zettel, J.: *Component-based Product Line Engineering with UML*. Addison-Wesley Component Software Series, Addison-Wesley, 2001

[ABC+96] Abowd, G.; Bass, L.; Clements, P.; Kazman, R.; Northrop, L.; Zaremski, A.: *Recommended Best Industrial Practice for Software Architecture Evaluation*. Technischer Bericht CMU/SEI-96-TR-025, Software Engineering Institute, Carnegie Mellon University, 1996

[ABK+02] Astesiano, E.; Bidoit, M.; Krieg-Brückner, B.; Mosses, P. D.; Sannella, D.; Tarlecki, A.: Casl: The Common Algebraic Specification Language. In: *Theoretical Computer Science* 286 (2002), Nr. 2, S. 153–196

[Ack96] Ackermann, P.: *Developing Object-Oriented Multimedia Software – Based on MET++ Application Framework*. dpunkt.verlag, 1996

[ACM01] *Computing Curricula 2001 Computer Science*. Technischer Bericht, The Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery, Dezember 2001

[ACM03] Alur, D.; Crupi, J.; Malks, D.: *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall, 2. Aufl., 2003

[AFG+00] Andrade, L. F.; Fiadeiro, J. L.; Gouveia, J.; Lopes, A.; Wermelinger, M.: Patterns for Coordination. In: *Coordination Languages and Models*, Springer-Verlag, 2000, Bd. 1906 von *Lecture Notes in Computer Science*, S. 317–322

[AG96] Allen, R.; Garlan, D.: *The Wright Architectural specification language*. Technischer Bericht CMU-CS-96-TBD, School of Computer Science, Carnegie Mellon University, 1996

[AG97] Allen, R.; Garlan, D.: A Formal Basis for Architectural Connection. In: *ACM Transactions on Software Engineering and Methodology* (1997)

[AGD97] Allen, R.; Garlan, D.; Douence, R.: Specifying Dynamism in Software Architectures. In: Leavens, G. T.; Sitaraman, M. (Hrsg.), *Proceedings of the Workshop on Foundations of Component-Based Software Engineering*, September 1997

[AH90] Agrawal, H.; Horgan, J. R.: Dynamic program slicing. In: *Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation*, ACM Press, 1990, S. 246–256

[AH01] de Alfaro, L.; Henzinger, T.: Interface Automata. In: *ESEC/FSE 01: Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2001, S. 109–120

[Ale79] Alexander, C.: *The Timeless Way of Building*. Oxford University Press, 1979

[All97] Allen, R.: *A Formal Approach to Software Architecture*. Dissertation, Carnegie Mellon School of Computer Science, Januar 1997

[AMR96] André, E.; Müller, J.; Rist, T.: WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring. In: *EUROMEDIA'96*, 1996, S. 95–102

[And04] Andresen, A.: *Komponentenbasierte Softwareentwicklung mit MDA, UML-2 und XML*, Bd. 2., neu bearbeitete Auflage. Hanser Verlag, 2004

[App05] Apple, USA: QuickTime. 2005, URL `http://www.apple.com/quicktime/`

[Arc05] ArcWay AG: ArcWay AG Home Page. Website, 2005, URL `http://www.arcway.com`

[AUT] AUTOSAR-Konsortium: Automotive Open System Architecture. URL `http://www.autosar.org`

[B] B-Homepage. URL `http://vl.fmnet.info/b/`

[Bal97] Balzert, H.: *Lehrbuch der Softwaretechnik, Bd. 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, 1997

[Bal00] Balzert, H.: *Lehrbuch der Software-Technik, Bd. 1: Software-Entwicklung*. Spektrum Akademischer Verlag, 2. Aufl., 2000

[BB99] Bentsson, P.-O.; Bosch, J.: Haemo Dialysis Software Architecture Design Experiences. In: *Proceedings of the 21st International Conference on Software Engineering*, May 1999, S. 516–526

[BBK+99] Balzert, H.; Behle, A.; Kelter, U.; Nagl, M.; Pauen, P.; Schäfer, W.; Six, H.; Voss, J.; Wadsack, J.; Weidauer, C.; Westfechtel, B.: Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme. September 1999

[BBS03] Balsamo, S.; Bernardo, M.; Simeoni, M.: Performance evaluation at the software architecture level. In: Bernado und Inveradi [BI03], S. 207–258

[BC89] Beck, K.; Cunningham, W.: A laboratory for teaching object oriented thinking. In: *OOPSLA '89: Conference proceedings on Object-oriented programming systems, languages and applications*, New York, NY, USA: ACM Press, 1989, S. 1–6

[BC00] Baldwin, C. Y.; Clark, K. B.: *The Power of Modularity*, Bd. 1 von *Design Rules*. MIT Press, 2000

[BCD00] Bernardo, M.; Ciancarini, P.; Donatiello, L.: AEMPA: A Process Algebraic Description Language for the Performance Analysis of Software Architectures. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2000, S. 1–11

[BCDW04] Bradbury, J.; Cordy, J.; Dingel, J.; Wermelinger, M.: A Survey of Self Management in Dynamic Software Architecture Specifications. In: *Proceedings of the ACM SIGSOFT International Workshop on Self-Managed Systems (WOSS'04)*, ACM Press, 2004

[BCH+02] Baron, A.; Clarke, B.; Hertroys, P.; von Oosterom, N.; Hinley, D.: *Application Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2002

[BCJ+02] Berry, C.; Carnell, J.; Juric, M.; Kunnumpurath, M.; Nashi, N.; Romanosky, S.: *J2EE Design Patterns Applied*. Wrox Press, 2002

[BCK03] Bass, L.; Clements, P.; Kazman, R.: *Software Architecture in Practice*. SEI Series in Software Engineering, Addison-Wesley, 2. Aufl., 2003

[BCR94] Basili, V. R.; Caldiera, G.; Rombach, H. D.: Goal Question Metric Paradigm. In: Marciniak, J. J. (Hrsg.), *Encyclopedia of Software Engineering*, John Wiley & Sons, Bd. 1, Kap. Goal Question Metric Paradigm, 1994, S. 528–532

[BD00] Bruegge, B.; Dutoit, A.: *Object-Oriented Software Engineering – Conquering Complex and Changing Systems*. Prentice Hall, 2000

[BDIS04] Balsamo, S.; DiMarco, A.; Inverardi, P.; Simeoni, M.: Model-Based Performance Prediction in Software Development: A Survey. In: *IEEE Transactions on Software Engineering* 30 (2004), Nr. 5, S. 295–310

[BDM02] Bernardi, S.; Donatelli, S.; Merseguer, J.: From UML sequence diagrams and state-charts to analysable Petri net models. In: *Proceedings of WOSP2002*, 2002

[BDW98] Briand, L. C.; Daly, J. W.; Wuest, J.: A Unified Framework for Cohesion Measurement in Object-Oriented Systems. In: *Empirical Software Engineering* 3 (1998), Nr. 1, S. 65–117

[BE93] Beck, J.; Eichmann, D.: Program and Interface Slicing for Reverse Engineering. In: *Proceedings: 15th International Conference on Software Engineering*, IEEE Computer Society Press / ACM Press, 1993, S. 509–518

[BEA] BEA: BEA Weblogic.

URL http://www.bea.com/products/weblogic/

[Bec97] Beck, K.: *Smalltalk – praxisnahe Gebrauchsmuster*. Markt und Technik, 1997

[Bec03] Beck, K.: *Extreme Programming – die revolutionäre Methode für Softwareentwicklung in kleinen Teams*. Programmer's choice, Addison-Wesley, 2003

[BEJV96] Binns, P.; Englehart, M.; Jackson, M.; Vestal, S.: Domain-Specific Software Architectures for Guidance, Navigation and Control. In: *International Journal of Software Engineering and Knowledge Engineering* 6 (1996), Nr. 2, S. 201–227

[Ber98] Bernstein, P. A.: Repositories and Object Oriented Databases. In: *SIGMOD Record* 27 (1998), Nr. 1, S. 88–96

[Béz05] Bézivin, J.: On the Unification Power of Models. In: *Software and Systems Modeling* 4 (2005), Nr. 2, S. 171–188

[BFF+97] Bordegoni, M.; Faconti, G.; Feiner, S.; Maybury, M. T.; Rist, T.; Ruggieri, S.; Trahanias, P.; Wilson, M.: A standard reference model for intelligent multimedia presentation systems. In: *Computer standards & interfaces* 18 (1997), Nr. 6-7, S. 477–496

[BFG+04] Becker, S.; Firus, V.; Giesecke, S.; Hasselbring, W.; Overhage, S.; Reussner, R.: Towards a Generic Framework for Evaluating Component-Based Software Architectures. In: Turowski, K. (Hrsg.), *Architekturen, Komponenten, Anwendungen – Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen (AKA 2004), Universität Augsburg*, Bonner Köllen Verlag, Dezember 2004, Bd. 57 von *GI-Edition Lecture Notes in Informatics*, S. 163–180

[BG98] Bernardo, M.; Gorrieri, R.: A Tutorial on EMPA: A Theory of Concurrent Processes with Nondetermism, Priorities, Probabilities and Time. In: *Theoretical Computer Science* 202 (1998), S. 1–54

[BG04] Bauer, A.; Günzel, H. (Hrsg.): *Data-Warehouse-Systeme – Architektur, Entwicklung, Anwendung*. dpunkt.verlag, 2. Aufl., 2004

[BGM03] Balsamo, S.; Grosso, M.; Marzolla, M.: *Towards Simulation-Based Performance Modeling of UML Specifications*. Technischer Bericht CS-2003-2, Dep. di Informatica, Universtita Ca' Foscari Venezia, Italy, 2003

[BGMT99] Bolch, G.; Greiner, S.; Meer, H.; Trivedi, K.: *Queueing Networks and Markov Chains*. John Wiley and Sons, 1999

[BH04a] Bischofs, L.; Hasselbring, W.: A Hierarchical Super Peer Network for Distributed Software Development. In: de Lucia, A.; Gall, H. C.; Dustdar, S. (Hrsg.), *Proceedings of the Workshop on Cooperative Support for Distributed Software Engineering*

Processes (CSSE 2004), Linz, Austria: Austrian Computer Society, September 2004, S. 99–106

[BH04b] Buhl, H.-U.; Heinrich, B.: Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Information6ssystemen. In: *Wirtschaftsinformatik* 46 (2004), Nr. 4, S. 311–322

[BHLP04] Bühne, S.; Halmans, G.; Lauenroth, K.; Pohl, K.: Variabilität in Software-Produktlinien. In: Böckle et al. [BKPS04], S. 13–24

[BHN+04] Bischofs, L.; Hasselbring, W.; Niemann, H.; Schuldt, H.; Wurz, M.: Verteilte Architekturen zur intra- und inter-institutionellen Integration von Patientendaten. In: Ammenwerth, E.; Gaus, W.; Haux, R.; Lovis, C.; Pfeiffer, K.; Tilg, B.; Wichmann, H. (Hrsg.), *Tagungsband der 49. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds2004)*, Innsbruck, österreich: videel, September 2004, S. 87–89

[BHSS04] Bischofs, L.; Hasselbring, W.; Schlegelmilch, J.; Steffens, U.: A Hierarchical Super Peer Network for Distributed Artifacts. In: Agosti, M.; Schek, H.-J.; Türker, C. (Hrsg.), *Pre-proceedings of the Sixth Thematic Workshop of the EU Network of Excellence DELOS*, Cagliari, Italy: Edizioni Libreria Progetto, Padova, 2004, S. 105–114

[BHTV03] Baresi, L.; Heckel, R.; Thöne, S.; Varró, D.: Modeling and validation of service-oriented architectures: Application vs. style. In: *Proc. of the European Software Engineering Conference/Foundations of Software Engineering*, ACM Press, 2003, S. 68–77

[BHTV04] Baresi, L.; Heckel, R.; Thöne, S.; Varró, D.: Style-Based Refinement of Dynamic Software Architectures. In: *Proc. of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA4)*, IEEE Computer Society Press, 2004, S. 155–164

[BI03] Bernado, M.; Inveradi, P. (Hrsg.): *Formal Methods for Software Architecture*, Bd. 2804 von *Lecture Notes in Computer Science*, Springer-Verlag, 2003

[Bie02] Bien, A.: *J2EE Patterns – Entwurfsmuster für die J2EE*. Programmer's choice, Addison-Wesley, 2002

[Bir99] Birolini, A.: *Reliability Engineering: Theory and Practice*. Springer-Verlag, 3. Aufl., 1999

[BK03a] Björkander, M.; Kobryn, C.: Architecting Systems with UML-2.0. 2003, URL `http://www.uml-forum.com/papers.htm`

[BK03b] Block, M.; Konrad, S.: *Personalized and multimedia Webservice – Sightseeing4U*. Individuelle Projekte, Carl von Ossietzky

Universität Oldenburg, Fakultät II, Department für Informatik, September 2003

[BKPS04] Böckle, G.; Knauber, P.; Pohl, K.; Schmid, K. (Hrsg.): *Software-Produktlinien – Methoden, Einführung und Praxis*. dpunkt.verlag, 2004

[BKS04] Boll, S.; Krösche, J.; Scherp, A.: Personalized Mobile Multimedia meets Location-Based Services. In: Dadam, P.; Reichert, M. (Hrsg.), *Multimedia-Informationssysteme Workshop im Rahmen der 34. Jahrestagung der Gesellschaft für Informatik (Informatik 2004), Bd. 2*, Ulm, Deutschland: GI, September 2004, Bd. 51 von *LNI*, S. 64–69

[BKW03] Boll, S.; Krösche, J.; Wegener, C.: Paper chase revisited – a real world game meets hypermedia. In: *Proc. der Intl. Conference on Hypertext (HT03)*, ACM, 2003, S. 126–127

[BLBV04] Bengtsson, P.; Lassing, N.; Bosch, J.; van Vliet, H.: Architecture-level modifiability analysis (ALMA). In: *Journal of Systems & Software* 69 (2004), Nr. 1-2, S. 129–147

[Blo01] Bloch, J.: *Effective Java*. Addison-Wesley, 2001

[BM98] Baniassad, E. L. A.; Murphy, G. C.: Conceptual module querying for software reengineering. In: *Proceedings of the 20th international conference on Software engineering*, IEEE Computer Society Press, 1998, S. 64–73

[BM04] Bertolino, A.; Mirandola, R.: CB-SPE Tool: Putting Component-Based Performance Engineering into Practice. In: Crnkovic, I.; Stafford, J. A.; Schmidt, H. W.; Wallnau, K. C. (Hrsg.), *CBSE 2004*, Springer-Verlag, 2004, Bd. 3054 von *Lecture Notes in Computer Science*, S. 233–248

[BMM+99] Bosch, J.; Molin, P.; Mattsson, M.; Bengtsson, P.; Fayad, M. E.: Framework Problems and Experiences. In: Fayad et al. [FSJ99a], S. 55–82

[BMR+96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M.: *A System of Patterns (Pattern-Oriented Software Architecture, vol. 1)*. Wiley Series in Software Design Patterns, John Wiley & Sons, 1. Aufl., Juli 1996

[BMW94] Biggerstaff, T. J.; Mitbander, B. G.; Webster, W.: Program Understanding and the Concept Assignment Problem. In: *Communications of the ACM* 37 (1994), Nr. 5, S. 72–82

[BN03] Bruns, K.; Neidhold, B.: *Audio-, Video- und Grafikprogrammierung*. München, Wien: Fachbuchverlag Leipzig im Carl Hanser Verlag, 2003

[BO02] Bolusset, T.; Oquendo, F.: Formal refinement of dynamic software architectures based on rewriting logic. In: *Proc. RCS02, Int. Workshop on Refinement of Critical Systems*, 2002

[Boe88] Boehm, B. W.: A Spiral Model of Software Development and Enhancement. In: *Computer* May 1988 (1988), S. 61–72

[Bol03] Boll, S.: Vienna 4 U – What Web Services can do for personalized multimedia applications. In: *Seventh Multi-Conference on Systemics (SCI 2003), Cybernetics and Informatics*, Juli 2003, S. 220–225

[Bor03] Borchers, B.: Verursacherbedingt verspätet – Das »fortschrittlichste Mautsystem der Welt« und die Realität. In: *C't – Magazin für Computer-Technik* (2003), Nr. 22

[BOR04] Bastarrica, M. C.; Ochoa, S. F.; Rossel, P. O.: Integrated Notation for Software Architecture Specification. In: *Proc. of the XXIV International Conference of the SCCC*, 2004, S. 26–35

[Bos00] Bosch, J.: *Design and Use of Software Architectures – Adopting and evolving a product-line approach*. Addison-Wesley, 2000

[Bos02] Bosch, J.: Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. In: Chastek, G. J. (Hrsg.), *Proceedings of Software Product Lines, Second International Conference (SPLC2)*, Springer-Verlag, August 2002, Bd. 2379 von *Lecture Notes in Computer Science*

[Bos04] Bosch, J.: Software Variability Management. Tutorial at 4th Working IEEE/IFIP Conference on Software Architecture (WICSA), Juni 2004

[BP93] Burns, D. J.; Pitblado, R. M.: A Modified Hazop Methodology for Safety Critical System Assessment. In: Redmill, F.; Anderson, T. (Hrsg.), *Directions in Safety-Critical Systems*, Springer-Verlag, 1993, S. 232

[BPM05] *BPML*. Technischer Bericht, Business Process Management Initiative, 2005, URL `http://www.bpmi.org/BPML.htm`

[BR90] Basili, V. R.; Rombachf, H. D.: *Towards a comprehensive framework for reuse: model-based reuse characterization schemes*. Technischer Bericht, College Park, MD, USA, 1990

[BR91] Basili, V. R.; Rombach, H. D.: Support for comprehensive reuse. In: *Software Engineering Journal* 6 (1991), Nr. 5, S. 303–316

[Bra98] Braband, J.: RAMS-Management nach CENELEC. In: *Signal + Draht* 98 (1998), Nr. 12, S. 20–24

[Bro00] Brown, A. W.: *Large-Scale Component-Based Development*. Prentice Hall, Englewood Cliffs, NJ, USA, 2000

[Bro02] Broemmer, D.: *J2EE Best Practices – Java Design Patterns, Automation, and Performance*. John Wiley & Sons, 2002

[BS95] Brodie, M.; Stonebraker, M.: *Migrating Legacy Systems – Gateways, Interfaces and The Incremental Approach*. San Francisco, CA, USA: Morgan Kaufmann, 1995

[BS97] Bellin, D.; Simone, S. S.: *The CRC Card Book*. Addison-Wesley, 1997

[BS01] Broy, M.; Stølen, K.: *Specification and Development of Interactive Systems. Focus on Streams, Interfaces and Refinement*. Springer-Verlag, 2001

[BS02] Broy, M.; Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung – Eine kritische Einschätzung. In: *Informatik-Spektrum* 25 (2002), Nr. 1, S. 3–11

[BSB+04] Beneken, G.; Seifert, T.; Baehr, N.; Hanschke, I.; Rauch, O.: Referenzarchitekturen und MDA. In: Dadam, P.; Reichert, M. (Hrsg.), *INFORMATIK 2004 – Informatik verbindet, Bd. 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, Gesellschaft für Informatik, 2004, Bd. 51 von *GI-Edition Lecture Notes in Informatics*, S. 101–105

[BSF02] Boger, M.; Sturm, T.; Fragemann, F.: Refactoring Browser for UML. In: Wells und Williams [WW02], S. 77–81

[BST02] Bidoit, M.; Sannella, D.; Tarlecki, A.: Architectural Specifications in Casl. In: *Formal Aspects of Computing* 13 (2002), S. 252–273

[Buh98] Buhr, R. J. A.: Use Case Maps as Architectural Entities for Complex Systems. In: *IEEE Trans. Softw. Eng.* 24 (1998), Nr. 12, S. 1131–1155

[BVH+03] Burnett, I.; Van de Walle, R.; Hill, K.; Bormans, J.; Pereira, F.: MPEG-21: Goals and Achievements. In: *IEEE MultiMedia* 10 (2003), Nr. 4, S. 60–70

[CBB+03] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; Stafford, J.: *Documenting Software Architectures – Views and Beyond*. SEI Series in Software Engineering, Addison-Wesley, 2003

[CC92] Chikofsky, E. J.; Cross, II, J. H.: Reverse Engineering and Design Recovery: A Taxonomy. In: *Software Reengineering*, IEEE Computer Society Press, 1992, S. 54–58

[CC02] Carey, J.; Carlson, B.: *Framework Process Patterns – Lessons Learned Developing Application Frameworks*. Addison-Wesley, 2002

[CCMT93] Canfora, G.; Cimitile, A.; Munro, M.; Taylor, T.: Extracting Abstract Data Types from C Programs: A Case Study. In: *Proceedings of the International Conference on Software Maintenance 1993*, IEEE Computer Society Press, September 1993, S. 200–209

[CE00] Czarnecki, K.; Eisenecker, U. W.: *Generative Programming – Methods, Tools and Applications*. Addison-Wesley, 2000

[CEN00] CENELEC (European Committee for Electro-technical Standardisation): CENELEC EN 50128: Railway Applications: Software for Railway Control and Protection Systems CENELEC, Brussels. 2000

[Che76] Chen, P.: The Entity–Relationship Model — Towards a Unified View of Data. In: *ACM Transactions on Database Systems* 1 (1976), Nr. 1, S. 9–36

[CHKT06] Conrad, S.; Hasselbring, W.; Koschel, A.; Tritsch, R.: *Enterprise Application Integration*. Elsevier Spektrum Akademischer Verlag, 2006

[CKK02] Clements, P.; Kazman, R.; Klein, M.: *Handbook of Software Architecture Evaluation*. Addison-Wesley, 2002

[Cle96] Clements, P. C.: A Survey of Architecture Description Languages. In: *IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design, Washington, DC, USA*, IEEE Computer Society Press, 1996, S. 16–25

[CMR+97] Corradini, A.; Montanari, U.; Rossi, F.; Ehrig, H.; Heckel, R.; Löwe, M.: Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. In: Rozenberg, G. (Hrsg.), *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*, World Scientific, Kap. 3, 1997

[CN02] Clements, P.; Northrop, L.: *Software Product Lines – Practices and Patterns*. SEI Series in Software Engineering, Addison-Wesley, 2002

[COB] *COBIT-Homepage*. Technischer Bericht, IT Governance Institute, URL `http://www.isaca.org/cobit.htm`

[Coc02] Cockburn, A.: *Agile Software-Entwicklung*. Addison-Wesley, 2002

[Com] Apache Jakarta Commons. URL `http://jakarta.apache.org/commons/`

[Con02] Conn, R.: Developing Software Engineers at the C-130J Software Factory. In: *IEEE Software* 19 (2002), Nr. 5, S. 25–29

[Cop92] Coplien, J.: *Advanced C++ Styles and Idioms*. Addison-Wesley, 1992

[Cor03] Cordy, J.: Comprehending Reality: Practical Challenges to Software Maintenance Automation. In: *Keynote address at IEEE 11th International Workshop on Program Comprehension, Portland*, IEEE Computer Society Press, 2003

[CPT99] Canal, C.; Pimentel, E.; Troya, J. M.: Specification and refinement of dynamic software architectures. In: *Software Architecture*, Kluwer Academic Publishers, 1999, S. 107–126

[CRW98] Clayton, R.; Rugaber, S.; Wills, L.: On the Knowledge Required to Understand a Program. In: *Proceedings of WCRE98*, IEEE Computer Society Press, Oktober 1998, S. 69–78

[CSWH01] Clarke, I.; Sandberg, O.; Wiley, B.; Hong, T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Federrath, H. (Hrsg.), *Designing Privacy Enhancing Technologies – International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, Springer-Verlag, Juli 2001, Bd. 2009 von *Lecture Notes in Computer Science*, S. 46–66

[CV95] Cimitile, A.; Visaggio, G.: Software Salvaging and the Call Dominance Tree. In: *Journal of Systems & Software* 28 (1995), S. 117–127

[CW04] Correa, A. L.; Werner, C. M. L.: Applying Refactoring Techniques to UML/OCL Models. In: Baar, T.; Strohmeier, A.; Moreira, A. M. D.; Mellor, S. J. (Hrsg.), *UML*, Springer-Verlag, 2004, Nr. 3273 in Lecture Notes in Computer Science, S. 173–187

[Daw01] Dawis, E. P.: Architecture of an SS7 Protocol Stack on a Broadband Switch Platform using Dualistic Petri Nets. In: *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2001, S. 323–326

[DDK01] Dawis, E. P.; Dawis, J. F.; Koo, W. P.: Architecture of Computer-based Systems using Dualistic Petri Nets. In: *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, Oktober 2001

[DDN00] Demeyer, S.; Ducasse, S.; Nierstrasz, O.: Finding refactorings via change metrics. In: *OOPSLA*, 2000, S. 166–177

[DEM+99] Depke, R.; Engels, G.; Mehner, K.; Sauer, S.; Wagner, A.: Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen. In: *Informatik – Forschung und Entwicklung* 14 (1999), Nr. 2, S. 83–94

[Der03] Dern, G.: *Management von IT-Architekturen – Informationssysteme im Fokus von Architekturplanung und -entwicklung*. Edition CIO, Wiesbaden: Vieweg Verlag, 2003

[Deu02] van Deursen, A. (Hrsg.): *Proceedings / Ninth Working Conference on Reverse Engineering (WCRE02)*, IEEE Computer Society Press, 2002

[DF98] Diaconescu, R.; Futatsugi, K.: *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*. World Scientific, 1998

[DFI99] Diaconescu, R.; Futatsugi, K.; Iida, S.: Component-Based Algebraic Specification and Verification in CafeOBJ. In: *World Congress on Formal Methods (2)*, 1999, S. 1644–1663

[DHM99] Duke, D. J.; Herman, I.; Marschall, M. S.: *PREMO: A Framework for Multimedia Middleware – Specification, Rationale, and Java Binding*. LNCS 1591, Springer-Verlag, 1999

[Die03] Dietzsch, A.: Positionierung eines Unternehmensarchitektur-Ansatzes: Erfahrung der Schweizerischen Mobiliar im Architekturmanagement. In: *Enterprise Architecture und Enterprise Application Integration (EAI) – Proceedings des GI-Arbeitskreis Enterprise Architecture Frühjahrskonferenz 2003*, St. Gallen: Institut für Wirtschaftsinformatik IWI-HSG, 2003, S. 50–61

[Die04] Dietrich, A.: Puff in der Landschaft. In: *NZZ-Folio* (2004), Nr. 5/04, S. 30–37

[Dij70] Dijkstra, E. W.: Notes on Structured Programming. April 1970, URL `http://www.cs.utexas.edu/users/EWD/`

[DIN68] DIN: *Din 66200: Betrieb von Rechensystemen – Begriffe, Auftragsbeziehungen.* Technischer Bericht, Deutsches Institut für Normung e.V., 1968. Beuth Verlag Berlin

[DIN90a] DIN: *DIN 25424: Fault Tree Analysis: Part 1 (Method and graphical symbols) and Part 2 (Manual: calculation procedures for the evaluation of a fault tree)*. Technischer Bericht, Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 1981/1990

[DIN90b] DIN: *DIN V VDE 0801 –Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben*. Technischer Bericht, Deutsches Institut für Normung e.V., 1990

[DIN91] DIN: *DIN 9126 – Information Technology, Software Product Evaluation, Quality, Characteristics and Guidelines for their Use*. Technischer Bericht, Deutsches Institut für Normung e.V., 1991

[DK99] van Deursen, A.; Kuipers, T.: Identifying Objects Using Cluster and Concept Analysis. In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, Los Angeles, USA: ACM, Mai 1999, S. 246–255

[DKK+01] Dabek, F.; Kaashoek, M. F.; Karger, D.; Morris, R.; Stoica, I.: Wide-area cooperative storage with CFS. In: *Proceedings of the eighteenth ACM symposium on Operating systems principles*, ACM Press, 2001, S. 202–215

[DL04] Dyson, P.; Longshaw, A.: *Architecting Enterprise Solutions: Patterns for High-Capability Internet-Based Systems*. John Wiley & Sons, 2004

[Dou99] Douglass, B. P.: *Doing Hard Time – Developing Real-time Systems with UML, Objects, Frameworks, and Patterns*. Object Technology Series, Addison-Wesley, 1999

[DRD99] Ducasse, S.; Rieger, M.; Demeyer, S.: A Language Independent Approach for Detecting Duplicated Code. In: *ICSM*, 1999, S. 109–118

[DS99] DeBaud, J.-M.; Schmid, K.: A Systematic Approach to Derive the Scope of Software Product Lines. In: *IEEE Computer Society,*

Technical Council on Software Engineering; ACM, Special Interest Group on Software Engineering -SIGSOFT-: International Conference on Software Engineering, ACM Press, 1999, S. 34–43

[DW99a] Dröschel, W.; Wiemers, M.: *Das V-Modell 97 – Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. Oldenbourg-Verlag, 1999

[DW99b] D'Souza, D. F.; Wills, A. C.: *Objects, Components, and Frameworks with UML – The Catalysis Approach*. Addison-Wesley, 1999

[Dym02] Dymond, K. M.: *CMM Handbuch – Das Capability Maturity Model für Software*. Xpert.press, Springer-Verlag, 2002

[EBK+05] Ehrig, H.; Braatz, B.; Klein, M.; Orejas, F.; Pérez, S.; Pino, E.: Object-Oriented Connector-Component Architectures. In: *Proc. Of FESCA, ETAPS satellite*, 2005

[EFRV86] Estrin, G.; Fenchel, R. S.; Razouk, R. R.; Vernon, M. K.: SARA (system architects apprentice): modeling, analysis, and simulation support for design of concurrent systems. In: *IEEE Transactions on Software Engineering* 12 (1986), Nr. 2, S. 293–311

[EJ02] Eden, A. H.; Jahnke, J. H.: Coordinating Software Evolution via Two-Tier Programming. In: *Proceedings of the 5th International Conference on Coordination Models and Languages*, Springer-Verlag, 2002, S. 149–157

[EM90] Ehrig, H.; Mahr, B.: *Fundamentals of Algebraic Specification, Vol. 2: Module Specifications and Constraints*, Bd. 21 von *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1990

[EM02] van Emden, E.; Moonen, L.: Java Quality Assurance by Detecting Code Smells. In: van Deursen [Deu02]

[End93] Endler, M.: A model for distributed management of dynamic changes. In: *4th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, 1993

[EOB+02] Ehrig, H.; Orejas, F.; Braatz, B.; Klein, M.; Piirainen, M.: A Generic Component Concept for System Modeling. In: *Proc. FASE '02*, Springer-Verlag, 2002, Bd. 2306 von *Lecture Notes in Computer Science*

[EPB+04] Ehrig, H.; Padberg, J.; Braatz, B.; Klein, M.; Orejas, F.; Pérez, S.; Pino, E.: A Generic Framework for Connector Architectures based on Components and Transformations. In: *Proc. FESCA'04, satellite of ETAPS'04, Barcelona, ENTCS*, 2004, Bd. 108, S. 53–67

[ER03] Endres, A.; Rombach, D.: *A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories*. Addison Wesley, 2003

[ESN03] Engels, G.; Sauer, S.; Neu, B.: Integrating Software Engineering and User-centred Design for Multimedia Software Developments. In: *In Proc. IEEE Symposia on Human-Centric Computing Languages and Environments – Symposium on Visual/Multimedia Software Engineering*, Auckland, New Zealand: IEEE Computer Society Press, Oktober 2003

[Fak04] Fakultätentag Informatik: *Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten*. Technischer Bericht, November 2004, URL `http://www.ft-informatik.de`

[FH84] Fjeldstadt, R.; Hamlen, W.: Application Program Maintenance Study: Report to Our Respondents. In: *Proc. IBM GUIDE Conference, no. 48*, April 1984

[FH00] Fahmy, H.; Holt, R.: Using Graph Rewriting to Specify Software Architectural Transformations. In: *Proc. of Automated Software Engineering (ASE 2000)*, 2000, S. 187–196

[FHK+97] Finnigan, P. J.; Holt, R. C.; Kalas, I.; Kerr, S.; Kontogiannis, K.; Müller, H. A.; Mylopoulos, M.; Perelgut, S. G.; Stanley, M.; Wong, W.: The software bookshelf. In: *IBM Systems Journal* 36 (1997), Nr. 4, S. 564–593

[FHLS99] Froehlich, G.; Hoover, H. J.; Liu, L.; Sorenson, P.: Reusing Hooks. In: Fayad et al. [FSJ99a], S. 219–236

[FHM+95] Franks, G.; Hubbard, A.; Majumdar, S.; Petriu, D.; Rolia, J.; Woodside, M.: A toolset for Performance Engineering and Software Design of Client-Server Systems. In: *Performance Evaluation* 24 (1995), Nr. 1-2, S. 117–135

[FI99] Futatsugi, K.; Iida, S.: Formal Architecture Description Techniques for Software Evolution. In: *Proc. Int. Workshop on the Principles of Software Evolution*, 1999

[FK98] Frolund, S.; Koistinen, J.: *QML: A Language for Quality of Service Specification*. Technischer Bericht HPL-98-10, Hewlett-Packard Laboratories, 1998

[FKB+05] Firus, V.; Koziolek, H.; Becker, S.; Reussner, R.; Hasselbring, W.: Empirische Bewertung von Performanz-Vorhersageverfahren für Software-Architekturen. In: *Tagungsband Software Engineering 2005 – Fachtagung des GI-Fachbereichs Softwaretechnik, Bd. P-64 der Reihe Lecture Notes in Informatics*, März 2005, S. 55–66

[FM93] Fenelon, P.; McDermid, J. A.: An Integrated Toolset For Software Safety Analysis. In: *Journal of Systems & Software* 21 (1993), Nr. 3, S. 279–290

[FM97] Fiadeiro, J. L.; Maibaum, T.: Categorical semantics of parallel program design. In: *Science of Computer Programming* 28 (1997), S. 111–138

[FMC05] FMC Consortium: Fundamental Modeling Concepts Home Page. Website, 2005, URL http://f-m-c.org/

[FMNP94] Fenelon, P.; McDermid, J.; Nicholson, M.; Pumfrey, D. J.: Towards Integrated Safety Analysis and Design. In: *ACM SIGAPP Applied Computing Review* 2 (1994), Nr. 1, S. 21–32

[Foe03] Foegen, M.: Architektur und Architekturmanagement. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 57–65

[Fou] Four J's: Four J's Development Tools. URL http://www.4js.com

[Fow97] Fowler, M.: *Analysis Patterns*. Addison-Wesley, 1997

[Fow00] Fowler, M.: *Refactoring – Wie Sie das Design vorhandener Software verbessern*. Addison-Wesley, 2000

[Fow03] Fowler, M.: *Patterns für Enterprise-Application-Architekturen*. Bonn: mitp, 2003

[FP97] Fenton, N. E.; Pfleeger, S. H.: *Software Metrics – a rigorous and practical approach*. International Thomson Computer Press, 1997

[FPR00] Fontoura, M.; Pree, W.; Rumpe, B.: UML-F: A Modeling Language for Object-Oriented Frameworks. In: *ECOOP*, Springer-Verlag, Bd. 1850 von Lecture Notes in Computer Sciences, 2000, S. 63–82

[FPR02] Fontoura, M.; Pree, W.; Rumpe, B.: *The UML Profile for Framework Architectures*. Object Technology Series, Addison-Wesley, 2002

[Fra99] Franks, G.: *Performance Analysis of Distributed Server Systems*. Dissertation, Carleton University, Ottawa, 1999

[FRF+02] Fowler, M.; Rice, D.; Foemmel, M.; Hieatt, E.; Mee, R.; Stafford, R. (Hrsg.): *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002

[FSJ99a] Fayad, M. E.; Schmidt, D. C.; Johnson, R. E. (Hrsg.): *Building Application Frameworks – Object-Oriented Foundations of Framework Design*. Wiley computer publishing, John Wiley & Sons, 1999

[FSJ99b] Fayad, M. E.; Schmidt, D. C.; Johnson, R. E. (Hrsg.): *Implementing Application Frameworks: Object-Oriented Frameworks at Work*. Wiley computer publishing, John Wiley & Sons, 1999

[FV03] Faust, D.; Verhoef, C.: Software Product Line Migration and Deployment. In: *Software – Practice and Experience* 33 (2003), S. 933–955

[Gam92] Gamma, E.: *Objektorientierte Software-Entwicklung am Beispiel von ET++ – Design-Muster, Klassenbibliothek, Werkzeuge*. Springer-Verlag, 1992

[Gar03] Garlan, D.: Formal modeling and analysis of software architecture: Components, connectors, and events. In: Bernado und Inveradi [BI03], S. 1–24

[GBS01] van Gurp, J.; Bosch, J.; Svahnberg, M.: On the Notion of Variability in Software Product Lines. In: Kazman, R.; Kruchten, P.; Verhoef, C.; van Vliet, H. (Hrsg.), *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01)*, IEEE Computer Society Press, 2001, S. 45–54

[GFd98] Griss, M.; Favaro, J.; d'Alessandro, M.: Integrating Feature Modeling with the RSEB. In: *Proceedings of the Fifth International Conference on Software Reuse*, Vancouver, BC, Canada, 1998, S. 76–85

[GH94] Gilmore, S.; Hillston, J.: The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling. In: *Proceedings of the Seventh International Conference for Modelling Techniques and Tools for Performance Evaluation*, 1994, S. 353–368

[GHJV04] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. Programmer's Choice, Addison-Wesley, 2004

[GHOS96] Gray, J.; Helland, P.; O'Neil, P.; Shasha, D.: The Dangers of Replication and a Solution. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, ACM Press, 1996, S. 173–182

[GHR+02] Graham, P.; Hulzinga, S.; Rudd, C.; van Dijk, A.; von Winden, R.; Bekenkamp, P.; Doeff, I.; Hinley, D.; Leigh, C.; Stringfellow, I.; Wielinga, R.: *ICT Infrastructure Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2002

[Gia95] Giannakopoulou, D.: *The TRACTA Approach for Behaviour Analysis of Concurrent Systems*. Technical report DoC 95/16, Department of Computing, Imperial College of Science, Technology and Medicine, September 1995

[GKKS04] Gröne, B.; Knöpfel, A.; Kugel, R.; Schmidt, O.: *The Apache Modelling Project*. Institute Report 5, Hasso-Plattner-Institut für Softwaresystemtechnik, 2004, URL `http://apache.hpi.uni-potsdam.de`

[GKP05] Grunske, L.; Kaiser, B.; Papadopoulos, Y.: Model-Driven Safety Evaluation with State-Event-Based Component Failure Annotations. In: *CBSE Component-based Software Engineering*, 2005, S. 33–48

[GKR05] Grunske, L.; Kaiser, B.; Reussner, R.: Annotation of Component Specifications with Modular Analysis Models for Safety Properties.

In: *Embedded Software Development with Components – An Overview on Current Research Trends*, Springer-Verlag, 2005, S. 737–748

[Gla98a] Glass, R.: Maintenance: Less Is Not More. In: *IEEE Software* 15 (1998), Nr. 4, S. 67–68

[Gla98b] Glass, R. L.: *Software Runaways. Lessons learned from Massive Software Project Failures*. Prentice Hall, 1998

[Gla02a] Glass, R. L.: *Facts and Fallacies of Software Engineering*. Addison-Wesley, 2002

[Gla02b] Glass, R. L.: The Naturalness of Object Orientation: Beating a Dead Horse? In: *IEEE Software* 19 (2002), Nr. 3, S. 103–104

[Gla04] Glass, R. L.: Learning to Distinguish a Solution from a Problem. In: *IEEE Software* 21 (2004), Nr. 3, S. 111–112

[GM02] Grassi, V.; Mirandola, R.: PRIMAmob-UML: A Methodology for Performance Analysis of Mobile Software Architectures. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2002, S. 262–274

[GMW97] Garlan, D.; Monroe, R.; Wile, D.: Acme: An Architecture Description Interchange Language. In: *Proc. of CASCON'97*, 1997, S. 169–183

[Gol04] Golm, M.: Offene Systemarchitekturen in der Automobilsoftware. In: *Objektspektrum* (2004), Nr. 5, S. 60–63

[Gom04] Gomaa, H.: *Designing Software Product Lines with UML – From Use-Cases to Pattern-Based Software-Architectures*. Addison-Wesley, August 2004

[Gov99] Govoni, D.: *Java Application Frameworks*. John Wiley & Sons, 1999

[GP02] Gu, G.; Petriu, D.: XSLT Transformations from UML Models to LQN Performance Models. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2002

[Gra] GraTra: Graph Transformations & Graph Grammars. URL `http://www.gratra.org/`

[Gru04] Grunske, L.: *Strukturorientierte Optimierung der Qualitätseigenschaften von softwareintensiven technischen Systemen im Architekturentwurf*. Dissertation, Universität Potsdam, 2004

[GSCK04] Greenfield, J.; Short, K.; Cook, S.; Kent, S.: *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. John Wiley & Sons, 2004

[GSMD03] van Gorp, P.; Stenten, H.; Mens, T.; Demeyer, S.: Towards automating source-consistent UML Refactorings. In: *Proceedings of UML 03*, 2003

[Hab92] Habel, A.: *Hyperedge replacement: Grammars and Languages*, Bd. 643 von *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 1992

[Hag03] Hagen, C.: Integrationsarchitektur der Credit Suisse. In: Aier, S.; Schönherr, M. (Hrsg.), *Enterprise Application Integration*, Berlin: GITO-Verlag, 2003, S. 61–81

[Has97] Hasselbring, W.: Federated Integration of Replicated Information within Hospitals. In: *International Journal on Digital Libraries* 1 (1997), Nr. 3, S. 192–208

[Has00] Hasselbring, W.: Information System Integration. In: *Communications of the ACM* 43 (2000), Nr. 6, S. 33–38

[Has02] Hasselbring, W.: Component-Based Software Engineering. In: Chang, S. (Hrsg.), *Handbook of Software Engineering and Knowledge Engineering*, New Jersey: World Scientific Publishing, 2002, S. 289–305

[HB85] Hutchens, D.; Basili, V.: System structure analysis: clustering with data bindings. In: *IEEE Transactions on Software Engineering* SE-11 (1985), Nr. 8, S. 749–757

[Hei02] Heinrich, L. J.: *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*. München, Wien: Oldenbourg-Verlag, 7. Aufl., 2002

[HFS04] Hein, A.; Fischer, T.; Stiel, S.: Fahrerassistenzsystem bei der Robert Bosch GmbH. In: Böckle et al. [BKPS04], S. 193–205

[HH03] Hochstein, A.; Hunziker, A.: Serviceorientierte Referenzmodelle des IT-Managements. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 46–56

[HHK+00] Hermanns, H.; Herzog, U.; Klehmet, U.; Mertsiotakis, V.; Siegle, M.: Compositional Performance Modelling with the TIPPtool. In: *Performance Evaluation* 39 (2000), Nr. 1-4, S. 5–35

[HHK02] Hermanns, H.; Herzog, U.; Katoen, J.: Process Algebra for Performance Evaluation. In: *Theoretical Computer Science* 274 (2002), Nr. 1-2, S. 43–87

[HHKS97] Heuer-Hasenplatt, H.; Hollunder, B.; Kittlaus, H.-B.; Schumacher, N.: Bausteinorientierte Anwendungsentwicklung: Voraussetzungen, Anforderungen und Auswirkungen. In: *Objektspektrum*, Nr. 3, 1997, S. 40–51

[Hil93] Hillston, J.: *PEPA – Performance Enhanced Process Algebra*. Technischer Bericht, Dept. of Computer Science, University of Edinburgh, 1993

[HIM99] Hirsch, D.; Inverardi, P.; Montanari, U.: Modeling Software Architectures and Styles with Graph Grammars and Constraint Solving. In: *Proc. Working IFIP Conference on Software Architecture*, 1999

[HK03] Hitz, M.; Kappel, G.: *UML Work: Von der Analyse zur Realisierung*. dpunkt.verlag, 2. Aufl., 2003

[HKKI04] Higo, Y.; Kamiya, T.; Kusumoto, S.; Inoue, K.: Refactoring Support Based on Code Clone Analysis. In: Bomarius, F.; Iida, H. (Hrsg.), *PROFES*, Springer-Verlag, 2004, Nr. 3009 in Lecture Notes in Computer Science, S. 220–233

[HM99] Hirsch, D.; Montanari, U.: Consistent transformations for software architecture styles of distributed systems. In: *Electronic Notes in Theoretical Computer Science* 28 (1999), S. 23–40

[HN90] Harandi, M. T.; Ning, J. Q.: Knowledge-Based Program Analysis. In: *IEEE Software* 7 (1990), Nr. 1, S. 74–81

[HNS00] Hofmeister, C.; Nord, R.; Soni, D.: *Applied Software Architecture*. Object technology series, Addison-Wesley, 2000

[Hoa85] Hoare, C. A. R.: *Communicating Sequential Processes*. Prentice-Hall international series in computer science, Prentice Hall, 1985

[HPR89] Horwitz, S.; Pfeiffer, P.; Reps, T.: Dependence analysis for pointer variables. In: *Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation*, ACM Press, 1989, S. 28–40,

[HRB88] Horwitz, S.; Reps, T.; Binkley, D.: Interprocedural Slicing Using Dependence Graphs. In: *Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation*, 1988, S. 35–46

[HRJ+04] Hasselbring, W.; Reussner, R.; Jaekel, H.; Schlegelmilch, J.; Teschke, T.; Krieghoff, S.: The Dublo Architecture Pattern for Smooth Migration of Business Information Systems: An experience report. In: *Proceedings of the 26th International Conference on Software Engeneering (ICSE 2004)*, IEEE Computer Society Press, Mai 2004, S. 117–126

[Hub04] Huber-Graul, M.: Software-Wartung als Herausforderung. 2004, URL `http://www.nzz.ch/2003/02/04/hy/page-article8NCG9.html`

[Hüs94] Hüsener, T.: *Entwurf komplexer Echtzeitsysteme: State of the Art*. Nr. 11 in Angewandte Informatik, BI Wiss.-Verlag, 1994

[HW03] Hohpe, B.; Woolf, G.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003

[HW04] Heinrich, B.; Winter, R.: A Strategy Modelling Technique for Financial Services. In: *The European IS Profession in the Global Networking Environment, Proc. 12th European Conference on Information Systems*, 2004

[HW05] Hafner, M.; Winter, R.: Vorgehensmodell für das Management der unternehmensweiten Applikationsarchitektur. In: Ferstl, O. K.; Sinz, E. J.; Eckert, S.; Isselhorst, T. (Hrsg.), *Wirtschaftsinformatik 2005: eEconomy – eGovernment – eSociety, Bamberg, 23.02.2005*, Heidelberg: Physica, 2005, S. 627–646

[IBM] IBM: Informix 4GL product family. URL `http://www-3.ibm.com/software/data/informix/tools/4gl/`

[IBM04a] IBM: QBIC Home Page. 2004, URL `http://wwwqbic.almaden.ibm.com/`

[IBM04b] IBM: Rational Unified Process Evaluation V6.13, 2004 URL `http://www-306.ibm.com/software/awdtools/`

[IEC90] IEC: *Fault-Tree-Analysis (FTA)*. Standard IEC 61025, International Electrotechnical Commission, Genf, Schweiz, 1990

[IEE90] IEEE: *Standard Glossary of Software Engineering Terminology*. Standard IEEE 610.12-1990, IEEE, 1990

[IEE00] IEEE: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems – Description*. Standard IEEE 1471–2000, ANSI/IEEE, 2000

[ISO98] ISO/IEC: *Reference Model for Open Distributed Processing – Part 1: Overview*. ISO Standard 10746-1, International Organization for Standardization, 1998

[ISO99] ISO/IEC: *Interface Definition Language*. ISO Standard 14750, International Organization for Standardization, 1999

[ISO01] ISO/IEC: *Software Engineering – Product Quality – Quality Model*. ISO Standard 9126-1, International Organization for Standardization, 2001

[ite04] iteratec: iteratec e-Business Referenzarchitektur, Version 1.4. 2004, URL `http://www.iteratec.de/dokumente/iteratec_ebusiness_Referenzarchitektur.pdf`

[ITI] Official ITIL Webpages. URL `http://www.ogc.gov.uk/index.asp?id=2261`

[ITU99] ITU: Message Sequence Charts, ITU-T Recommendation. 1999

[JAD] JADE: Website, URL `http://jade.dautelle.com/`

[Jah04] Jahnke, J. H.: Reverse engineering software architecture using rough clusters. In: *Processing IEEE Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS'04)*, IEEE Computer Society Press, 2004, S. 4–9

[JBo04] JBoss Inc.: The JBoss Application Server. 2004, URL `http://www.jboss.org/products/jbossas`

[JBu] JBuilder. URL `http://www.borland.com/jbuilder/`

[JDB] JDBC: Website,
URL http://java.sun.com/products/jdbc/

[JF88] Johnson, R.; Foote, B.: Designing Reusable Classes. In: *Journal of Object-Oriented Programming* 1 (1988), Nr. 2, S. 22–35

[JLMM04] Jablonski, S.; Lay, R.; Meiler, C.; Müller, S.: Process Based Data Logistics: a Solution for Clinical Integration Problems. In: *First International Workshop on Data Integration in the Life Sciences (DILS 2004), Leipzig, Germany*, Springer-Verlag, 2004, Nr. 2994 in Lecture Notes in Bioinformatics

[JN99] Jacobson, E. E.; Nowack, P.: Frameworks and Patterns – Architectural Abstractions. In: Fayad et al. [FSJ99a], S. 29–54

[JO93] Johnson, R. E.; Opdyke, W. F.: Refactoring and Aggregation. In: Nishio, S.; Yonezawa, A. (Hrsg.), *ISOTAS*, Springer-Verlag, 1993, Nr. 742 in Lecture Notes in Computer Science, S. 264–278

[JPM03] Jablonski, S.; Petrov, I.; Meiler, C.: An Architectural Framework for Web Applications. In: *Proceedings of ICEIS 2003, 5th International Conference on Enterprise Information Systems, Angers, France*, 2003, Bd. 1, S. 285–293

[JR01] Jacobi, C.; Rumpe, B.: Hierarchical XP. Improving XP for Large-Scale Projects in Analogy to Reorganization Processes. In: Succi, G.; Marchesi, M. (Hrsg.), *Extreme Programming Examined*, Addison-Wesley, 2001, S. 83–102

[JRL00] Jazayeri, M.; Ran, A.; van der Linden, F. (Hrsg.): *Software Architecture for Product Families – Principles and Practices*. Addison-Wesley, 2000

[JSZ97] Jahnke, J.; Schäfer, W.; Zuendorf, A.: Generic Fuzzy Reasoning Nets as a Basis for Reverse Engineering Relational Database Applications. In: Jazayeri, M.; Schauer, H. (Hrsg.), *ESEC/FSE '97*, Springer-Verlag, 1997, Bd. 1301 von *Lecture Notes in Computer Science*, S. 193–210

[Kan92] Kant, K.: *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992

[KBAW94] Kazman, R.; Bass, L.; Abowd, G.; Webb, M.: SAAM: A Method for Analyzing the Properties Software Architectures. In: *Proceedings of the 16th International Conference on Software Engineering, (Sorrento, Italy)*, 1994, S. 81–90

[KBS] KBSt: Standards und Architekturen für eGovernment-Anwendungen.
URL http://www.kbst.bund.de/saga

[KCH+90] Kang, K. C.; Cohen, S. G.; Hess, J. A.; Novak, W. E.; Peterson, A. S.: *Feature-Oriented Domain Analysis (FODA) – Feasibility Study*. Technischer Bericht CMU/SEI-90-TR-21 ESD 90-TR-222,

Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1990

[Kel03] Keller, F.: *Über die Rolle von Architekturbeschreibungen im Software-Entwicklungsprozess*. Dissertation, Universität Potsdam, August 2003

[Ker05] Kerievsky, J.: *Refactoring to Patterns*. Addison-Wesley signature series, Addison-Wesley, 2005

[KGT06] Knöpfel, A.; Gröne, B.; Tabeling, P.: *Fundamental Modeling Concepts*. John Wiley & Sons, 2006

[KJ04] Kirchner, M.; Jain, P.: *Patterns for resource management (Pattern-oriented Software Architecture, vol. 3)*. Wiley series in software design patterns, John Wiley & Sons, 2004

[KKB+98] Kazman, R.; Klein, M.; Barbacci, M.; Longstaff, T.; Lipson, H.; Carriere, J.: The Architecture Tradeoff Analysis Method. In: *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 1998, S. 68–78

[KKC00] Kazman, R.; Klein, M.; Clements, P.: *ATAM: Method for 136 Architecture Evaluation*. Technischer Bericht CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000

[Kle75] Kleinrock, L.: *Queueing Systems, Volume 1: Theory*. A Wiley-Interscience publication, John Wiley & Sons, 1975

[Kle99] Kleis, W.: *Konzepte zur verständlichen Beschreibung objektorientierter Frameworks*. Berichte aus der Informatik, Aachen: Shaker Verlag, 1999

[KLM+97] Kiczales, G.; Lamping, J.; Menhdhekar, A.; Maeda, C.; Lopes, C.; Loingtier, J.-M.; Irwin, J.: Aspect-Oriented Programming. In: Akşit, M.; Matsuoka, S. (Hrsg.), *ECOOP'97 – Object-Oriented Programming, 11th European Conference*, Springer-Verlag, Bd. 1241 von *Lecture Notes in Computer Science*, 1997, S. 220–242

[KLM03] Kaiser, B.; Liggesmeyer, P.; Mäckel, O.: A New Component Concept for Fault Trees. In: *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS'03)*, Adelaide, 2003

[KMO+05] Kreowski, H.-J.; Montanari, U.; Orejas, F.; Rozenberg, G.; Taentzer, G. (Hrsg.): *Formal methods in software and systems modeling – essays dedicated to Hartmut Ehrig on the occasion of his 60th birthday*, Bd. 3393 von *Lecture Notes in Computer Science*. Springer-Verlag, 2005

[KMRT02] Kienle, A.; Mambrey, P.; Reiband, N.; Tan, D.: Erfolgsfaktoren und Hindernisse bei Wissensmanagementlösungen. In: *GI Jahrestagung 2002*, Gesellschaft für Informatik, 2002, Bd. 19 von *GI-Edition Lecture Notes in Informatics*, S. 709–712

[KMU03] Kramer, J.; Magee, J.; Uchitel, S.: Software architecture modeling & analysis: A rigorous approach. In: Bernado und Inveradi [BI03], S. 44–51

[Kos00] Koschke, R.: *Atomic Architectural Component Recovery for Program Understanding and Evolution*. Dissertation, Universität Stuttgart, 2000

[Koz04] Koziolek, H.: *Empirische Bewertung von Performance-Analyseverfahren für Software-Architekturen*. Diplomarbeit, Universität Oldenburg, Fakultät II, Department für Informatik, Okt. 2004

[KP00] King, P. J. B.; Pooley, R. J.: Derivation of Petri net performance models from UML specifications of communications software. Springer-Verlag, 2000, Bd. 2047 von *Lecture Notes in Computer Science*, S. 262–276

[KPW04] King, R.; Popitsch, N.; Westermann, U.: METIS: a flexible database foundation for unified media management. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, ACM Press, 2004, S. 744–745

[KR05] Krahn, H.; Rumpe, B.: *Evolution von Software-Architekturen*. Technischer Bericht Informatik-Bericht 2005-04, Technische Universität Braunschweig, Carl-Friedrich-Gauß-Fakultät für Mathematik und Informatik, Mai 2005

[Krc90] Krcmar, H.: Bedeutung und Ziele von Informationssystemarchitekturen. In: *Wirtschaftsinformatik* 32 (1990), Nr. 5, S. 395–402

[Kru95] Kruchten, P.: The 4+1 View Model of Architecture. In: *IEEE Software* 12 (1995), Nr. 6, S. 42–50

[KTG+02] Keller, F.; Tabeling, P.; Gröne, B.; Knöpfel, A.; Schmidt, O.; Kugel, R.; Apfelbacher, R.: Improving knowledge transfer at the architectural level: Concepts and notations. In: Arabnia, H. R.; Mun, Y. (Hrsg.), *Proceedings of the SERP'02, the International Conference on Software Engineering Research and Practice, Las Vegas*, CSREA Press, Juni 2002, S. 101–107

[KW99] Kullbach, B.; Winter, A.: Querying as an Enabling Technology in Software Reengineering. In: Verhoef, C.; Nesi, P. (Hrsg.), *Proceedings of the 3rd Euromicro Conference on Software Maintenance and Reengineering*, Los Alamitos: IEEE Computer Society Press, 1999, S. 42–50

[LBHB99] Lundberg, L.; Bosch, J.; Häggander, D.; Bengtsson, P.-O.: Quality Attributes in Software Architecture Design. In: *Proceedings of the IASTED 3rd International Conference on Software Engineering and Applications*, October 1999, S. 353–362

[LBVB02] Lassing, N.; Bengtsson, P.; van Vliet, H.; Bosch, J.: Experiences with ALMA: architecture-level modifiability analysis. In: *Journal of Systems & Software* 61 (2002), Nr. 1, S. 47–57

[Lea99] Lea, D.: *Concurrent Programming in Java, Design Principles and Patterns*. Addison-Wesley, 1999

[Lev95] Leveson, N. G.: *SAFEWARE: System Safety and Computers*. Addison-Wesley, 1995

[LF99] Lopes, A.; Fiadeiro, J. L.: Using explicit state to describe architectures. In: *Proc. of Fundamental Approaches to Software Engineering*, Springer-Verlag, 1999, Bd. 1577 von *Lecture Notes in Computer Science*, S. 144–160

[LH96] Larsen, L.; Harrold, M. J.: Slicing object-oriented software. In: *Proceedings of the 18th international conference on Software engineering*, IEEE Computer Society Press, 1996, S. 495–505

[Lig00] Liggesmeyer, P.: *Qualitätssicherung softwareintensiver technischer Systeme*. Spektrum Akademischer Verlag, 2000

[Lin03] Linthicum, D.: *Next Generation Application Integration: From Simple Information to Web Services*. Addison-Wesley, 2003

[LN95] Landin, N.; Niklasson, A.: *Development of Object-Oriented Frameworks*. Diplomarbeit, Department of Communication Systems, Lund Institute of Technology, Lund University, Lund, Sweden, 1995

[LRR03] Lichtenegger, R.; Rohloff, M.; Rosauer, B.: Beschreibung von Unternehmensarchitekturen: Sichten und Abhängigkeiten am Beispiel der IT-Infrastrukturarchitektur. In: Dittrich, K.; König, W.; Oberweis, A.; Rannenberg, K.; Wahlster, W. (Hrsg.), *Informatik 2003*, Bonn: Köllen Druck+Verlag, 2003, Nr. P-35 in GI-Edition Lecture Notes in Informatics, S. 426–434

[LS97] Lindig, C.; Snelting, G.: Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis. In: *Proceedings of the 1997 International Conference on Software Engineering*, ACM Press, 1997, S. 349–359

[LSA04] Leser, F.; Scheibehenne, R.; Alt, R.: Ansatz zur Bestimmung des Architekturnutzens bei der Deutschen Telekom. In: Alt, R.; Österle, H. (Hrsg.), *Real-time Business. Lösungen, Bausteine und Potenziale des Business Networking*, Berlin et al.: Springer-Verlag, 2004, S. 233–253

[LT87] Lynch, N.; Tuttle, M.: Hierarchical Correctness Proofs for Distributed Algorithms. In: *Proc. of the 6th Annual ACM Symposium on Principles of Distributed Computing*, 1987, S. 137–151

[LT89] Lynch, N.; Tuttle, M.: An Introduction to Input/Output automata. In: *CWI quarterly* 2 (1989), Nr. 3, S. 219–246

[LTP03] Lethbridge, T. C.; Tichelaar, S.; Ploederede, E.: The Dagstuhl Middle Metamodel: A Schema For Reverse Engineering. In: *Proceedings of the International Workshop on Meta-Models and Schemas for Reverse Engineering (ateM 2003)*, Springer-Verlag, 2003, Electronic Notes in Computer Science, S. 7–18

[LV95] Luckham, D. C.; Vera, J.: An Event-Based Architecture Definition Language. In: *IEEE Transactions on Software Engineering* 21 (1995), Nr. 9, S. 717–734

[LVB+93] Luckham, D. C.; Vera, J.; Bryan, D.; Augustin, L.; Belz, F.: Partial Orderings of Event Sets and Their Application to Prototyping Concurrent, Timed Systems. In: *Journal of Systems & Software* 21 (1993), Nr. 3, S. 253–265

[LW97] Lutz, R. R.; Woodhouse, R. M.: Requirements Analysis Using Forward and Backward Search. In: *Annals of Software Engineering* 3 (1997), Nr. 1, S. 459–475

[Mac05] Macromedia, Inc., USA: Macromedia – Flash MX 2004. 1995–2005, URL `http://www.macromedia.com/software/flash/`

[MAD94] Menasce, D. A.; Almeida, V. A. F.; Dowdy, L. W.: *Capacity Planning and Performance Modeling – from mainframes to client-server systems*. Prentice Hall, 1994

[MAD04] Menasce, D. A.; Almeida, V. A.; Dowdy, L. W.: *Performance by Design*. Prentice Hall, 2004

[Mar02] Marinescu, F.: *EJB Design Patterns: Advanced Patterns, Processes, and Idioms*. John Wiley & Sons, 2002

[Mat03] Mattern, T.: The New Architecture of Integration. In: *SAP Info* online Journal, 2003

[Mau01] Mauri, G.: *Integrating Safety Analysis Techniques, Supporting Identification of Common Cause Failures*. Dissertation, Department of Computer Science, University of York, 2001

[MB01] Mertens, P.; Bodendorf, F.: *Programmierte Einführung in die Betriebswirtschaftslehre – Institutionenlehre*. Wiesbaden: Gabler Verlag, 11. Aufl., 2001

[MBC84] Marsan, M. A.; Balbo, G.; Conte, G.: A class of Generalized Stochastic Petri Nets for the performance evaluation of mulitprocessor systems. In: *ACM Transactions on Computer Systems* (1984)

[MBP+04] Moll, K.-R.; Broy, M.; Pizka, M.; Seifert, T.; Bergner, K.; Rausch, A.: Erfolgreiches Management von Software-Projekten. In: *Informatik-Spektrum* (2004)

[MDEK95] Magee, J.; Dulay, N.; Eisenbach, S.; Kramer, J.: Specifying Distributed Software Architectures. In: Schäfer, W.; Botella, P. (Hrsg.), *Proc. 5th European Software Engineering Conf.*

(ESEC 95), Sitges, Spain, Springer-Verlag, 1995, Nr. 989 in Lecture Notes in Computer Science, S. 137–153

[MDJ02] Mens, T.; Demeyer, S.; Janssens, D.: Formalising Behaviour Preserving Program Transformations. In: *ICGT 2002*, 2002, Nr. 2505 in Lecture Notes in Computer Science

[Mey90] Meyer, B.: Lessons from the design of the Eiffel libraries. In: *Communications of the ACM* 33 (1990), Nr. 9, S. 68–88

[Mey97] Meyer, B.: *Object-Oriented Software Construction*. Prentice Hall, 2. Aufl., 1997

[Mey99] Meyer, B.: The Unity of Software and the Power of Roundtrip Engineering. In: *Proceedings of the Technology of Object-Oriented Languages and Systems*, IEEE Computer Society Press, 1999, S. 2

[MHR04] Matevska-Meyer, J.; Hasselbring, W.; Reussner, R.: Software Architecture Description supporting Component Deployment and System Runtime Reconfiguration. In: *Proceedings of Workshop on Component-Oriented Programming (WCOP 2004)*, Juni 2004

[Mica] Microsoft Coorperation: Domain Specific Language (DSL) Tools. URL `http://lab.msdn.microsoft.com/vs2005/teamsystem/workshop/dsltools/`

[Micb] Microsystems, S.: Java 2 Platform, Enterprise Edition (J2EE). URL `http://java.sun.com/j2ee/`

[Mic05a] Microsoft Corporation, USA: About the Microsoft Foundation Classes. 2005, URL `http://msdn.microsoft.com/`

[Mic05b] Microsoft Corporation, USA: Microsoft DirectShow 9.0. 2005, URL `http://msdn.microsoft.com/`

[Mic05c] Microsoft Corporation, USA: .NET Development. 2005, URL `http://msdn.microsoft.com/`

[Mil67] Milgram, S.: The Small World Problem. In: *Psychology Today* 61 (1967), S. 60–67

[Mil80] Milner, R.: *A calculus of communicating systems*. Nr. 92 in Lecture Notes in Computer Science, Springer-Verlag, 1980

[Mil89] Milner, R.: *Communication and Concurrency*. Prentice Hall, 1989

[MJS+00] Müller, H. A.; Jahnke, J. H.; Smith, D.; Storey, M.; Tilley, S. R.; Wong, K.: Reverse Engineering: A Roadmap. In: Finkelstein, A. (Hrsg.), *The Future of Software Engineering*, ACM Press, 2000, S. 47–60

[MKG97] Magee, J.; Kramer, J.; Giannakopoulou, D.: Analysing the behaviour of distributed software architectures: a case study. In: *Proceedings of the Sixth IEEE Computer Society Workshop on Distributed Computing Systems*, IEEE Computer Society, 1997

[MM01] Maletic, J. I.; Marcus, A.: Supporting program comprehension using semantic and structural information. In: *Proceedings of the*

23rd International Conference on Software Engineering, IEEE Computer Society Press, 2001, S. 103–112

[MM03] Miller, J.; Mukerji, J.: MDA Guide Version 1.0.1 (omg/2003-06-01). Juni 2003, URL `http://www.omg.org/docs/omg/03-06-01.pdf`

[MMT70] Mesarovic, M. D.; Macko, D.; Takahara, Y.: *Theory of Hierarchical, Multilevel Systems*. New York, London: Academic Press, 1970

[MNP+04] McCoy, D. W.; Natis, Y. V.; Pezzini, M.; Sinur, J.; Schulte, R. W.; Thompson, J.; Lhereux, B. J.; Kenney, F.; Haight, C.; Friedman, B.; Phifer, G.; James, G. A.; Blechar, M. J.; Vecchio, D.: Hype Cycle for Application Integration and Platform Middleware. 2004

[MNS95] Murphy, G. C.; Notkin, D.; Sullivan, K.: Software reflexion models: bridging the gap between source and high-level models. In: *Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering*, ACM Press, 1995, S. 18–28

[Mon98] Montenegro, S.: Formale Methoden in der Softwareentwicklung Heute und Morgen. In: *Elektronik Zeitschrift* 21 (1998)

[MP94] McDermid, J. A.; Pumfrey, D. J.: A Development of Hazard Analysis to Aid Software Design. In: *Compass'94: 9th Annual Conference on Computer Assurance*, Gaithersburg, MD: National Institute of Standards and Technology, 1994, S. 17–26

[MQ94] Moriconi, M.; Qian, X.: Correctness and Composition of Software Architectures. In: *Proceedings of ACM SIGSOFT'94: Symposium on Foundations of Software Engineering*, 1994, S. 164–174.

[MQR95] Moriconi, M.; Qian, X.; Riemenschneider, R. A.: A formal approach to correct refinement of software architectures. In: *Proceedings of ACM SIGSOFT'94: Symposium on Foundations of Software Engineering*, 1995, Bd. 21(4), S. 356–372

[MR97] Moriconi, M.; Riemenschneider, R.: *Introduction to Safl 1.0: A language foe specifying software architecture hierarchies*. Technischer Bericht SRI-CSL-97-01, SRI International, 1997

[MR04] Mortensen, K. H.; Rölke, H.: Petri net tool database. 2004, URL `http://www.daimi.au.dk/PetriNets`

[MRR04] Meister, J.; Reussner, R.; Rohde, M.: Managing Product Line Variability by Patterns. In: Weske, M.; Liggesmeyer, P. (Hrsg.), *Proceedings of 5th Intl. Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World, Net.ObjectDays 2004, Erfurt, Germany*, Springer-Verlag, September 2004, Bd. 3263 von *Lecture Notes in Computer Science*, S. 153–168

[MSUW04] Mellor, S. J.; Scott, K.; Uhl, A.; Weise, D.: *MDA Distilled*. Addison-Wesley, 2004

[MT00] Medvidovic, N.; Taylor, R. N.: A Classification and Comparison Framework for Software Architecture Description Languages. In: *IEEE Transactions on Software Engineering* 26 (2000), Nr. 1, S. 70–93

[MT04] Mens, T.; Tourwé, T.: A Survey of Software Refactoring. In: *IEEE Transactions on Software Engineering* 30 (2004), Nr. 2, S. 126–139

[MW01] Mens, T.; Wermelinger, M.: *Proceedings of the Workshop on Formal Foundations of Software Evolution*. Technical Report UNL-DI-1-2001, Departamento de Informatica Faculdade de Ciencias e Tecnologia Universidade Nova de Lisboa, 2001

[MW02] Mens, T.; Wermelinger, M.: Separation of concerns for software evolution. In: *Journal of software maintenance and evolution – research and practice* 14 (2002), Nr. 5, S. 311–315

[MW04] Melchert, F.; Winter, R.: The Enabling Role of Information Technology for Business Performance Management. In: *Decision Support in an Uncertain and Complex World, Proc. of the 2004 IFIP International Conference on Decision Support Systems (DSS2004)*, 2004, S. 535–546

[MWT95] Müller, H. A.; Wong, K.; Tilley, T.: Understanding Software Systems Using Reverse Engineering Technology. In: Alagar, V. S.; Missaoui, R. (Hrsg.), *Object-Oriented Technology for Database and Software Systems*, World Scientific, 1995, S. 240–252

[MZK03] Meyer, M.; Zarnekow, R.; Kolbe, L. M.: IT-Governance – Begriffe, Status quo und Bedeutung. In: *Wirtschaftsinformatik* 45 (2003), Nr. 4, S. 445–448

[Neu02] Neuhaus, U.: Service Level Agreements als Basis der Qualitätssicherung für einen IT-Betrieb. In: von Knop, J.; Haverkamp, W. (Hrsg.), *Zukunft der Netze*, Gesellschaft für Informatik, 2002, Nr. P-17 in GI-Edition Lecture Notes in Informatics, S. 309–316

[NHW+02] Niemann, H.; Hasselbring, W.; Wendt, T.; Winter, A.; Meierhofer, M.: Kopplungsstrategien für Anwendungssysteme im Krankenhaus. In: *Wirtschaftsinformatik* 44 (2002), Nr. 5, S. 425–434

[Nie02] Niere, J.: Fuzzy logic based interactive recovery of software design. In: *Proceedings of the 24th International Conference on Software Engineering (ICSE-02)*, ACM Press, Mai 2002, S. 727–728

[NMM05] *Network-Integrated Multimedia Middleware*. Technischer Bericht, Computer Graphics Lab, Saarland Universität, Saarbrücken, 2005, URL `http://www.networkmultimedia.org/`

[NNZ00] Nickel, U.; Niere, J.; Zundorf, A.: The FUJABA environment. In: *Proc. of 22nd International Conference on Software Engineering (ICSE-22)*, 2000, S. 742–745

[NP90] Nosek, J. T.; Palvia, P.: Software maintenance management: changes in the last decade. In: *Journal of Software Maintenance* 2 (1990), Nr. 3, S. 157–174

[NW00] Noble, J.; Weir, C.: *Small Memory Software: Patterns for Systems with Limited Memory*. Addison-Wesley, 2000

[OAS04] OASIS: Business Transaction Protocol (BTP), Version 1.0.9.1. 2004, URL `http://www.oasis-open.org/committees/document.php?document_id=6634`

[ÖB03] Österle, H.; Blessing, D.: Business Engineering Model. In: Österle und Winter [ÖW03b], S. 65–86

[ÖBH92] Österle, H.; Brenner, W.; Hilbers, K.: *Unternehmensführung und Informationssystem – Der Ansatz des St. Galler Informationssystem-Managements*. Informatik und Unternehmensführung, Stuttgart: Teubner, 1992

[OMGa] OMG: *Meta Object Facilitiy (MOF) Specification*. Technischer Bericht, Object Management Group, URL `http://www.omg.org/technology/documents/formal/mof.htm`

[OMGb] OMG: *Model-Driven Architecture*. Technischer Bericht, Object Management Group, URL `http://www.omg.org/mda/`

[OMGc] OMG: *Object Management Architecture – Ressource Page*. Online-ressource, Object Management Group, URL `http://www.omg.org/oma/`

[OMGd] OMG: *XML Metadata Interchange*. Technischer Bericht, Object Management Group, URL `http://www.omg.org/technology/documents/formal/xmi.htm`

[OMG01] OMG: *General Ledger*. OMG Specification, Version 1.0 01-02-67, Object Management Group, 2001

[OMG02] OMG: *Request for Proposals: MOF 2.0 Query / Views / Transformations*. Technischer Bericht, Object Management Group, 2002, URL `http://www.omg.org/`

[OMG03a] OMG: UML Profile for Schedulability, Performance and Time. 2003, URL `http://www.omg.org/`

[OMG03b] OMG: Unified Modeling Language: Superstructure – version 2.0. August 2003, URL `http://www.omg.org/uml/`

[OMG03c] OMG: The Unified Modeling Language, Version 1.5. 2003, URL `http://www.omg.org/`

[OMG04] OMG: *Request for Proposals: MOF Model to Text Transformation Language*. Technischer Bericht, Object Management Group, 2004, URL `http://www.omg.org/docs/ad/04-04-07.pdf`

[Opd92] Opdyke, W.: *Refactoring Object-Oriented Frameworks*. Technischer Bericht UIUCDCS-R 92-1759, Univ. of Illinois at Urbana-Champaign, Dept. of Computer Science, 1992

[Oqu04] Oquendo, F.: π-ADL: an Architecture Description Language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures. In: *Software Engineering Notes* 29 (2004), Nr. 3, S. 1–14

[Ora] Oracle Corporation: OCI. URL `http://otn.oracle.com/tech/oci`

[Ora04] Oracle Corporation: *interMedia*. Technischer Bericht, 2004, URL `http://www.oracle.com/technology/products/intermedia/index.html`

[Ora05] Oracle Corporation: *interMedia Documentation*. Technischer Bericht, 2005, URL `http://www.oracle.com/technology/documentation/intermedia.html`

[OSC] Online Services Computer Interface. URL `http://www.osci.de`

[Öst95] Österle, H.: *Business Engineering: Prozess- und Systementwicklung – Entwurfstechniken*, Bd. 1. Berlin et al.: Springer-Verlag, 2. Aufl., 1995

[OT89] Ott, L. M.; Thuss, J. J.: The relationship between slices and module cohesion. In: *Proceedings of the 11th international conference on Software engineering*, ACM Press, 1989, S. 198–204

[Ous94] Ousterhout, J.: *Tcl and the Tk Toolkit*. Addison-Wesley, 1994

[ÖW03a] Österle, H.; Winter, R.: Business Engineering. In: *Business Engineering* [ÖW03b], S. 3–20

[ÖW03b] Österle, H.; Winter, R. (Hrsg.): *Business Engineering – Auf dem Weg zum Unternehmen des Informationszeitalters*. Business Engineering, Springer-Verlag, 2. Aufl., 2003

[Pad02a] Padberg, J.: Basic Ideas for Transformations of Specification Architectures. In: Heckel, R.; Mens, T.; M., W. (Hrsg.), *Proc. Workshop on Software Evolution through Transformations (SET 02), Satellite Event of ICGT'02*, Oktober 2002, Bd. 72 von *Electronic Notes in Theoretical Computer Science (ENTCS)*

[Pad02b] Padberg, J.: Petri Net Modules. In: *Journal on Integrated Design and Process Technology* 6 (2002), Nr. 4, S. 121–137

[PAMA00] Petriu, D.; Amer, H.; Majumdar, S.; Abdull-Fatah, I.: Using Analytic Models for Predicting Middleware Performance. In:

Proceedings of the Second International Workshop on Software and Performance (WOSP), 2000

[Par72] Parnas, D. L.: On the Criteria To Be Used in Decomposing Systems into Modules. In: *Communications of the ACM* 15 (1972), Nr. 12, S. 1053–1058

[PAS98] Pree, W.; Althammer, E.; Sikora, H.: Framelets als handliche Architekturbausteine. In: *Softwaretechnik 98*, Paderborn, September 1998

[PB04] Pletschen, W.; Böckmann, F.-J.: Infrastruktur-Management als Erfolgsfaktor. In: Dietrich, L.; Schirra, W. (Hrsg.), *IT im Unternehmen – Leistungssteigerung bei sinkenden Budgets*, Springer-Verlag, Xpert.press, 2004, S. 103–137

[PBG04] Posch, T.; Birken, K.; Gerdom, M.: *Basiswissen Softwarearchitektur – Verstehen, entwerfen, bewerten und dokumentieren*. dpunkt.verlag, 2004

[Pet62] Petri, C. A.: *Kommunikation mit Automaten*. Dissertation, Technische Hochschule Darmstadt, 1962

[Pet95] Petre, M.: Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. In: *Communications of the ACM* 38 (1995), Nr. 6, S. 33–44

[PJC00] de Paula, V.; Justo, G.; Cunha, P.: Specifying and verifying reconfigurable software architectures. In: *International Symposium on Software Engineering for Parallel and Distributed Systems*, 2000, S. 21–31

[PK99] Pree, W.; Koskimies, K.: Framelets – Small is Beautiful. In: Fayad et al. [FSJ99a], S. 411–413

[PK00] Pree, W.; Koskimies, K.: Framelets – small and loosely coupled frameworks. In: *ACM Computing Surveys* 32 (2000), Nr. 1es, S. 6

[PLV97] Posnak, E.; Lavender, R.; Vin, H.: An adaptive framework for developing multimedia software components. In: *Communications of the ACM* 40 (1997), Nr. 10, S. 43–47

[PMSH01] Papadopoulos, Y.; McDermid, J. A.; Sasse, R.; Heiner, G.: Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. In: *Int. J. of Reliability Engineering and System Safety* 71 (2001), Nr. 3, S. 229–247

[PNW] Petri Nets World.
URL `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`

[Poo99] Pooley, R.: Using UML to Derive Stochastic Process Algebra Models. In: *Proceedings of the 25th UK Performance Engineering Workshop*, 1999, S. 23–34

[Pre94] Pree, W.: Meta Patterns — A Means for Capturing the Essentials of Reusable Object-Oriented Design. In: *Lecture Notes in Computer Science* 821 (1994)

[Pre95] Pree, W.: *Design Patterns for Object-Oriented Software Development*. ACM Press Books, Addison-Wesley, 1995

[Pre96a] Pree, W.: *Framework Patterns*. SIGS Books and Multimedia, 1996

[Pre96b] Pree, W.: Frameworks – Past, present, future. In: *Object magazine – improving software quality through object development & reuse* 6 (1996), Nr. 3

[Pre97a] Pree, W.: Component-Based Software Development – A New Paradigm in Software Engineering? In: *Software – Concepts and Tools* 18 (1997), Nr. 4, S. 169–174

[Pre97b] Pree, W.: Essential Framework Design Patterns. In: *Object magazine – improving software quality through object development & reuse* 7 (1997), Nr. 1

[Pre97c] Pree, W.: *Komponentenbasierte Softwareentwicklung mit Frameworks*. dpunkt.verlag, 1997

[Pre97d] Pree, W.: Object-Oriented Design Patterns and Hot Spot Cards. In: *IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS97)*, Como, Italy, September 1997

[Pre99] Pree, W.: Hot-Spot-Driven Framework Development. In: Fayad et al. [FSJ99a], S. 379–393

[PS02] Petriu, D. C.; Shen, H.: Applying the UML Performance Profile: Graph Grammar-Based Derivation of LQN Models from UML Specifications. In: *TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, London, UK: Springer-Verlag, 2002, S. 159–177

[Pum99] Pumfrey, D.: *The Principled Design of Computer System Safety Analyses*. Dissertation, Department of Computer Science, University of York, 1999

[PVL96] Posnak, E.; Vin, H.; Lavender, R.: Presentation Processing Support for Adaptive Multimedia Applications. In: *Proc. of Multimedia Computing and Networking 1996 (MMCN96)*, Januar 1996, S. 234–245

[PW92] Perry, D. E.; Wolf, A. L.: Foundations for the Study of Software Architecture. In: *Software Engineering Notes* 17 (1992), Nr. 4, S. 40–52

[PW02] Petriu, D.; Woodside, C.: Software Performance Models from System Scenarioes in Use Case Maps. In: *Proceedings of the 12th International Conference for Modelling Tools and Techniques for Computer and Comm. System Performance Evaluation*, 2002, S. 141–158

[QFD] QFD: Quality Function Deployment. QM-Lexikon. URL http://www.quality.de/lexikon/qfd.htm

[Qui94] Quilici, A.: A memory-based approach to recognizing programming places. In: *Communications of the ACM* 37 (1994), Nr. 5, S. 84–93

[Ran00] Ran, A.: ARES Architectural Framework for Software Architecture. In: Jazayeri et al. [JRL00], S. 1–29

[RBD+03] Rudd, C.; Bicket, D.; Diamon, P.; Bull, R.; Fröhlich, S.; Best, R.: *Software Asset Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2003

[RBJ97] Roberts, D.; Brant, J.; Johnson, R.: A Refactoring Tool for Smalltalk. In: *Theory and Practice of Object Systems* 3 (1997), S. 253–263

[RBSP02] Riebisch, M.; Böllert, K.; Streitferdt, D.; Phillipow, I.: Extending Feature Diagrams with UML Multiplicities. In: *Proceedings of Integrated Design and Process Technology*, Society for Design and Process Science, June 2002

[RCM99] Rajala, N.; Campara, D.; Mansurov, M.: inSight – Reverse Engineer CASE Tool. In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE99)*, IEEE Computer Society Press / ACM Press, 1999, S. 630–633

[RD03] Riva, C.; Del Rosso, C.: Experiences with Software Product Family Evolution. In: *Proceedings of the International Workshop on Principles of Software Evolution*, IEEE Computer Society Press, September 2003

[RE99] Rosel, A.; Erni, K.: Experiences with the Semantic Graphics Framework. In: Fayad et al. [FSJ99b], Kap. 27, S. 629–657

[Rea05] RealNetworks: *Helix Community*. Technischer Bericht, 2005, URL https://helixcommunity.org/

[Rei86] Reisig, W.: *Petrinetze*. Springer-Verlag, 2. Aufl., 1986

[RHMM04] Roshandel, R.; van der Hoek, A.; Mikic-Rakic, M.; Medvidovic, N.: Mae – A System Model and Environment for Managing Architectural Evolution. In: *ACM Transactions on Software Engineering and Methodology* 13 (2004), Nr. 2, S. 240–276

[RHS05] Richter, J.-P.; Haller, H.; Schrey, P.: Serviceorientierte Architektur — Das aktuelle Schlagwort. In: *Informatik-Spektrum* 28 (2005), Nr. 6

[Ric03] Richter, C.: *Entwurf und Realisierung eines webbasierten personalisierten multimedia Musik-Newsletters*. Individuelles Projekt, Carl von Ossietzky Universität Oldenburg, Juli 2003

[Rie96] Riel, A. J.: *Object-Oriented Design Heuristics*. Addison-Wesley, 1996

[Rie00] Riehle, D.: *Framework Design: A Role Modeling Approach*. Dissertation, Swiss Federal Institute of Technology Zurich, 2000

[Ris00] Rising, L.: *Pattern Almanac 2000*. Addison-Wesley, 2000

[RJ97] Roberts, D.; Johnson, R.: Evolving Frameworks – A Pattern Language for Developing Object-Oriented Frameworks. In: *Pattern Languages of Program Design 3*, Illinois, USA: Addison-Wesley, 1997

[RJB98] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The Unified Modeling Language Reference Manual*. Addison Wesley, Dezember 1998

[RL04] Roock, S.; Lippert, M.: *Refactorings in großen Softwareprojekten*. dpunkt.verlag, 2004

[Rog97] Rogers, G. F.: *Framework-Based Software Development in C++*. Prentice Hall Series on Programming Tools and Methodologies, Prentice Hall, 1997

[RR03] Ravichandran, T.; Rothenberger, M.: Software reuse strategies and component markets. In: *Communications of the ACM* 46 (2003), Nr. 8, S. 109–114

[RS95] Rolia, J.; Sevcik, K.: The Method of Layers. In: *IEEE Transactions on Software Engineering* 21 (1995), Nr. 8, S. 682–688

[RS99] Rout, T.; Sherwood, C.: Software Engineering Standards and the Development of Multimedia-Based Systems. In: *In Fourth IEEE International Symposium and Forum on Software Engineering Standards*, Mai 1999

[RS02] Rumpe, B.; Schröder, A.: Quantitative Survey on Extreme Programming Projects. In: Wells und Williams [WW02], S. 43–46

[RSP04] Riebisch, M.; Streitferdt, D.; Pashov, I.: Modeling Variability for Object-Oriented Product Lines. In: Buschmann, F.; Buchmann, A. P.; Cilia, M. (Hrsg.), *Object-Oriented Technology. ECOOP 2003 Workshop Reader*, Springer-Verlag, 2004, Bd. 3013 von *Lecture Notes in Computer Science*, S. 165–178

[Rue02] Rueegg-Stuerm, J.: *Das neue St. Galler Management-Modell: Grundkategorien einer integrierten Managementlehre: der HSG-Ansatz*. Bern: Haupt, 2002

[Rum96] Rumpe, B.: *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, 1996

[Rum04] Rumpe, B.: *Modellierung mit UML – Sprache, Konzepte und Methodik*. Xpert.press, Springer-Verlag, 2004

[Rum05] Rumpe, B.: *Agile Modellierung mit UML – Codegenerierung, Testfälle, Refactoring*. Xpert.press, Springer-Verlag, 2005

[RWP04] Ross, K.; Westermann, G. U.; Popitsch, N.: METIS – A Flexible Database Solution for the Management of Multimedia Assets. In: *Proc. of the 10th International Workshop on Multimedia Information Systems (MIS 2004)*, August 2004

[Saw95] Sawhney, M.: *Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystems für eine Universität*.

Diplomarbeit, Universität Osnabrück, Fachbereich Wirtschaftswissenschaften, Osnabrück, Juni 1995

[SB99] van Solingen, R.; Berghout, E.: *The Goal/Question/Metric Method, A Practical Method for Quality Improvement of Software Development*. McGraw-Hill, 1999

[SB04] Scherp, A.; Boll, S.: Generic support for personalized mobile multimedia tourist applications. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia – Technische Demonstration*, New York, NY, USA: ACM Press, Oktober 2004, S. 178–179

[SB05a] Scherp, A.; Boll, S.: A lightweight process model and development methodology for component frameworks. In: *Proceedings of the tenth International Workshop on Component-Oriented Programming*, Juli 2005

[SB05b] Scherp, A.; Boll, S.: MM4U – A framework for creating personalized multimedia content. In: Srinivasan, U.; Nepal, S. (Hrsg.), *Managing Multimedia Semantics*, Hershey, PA, USA: IRM Press, Kap. 11, 2005

[SB05c] Scherp, A.; Boll, S.: Paving the Last Mile for Multi-Channel Multimedia Presentation Generation. In: Chen, Y.-P. P. (Hrsg.), *Procedings of the 11th Multimedia Modeling (MMM) Conference*, Melbourne, Australia: IEEE Computer Society, Januar 2005, S. 190–197

[Sch97] Scheer, A.-W.: *Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse*. Springer-Verlag, 2. Aufl., 1997

[Sch99a] Schmid, H. A.: Framework Design by Systematic Generalization. In: Fayad et al. [FSJ99a], Kap. 15, S. 353–378

[Sch99b] Schwanitz, D.: *Alles was man wissen muss*. Eichborn Verlag, 1999

[Sch01a] Scholl: Napster Messages. April 2001, URL `http://opennap.sourceforge.net/napster.txt`

[Sch01b] Schollmeier, R.: A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In: *First International Conference on Peer-to-Peer Computing (P2P'01)*, Linköping, Schweden, August 2001, S. 101–102

[Sch04] Schott, A.: Architekturzentrierte Software-Entwicklung – elitäre Technik-Disziplin oder ökonomische Notwendigkeit? In: Dadam, P.; Reichert, M. (Hrsg.), *INFORMATIK 2004 – Informatik verbindet, Bd. 2*, Gesellschaft für Informatik, 2004, Bd. 51 von *GI-Edition Lecture Notes in Informatics*, S. 116–120

[SEH03] Sim, S. E.; Easterbrook, S.; Holt, R. C.: Using benchmarking to advance research: a challenge to software engineering. In: *ICSE '03: Proceedings of the 25th International Conference on Software*

Engineering, Washington, DC, USA: IEEE Computer Society, 2003, S. 74–83

[SEI05] SEI: *What are the Duties of a Chief Software Architect?* Technischer Bericht, Software Engineering Institute, Carnegie Mellon University, 2005, URL `http://www.sei.cmu.edu/architecture/`

[SES02] Stroulia, E.; El-Ramly, M.; Sorenson, P.: From Legacy to Web through Interaction Modeling. In: *Proceedings of the International Conference on Software Maintenance (ICSM '02)*, Montreal, Kanada: IEEE Press, Oktober 2002, S. 320–329

[SG95] Shaw, M.; Garlan, D.: Formulations and Formalisms in Software Architecture. In: van Leeuwen, J. (Hrsg.), *Computer Science Today: Recent Trends and Developments*, Springer-Verlag, Bd. 1000 von *Lecture Notes in Computer Science*, 1995, S. 307–323

[SG96] Shaw, M.; Garlan, D.: *Software Architecture – Perspectives on an Emerging Discipline*. An Alan R. Apt book, Prentice Hall, 1996

[SGM02] Szyperski, C.; Gruntz, D.; Murer, S.: *Component Software – Beyond Object-Oriented Programming*. Addison-Wesley Component Software Series, Addison-Wesley, 2. Aufl., 2002

[SH99] Scheer, A.-W.; Hoffmann, M.: From Business Process Model to Application System – Developing an Information System with the House of Business Engineering (HOBE). In: *Advanced Information Systems Engineering, 11th International Conference CAiSE 1999, Heidelberg, Germany*, Springer-Verlag, 1999, Bd. 1626 von *Lecture Notes in Computer Science*, S. 2–9

[Shi00] Shirky, C.: What Is P2P... And What Isn't? 2000, URL `http://www.openp2p.com/`

[Sie04] Siedersleben, J.: *Moderne Softwarearchitektur – Umsichtig planen, robust bauen mit Quasar*. dpunkt.verlag, 2004

[Sih01] Sihling, M.: *Methodische Entwicklung und rollenbasierte Integration von Komponentenframeworks*. Dissertation, Technische Universität München – Institut für Informatik, 2001

[SM95] Storey, M.-A. D.; Muller, H. A.: Manipulating and documenting software structures using SHriMP views. In: *International Conference on Software Maintenance*, IEEE Computer Society Press, Oktober 1995, S. 275–284

[Smi90] Smith, C. U.: *Performance Engineering of Software Systems*. Addison-Wesley, 1990

[Smi02] Smith, C. U.: *Performance Solutions: A Practical Guide To Creating Responsive, Scalable Software*. Addison-Wesley, 2002

[SMK+01] Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. F.; Balakrishnan, H.: Chord – A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proceedings of the 2001 Conference on*

Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM Press, 2001, S. 149–160

[SMM02] Shokoufandeh, A.; Mancoridis, S.; Maycock, M.: Applying Spectral Methods to Software Clustering. In: van Deursen [Deu02], S. 3–12

[SN95] Steinmetz, R.; Nahrstedt, K.: *Multimedia: Computing, Communications and Applications*. Prentice Hall, 1995

[Sof04] Bericht: Software-Projekt für Finanzämter gescheitert. Juli 2004, URL `http://www.heise.de/newsticker/meldung/48843`

[Som04] Sommerville, I.: *Software Engineering*. Addison-Wesley, 7. Aufl., 2004

[SPL03] Seacord, R. C.; Plakosh, D.; Lewis, G. A.: *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Addison-Wesley, 2003

[SS04] Streckenbach, M.; Snelting, G.: Refactoring Class Hierarchies with KABA. In: *Proceedings of OOPSLA 04*, 2004

[SS05a] Schelp, J.; Schwinn, A.: Extending the Business Engineering Framework for Application Integration Purposes. In: *Applied Computing – Proceedings of the 2005 ACM Symposium on Applied Computing*, New York, NY, USA: ACM Press, 2005, Bd. 2, S. 1333–1337

[SS05b] Schwinn, A.; Schelp, J.: Design Patterns for data integration. In: *Journal of Enterprise Information Management* 18 (2005), Nr. 4, S. 471–482

[SSL01] Simon, F.; Steinbückner, F.; Lewerentz, C.: Metrics Based Refactoring. In: *Proceedings of European Conference Software Maintenance and Reengineering*, 2001, S. 157–169

[SSN02] Sharma, R.; Stearns, B.; Ng, T.: *J2EE Connector Architecture and Enterprise Application Integration*. Addison-Wesley, 2002

[SSRB00] Schmidt, D.; Stal, M.; Rohnert, H.; Buschmann, F.: *Patterns for Concurrent and Networked Objects (Pattern-oriented Software Architecture, vol. 2)*. Wiley series in software design patterns, John Wiley & Sons, 2000

[Sta73] Stachowiak, H.: *Allgemeine Modelltheorie*. Wien: Springer-Verlag, 1973

[Sta03] Starke, G.: Architektur und Flexibilität – Ein Widerspruch? In: *IT FOKUS* o. A. (2003), Nr. 3, S. 22–26

[Ste00] Steinmetz, R.: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, 3. Aufl., 2000

[Sun] Sun Java Center J2EE Patterns. URL `http://java.sun.com/blueprints/patterns/`

[Sun04] Sun Microsystems: EJB 2.1 Specification Final Release 2.1. 2004, URL `http://java.sun.com/products/ejb/index.jsp`

[Sun05a] Sun Microsystems: *Java Foundation Classes (JFC/Swing)*. Technischer Bericht, 1994–2005, URL `http://java.sun.com/products/jfc/`

[Sun05b] Sun Microsystems: *Java Media Framework API (JMF)*. Technischer Bericht, 1994–2005, URL `http://java.sun.com/products/java-media/jmf/`

[Sun05c] Sun Microsystems: *JMF 2.0 Documentation Downloads*. Technischer Bericht, 1994–2005, URL `http://java.sun.com/products/java-media/jmf/`

[Sun05d] Sun Microsystems: JXTA v2.3.x: Java Programmer's Guide. Januar 2005

[SW04] Steinmetz, R.; Wehrle, K.: Peer-to-Peer-Networking und -Computing. In: *Informatik-Spektrum* 27 (2004), Nr. 1, S. 51–54

[SWSS03] Schuler, C.; Weber, R.; Schuldt, H.; Schek, H.-J.: Peer-to-Peer Process Execution with Osiris. In: Orlowska, M. E.; Weerawarana, S.; Papazoglou, M. P.; Yang, J. (Hrsg.), *Service-Oriented Computing (ICSOC 2003)*, Springer-Verlag, 2003, Bd. 2910 von *Lecture Notes in Computer Science*, S. 483–498

[SWZ99] Schürr, A.; Winter, A.; Zündorf, A.: The PROGRES-Approach: Language and Environment. In: Ehrig, H.; Engels, G.; Kreowski, J.-J.; Rozenberg, G. (Hrsg.), *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages and Tools*, World Scientific, 1999

[Tab00] Tabeling, P.: *Der Modellhierarchieansatz zur Beschreibung nebenläufiger, verteilter und transaktionsverarbeitender Systeme*. Berichte aus der Informatik, Shaker Verlag, 2000

[Tab01] Tabeling, P.: Wissensorientierte Beschreibung großer Software-Systeme – ein Ansatz jenseits softwareorientierter Konzepte. In: *Knowtech Konferenzband*, Dresden: Knowtech Conference, 2001

[Tab02] Tabeling, P.: Ein Metamodell zur architekturorientierten Beschreibung komplexer Systeme. In: *Proceedings of Modellierung 2002, GI-Lecture Notes in Informatics, Tutzing*, 2002

[Tab04a] Tabeling, P.: Architectural Description with Integrated Data Consistency Models. In: *Proceedings of the IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2004

[Tab04b] Tabeling, P.: Eine Entwicklungsplattform für die architekturorientierte Programmierung. In: *Lecture Notes in Informatics (LNI) – Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen*, 2004

[Tab05] Tabeling, P.: *Software-Systeme und ihre Modellierung – Grundlagen, Methoden und Techniken*. Springer-Verlag, 2005

[Tau03] Taubner, D.: Effizientes Software-Engineering: Vorgehen für wirtschaftliche Projekte. In: *IM – Die Fachzeitschrift für Information Management & Consulting* (2003), Nr. 18, S. 14–18

[TB01] Tokuda, L.; Batory, D.: Evolving Object-Oriented Designs with Refactorings. In: *Journal of Automated Software Engineering* 8 (2001), S. 89–120

[TDDN00] Tichelaar, S.; Ducasse, S.; Demeyer, S.; Nierstrasz, O.: A Meta-model for Language-Independent Refactoring. In: *Proceedings ISPSE*, 2000, IEEE Computer Society Press

[TDDN01] Tichelaar, S.; Ducasse, S.; Demeyer, S.; Nierstrasz, O.: Refactoring UML models. In: *Proceedings of UML 01*, Springer-Verlag, 2001, Nr. 2185 in Lecture Notes in Computer Science

[TG03] Tabeling, P.; Gröne, B.: Mappings between Object-oriented Technology and Architecture-based models. In: Al-Ani, B.; Arabnia, H. R.; Mun, Y. (Hrsg.), *Proceedings of the SERP'03, the international conference on software engineering research and practice, Las Vegas*, CSREA Press, Juni 2003, Bd. II, S. 568–574

[TH00] Tzerpos, V.; Holt, R. C.: On the Stability of Software Clustering Algorithms. In: *International Workshop on Program Comprehension*, IEEE Computer Society Press, 2000

[TJK+04] Teschke, T.; Jaekel, H.; Krieghoff, S.; Langnickel, M.; Hasselbring, W.; Reussner, R.: Funktionsgetriebene Integration von Legacy-Systemen mit Web Services. In: Hasselbring, W.; Reichert, M. (Hrsg.), *Proc. Workshop Enterprise Application Integration (EAI 2004)*, Berlin: GITO Verlag, Februar 2004, S. 19–28

[TK01] Tolvanen, J.-P.; Kelly, S.: Domain-Specific Modeling: 10 times faster than UML. In: *Proceedings of Embedded Systems Conference, Stuttgart, Germany*, 2001

[TK02] Tin, R.; Keller, R. K.: Program comprehension by visualization in contexts. In: *Proceedings of the International Conference on Software Maintenance*, IEEE Computer Society Press, 2002, S. 332–341

[TM03] Tourwé, T.; Mens, T.: Identifying Refactoring Opportunities Using Logic Meta Programming. In: *Seventh European Conference on Software Maintenance and Reengineering (CSMR'03)*, 2003

[TMQ+03] Trowbridge, D.; Mancini, D.; Quick, D.; Hohpe, G.; Newkirk, J.; Lavigne, D.: *Enterprise Solution Patterns Using Microsoft .NET: Version 2.0 – Patterns & Practices*. Microsoft Press, 2003

[TOGa] TOGAF: *TOGAF 8 Documentation*. Technischer Bericht, The Open Group, URL `http://www.opengroup.org/architecture/togaf8-doc/arch/`

[Togb] Together:

URL http://www.borland.com/together/

[Tro05] Trolltech: *Trolltech – Qt Product Overview*. Technischer Bericht, 2005,
URL http://www.trolltech.com/products/qt/

[TX00] Tsai, J. J. P.; Xu, K.: A comparative study of formal verification techniques for software architecture specifications. In: *Annals of Software Engineering* 10 (2000), Nr. 1–4, S. 207–223

[van04] *The Cuypers Multimedia Transformation Engine*. Technischer Bericht, CWI, 2004,
URL http://media.cwi.nl:8080/demo/

[VDI03] VDI: VDI-Richtlinie 3633. 2003,
URL http://www.vdi.de/vdi/vrp/richtliniendetails/index.php?ID=9509528

[VGRH96] Vesely, W. E.; Goldberg, F. F.; Roberts, N. H.; Haasl, D. F.: Fault Tree Handbook. U. S. Nuclear Regulatory Commission. 1996

[vHW03] van der Aalst, W. M.; Hofstede, A. H. t.; Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.; ter Hofstede, A.; Weske, M. (Hrsg.), *Business Process Management: International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, Springer-Verlag, 2003, Bd. 2678, S. 1–12

[Vit03] Vitharana, P.: Risks and challenges of component-based software development. In: *Communications of the ACM* 46 (2003), Nr. 8, S. 67–72

[VSW02] Völter, M.; Schmid, A.; Wolff, E.: *Server Component Patterns: Component Infrastructures illustrated with EJB*. John Wiley & Sons, 2002

[W3C02] W3C: Web Services Architecture. 2002,
URL http://www.w3.org/

[W3C04] Scalable Vector Graphics (SVG) 1.2. 2004
URL http://www.w3.org/

[W3C05] Synchronized Multimedia Integration Language (SMIL 2.0), 2005,
URL http://www.w3.org/

[WAGL99] Wedekind, H.; Albrecht, J.; Günzel, H.; Lehner, W.: Repositories for Data Warehouse Systems in a Middleware Environment. In: *Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis, ISAS'99, Orlando (FL)*, 1999, S. 298–305

[WBF97] Wiggerts, T.; Bosma, H.; Fielt, E.: Scenarios for the Identification of Objects in Legacy Systems. In: Baxter, I. D.; Quilici, A.; Verhoef, C. (Hrsg.), *Proceedings of the 4th Working Conference on Reverse Engineering (WCRE '97)*, Amsterdam, Niederlande: IEEE Computer Society Press, Oktober 1997, S. 24–32

[WBS04] Wurz, M.; Brettlecker, G.; Schuldt, H.: Data Stream Management and Digital Library Processes on Top of a Hyperdatabase and Grid Infrastructure. In: Agosti, M.; Schek, H.-J.; Türker, C. (Hrsg.), *DELOS Workshop: Digital Library Architectures*, 2004, S. 37–48

[WC01] Westfechtel, B.; Conradi, R.: Software Architecture and Software Configuration Management. In: *SCM*, Springer-Verlag, 2001, Bd. 2649, S. 24–39

[WE02] Weber, M.; Eisenführ, F.: *Rationales Entscheiden*. Springer-Verlag, 2002

[Wei84] Weiser, M.: Program Slicing. In: *IEEE Transactions on Software Engineering* 10 (1984), Nr. 4, S. 352–357

[Wen80] Wendt, S.: Modified Petri Nets as Flowcharts for Recursive Programs. In: *Software – Practice and Experience* 10 (1980), S. 935–942

[Wen82a] Wendt, S.: Der Kommunikationsansatz in der Software-Technik. In: *Data Report* 17 (1982), Nr. 4

[Wen82b] Wendt, S.: Einführung in die Begriffswelt allgemeiner Netzsysteme. In: *Regelungstechnik* 30 (1982), Nr. 1

[Wen91] Wendt, S.: *Nichtphysikalische Grundlagen der Informationstechnik – Interpretierte Formalismen*. Springer-Verlag, 2. Aufl., 1991

[WF98] Wijegunaratne, I.; Fernandez, G.: *Distributed Applications Engineering – Building new applications and managing legacy applications with distributed technologies*. Practitioner series, Springer-Verlag, 1998

[WF99] Wermelinger, M.; Fiadeiro, J. L.: Algebraic Software Architecture Reconfiguration. In: *Proceedings ESEC/FSE'99*, Springer-Verlag, 1999, Bd. 1687 von *Lecture Notes in Computer Science*, S. 393–409

[WF00] Wermelinger, M.; Fiadeiro, J. L.: Graph Transformation Approach to Software Architecture Reconfiguration. In: *Proc. of the Joint APPLIGRAPH and GETGRATS Workshop on Graph Transformation Systems*, 2000, S. 134–141

[WF04] Wang, G.; Fung, C. K.: Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems. In: *Proc. 37th Hawaii International Conference on Systems Sciences (HICSS 2004)*, IEEE Computer Society Press, 2004, S. 90272a

[WHH94] Weide, B. W.; Heym, W. D.; Hollingsworth, J. E.: *Reverse Engineering of Legacy Code is Intractable*. Technischer Bericht OSU-CISRC-10/94-TR55, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio, Oktober 1994

[Wil96] Wills, L. M.: Using Attributed Flow Graph Parsing to Recognize Clichés in Programs. In: Cuny, J. E.; Ehrig, H.; Engels, G.;

Rozenberg, G. (Hrsg.), *Proc. 5th Int. Workshop on Graph Grammars and their Application to Computer Science*, Springer-Verlag, 1996, Bd. 1073 von *Lecture Notes in Computer Science*, S. 170–184

[Win03a] Winter, R.: An Architecture Model for Supporting Application Integration Decisions. In: *Proc. 11th European Conference on Information Systems*, 2003

[Win03b] Winter, R.: Modelle, Techniken und Werkzeuge im Business Engineering. In: Österle und Winter [ÖW03b], S. 87–118

[WJ90] Wirfs-Brock, R. J.; Johnson, R. E.: Surveying current research in object-oriented design. In: *Communications of the ACM* 33 (1990), Nr. 9, S. 104–124

[WL99] Weiss, D. M.; Lai, C. T. R.: *Software Product-Line Engineering – A Family-Based Sorftware Development Process*. Addison-Wesley, 1999

[WLF01] Wermelinger, M.; Lopes, A.; Fiadeiro, J. L.: A graph based architectural (Re)configuration language. In: *Software Engineering Notes* 26 (2001), Nr. 5, S. 21–32

[WM81] Wedekind, H.; Müller, T.: Stücklistenorganisation bei einer großen Variantenzahl. In: *Angewandte Informatik* 23 (1981), Nr. 9, S. 377–383

[WMSR00] Walker, R. J.; Murphy, G. C.; Steinbok, J.; Robillard, M. P.: Efficient mapping of software system traces to architectural views. In: *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, IBM Press, 2000, S. 12

[Woo03] Woods, D.: *Enterprise Services Architecture*. O'Reilly & Associates, 2003

[WS98a] Watts, D. J.; Strogatz, S. H.: Collective dynamics of small-world networks. In: *Nature* 393 (1998), Nr. 6684, S. 440–442

[WS98b] Williams, L. G.; Smith, C. U.: Performance evaluation of software architectures. In: *WOSP '98: Proceedings of the first international workshop on Software and performance*, New York, NY, USA: ACM Press, 1998, S. 164–177

[WW02] Wells, D.; Williams, L. A. (Hrsg.): *Extreme Programming and Agile Methods*, Nr. 2418 in Lecture Notes in Computer Science, Springer-Verlag, 2002

[WY96] Woods, S.; Yang, Q.: The program understanding problem: Analysis and a heuristic approach. In: *Proceedings of the 18th International Conference on Software Engineering*, IEEE Computer Society Press, 1996, S. 6–15

[YC79] Yourdon, E.; Constantine, L. L.: *Structured Design – Fundamentals of a Discipline of Computer Program and System Design*. Prentice Hall, 1979

[Z] The Z notation.
URL http://vl.zuser.org/

[Zac87] Zachman, J. A.: A Framework for Information Systems Architecture. In: *IBM Systems Journal* 26 (1987), Nr. 3, S. 276–292

[ZB03] Zarnekow, R.; Brenner, W.: Auf dem Weg zu einem produkt- und dienstleistungsorientierten IT-Management. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 7–16

[ZG04] Zhang, X.; Gupta, R.: Cost effective dynamic program slicing. In: *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation*, ACM Press, 2004, S. 94–106

[ZLX04] Zhou, Y.; Lu, J.; Xu, H. L. B.: A comparative study of graph theory-based class cohesion measures. In: *Software Engineering Notes* 29 (2004), Nr. 2, S. 13

[Zül05] Züllighoven, H. (Hrsg.): *Object-Oriented Construction Handbook – Developing Application-Oriented Software with the Tools and Materials Approach*. Morgan Kaufman Publ./dpunkt.verlag, 2005

[ZWDZ04] Zimmermann, T.; Weisgerber, P.; Diehl, S.; Zeller, A.: Mining Version Histories to Guide Software Changes. In: *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society Press, 2004, S. 563–572

[ZXZY02] Zhou, Y.; Xu, B.; Zhao, J.; Yang, H.: ICBMC: an improved cohesion measure for classes. In: *Proceedings International Conference on Software Maintenance*, IEEE Computer Society Press, 2002, S. 44–53

[Zyl02] van Zyl, J.: A perspective on service based architecture: the evolutionary concept that assists technology providers in dealing with a changing environment. In: *SAICSIT '02: Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2002, S. 249

[ZYXX02] Zhao, J.; Yang, H.; Xiang, L.; Xu, B.: Change impact analysis to support architectural evolution. In: *Journal of software maintenance and evolution – research and practice* 14 (2002), Nr. 5, S. 317–333