19 Framework-Entwurf

Ansgar Scherp, Susanne Boll

Die Wiederverwendung von Software-Architekturen ist einer der wichtigsten Faktoren im Software-Entwicklungsprozess, um Entwicklungsaufwand und deren Kosten zu reduzieren und gleichzeitig die Zuverlässigkeit und Qualität von Software zu steigern [BR91, BR90, SGM02].

Definition (Software-Framework) Ein Software-Framework stellt typischerweise ein halbfertiges Architekturgerüst für einen (komplexen) Anwendungsbereich dar, das auf die Bedürfnisse und Anforderungen einer konkreten Anwendung aus diesem Anwendungsbereich angepasst werden kann.

Die Motivation, die zur Konzeption und Erstellung eines Frameworks führt, ist oft ein ausgeprägtes Domänenwissen. Die langjährigen Erfahrungen bei der Entwicklung von Software-Anwendungen in einer bestimmten Domäne werden mithilfe eines Frameworks in ein einheitliches Architekturgerüst gegossen. Frameworks erweisen sich gerade dann als nützlich, wenn flexible und dennoch einfach wiederzuverwendende Software-Architekturen zu entwickeln sind [Pre97c], und stellen einen erprobten Ansatz zur Wiederverwendung von Software-Architekturen im Großen dar [DW99b]. Der Aufwand zur Entwicklung von Frameworks ist signifikant höher als bei der konventionellen Anwendungsentwicklung. Daher stellen sie eine langfristige Investition dar, die sich erst dann auszahlt, wenn das Framework in mehreren Anwendungen eingesetzt wird [Pre97a].

In Abschnitt 19.1 werden zunächst die generellen Eigenschaften von Frameworks vorgestellt. Darauf basierend wird in Abschnitt 19.2 der Begriff des Frameworks geklärt. Dabei wird zwischen objektorientierten und komponentenbasierten Frameworks unterschieden. Zudem wird das Konzept des Frameworks gegen verwandte Konzepte abgegrenzt. Frameworks entstehen nicht von heute auf morgen, sondern sind das Ergebnis eines längeren Entwicklungsprozesses. In Abschnitt 19.3 wird daher auf den Entwicklungsprozess von Frameworks eingegangen. In Abschnitt 19.4 wird der Entwurf von objektorientierten Frameworks beschrieben. Dabei wird als Beispiel der Hotspot-getriebene Entwurf im Detail vorgestellt sowie kurz auf andere Entwurfsmethoden eingegangen. Abschließend

wird in Abschnitt 19.5 der Entwurf von Komponenten-Frameworks betrachtet, bevor das Kapitel mit einem Fazit schließt.

19.1 Eigenschaften von Frameworks

Frameworks zeichnen sich typischerweise durch folgende drei Eigenschaften aus: die Umkehrung des Kontrollflusses, die Vorgabe einer konkreten Anwendungsarchitektur und ihre Anpassbarkeit durch vordefinierte Variationspunkte. Diese Eigenschaften werden im Folgenden kurz diskutiert.

19.1.1 Umkehrung des Kontrollflusses

Konventionell entwickelte Anwendungen nutzen typischerweise eine Reihe von Klassenbibliotheken, um die eigenen Funktionalitäten möglichst einfach zu realisieren. Der Kontrollfluss solcher Anwendungen, also die Reihenfolge, in der die Operationen ausgeführt werden [BD00], wird typischerweise von ihnen selbst gesteuert. Die Klassenbibliotheken werden lediglich von der sie nutzenden Anwendung aufgerufen, übernehmen jedoch niemals den Hauptkontrollfluss der Anwendung. Dieser Aufruf externer Klassenbibliotheken wird Call-down-Prinzip genannt und ist in Abbildung 19.1(a) dargestellt. Der obere Teil der Grafik symbolisiert die Anwendung und der untere Bereich die verschiedenen Funktionalitäten, die von externen Klassenbibliotheken zur Verfügung gestellt werden. Immer, wenn die Anwendung Funktionalitäten einer Klassenbibliothek benötigt, ruft sie die dort zur Verfügung gestellten Funktionalitäten auf. Klassenbibliotheken liefern keine Anwendungsstruktur mit, d. h., die Anwendungsentwickler müssen die Architektur ihrer Anwendung selbst entwerfen.

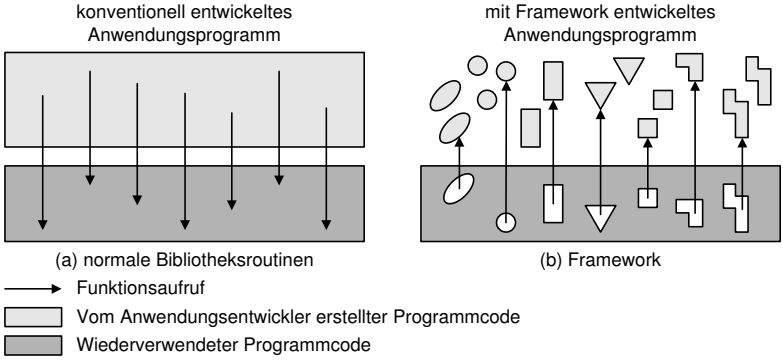

Abbildung 19.1: Darstellung des (a) Call-down-Prinzips und (b) Call-back-Prinzips (nach [Ack96, FPR02])

Frameworks invertieren den Kontrollfluss einer Anwendung [Has02]. Im Gegensatz zu konventionell entwickelten Anwendungen wird der Hauptkontrollfluss einer Anwendung, die mithilfe eines Frameworks entwickelt wurde, vom Framework gesteuert. Dieser Aufrufmechanismus wird Call-back- oder auch Hollywood-Prinzip (»Don't call us, we'll call you!«) genannt und ist in Abbildung 19.1(b) dargestellt. Zentrale Bestandteile der Anwendungsarchitektur werden bereits vom Framework vorgegeben. Die anwendungsspezifischen Funktionalitäten einer Anwendung, die das Framework nutzt, werden aus dem Framework heraus aufgerufen. So hat das Framework den Hauptkontrollfluss über die Anwendung inne [JF88].

19.1.2 Vorgabe einer konkreten Anwendungsarchitektur

Gute Frameworks definieren bereits einen Großteil der Anwendungsarchitektur [Pre97c] und halten sie nur noch an bestimmten Punkten flexibel. Das sind die Stellen, an denen anwendungsspezifische Funktionalitäten integriert und aufgerufen werden (siehe Abschnitt 19.1.3 sowie [FPR02, Pre97c, Ack96]).

Die Invertierung des Kontrollflusses gibt Frameworks die Möglichkeit, als generische, erweiterbare, halbfertige Software-Architekturen (»Skelette«) zu dienen. Die Funktionalitäten, die durch die Anwendungsentwickler hinzugefügt werden, passen den generischen Algorithmus, der in einem Framework definiert und implementiert ist, an die Bedürfnisse und Anforderungen einer speziellen Anwendung an [JF88].

19.1.3 Anpassbarkeit durch Variationspunkte

Um für eine Vielzahl von Anwendungen in einer bestimmten Domäne eingesetzt werden zu können, stellen Frameworks flexible Variationspunkte (engl. variation points) zur Verfügung, an denen das Framework an die Anforderungen einer konkreten Anwendung angepasst und erweitert werden kann. Genauer gesagt handelt es sich hierbei um sogenannte offene Variationspunkte, da die Anzahl der konkreten Ausprägungen der Variationspunkte des Frameworks nicht vorgegeben ist (siehe dazu die Diskussion um offene und geschlossene Variationspunkte im Zusammenhang mit Software-Produktlinien in Abschnitt 18.3.2). Offene Variationspunkte werden auch Erweiterungspunkte genannt. Sie sind die Stellen in der Framework-Architektur, an denen eine Entscheidung für eine bestimmte Funktionalität zwar bereits typisiert ist, die letztendliche Ausprägung dieser Funktionalität aber offen gelassen und erst durch die jeweilige konkrete Anwendung bestimmt wird (vgl. [CBB+03]).

19.2 Arten von Frameworks

Neben der in der Einleitung angegebenen Definition, dass ein Framework ein halbfertiges Architekturgerüst für einen komplexen Anwendungsbereich darstellt, finden sich in der Literatur viele weitere und unterschiedliche Definitionen. Im Rahmen dieses Buchkapitels ist eine Differenzierung von Frameworks hinsichtlich der verwendeten Technologie zur Anpassung an die Anforderungen einer konkreten Anwendung von Bedeutung. Dazu sind historisch bedingt die objektorientierten Frameworks und die komponentenbasierten Frameworks zu unterscheiden: Im Falle von objektorientierten Frameworks geschieht die Anpassung durch Vererbung und (partielle) Überschreibung von abstrakten Klassen oder durch unterschiedliche Implementierungen von vordefinierten Schnittstellen. Bei komponentenbasierten Frameworks erfolgt die Anpassung ausschließlich durch Komposition von Komponenten, die die vordefinierten Schnittstellen des Komponenten-Frameworks implementieren. Neben diesen beiden Arten von Frameworks gibt es auch Mischformen von objektorientierten und komponentenbasierten Frameworks, die beide Technologien zur Anpassung verwenden.

Aus den Definitionen in der Literatur geht nicht immer eindeutig hervor, welche Art von Framework adressiert wird. Oftmals ist jedoch implizit eine der beiden Arten gemeint, wie beispielsweise in [GHJV04, Gam92, And04, HK03b, Bal00, Pre95, Rog97]. Diese Framework-Definitionen wurden analysiert und in eine der beiden Framework-Arten einsortiert. Darauf basierend wird in den folgenden beiden Abschnitten 19.2.1 und 19.2.2 die in der Einleitung dieses Buchteils auf Seite 319 angegebene Definition objektorientierter und komponentenbasierter Frameworks hergeleitet. Zudem werden die Unterschiede zwischen objektorientierten und komponentenbasierten Frameworks dargestellt. Abschließend wird in Abschnitt 19.2.3 kurz auf Mischformen objektorientierter und komponentenbasierter Frameworks eingegangen.

19.2.1 Objektorientierte Frameworks

Seit der Einführung des Framework-Konzeptes Ende der 80er-Jahre haben objektorientierte Frameworks große Aufmerksamkeit erhalten. Heute existieren objektorientierte Frameworks für eine Vielzahl von Anwendungsbereichen [Bos00]. Was die Definition objektorientierter Frameworks betrifft, stimmen die meisten Autoren wie beispielsweise [BMM+99, Ack96] darin überein, dass ein objektorientiertes Framework eine wiederverwendbare Software-Architektur darstellt, die sowohl die Wiederverwendung des Entwurfs als auch des Programmcodes einschließt. Von dieser Übereinstimmung abgesehen unterscheiden sich viele Definitionen. So werden zum Beispiel in [Bos00] objektorientierte Frameworks in kleinere Bestandteile zerlegt, und in [Gov99] werden Unter- bzw. Teilarten von objektorientierten Frameworks definiert. Die nach [BMM+99] wahrscheinlich am häufigsten zitierte und im Rahmen dieses Buchkapitels

verwendete Definition ist die von Johnson und Foote [JF88], nach der ein objektorientiertes Framework aus einer Menge von Klassen besteht, die einen abstrakten Lösungsentwurf für eine Familie verwandter Probleme darstellt. Ein objektorientiertes Framework besteht aus einer Menge von konkreten und – insbesondere – abstrakten Klassen [Bal00, WJ90], die ein Software-System in Form einer generischen Anwendung für eine spezielle Domäne bereitstellen [Pre95]. Es handelt sich also um einen unvollständigen Entwurf und eine unvollständige Implementierung für Anwendungen in einer speziellen Domäne bzw. einem Problembereich [Bos00]. Die Variationspunkte objektorientierter Frameworks sind dabei die abstrakten Framework-Klassen. Zahlreiche Autoren folgen dieser Definition, wie zum Beispiel [SGM02, GHJV04, Pre95, BD00]. Das objektorientierte Framework definiert die Struktur der Klassen und Objekte und deren Verantwortlichkeiten. Es legt fest, wie Klassen und Objekte zusammenarbeiten (Kollaborationen) und wie der Kontrollfluss aussieht [Bal00, GHJV04, Pre95]. Ein objektorientiertes Framework gibt die Architektur einer Anwendung vor, damit sich die Anwendungsentwickler auf die Details der Anwendung konzentrieren können [Bal00]. Die Anwendungsentwickler erweitern das objektorientierte Framework um anwendungsspezifische Funktionalitäten durch Definieren konkreter Unterklassen der abstrakten Framework-Klassen [Bos00, Bal00]. Diese selbst definierten Unterklassen werden vom Framework nach dem Call-back-Prinzip aufgerufen (siehe Abschnitt 19.1.1).

Das Offen/Geschlossen-Prinzip Ein objektorientiertes Framework soll dem Offen/Geschlossen-Prinzip (engl. open/close principle) genügen [Mey97]. Das heißt, es muss offen sein in Bezug auf (zukünftige) Erweiterungen und geschlossen hinsichtlich Modifikationen. Offen in Bezug auf (zukünftige) Erweiterungen bedeutet, dass das objektorientierte Framework um neue Funktionalitäten erweitert werden kann, auch um solche, die beim Entwurf des Frameworks noch nicht vorhersehbar waren. Geschlossen hinsichtlich Modifikation bedeutet, dass die abstrakte Anwendungslogik, die vom objektorientierten Framework den konkreten Anwendungen zur Verfügung gestellt wird, nicht durch Bildung konkreter Unterklassen modifiziert werden kann, um ein anderes Programmverhalten zu erzeugen, als von den Framework-Entwicklern vorgesehen war. Das bedeutet, dass die konkreten Unterklassen ausschließlich die abstrakten Methoden der Oberklassen des objektorientierten Frameworks implementieren. Die im Framework bereits implementierten Klassen und Methoden dürfen nicht überschrieben und modifiziert werden.

In der Praxis wird das Offen/Geschlossen-Prinzip nicht immer eingehalten: So findet man in einigen Frameworks, wie zum Beispiel in .NET [MC05c] von Microsoft und der grafischen Benutzungsoberfläche Swing [Sun08c] von Sun Microsystems, eine Defaultimplementierung vor, die von den Anwendungsentwicklern überschrieben wird.

Wiederverwendung von objektorientierten Frameworks Hinsichtlich der Wiederverwendung objektorientierter Frameworks wird zwischen der Whitebox- und Blackbox-Nutzung unterschieden. Wird das Verhalten des objektorientierten Frameworks durch Vererbung von abstrakten Klassen aus der Klassenhierarchie des Frameworks angepasst, indem die abstrakten Methoden der Oberklassen des Frameworks in den konkreten Unterklassen der Anwendung überschrieben werden [JF88], so spricht man von einer Whitebox-Nutzung des Frameworks bzw. von einem Whitebox-Framework [BD00]. Diese Variante zur Nutzung objektorientierter Frameworks hat ihren Namen von der Eigenschaft, dass zur Umsetzung der anwendungsspezifischen Funktionalitäten durch Vererbung ein Einblick in die Implementierung des objektorientierten Frameworks möglich ist. Teile der Implementierung des Whitebox-Frameworks liegen zur Inspektion offen, ohne aber eine Modifikation an der internen Implementierung zu erlauben (vgl. [SGM02], S. 555).

Im Gegensatz zu Whitebox-Frameworks basiert das Anpassungskonzept von Blackbox-Frameworks nicht auf abstrakten Klassen, sondern auf einer Anzahl fertiger, d. h. implementierter Klassen. Änderungen am Verhalten des Frameworks werden durch Komposition dieser Klassen erreicht und nicht durch Vererbung [Pre97c]. Eine Blackbox-Wiederverwendung findet ausschließlich auf Basis von wohldefinierten Schnittstellen der Klassen und deren »vertraglichen« Spezifikationen statt [BD00]. Dazu implementieren die Anwendungsentwickler neue, anwendungsspezifische Funktionalitäten gegen die externen Framework-Schnittstellen. Folglich ist bei der Blackbox-Wiederverwendung die Qualität der Schnittstellenspezifikation von entscheidender Bedeutung [Pre97c].

Durch partielles Öffnen eines Blackbox-Frameworks zur Whitebox-Wiederverwendung können beliebige »Graustufen« gebildet werden [SGM02]. Tatsächlich sind gängige objektorientierte Frameworks weder reine Whitebox- noch reine Blackbox-Frameworks. Typischerweise entwickeln sich Whitebox-Frameworks mit der Zeit, d. h. mit zunehmenden Reifegrad, zu Blackbox-Frameworks [RJ97, JF88]. Diese Entwicklung ergibt sich vornehmlich durch häufiges Wiederverwenden des Whitebox-Frameworks in konkreten Anwendungen.

Komposition von objektorientierten Frameworks Während ursprünglich angenommen wurde, dass zur Konstruktion einer Anwendung nur ein einzelnes objektorientiertes Framework verwendet wird, ist in den letzten Jahren zu beobachten, dass zunehmend mehrere Frameworks für die Konstruktion einer Anwendung zum Einsatz kommen [Bos00, SGM02]. Der Grund dafür liegt darin, dass heutige Anwendungen zunehmend mehrere Domänen abdecken (müssen), ein objektorientiertes Framework aber immer nur für genau einen Anwendungsbereich konzipiert ist. Werden mehrere unabhängig voneinander entwickelte objektorientierte Frameworks für die Konstruktion einer Anwendung kombiniert, so führt dies oft zu einer Fülle von Problemen. So beabsichtigt jedes objektorientierte Framework, bedingt durch seine Natur, den Hauptkontroll-

fluss der Anwendung zu übernehmen (siehe Abschnitt 19.1.1). Zudem kann es passieren, dass sich die verwendeten Frameworks in ihrer Funktionalität überlappen [BMM+99]. Bestimmte Teile der Anwendungsdomäne werden dann in mehreren Frameworks modelliert. Dies betrifft oftmals auch die Basisklassen der objektorientierten Frameworks. Versucht man ein objektorientiertes Framework durch Änderungen an den Basisklassen mit anderen Frameworks kompatibel zu machen, so kann das zahlreiche Änderungen an abgeleiteten Klassen zur Folge haben und grundlegende Entwurfsentscheidungen verwerfen lassen [Pre96b].

Die Interaktion zwischen verschiedenen objektorientierten Frameworks kann nur auf einer noch höheren Abstraktionsebene gelöst werden. Dazu bedarf es einer Ebene zur Nutzung objektorientierter Frameworks, die beschreibt, wo, wann und wie sich Teile von Frameworks überlappen oder interagieren [SGM02]. Wie ein normales objektorientiertes Framework die Kombination konkreter Klassen unterstützt, so könnten höherwertige »Teilsystem-Frameworks«, sogenannte Frameworks zweiter Ordnung, dazu dienen, komplexere Teilsysteme zu kombinieren. Jedes dieser Teilsysteme kann als traditionelles objektorientiertes Framework (erster Ordnung) strukturiert und verstanden werden [SGM02]. Die Erarbeitung eines allgemeinen Ansatzes zur Bildung solcher Framework-Hierarchien ist Gegenstand aktueller Forschung [Pre97c]. Bestrebungen in diese Richtung sind Komponenten-Frameworks, die im folgenden Abschnitt vorgestellt werden.

19.2.2 Komponentenbasierte Frameworks

Während sich objektorientierte Frameworks über die Struktur ihrer Klassen und Objekte und deren Verantwortlichkeiten definieren, stehen bei den komponentenbasierten Frameworks bzw. Komponenten-Frameworks die einzelnen Software-Komponenten und deren Schnittstellen im Vordergrund. Eine recht abstrakte Definition sieht Komponenten-Frameworks als Sammlung von verschiedenen einzelnen Komponenten mit vordefiniertem Kooperationsverhalten an mit dem Zweck, Aufgaben in einem bestimmten Anwendungsbereich zu erfüllen [And04, Pre97a]. Szyperski et al. präzisieren das, indem sie Komponenten-Frameworks als Sammlung von Regeln und Schnittstellen definieren [SGM02], die die Interaktion von Komponenten eines Komponenten-Frameworks regeln (siehe auch [FSJ99b]). Diese Regeln und Schnittstellen stellen vetragsähnliche Vereinbarungen zwischen einem Anbieter eines »Dienstes« (die Software-Komponente) und dessen »Nutzer« (das Komponenten-Framework bzw. die konkrete Anwendung) dar, die neben funktionalen auch nicht funktionale Aspekte des Dienstes beinhalten kann. Ein Komponenten-Framework erlaubt es, Instanzen von Komponenten, die diesen Verträgen genügen, an bestimmten, durch diese Verträge typisierten Erweiterungspunkten in das Komponenten-Framework einzuhängen. Die Interaktion zwischen diesen Instanzen wird dabei vom Komponenten-Framework reguliert. Dies bedeutet, dass der Hauptkontrollfluss der Anwendung – wie

bei den objektorientierten Frameworks – beim Komponenten-Framework liegt. Die einzelnen Komponenten eines Komponenten-Frameworks geben noch keine Anwendungsarchitektur vor, wohl aber das Komponenten-Framework selbst, also die Organisation und Strukturierung der einzelnen Komponenten, die sich aus den Anforderungen des betrachteten Anwendungsbereiches ergeben (vgl. [Sih01], S. 70). Im Gegensatz zu den objektorientierten Frameworks erfolgt die Anpassung des Verhaltens eines Komponenten-Frameworks an die Anforderungen einer konkreten Anwendung ausschließlich durch Komposition. Anstelle einzelner konkreter Klassen werden jedoch verschiedene Instanzen der Framework-Komponenten (und deren Klassen) komponiert. Folglich stellen Komponenten die Variationspunkte eines Komponenten-Frameworks dar.

Die einzelnen Komponenten eines Komponenten-Frameworks werden oftmals mithilfe objektorientierter Frameworks entwickelt [Bos00]. Dabei sollte ein objektorientiertes Framework immer nur zur Entwicklung von genau einer Komponente verwendet und nicht über mehrere Komponenten verteilt werden, um keine Vererbungsbeziehungen von Klassen verschiedener Komponenten zu erzeugen. Andernfalls ist eine der wichtigsten Eigenschaften von Software-Komponenten verletzt, nämlich die, dass eine Komponente independently deployable sein soll, d. h. als in sich geschlossene Einheit ausgeliefert (und eingesetzt) werden kann [SGM02].

Ein Komponenten-Framework kann alleine eingesetzt werden oder wieder mit anderen Komponenten oder Komponenten-Frameworks kooperieren. Im ersteren Fall wird die Anwendung durch genau ein Komponenten-Framework bestimmt. Um mit anderen Komponenten-Frameworks kooperieren zu können, sollten Komponenten-Frameworks ebenfalls als Komponenten modelliert werden. Das Konzept der Komponenten-Frameworks kann dann auch hierarchisch angewendet werden. Komponenten-Frameworks, die selbst wieder als Komponenten realisiert sind, sind im Gegensatz zu objektorientierten Frameworks gleich wieder auf das Zusammenspiel mit anderen Komponenten-Frameworks ausgelegt und lassen sich damit leichter integrieren [Sih01]. Komponenten-Frameworks werden als die derzeit höchste Stufe zur Wiederverwendung von Software-Architekturen angesehen [Bos00].

19.2.3 Mischformen von Frameworks

Neben objektorientierten und komponentenbasierten Frameworks sind auch Mischformen von Frameworks möglich. Diese hybriden Frameworks haben sowohl die Eigenschaften von objektorientierten als auch von komponentenbasierten Frameworks. In diesem Fall geschieht ein Teil der Anpassungen des Frameworks an die konkrete Anwendung über Vererbung, während ein anderer Teil über Implementierung von Schnittstellen realisiert wird. Durch Veränderung des Anteils, der durch Vererbung an die konkrete Anwendung angepasst wird, bzw. durch Veränderung des Anteils, der durch Implementierung

von Schnittstellen realisiert wird, sind beliebige Varianten von Mischformen möglich.

19.2.4 Abgrenzung von Frameworks zu anderen Konzepten

In den vorangegangenen Abschnitten wurde der Begriff des Frameworks eingeführt. Es wurde eine Unterscheidung zwischen objektorientierten und komponentenbasierten Frameworks vorgenommen und in Beziehung zueinander gestellt. In diesem Abschnitt wird das Konzept des Frameworks gegen die verwandten Konzepte der Klassenbibliothek, Framelet, Entwurfsmuster, Software-Produktlinie und Referenzarchitektur abgegrenzt.

Eine Klassenbibliothek ist eine organisierte Sammlung von Klassen, aus denen der Entwickler nach Bedarf beliebige Einheiten verwenden kann. Im Gegensatz zu Frameworks erzwingen Klassenbibliotheken keine bestimmte Anwendungsarchitektur und übernehmen auch nicht den Hauptkontrollfluss einer Anwendung. Klassenbibliotheken, wie beispielsweise Qt [TA07] und Microsoft Foundation Classes [MC05a], werden immer von der konkreten Anwendung nach dem Call-down-Prinzip (siehe Abschnitt 19.1.1) aufgerufen und nicht umgekehrt, wie es bei den Frameworks der Fall ist.

Das Konzept der Software-Muster wird ausführlich in Kapitel 16 vorgestellt. Die Abgrenzung von Software-Muster gegen Frameworks ist in der Einleitung zu diesem Buchteil zu finden. Neben den in Kapitel 16 vorgestellten Software-Mustern gibt es auch noch sogenannte Metamuster. Metamuster können zur Beschreibung des strukturellen Teils der Entwurfslösungen von Entwurfsmustern verwendet werden und dienen zur Konstruktion objektorientierter Frameworks [JN99]. Sie werden ausführlich im Zusammenhang mit dem Entwurf objektorientierter Frameworks in Abschnitt 19.4.4 vorgestellt.

Framelets repräsentieren wie objektorientierte Frameworks flexible objektorientierte Software-Architekturen. Während objektorientierte Frameworks jedoch typischerweise abstrakte Architekturgerüste für komplexe Anwendungsbereiche darstellen, sind Framelets kleinerer Natur. Sie können als Kombination einiger Entwurfsmuster zu einem wiederverwendbaren Architekturbaustein betrachtet werden und realisieren generisch einen Systemaspekt, der in verschiedenen Anwendungen benötigt wird [PAS98]. Im Gegensatz zu konventionellen objektorientierten Frameworks stellen Framelets kleinere Architekturbausteine dar [PK00, PK99, PAS98], die aus weniger als zehn Klassen bestehen. Framelets ziehen auch nicht den Hauptkontrollfluss der Anwendung an sich.

Das Konzept der Software-Produktlinie wird ausführlich in Kapitel 18 und das Konzept der Referenzarchitektur in Kapitel 17 diskutiert. Eine Abgrenzung zu Frameworks ist wie bereits für die Software-Muster der Einleitung dieses Buchteils zu entnehmen.

19.3 Der Entwicklungsprozess von Frameworks

Die Entwicklung eines Frameworks, unabhängig davon, ob es sich um ein objektorientiertes oder komponentenbasiertes Framework handelt, ist eine anspruchsvolle Aufgabe und lohnt sich erst, wenn mehrere Anwendungen mit diesem Framework entwickelt werden. Erfahrungsgemäß sind mindestens drei Anwendungen notwendig, damit sich der höhere Entwicklungsaufwand eines Frameworks im Vergleich zur normalen Anwendungsentwicklung lohnt. Ein qualitativ hochwertiges Framework ist typischerweise das Ergebnis eines langen Entwicklungsprozesses und sollte mittels eines geeigneten Vorgehensmodells mit entsprechender Entwicklungsmethodik durchgeführt werden [CC02]. Aufgrund sich ändernder Anforderungen und der Komplexität von Frameworks ist der Entwicklungsprozess gekennzeichnet durch eine iterative und inkrementelle Vorgehensweise (vgl. [BMM+99]). Aus Beispielanwendungen und anderen existierenden Software-Produkten werden die relevanten Framework-Funktionalitäten für den Anwendungsbereich schrittweise abstrahiert. Dies schließt auch die Strukturierung des Frameworks hinsichtlich der unterstützten Anpassungsmöglichkeiten mit ein, d. h. die Bereitstellung von flexiblen Variationspunkten, mittels derer das Framework angepasst und erweitert werden kann. Diese Variationspunkte werden im iterativen Entwicklungsprozess ermittelt und verfeinert [JN99]. Die Erfahrungen, die durch den Einsatz des Frameworks in Beispielanwendungen gewonnen werden, sind dabei die Eingabeparameter für die nächste Iteration des Framework-Entwicklungsprozesses. Diese iterative Entwicklung von Frameworks kann grob in die folgenden Phasen eingeteilt werden (siehe [BMM+99]): Entwicklung, Nutzung, Komposition sowie Evolution und Wartung von Frameworks. Die Eigenschaften und Herausforderungen dieser Phasen werden in den nächsten Abschnitten näher betrachtet.

19.3.1 Entwicklung von Frameworks

Die Entwicklung von Frameworks unterscheidet sich stark von der konventionellen Anwendungsentwicklung. Das liegt hauptsächlich daran, dass ein Framework alle Konzepte einer bestimmten Domäne betrachten und unterstützen soll, während bei der Entwicklung einer speziellen Anwendung nur die Konzepte von Interesse sind, die zur Umsetzung dieser einen Anwendung notwendig sind. Die Entwicklung eines Frameworks umfasst dabei die Aktivitäten, die typischerweise auch in Vorgehensmodellen für die klassische Software-Entwicklung zu finden sind (siehe [BMM+99]): Analyse der Anwendungsdomäne, Festlegen der Architektur, Entwurf, Implementierung, Testen und Dokumentation des Frameworks. Die Entwicklungsmethoden für die normale Anwendungsentwicklung unterstützen den Entwurf und die Entwicklung von Frameworks jedoch nicht in einem ausreichenden Maße. Der Entwurf von Frameworks erfordert einige neue Konzepte, die methodisch unterstützt werden müssen [BMM+99]. So muss zum

Beispiel während der Analyse der Anwendungsdomäne berücksichtigt werden, dass sich im Gegensatz zur normalen Anwendungsentwicklung das Verhalten und die Konfiguration des Frameworks in verschiedenen Anwendungsfällen ändern können.

Bei der Entwicklung eines Frameworks ist die Festlegung der betrachteten Domäne (engl. domain scope) von entscheidender Bedeutung. Da ein Framework oft in vielen unterschiedlichen Szenarien einsetzbar sein soll, ist häufig eine Auseinandersetzung mit verwandten Themen erforderlich, die nicht zwangsläufig zu den Kernkompetenzen der Framework-Entwickler gehören [Sih01]. Wird die Domäne des Frameworks zu groß gewählt, so hat das Entwicklerteam nicht die notwendigen Erfahrungen und das Wissen in dieser Domäne, um dafür ein Framework zu entwickeln. Dies kann höhere Entwicklungskosten zur Folge haben sowie die Anwendbarkeit und den Nutzen des Frameworks beeinträchtigen. Die Entwicklung von kleinen Frameworks ist zwar finanziell weniger riskant und einfacher durchzuführen, dafür besteht die Gefahr, anfällig gegenüber Änderungen der Anwendungsdomäne zu sein [BMM+99]. So können Anwendungen, die ein solches Framework einsetzen, schnell über dessen Grenzen hinauswachsen [BMM+99] und nur aufwendig um neue Funktionalitäten erweitert werden. Aus diesem Grund muss die Domäne eines Frameworks sorgfältig ausgewählt und festgelegt werden. Das Entwicklerteam sollte in der Lage sein, gute Hypothesen über die zu erwartenden Variabilitäten des Frameworks zu erstellen, um das Risiko zu minimieren.

19.3.2 Nutzung von Frameworks

Unter der Nutzung eines Frameworks wird dessen Einsatz und Anpassung für eine konkrete Anwendung verstanden. Das Erlernen der Nutzung eines Frameworks kann unter Umständen lange dauern [BMM+99] und hängt direkt von der Größe und der Anzahl der zur Verfügung stehenden Variationspunkte ab (vgl. [Sih01]). Werden Frameworks externer Organisationen für die Entwicklung einer Anwendung eingesetzt, so ist zu prüfen, ob die jeweiligen Frameworks die Anforderungen der Anwendung erfüllen. Ein besonderes Augenmerk bei dieser Auswahl ist auf die Interoperabilität der Frameworks zu legen (siehe Abschnitt 19.3.3).

Eine wesentliche Voraussetzung für den erfolgreichen Einsatz von Frameworks ist eine gute Dokumentation. Eine Möglichkeit, die Software-Entwickler bei der Anwendung eines Frameworks und dessen Anpassung an eine konkrete Anwendungsdomäne zu unterstützen, sind sogenannte Framework-Kochbücher (engl. framework cookbooks) [Pre97d]. Die »Rezepte« dieser »Kochbücher« beschreiben in Form von Prüflisten die Arbeitsschritte, die notwendig sind, um typische Aufgaben zur Anwendung und Anpassung des Frameworks durchzuführen. Eine andere Möglichkeit ist die Unterstützung durch einen Mentor, der idealerweise an der Entwicklung des Frameworks beteiligt war.

19.3.3 Komposition von Frameworks

Bei der Komposition von Frameworks werden mehrere Frameworks für die Entwicklung einer konkreten Anwendung eingesetzt. Die verwendeten Frameworks müssen dabei an die Architektur der Anwendung sowie der anderen Frameworks angepasst werden, damit diese zusammenarbeiten können. Die Komposition von Frameworks kann zu einer Fülle von Problemen führen, da diese naturgemäß bestrebt sind, die volle Kontrolle über die Anwendung innezuhaben (siehe dazu auch Abschnitt 19.2.1). Im Extremfall sind die Architekturstile, auf denen die Frameworks basieren, so unterschiedlich, dass es vielleicht unmöglich ist, sie miteinander zu komponieren. Diese Inkompatibilitäten von Architekturen werden als »architectural mismatch« bezeichnet. Ein erster Schritt zur Lösung dieses Problems scheint es zu sein, den Architekturstil, dem ein Framework unterliegt, explizit zu spezifizieren [BMM+99]. Werden zur Entwicklung einer Anwendung mehrere Frameworks eingesetzt, so kann es auch passieren, dass sich die Frameworks in ihrer Funktionalität teilweise überlappen. Dann werden bestimmte Teile der Anwendung in mehreren Frameworks jeweils aus ihrem eigenen Blickwinkel modelliert und müssen zusammengeführt werden (siehe dazu [BMM+99], S. 73).

19.3.4 Evolution und Wartung von Frameworks

Die Entwicklung von Frameworks muss als eine langfristige Investition betrachtet werden. Als solche müssen Frameworks entsprechend weiterentwickelt und gewartet werden. Unter der Evolution bzw. Wartung von Frameworks wird die Modifikation und Anpassung von Frameworks an veränderte Anforderungen der sie nutzenden Anwendungen und der betrachteten Domäne verstanden (nach [Som04]). Evolution und Wartung eines Frameworks beginnen bereits während der Entwicklung und nicht erst nach dem ersten Release. Schon früh in der Entwicklung eines Frameworks sind viele Iterationen bezüglich des Designs notwendig. Das liegt gerade am Ziel der Wiederverwendbarkeit von Frameworks. Die einzige Möglichkeit, dies zu prüfen, besteht darin, das Framework in konkreten Beispielanwendungen wiederzuverwenden und die Schwächen am Entwurf zu identifizieren und zu verbessern (vgl. [BMM+99]). Die Grundlagen der Evolution von Software-Architekturen im Allgemeinen werden in Kapitel 8 beschrieben.

19.4 Entwurf objektorientierter Frameworks

Ein objektorientiertes Framework zu entwerfen, erfordert sehr viel Erfahrung und Experimentieren [JF88]. Ein gut entworfenes objektorientiertes Framework ist typischerweise das Ergebnis von zahlreichen Entwurfsiterationen und sehr viel harter Arbeit [WJ90]. Das Streben nach Flexibilität um ihrer selbst willen

liefert allerdings nicht automatisch ein erstklassiges Framework. Vielmehr gilt, dass Flexibilität wohldosiert und in einer den Anforderungen des Anwendungsbereiches angemessenen Weise in ein Framework eingebaut werden sollte [Pre97c]. Unnötige Flexibilität erhöht dagegen sogar deutlich die Komplexität des Frameworks [FPR02]. Zur Entwicklung eines objektorientierten Frameworks sollte ein geeignetes Vorgehensmodell mit entsprechender Entwicklungsmethodik verwendet werden. Das Vorgehensmodell beschreibt, welche Aktivitäten zur Entwicklung des Frameworks wann durchzuführen sind. Die Entwicklungsmethodik unterstützt den Framework-Entwickler bei der Identifikation und Spezifikation der Flexibilitätsanforderungen an das Framework. In Abschnitt 19.4.1 bis Abschnitt 19.4.5 wird mit dem Hotspot-getriebenen Ansatz von Pree ein erprobtes Vorgehensmodell mit entsprechender Entwicklungsmethodik für objektorientierte Frameworks ausführlich vorgestellt. In Abschnitt 19.4.6 werden außerdem noch kurz andere Entwurfsmethoden für objektorientierte Frameworks genannt.

19.4.1 Der Hotspot-getriebene Entwurf von objektorientierten Frameworks

Der in Abbildung 19.2 gezeigte Hotspot-getriebene Ansatz stellt ein bekanntes und erprobtes Verfahren zur Entwicklung objektorientierter Frameworks dar und ist in zahlreichen Büchern und wissenschaftlichen Artikeln veröffentlicht worden [FPR02, Pre99, Pre97c, Pre97b, Pre96a, Pre95]. Da die Qualität eines objektorientierten Frameworks, wie in der Einleitung von Abschnitt 19.4 beschrieben, direkt von den Flexibilitätseigenschaften hinsichtlich der betrachteten Anwendungsdomäne abhängt, sieht der Hotspot-getriebene Entwurf eine explizite Aktivität zur Identifikation von geeigneten Variationspunkten vor [FPR02]. Eine sorgfältig durchgeführte Identifikation von Variationspunkten, den sogenannten Hotspots, kann eine wertvolle Hilfe bei der Entwicklung eines guten Frameworks sein [Pre97c]. Im Folgenden werden die einzelnen Aktivitäten des Hotspot-getriebenen Entwicklungsprozesses beschrieben. Als Notation für die Klassendiagramme wird dabei die UML-F [FPR02, FPR00] verwendet, eine Erweiterung der Unified Modeling Language (UML) zur Beschreibung von Frameworks.

19.4.2 Definition eines speziellen Objektmodells

Der kritische erste Schritt bei der Entwicklung eines objektorientierten Frameworks ist die Identifikation der wichtigsten Abstraktionen, der sogenannten Schlüsselabstraktionen (auch Archetyp [BB99, LBHB99]), des betrachteten Anwendungsbereiches [FPR02]. Die Objektmodellierung erfordert hauptsächlich anwendungsspezifisches Wissen. Die Software-Techniker unterstützen die Anwendungsexperten bei der Durchführung dieser Aufgabe. Die Unterscheidung zwischen diesen beiden Rollen ist allerdings nur eine hypothetische und soll

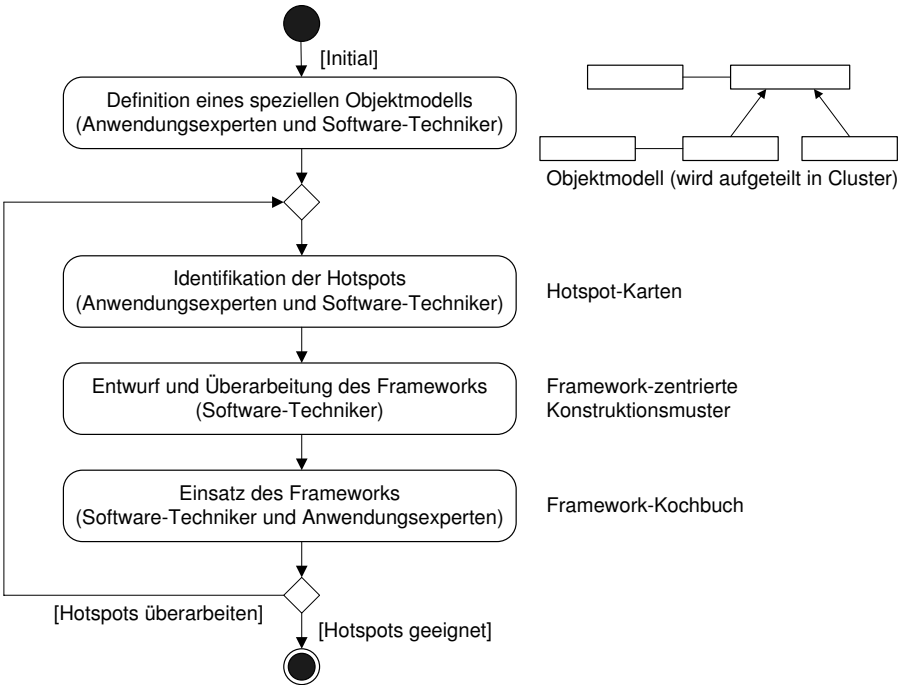

Abbildung 19.2: Der Hotspot-getriebene Entwurf objektorientierter Frameworks (nach [FPR02, Pre97c, Pre95])

ausdrücken, dass verschiedene Arten von Wissen zur Durchführung dieser Aktivität notwendig sind. So haben Software-Techniker in der Regel nur ein eingeschränktes Wissen über den zu modellierenden Anwendungsbereich.

Das initiale Objektmodell bildet die Basis des Hotspot-getriebenen Framework-Entwurfs. Zum Erstellen dieses Objektmodells können zum Beispiel CRC-Karten (engl. Class Responsibility Collaboration cards) eingesetzt werden [FPR02]. CRC-Karten [BC89] eignen sich generell zum Auffinden von Klassen, Objekten und ihren Assoziationen [Pre97c]. Neben den CRC-Karten für das gesamte Objektmodell empfiehlt es sich, zudem einen separaten Satz an Karten zu erstellen, die nur die abstrakten Klassen des zu entwickelnden objektorientierten Frameworks umfassen. Diese werden als aCiRC-Karten bezeichnet (engl. abstract Class/interface Responsibility Collaboration cards). Das initiale Objektmodell wird auf Basis der identifizierten Schlüsselabstraktionen in einzelne Cluster gegliedert [Mey90]. Ein Cluster besteht aus einer Gruppe von Klassen und/oder Schnittstellen. Der Framework-Entwicklungsprozess wird den Clustern entsprechend in mehrere, sich zeitlich überlappende und unabhängige Software-Lebenszyklen aufgeteilt. Die einzelnen Cluster können sich also in verschiedenen Entwicklungsphasen befinden, die zudem unabhängig davon sind, in welcher

Phase des Hotspot-getriebenen Entwurfs das zu entwickelnde Framework gerade ist. Diese Vorgehensweise spiegelt die Tatsache wider, dass der iterative Entwurf abstrakter Klassen und Schnittstellen erfordert, dass sich einige Cluster in der Implementierungsphase befinden. Wird während der Implementierung eines Clusters festgestellt, dass eine Abstraktion nicht passend ist, dann muss die Überarbeitung des Entwurfs dieser Abstraktion parallel mit der Implementierung dieses speziellen Clusters fortgeführt werden [FPR02].

Bevor nun mit dem Hauptzyklus des Framework-Entwicklungsprozesses begonnen wird, ist es hilfreich, mehrere Objektmodelle von ähnlichen Anwendungen in der betrachteten Domäne zur Verfügung zu haben. Dadurch wird das Auffinden von Gemeinsamkeiten und deren spätere Abstraktion durch das Framework einfacher. Diese Objektmodelle stehen in der Regel jedoch nicht zur Verfügung, da das Erstellen einer anwendungsspezifischen Software-Lösung eine komplexe und iterative Aufgabe ist, in der Objektmodelle erstellt und verfeinert werden müssen, bis sie den Anforderungen genügen.

19.4.3 Identifikation der Hotspots und Erstellung der Hotspot-Karten

Das Hauptproblem bei der Identifikation der Hotspots ist, dass Anwendungsexperten nicht gewohnt sind, von einem konkreten Software-System zu abstrahieren und zu generalisieren. In der Regel wissen Anwendungsexperten nichts über objektorientierte Konzepte, wie Klassen, Objekte, Vererbung, Entwurfsmuster und Frameworks. Stattdessen können sie meistens gut in Software-Funktionalitäten denken und wissen typischerweise auch, durch welche grundlegenden Elemente sich ihr Anwendungsbereich auszeichnet. Zur Kommunikation zwischen den Software-Technikern und Anwendungsexperten muss daher eine gemeinsame Basis geschaffen werden, die auf den Funktionalitäten des Frameworks beruht und nicht angibt, in welchen Klassen und Schnittstellen welche Funktionalitäten platziert werden [FPR02, Pre97c]. Eine Möglichkeit, um die Framework-Funktionalitäten zu kommunizieren, sind Hotspot-Karten (engl. variation point cards).

Hotspot-Karten sind eine weitere Variante der CRC-Karten und stellen ein einfaches, aber effektives Hilfsmittel zur Dokumentation und Kommunikation von Flexibilitätsanforderungen zwischen den Anwendern und Framework-Entwicklern dar [Pre97c]. Sie dienen im ersten Schritt dazu, die Hotspots des zu entwickelnden Frameworks zu identifizieren, und helfen im zweiten Schritt beim konkreten Entwurf des Frameworks. Prinzipiell können sowohl Funktionen als auch Daten als Hotspots identifiziert werden. Die meisten Hotspots stellen aber wahrscheinlich Funktionen dar [Pre97c]. In Abhängigkeit vom Typ des identifizierten Hotspots wird entweder eine Funktionen- oder Daten-Hotspot-Karte erstellt. Ihr Aufbau ist in Abbildung 19.3 dargestellt. Funktionen-Hotspot-Karten dokumentieren die Funktionalitäten, die an dem objektorientierten

Framework flexibel gehalten werden sollen. Auf Daten-Hotsport-Karten werden die Elemente der Anwendungsdomäne festgehalten, die im Zuge einer Framework-Implementierung verallgemeinert werden müssen. Für konkrete Beispiele von Funktionen- und Daten-Hotspot-Karten sei auf die Literatur verwiesen, wie zum Beispiel [Pre97c, Pre96a].

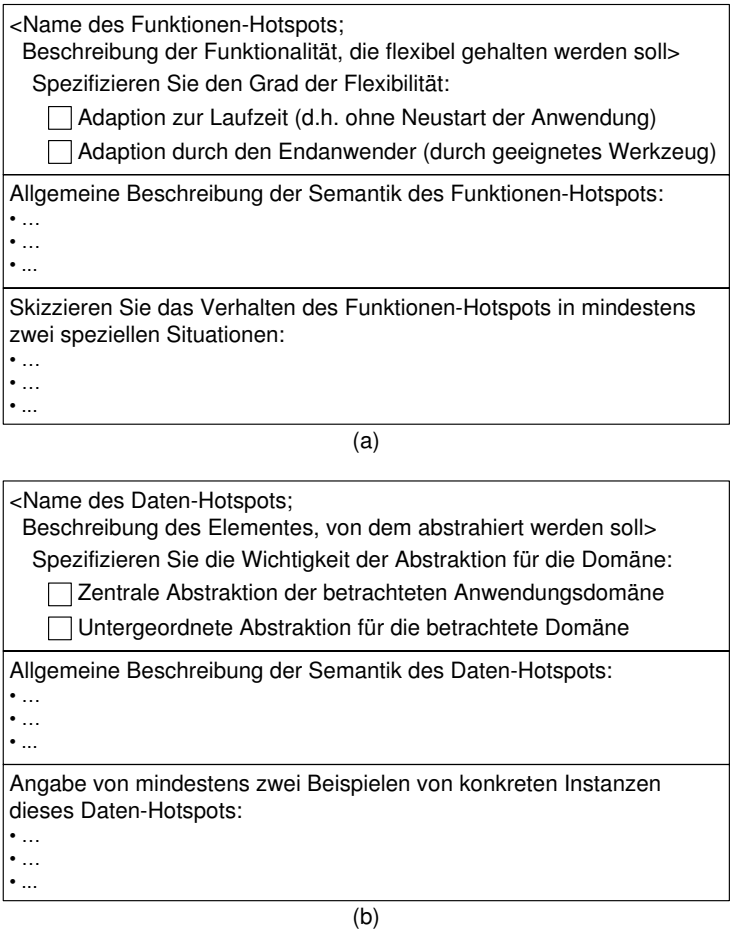

Abbildung 19.3: Aufbau einer (a) Funktionen-Hotspot-Karte und (b) Daten-Hotspot-Karte (nach [Pre97c, Pre96a])

Zur Realisierung einer Daten-Hotspot-Karte wird typischerweise eine abstrakte Klasse in das Objektmodell des zu entwickelnden Frameworks eingeführt. Daher kann die Dokumentation von Daten-Hotspots auf entsprechenden Karten auch weggelassen und das Objektmodell des Frameworks direkt manipuliert werden [Pre96a]. Im weiteren Verlauf dieses Kapitels konzentrieren wir uns deshalb auf

die Umsetzung der Funktionen-Hotspots. Zudem vereinbaren wir, dass mit einem Hotspot bzw. einer Hotspot-Karte im Folgenden implizit ein Funktionen-Hotspot bzw. eine Funktionen-Hotspot-Karte gemeint ist.

Um mithilfe von Hotspot-Karten ein Objektmodell in ein objektorientiertes Framework zu transformieren, sollte ihre Granularität einer Methode entsprechen. Diese Empfehlung weicht von der in der Literatur zum Hotspot-getriebenen Entwurf objektorientierter Frameworks [Pre97c, Pre95] angegebenen Empfehlung ab, die die Granularität einer Hotspot-Karte auf in etwa einer Methode festlegt. Die hier vorgenommene Festlegung der Granularität von Hotspot-Karten auf eine Methode begründet sich mit der Einführung sogenannter Gruppen-Hotspot-Karten in Kapitel 20. Diese Gruppen-Hotspot-Karten werden dort verwendet, um die Komponenten des MM4U-Frameworks (Abkürzung für »Multimedia for you«) zu identifizieren und die Flexibilitätsanforderungen an diese Komponenten zu spezifizieren.

19.4.4 Entwurf und Überarbeitung des objektorientierten Frameworks mit Metamustern

Nachdem die initialen Hotspots identifiziert und auf Hotspot-Karten dokumentiert wurden, wird das Objektmodell hinsichtlich der in den Hotspot-Karten festgehaltenen Flexibilitätsanforderungen von den Software-Technikern modifiziert. Dazu definieren sie für jeden Hotspot, der auf den Hotspot-Karten beschrieben ist, einen »Haken« (engl. hook) im Entwurf des objektorientierten Frameworks, d. h., sie bestimmen, wo die zu den Hotspot-Karten gehörenden Methoden in der Klassenhierarchie des Frameworks platziert werden sollen [FHLS99]. Diese Methoden werden im Folgenden als Einschubmethoden (engl. hook methods) bezeichnet. Dabei unterstützen Framework-zentrierte Konstruktionsmuster, die sogenannten Metamuster (engl. meta pattern), die Software-Techniker bei der konkreten Umsetzung der Hotspot-Karten [Pre95]. Metamuster stellen die essenziellen Konstruktionsprinzipien für objektorientierte Frameworks dar [Pre97c] und beschreiben, wie ein Framework zu entwerfen ist, unabhängig von einer bestimmten Domäne [Pre95]. Sie werden geformt aus Schablonenmethoden (engl. template methods) und den bereits genannten Einschubmethoden [Pre95]. Die Einschubmethoden stellen dabei abstrakte Platzhalter im objektorientierten Framework dar, die durch bereits implementierte Schablonenmethoden des Frameworks aufgerufen werden. Schablonenmethoden können abstraktes Verhalten, generischen Kontrollfluss oder auch das Interaktionsverhalten von Objekten des Frameworks beschreiben. Die prinzipielle Idee der Einschubmethoden ist, dass das Verhalten des objektorientierten Frameworks von den Anwendungsentwicklern durch Überschreiben dieser Einschubmethoden verändert wird [BD00]. Eine Spezialisierung, d. h. Anpassung, des objektorientierten Frameworks findet »ausschließlich« über die Einschubmethoden statt [Pre97c, Pre95]. Dadurch wird die Einhaltung des Offen/Geschlossen-Prinzips sichergestellt (siehe Abschnitt

19.2.1). Schablonenmethoden beschreiben die abstrakte Anwendungslogik des objektorientierten Frameworks, die nicht mehr für Anpassungen modifiziert werden kann. Schablonenmethoden werden daher auch Frozen-Spot genannt. Da neue Funktionalität nur über die Einschubmethoden in das objektorientierte Framework eingefügt werden kann, ist eine weite Voraussicht der Framework-Entwickler nötig, um die Einschubmethoden zu bestimmen [Pre97c]. Zur Modifizierung des Objektmodells reicht im Allgemeinen die Semantik eines Hotspots aus, um sowohl die Klasse als auch die Schablonenmethode zu finden, in die die Einschubmethode eingefügt werden soll [Pre97c].

Abbildung 19.4 stellt das Konzept der Schablonen- und Einschubmethoden beispielhaft dar. Die Schablonenmethode `t()` der Klasse `A` ruft, wie in Abbildung 19.4(a) gezeigt, die Einschubmethode `h()` auf. Die Klasse `B` in Abbildung 19.4(b) überschreibt die Einschubmethode `h()` durch Vererbung von Klasse `A`. Das dazugehörige UML-Klassendiagramm ist in Abbildung 19.4(c) zu sehen. Die Menge an Metamustern, die für den Entwurf von objektorientierten Frameworks nötig sind, kann aus den Kombinationen von Schablonen- und Einschubklassen abgeleitet werden. Schablonen- und Einschubklassen können prinzipiell jede Komplexität annehmen. An der Anzahl der essenziellen Metamuster ändert das allerdings nichts, sondern nur an deren Größe.

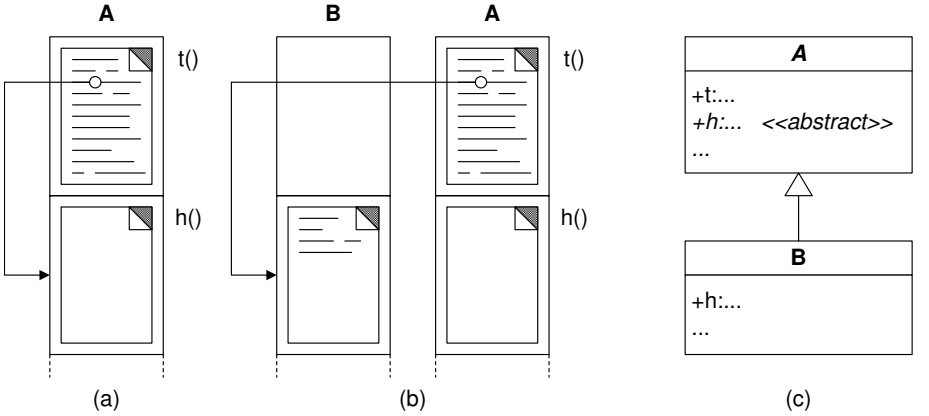

Abbildung 19.4: (a) Schablonen- und Einschubmethoden, (b) das Überschreiben von Einschubmethoden und (c) das dazugehörende UML-Klassendiagramm (nach [Pre97c])

Die Schablonen- und Einschubmethoden können, wie in Abbildung 19.4 dargestellt, entweder in einer gemeinsamen Klasse oder in zwei getrennten Klassen definiert werden. Eine Klasse, die Einschubmethoden enthält, wird als Einschubklasse (engl. hook class) bezeichnet, und eine Klasse, die Schablonenmethoden enthält, wird Schablonenklasse (engl. template class) genannt. Die Einschub-

klasse parametrisiert dabei die Schablonenklasse. Da die Klasse `A` in Abbildung 19.4(c) sowohl die Schablonen- als auch die Einschubmethode enthält, ist sie sowohl Schablonen- als auch Einschubklasse und parametrisiert sich selbst. Um die Einschub- und Schablonenmethoden im Entwurf eines objektorientierten Frameworks explizit kenntlich zu machen, sieht die UML-F zwei Tags vor. Wie in Abbildung 19.5 dargestellt ist, werden Einschubmethoden im Klassendiagramm mit dem Tag «`hook`» und Schablonenmethoden mit «`template`» versehen. Diese Tags werden auch zum Kennzeichnen von Einschub- und Schablonenklassen verwendet.

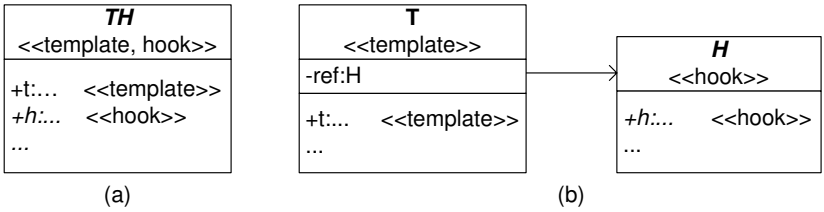

Abbildung 19.5: (a) Unifikation und (b) Separation von Schablonen- und Einschubmethoden (nach [Pre99])

Abhängig davon, ob eine Anpassung zur Laufzeit unterstützt werden soll oder nicht, werden die Einschub- und Schablonenmethoden in einer gemeinsamen Klasse oder in zwei getrennten Klassen definiert. Dazu wird entweder das Unifikationsprinzip (engl. unification principle) oder das Separationsprinzip (engl. separation principle) angewendet [Pre97c]. Beim Unifikationsprinzip werden, wie im Beispiel in Abbildung 19.5(a) dargestellt, die Schablonen- und Einschubmethode in der gemeinsamen Klasse `TH` definiert. Anpassungen sind hier durch Bildung entsprechender Unterklassen möglich und benötigen einen Neustart der Anwendung. Um eine Anpassung zur Laufzeit vornehmen zu können, werden die Einschub- und Schablonenmethoden nach dem Separationsprinzip in zwei Klassen getrennt, wie in Abbildung 19.5(b) dargestellt ist. Das Verhalten der Schablonenklasse `T` kann durch Komposition mit unterschiedlichen `H`-Objekten (also Instanzen von konkreten Klassen, die `H` implementieren) modifiziert werden. Die gerichtete Assoziation zwischen `T` und `H` drückt aus, dass ein `T`-Objekt auf ein `H`-Objekt verweist. Dies geschieht typischerweise über eine Instanzvariable in der Schablonenklasse `T`, hier `ref`. Die `H`-Objekte können so zur Laufzeit ausgetauscht werden, ohne die Anwendung neu starten zu müssen. An dieser Stelle ist darauf hinzuweisen, dass die abstrakte Klasse `H` zum Beispiel in der Programmiersprache Java auch ein Interface sein kann. Prinzipiell ist die Realisierung durch eine abstrakte Klasse oder durch ein Interface als gleichwertig zu betrachten [PAS98]. Ist nicht sicher, ob eine Anpassung zur Laufzeit erforderlich ist oder nicht, so ist im Zweifelsfall die Realisierung

über ein Interface vorzuziehen (vgl. [RJ97]). Ist nicht nur eine Anpassung der Funktionalität zur Laufzeit erforderlich, sondern soll die Anpassung auch durch den Endanwender vorgenommen werden können, so muss zusätzlich noch ein geeignetes Konfigurationswerkzeug entwickelt werden. Außerdem ist anzumerken, dass die Wahl zur Umsetzung der Hotspots mit dem Unifikations- oder Separationsprinzip nur für objektorientierte Frameworks existiert. Bei der komponentenbasierten Software-Entwicklung und damit auch bei den Komponenten-Frameworks (siehe Abschnitt 19.2.2) ist nur die Umsetzung nach dem Separationsprinzip möglich. Zudem müssen die Einschubmethoden in einem Interface definiert werden.

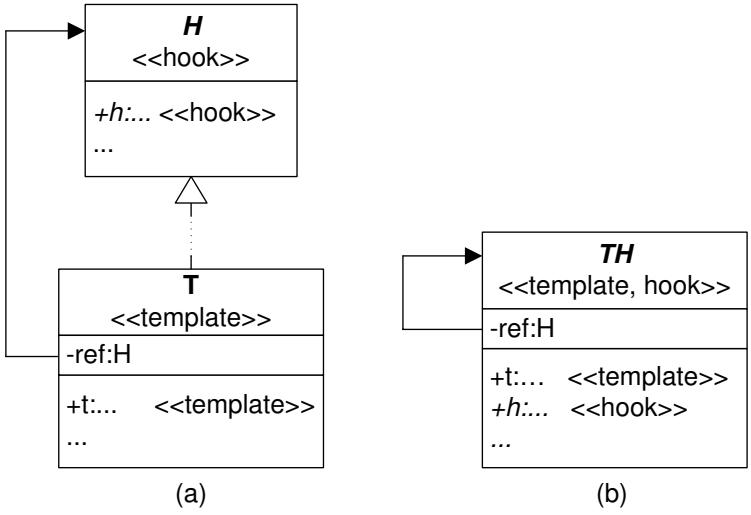

Abbildung 19.6: Rekursive Kombination von Schablonen- und Einschubklassen (aus [Pre99])

Neben den bisher vorgestellten Metamustern können die Schablonen- und Einschubklassen auch rekursiv miteinander kombiniert werden (rekursive Komposition). So kann, wie in Abbildung 19.6(a) dargestellt, die Schablonenklasse `T` ihrerseits eine Unterklasse der Einschubklasse `H` sein. Im Extremfall fallen Schablonen- und Einschubklasse zusammen und sind, wie in Abbildung 19.6(b) dargestellt, in einer Klasse vereinigt. Als Unterklasse von `H` kann die Schablonenklasse `T` nun eine weitere Instanz von sich selbst als Implementierung für die Einschubklasse `H` verwalten. Umgekehrt können aber Einschubklassen nicht als Unterklassen von Schablonenklassen definiert werden, da Einschubklassen nur aus den abstrakten Einschubmethoden bestehen. Dass die rekursive Verschachtelung nicht nur von theoretischem Interesse, sondern auch für die Praxis relevant ist, zeigen Beispiele in [Pre95, Pre96a]. Die Metamuster für

objektorientierte Frameworks existieren darüber hinaus auch in einer Variante, bei der die Schablonenklasse nicht nur eine Instanz der Einschubklasse, sondern auch über eine Liste von Objektreferenzen eine beliebige Menge von Instanzen der Einschubklasse verwalten kann (siehe dazu [Pre95, Pre94]).

Der Entwurf der Klassenschnittstellen durchläuft im Allgemeinen sehr viele Iterationsschritte. Dies wird vornehmlich durch die verschiedenen Verwendungen des objektorientierten Frameworks in unterschiedlichen konkreten Anwendungen verursacht. So kann während einer dieser Iterationen deutlich werden, dass die Schablonenmethoden zu starr angelegt sind und das Framework dadurch unflexibel wird. Infolgedessen müssen weitere Einschubmethoden definiert und eingesetzt werden. Andererseits können sich aus der wiederholten Anwendung des objektorientierten Frameworks auch neue sinnvolle Schablonenmethoden ergeben. Implementieren zum Beispiel mehrere Anwendungsentwickler beim Anpassen des Frameworks einen ähnlichen Kontrollfluss, dann sollte dieser als Schablonenmethode in das Framework integriert werden.

19.4.5 Einsatz des objektorientierten Frameworks

Ein objektorientiertes Framework muss mehrere Male wiederverwendet, d. h. in konkreten Anwendungen eingesetzt werden, um eventuelle Schwächen am Entwurf festzustellen (vgl. [BMM+99]). Die Schwächen am Entwurf eines Frameworks sind die Stellen, wo Einschubmethoden fehlen oder ungeeignet sind bzw. wo Schablonenmethoden zu wenig oder zu viel (abstraktes) Verhalten kapseln.

Der in Abbildung 19.2 angegebene Zyklus stellt diese evolutionäre Vorgehensweise beim Entwurf objektorientierter Frameworks mithilfe von Hotspot-Karten dar. Die explizite Identifikation von Hotspots trägt hier zu einer signifikanten Reduzierung der Iterationen bei.

19.4.6 Andere Entwurfsmethoden für objektorientierte Frameworks

In den vorangegangenen Abschnitten 19.4.1 bis 19.4.5 wurde ausführlich der Hotspot-getriebene Entwicklungsprozess von objektorientierten Frameworks vorgestellt. Daneben gibt es auch noch einige andere Ansätze für die Entwicklung objektorientierter Frameworks. So wird in [Sch99] der Entwurf von Frameworks durch systematische Generalisierung beschrieben, die Verwendung von Templates zur Spezifikation von Frameworks wird in [DW99b] vorgestellt, in [Rie00] wird ein rollenbasierter Ansatz zur Entwicklung objektorientierter Frameworks vorgeschlagen und in [LN95] wird ein Vorgehensmodell für objektorientierte Frameworks vorgestellt, das eine 71 Punkte umfassende Checkliste zur Framework-Entwicklung beinhaltet.

19.5 Entwurf von Komponenten-Frameworks

Zur komponentenbasierten Software-Entwicklung stehen entsprechende Vorgehensmodelle zur Verfügung, wie diejenigen zum Beispiel in [Som04, And04, Bos00, HHKS97]. Darüber hinaus existieren auch erprobte Verfahren zur Identifikation von Komponenten, wie zum Beispiel der in [Sie04] beschriebene Ansatz zur Bildung von Software-Kategorien. Im Gegensatz zu objektorientierten Frameworks, für die es mit dem Hotspot-getriebenen Entwicklungsprozess (siehe Abschnitt 19.4.1) bereits ein erprobtes Vorgehensmodell gibt, fehlt derzeit noch eine entsprechende Unterstützung zur Entwicklung von Komponenten-Frameworks. Folglich hängt die Qualität eines Komponenten-Frameworks letztendlich von der Erfahrung und dem Geschick der Framework-Entwickler ab [Sih01].

Eine Möglichkeit, die Entwicklung von komponentenbasierten Frameworks zu verbessern, ist, den Entwurf und die Entwicklung von Software-Komponenten gleich so auszurichten, dass flexible und anpassbare Komponenten entstehen. Die Komponenten werden also gleich so gebaut, dass daraus Komponenten-Frameworks gebildet werden können. Dadurch besteht die Hoffnung, durch Implementierung flexibler Komponenten zu einem geeigneten Komponenten-Framework zu gelangen. Die Anforderungen, die sich aus der Anwendungsdomäne eines potenziellen Komponenten-Frameworks ergeben, hat man dabei jedoch nicht vor Augen und damit auch nicht die Flexibilitätsanforderungen an die einzelnen Komponenten des potenziellen Frameworks. Hier besteht die Gefahr, Flexibilität um ihrer selbst willen in das Framework zu integrieren, die dem Anwendungsbereich nicht angemessen ist (vgl. [Pre97c]) und damit die Komplexität des Frameworks unnötig erhöht (siehe [FPR02]). Stattdessen sollte Flexibilität, wie bei den objektorientierten Frameworks, mithilfe eines geeigneten Vorgehensmodells identifiziert und wohldosiert in ein Komponenten-Framework implementiert werden (siehe auch Abschnitt 19.4).

Zur systematischen Unterstützung des Entwicklungsprozesses von Komponenten-Frameworks kann der in Abschnitt 20.4 vorgeschlagene Ansatz verwendet werden. Dieser stellt eine Modifikation des Hotspot-getriebenen Entwurfs objektorientierter Frameworks dar und integriert explizite Aktivitäten zur Identifikation der benötigten Framework-Komponenten und zur Spezifikation der Flexibilitätsanforderungen an diese Komponenten. Erprobt wurde dieser Entwicklungsprozess für Komponenten-Frameworks am Framework für personalisierte Multimedia-Anwendungen MM4U. Der Entwurf und die Implementierung dieses Frameworks werden in Kapitel 20 beschrieben.

19.6 Fazit

In diesem Kapitel wurden Frameworks als ein Konzept zur Wiederverwendung von Software-Architekturen eingeführt. Die Unterscheidung zwischen objektorientierten und komponentenbasierten Frameworks ist dabei von wesentlicher Bedeutung, da sich diese beiden Arten von Frameworks hinsichtlich der eingesetzten Technologie zur Erweiterung und Anpassung an die Anforderungen einer konkreten Anwendung unterscheiden. Da die Entwicklung eines Frameworks sehr aufwendig ist, stellt sie immer eine Zukunftsinvestition dar [Pre97c], die sich erst dann lohnt, wenn das Framework in mehreren Anwendungen genutzt und wiederverwendet wird [Pre97a]. Der höhere Entwicklungsaufwand eines Frameworks zahlt sich also erst auf lange Sicht aus. Im Gegenzug wird jedoch signifikant der Entwicklungsaufwand für die einzelnen konkreten Anwendungen reduziert.

Für die erfolgreiche Entwicklung von Frameworks sind neben dem Softwaretechnischen Wissen auch organisatorische Aspekte wichtig (vgl. [RE99]). Die Entwicklung eines Frameworks in einem Projekt bedeutet immer auch eine Änderung der Organisationsstruktur des Entwicklerteams. So sollte das Entwicklerteam in Framework- und Anwendungsentwickler aufgeteilt werden, also in die, die das Framework entwickeln, und die, die es einsetzen und damit gegen die Framework-Anforderungen testen.

Frameworks gehören heute leider immer noch nicht zur allgemeinen Projektkultur [Pre99], sodass wirklich gut entwickelte Frameworks nur schwer zu finden sind. Es ist jedoch festzustellen, dass als Konsequenz aus der zunehmenden Anzahl an anwendungsspezifischen Frameworks sich Software-Architekten zukünftig genauso viel mit den Zwängen, die durch die Wahl und den Einsatz eines Frameworks entstehen, wie mit der Neuentwicklung von Anwendungen beschäftigen werden [BCK03]. Insgesamt ist daher eine weitere Zunahme der Bedeutung von Frameworks für die Software-Entwicklung zu erwarten.

Literatur

[AA03] Abbas, A.; Abbas, A.: Grid Computing: A Practical Guide to Technology and Applications. Rockland, MA, USA: Charles River Media, Inc., 2003

[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

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

[ACKM03] Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V.: Web Services: Concepts, Architectures and Applications.: Concepts, Architectures and Applications. Berlin: Springer-Verlag, 2003

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

[AEW96] Ackermann, P.; Eichelberg, D.; Wagner, B.: Visual programming in an object-oriented framework. In: Proc. Swiss Computer Science Conference, Oktober 1996

[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

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

[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] Andre, 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, U.: QuickTime. 2005, URL http://www.apple.com/quicktime/

[Ars06] Arsanjani, A.: Service-oriented modeling and architecture. 2006, URL http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

[AS07] Adobe Systems, U., Inc.: Macromedia Flash (SWF) and Flash Video (FLV) File Format Specification (Version 8). 2007, URL http://www.adobe.com/licensing/developer/

[B+01] Beck, K.; et al.: Principles behind the Agile Manifesto. 2001, URL http://www.agilemanifesto.org/principles.html

[Bal97] Balzert, H.: Lehrbuch der Software-Technik, 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

[Bar03] Barry, D. K.: Web Services and Service-Oriented Architecture: Your Road Map to Emerging IT. San Francisco: Morgan Kaufmann, 2003

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

[BBF+06] Barton, T.; Basney, J.; Freeman, T.; Scavo, T.; Siebenlist, F.; Welch, V.; Ananthakrishnan, R.; Baker, M.; Goode, M.; Keahey, K.: Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy. In: Proceedings of the 5th Annual PKI R&D Workshop, April 2006

[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

[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.: Design Rules, Vol. 1: The Power of Modularity, Bd. 1. 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

[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

[BDT99] Bradford, R. W.; Duncan, J. P.; Tarcy, B.: Simplified Strategic Planning: A No-Nonsense Guide for Busy People Who Want Results Fast! Chandler House Press, 1999

[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. Prentice Hall, 1997

[Bec01] Becker, F.: Organisational agility and the knowledge infrastructure. In: Journal of Corporate Real Estate 3 (2001), Nr. 1, S. 28–37

[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

[Ber07] Berbner, R.: Dienstgüteunterstützung für Service-orientierte Workflows. Darmstadt, 2007. Dissertation an der Technischen Universität Darmstadt

[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

[BGGK07] Becker, S.; Goldschmidt, T.; Gruschko, B.; Koziolek, H.: A Process Model and Classification Scheme for Semi-Automatic Meta-Model Evolution. In: Steffens, U.; Addicks, J. S.; Streekmann, N. (Hrsg.), MDD, SOA und IT-Management (MSI 2007), GITO-Verlag, 2007

[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

[BGR+05] Berbner, R.; Grollius, T.; Repp, N.; Heckmann, O.; Ortner, E.; Steinmetz, R.: An approach for the Management of Service-oriented Architecture (SoA) based Application Systems. In: Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures (EMISA 2005), Oct 2005

[BGR+07] Berbner, R.; Grollius, T.; Repp, N.; Eckert, J.; Heckmann, O.; Ortner, E.; Steinmetz, R.: Management of Service-oriented Architecture (SoA)-based Application Systems. In: Enterprise Modelling and Information Systems Architectures 1 (2007), Nr. 2, S. 14–26

[BH04] Buhl, H.-U.; Heinrich, B.: Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. 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

[BHS07a] Buschmann, F.; Henney, K.; Schmidt, D. C.: Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing. Wiley, 2007

[BHS07b] Buschmann, F.; Henney, K.; Schmidt, D. C.: Pattern-Oriented Software Architecture: On Patterns and Pattern Languages. Wiley, 2007

[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/pubs.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

[BKR05] Becker, J.; Kugeler, M.; Rosemann, M.: Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung, Bd. 5. Springer, 2005

[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

[BL06] Bichler, M.; Lin, K.-J.: Service-Oriented Computing. In: Computer 39 (2006), Nr. 3, S. 99–101

[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

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

[Bol03] Boll, S.: Vienna 4 U – What Web Services can do for personalized multimedia applications. In: 7th 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: ct (2003), Nr. 22, S. 92

[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, 2nd International Conference (SPLC2), Springer-Verlag, August 2002, Bd. 2379 von Lecture Notes in Computer Science

[Bos04] Bosch, J.: Software Variability Management. Jun 2004, URL `http://www.janbosch.com/01SVM-Introduction.pdf`

[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] : BPMI. 2005, URL `http://www.bpmi.org`

[BR90] Basili, V. R.; Rombach, 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

[BR05] Blaha, M.; Rumbaugh, J.: Object-Oriented Modeling and Design with UML, Bd. 2. Prentice Hall International, 2005

[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

[BRS06] Bonati, B.; Regutzki, J.; Schroter, M.: Enterprise Service Architecture for Financial Services – Taking SOA to the next level. Bonn: Galileo Press, 2006

[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

[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

[BSR+07] Berbner, R.; Spahn, M.; Repp, N.; Heckmann, O.; Steinmetz, R.: WSQoSX – A QoS architecture for Web Service workflows. In: 5th International Conference on Service-Oriented Computing (ICSOC 2007), Demo track, LNCS, 2007, S. 623–624

[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

[CESW04] Clark, T.; Evans, A.; Sammut, P.; Willans, J.: Applied Metamodeling – A foundation for Language Driven Development. Technical report, Xactium Ltd., 2004

[CH06] Czarnecki, K.; Helsen, S.: Feature-based survey of model transformation approaches. In: IBM Systems Journal 45 (2006), Nr. 3, S. 621–645

[Cha04] Chappell, D.: Enterprise Service Bus. O'Reilly, 2004

[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

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

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

[Com01] Committee, A.: Accountability. 2001, URL `http://www.atis.org/tg2k/_accountability.html`

[Com02] Commission, I. I. E.: INTERNATIONAL STANDARD IEC 61970-301: Energy management system application program interface (EMS-API) Part 301: Common Information Model (CIM) Base. Technischer Bericht, IEC, 2002

[Com05] Commission, I. I. E.: INTERNATIONAL STANDARD IEC 61970-1 Ed. 1: Energy management system application program interface (EMS-API) – Part 1: Guidelines and general requirements. Technischer Bericht, IEC, 11 2005

[Com07] Commission, I. I. E.: INTERNATIONAL STANDARD IEC 61968-1: Application integration at electric utilities – System interfaces for distribution management – Part 1: Interface architecture and general requirements. Technischer Bericht, IEC, 4 2007

[Coo08] Coorperation, M.: Domain Specific Language (DSL) Tools. 2008, URL http://www.microsoft.com/downloads/details.aspx?FamilyId=57A14CC6-C084-48DD-B401-1845013BF834&displaylang=en

[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

[Cor04] Corporation, O.: interMedia. Technischer Bericht, 2004, URL http://www.oracle.com/technology/products/intermedia/index.html

[Cor05a] Corporation, O.: interMedia Documentation. Technischer Bericht, 2005, URL http://www.oracle.com/technology/documentation/intermedia.html

[Cor05b] Corporation, O.: Oracle interMedia User's Guide 10g Release 2 (10.2). Technischer Bericht B14302-01, Juni 2005, URL http://download.oracle.com/docs/pdf/B14302_01.pdf

[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

[Dav93] Davenport, T. H.: Process Innovation – Reegineering Work through Information Technology. Boston: Harvard Business School Press, 1993

[DD07] Durst, M.; Daum, M.: Erfolgsfaktoren serviceorientierter Architekturen. In: HMD – Praxis der Wirtschaftsinformatik 44 (2007), Nr. 253, S. 18–27

[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

[Dev99] Devaux, S. A.: Total Project Control: A Managers Guide to Integrated Project Planning, Measuring, and Tracking. Wiley, 1999

[DHM99] Duke, D. J.; Herman, I.; Marschall, M. S.: PREMO: A Framework for Multimedia Middleware – Specification, Rationale, and Java Binding, Bd. 1591 von Lecture Notes in Computer Science. Springer-Verlag, 1999

[DHT01] Dashofy, E. M.; van der Hoek, A.; Taylor, R. N.: A Highly-Extensible, XML-Based Architecture Description Language. In: Kruchten, P.; Verhoef, C.; Kazman, R.; van Vliet, H. (Hrsg.), Proceedings of The Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, 2001, S. 103–112

[DHT05] Dashofy, E. M.; van der Hoek, A.; Taylor, R. N.: A comprehensive approach for the development of modular software architecture description languages. In: ACM Trans. Softw. Eng. Methodol. 14 (2005), Nr. 2, S. 199–245

[Die03] Dietzsch, A.: Positionierung eines Unternehmensarchitektur-Ansatzes: Erfahrung der Schweizerischen Mobiliar im Architekturmanagement. In: Enterprise Architecture und Enterprise Application Integration (EAI) – Proceedings des GI-Arbeitskreises 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/ewd02xx/EWD249.PDF`

[Dij82] Dijkstra, E. W.: Selected Writings on Computing: A Personal Perspective. New York, NY, USA: Springer-Verlag New York, Inc., 1982

[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

[DJMZ05] Dostal, W.; Jeckle, M.; Melzer, I.; Zengler, B.: Service-orientierte Architekturen mit Web-Services. Konzepte – Standards – Praxis. spektrum, 2005

[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

[DLP97] Duguay, C. R.; Landry, S.; Pasin, F.: From mass production to flexible/agile production. In: International Journal of Operations & Production Management 17 (1997), Nr. 12, S. 1183–1195

[Dmi04] Dmitriev, S.: Language Oriented Programming – The Next Programming Paradigm. In: onBoard electronic magazine (2004), URL http://www.onboard.jetbrains.com/is1/articles/04/10/lop/

[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

[D'S01] D'Souza, D.: Model-Driven Architecture and Integration – Opportunities and Challenges. Version 1.1. Technical report, Kinetum, 2001, URL ftp://ftp.omg.org/pub/docs/ab/01-03-02.pdf

[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

[EAA+04] Endrei, M.; Ang, J.; Arsanjani, A.; Chua, S.; Comte, P.; Krogdahl, P.; Luo, M.: Patterns: Service-Oriented Architecture and Web Services. IBM International Technical Support Organization, 1. Aufl., 2004

[EAK06] Erradi, A.; Anand, S.; Kulkarni, N.: Evaluation of Strategies for Integrating Legacy Applications as Services in a Service Oriented Architecture. In: Conference on Services Computing, 2006

[EF07] Eclipse Foundation, I.: SWT: The Standard Widget Toolkit. Februar 2007, URL http://www.eclipse.org/swt/

[EHH+08] Engels, G.; Hess, A.; Humm, B.; Juwig, O.; Lohmann, M.; Richter, J.-P.; Voß, M.; Willkomm, J.: Quasar Enterprise – Anwendungslandschaften serviceorientiert gestalten. dpunkt.verlag, 2008

[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

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

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

[Erl05] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005

[Erl07] Erl, T.: SOA: Principles of Service Design. Upper Saddle River, NJ: Prentice Hall, 2007

[ESN03] Engels, G.; Sauer, S.; Neu, B.: Integrating Software Engineering and User-centred Design for Multimedia Software Developments. 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

[e.V07] e.V., K.: K Desktop Environment – Conquer your Desktop! 2007, URL http://www.kde.org/

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

[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

[FK98a] Foster, I.; Kesselmann, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998

[FK98b] 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

[FKS+06] Foster, I.; Kishimoto, H.; Savva, A.; Berry, D.; Grimshaw, A.; Horn, B.; Maciel, F.; Siebenlist, F.; Subramaniam, R.; Treadwell, J.; Reich, J. V.: The Open Grid Services Architecture, Version 1.5, Open Grid Forum Final Document GFD.80. Technischer Bericht, OGF, September 2006, URL http://www.ogf.org/documents/GFD.80.pdf

[FKT01] Foster, I.; Kesselman, C.; Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: Lecture Notes in Computer Science 2150 (2001)

[FKTT98] Foster, I.; Kesselman, C.; Tsudik, G.; Tuecke, S.: A Security Architecture for Computational Grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, November 1998, S. 83–91

[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

[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

[Fos02] Foster, I.: What is the Grid? – a three point checklist. In: GRIDtoday 1 (2002), Nr. 6, URL http://www.gridtoday.com/02/0722/100136.html

[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: 14th European Conference on Object-Oriented Programming: Sophia Antipolis and Cannes, France, Juni 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

[FSN+01] Flickner, M.; Sawhney, H.; Niblack, W.; Ashley, J.; Huang, Q.; Dom, B.; Gorkani, M.; Hafner, J.; Lee, D.; Petkovic, D.; Steele, D.; Yanker, P.: Query by image and video content: the QBIC system. In: Readings in multimedia computing and networking, San Francisco, CA, USA: Morgan Kaufmann, 2001, S. 255–264

[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

[Gan03] Gandossy, R.: The Need for Speed. In: Journal of Business Strategy 24 (2003), Nr. 1, S. 29–33

[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, Juli 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

[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

[GJS03] Gruner, K.; Jost, C.; Spiegel, F.: Erfolgsorientierte Steuerung in allen Phasen des Lifecycles (IT-Professional). Vieweg, 2003

[GKG03] Gouscos, D.; Kalikakis, M.; Georgiadis, P.: An approach to modeling Web service QoS and provision price. In: Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on, 2003, S. 121–130

[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

[Gla02] Glass, R. L.: Facts and Fallacies of Software Engineering. Addison-Wesley, 2002

[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, URL `http://www.cas.ibm.ca/cascon/cfp.html`

[GMW00] Garlan, D.; Monroe, R. T.; Wile, D.: Acme: architectural description of component-based systems. In: Foundations of component-based systems, New York, NY, USA: Cambridge University Press, 2000, S. 47–67

[GNP95] Goldman, S. L.; Nagel, R. N.; Preiss, K.: Agile competitors and virtual organizations: strategies for enriching the customer. New York, NY: Van Nostrand Reinhold, 1995

[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

[GPR06] Gruhn, V.; Pieper, D.; Röttgers, C.: MDA. Effektives Softwareengineering mit UML2 und Eclipse. Springer-Verlag, 2006

[GRP06] Grimm, C.; Reiser, H.; Pattloch, M.: Sicherheit in Grids. In: Praxis der Informationsverarbeitung und Kommunikation 29 (2006), Nr. 3, S. 159–164

[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

[GW05] Gimnich, R.; Winter, A.: Workflows der Software-Migration. In: Softwaretechnik-Trends 25 (2005), Nr. 2, S. 22–24

[Hag03] Hagen, C.: Enterprise Application Integration – Flexibilisierung komplexer Unternehmensarchitekturen, Berlin: GITO-Verlag, Kap. Integrationsarchitektur der Credit Suisse. 2003, S. 61–81

[Hal05] Haller, S.: Dienstleistungsmanagement: Grundlagen – Konzepte – Instrumente, Bd. 3. Gabler Verlag, 2005

[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

[HBG+08] Hasselbring, W.; Büdenbender, A.; Grasmann, S.; Krieghoff, S.; Marz, J.: Muster zur Migration betrieblicher Informationssysteme. In: Tagungsband Software Engineering 2008, München: Köllen Druck+Verlag, Februar 2008, Lecture Notes in Informatics

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

[Heu07] van den Heuvel, W.-J.: Aligning Modern Business Processes and Legacy Systems – A Component-Based Perspective. Cooperative Information Systems, Cambridge, MA, USA: MIT Press, 2007

[HF06] Huntley, K.; Filippo, D. S.: Enabling Aspects to Enhance Service-Oriented Architecture. In: The Architecture Journal (2006)

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

[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, SIGS-DATACOM GmbH, Nr. 3, 1997, S. 40–51

[HHP04] Heise, C.; Heise, A.; Persson, A.: Bericht: Software-Projekt für Finanzämter gescheitert. Juli 2004, URL `http://www.heise.de/newsticker/meldung/48843`

[HHV06] Hess, A.; Humm, B.; Voß, M.: Regeln für serviceorientierte Architekturen hoher Qualität. In: Informatik Spektrum 6 (2006)

[HHVE07] Hess, A.; Humm, B.; Voß, M.; Engels, G.: Structuring Software Cities – A Multidimensional Approach. In: Proceedings of the 11th IEEE International EDOC Conference (EDOC 2007) The Enterprise Computing Conference. Annapolis, Maryland, USA, Oktober 2007

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

[HK03a] Hedman, J.; Kalling, T.: The business model concept: theoretical underpinnings and empirical illustrations. In: European Journal of Information Systems 12 (2003), Nr. 1, S. 49–59

[HK03b] 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

[HLÖ06] Heutschi, R.; Legner, C.; Österle, H.: Serviceorientierte Architekturen: Vom Konzept zum Einsatz in der Praxis. In: Data Warehousing 2006 – Integration, Informationslogistik und Architektur, Lecture Notes in Informatics (LNI) – Proceedings, September 2006, Bd. P-90, S. 361–382

[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

[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

[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

[IAK01] IBM; Adams, J.; Koushik, S.: Patterns for E-Business: A Strategy for Reuse. IBM Press, 2001

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

[IBM84] IBM: Business Systems Planning – Information Systems Planning Guide. Working Report IBM-Form GE20-0527-4, IBM, Atlanta, 1984

[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/rup/

[IBM06] IBM: developerWorks : SOA and Web services. 2006, URL http://www-128.ibm.com/developerworks/webservices,2006.

[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 (IEEE Std 1471-2000). IEEE Std. 1471-2000, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, USA, 2000, URL http://www.standards.ieee.org/reading/ieee/std/se/1471-2000

[IEE01] IEEE: Computing Curricula 2001 Computer Science. Technischer Bericht, The Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery, Dezember 2001, URL http://www.acm.org/education/curricula.html

[Inf04] Informatik, F.: Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten. Technischer Bericht, November 2004, URL http://www.ft-informatik.de

[Int04] International Telecommunication Union: ITU-T Recommendation Z.120: Message Sequence Chart (MSC). April 2004

[ISA] ISACA: COBIT-Homepage. URL http://www.isaca.org/cobit.htm

[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

[ISO06] ISO: Recommended Practice for Architectural Description of Software-Intensive Systems. 2006. IEEE Standard 1471-2000, ISO/IEC DIS 25961.

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

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

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

[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

[jbo07] jboss.org: The JBoss Application Server. 2007, URL http://www.jboss.org/

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

[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

[JM] Jones, S.; Morris, M.: A Methodology for Service Architectures. URL http://www.oasis-open.org/committees/download.php/15071

[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

[JRH+03] Jeckle, M.; Rupp, C.; Hahn, J.; Zengler, B.; Queins, S.: UML 2 glasklar. München: Hanser Wissenschaft, 2003

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

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

[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

[JVV01] Jonge, M. D.; Visser, E.; Visser, J.: XT: a bundle of program transformation tools. In: Electronic Notes in Theoretical Computer Science 44 (2001)

[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

[KBM02] Krauter, K.; Buyya, R.; Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. In: Software Practice and Experience 32 (2002), Nr. 2, S. 135–164

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

[KBS04] Krafzig, D.; Banke, K.; Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004

[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

[Kel07] Keller, W.: IT-Unternehmensarchitektur. dpunkt.verlag, 2007

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

[KFG02] Krallmann, H.; Frank, H.; Gronau, N.: Systemanalyse im Unternehmen: Vorgehensmodelle, Modellierungsverfahren und Gestaltungsoptionen. Oldenbourg, 4. Aufl., 2002

[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

[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

[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

[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

[KR06] Kühnemann, M.; Rünger, G.: Modellgetriebene Transformation von Legacy Business-Software. In: 3. Workshop Reengineering Prozesse (RePro2006), Software Migration, Nov 2006, Nr. 2 in Mainzer Informatik-Berichte, S. 20–21

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

[KST07] Krallmann, H.; Schönherr, M.; Trier, M. (Hrsg.): Systemanalyse im Unternehmen. München: Oldenbourg Verlag, 5. Aufl., 2007

[KT00] Kelly, S.; Tolvanen, J.-P.: Visual domain-specific modelling – Benefits and experiences of using metaCASE tools. In: Proceedings of the 1st International Workshop on Model Engineering, Cannes, 2000

[KT05] Kuhlin, B.; Thielmann, H. (Hrsg.): The Practical Real-Time Enterprise: Facts and Perspectives. Berlin et al.: Springer-Verlag, 2005

[Küt06] Kütz, M.: Kennzahlen in der IT – Werkzeuge für Controlling und Management. dpunkt.verlag, 2006

[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

[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

[LH07] Legner, C.; Heutschi, R.: SOA Adoption in Practice – Findings from Early SOA Implementations. In: Österle, H.; Schelp, J.; Winter, R. (Hrsg.), Proceedings of the 15th European Conference on Information Systems, 2007

[Lie07] Liebhart, D.: SOA goes real – Service-orientierte Architekturen erfolgreich planen und einführen. Hanser Verlag, 2007

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

[Lin00] Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley Information Technology Series, Harlow et al.: Addison Wesley, 2000

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

[LKN06] Luftman, J. N.; Kempaiah, R.; Nash, E.: Key Issues for IT Executives 2005. In: MISQ Executive 5 (2006), Nr. 2, S. 81–99

[LM04] Luftman, J. N.; McLean, E. R.: Key Issues for IT Executives. In: MISQ Executive 3 (2004), Nr. 2, S. 89–104

[LMS06] Lewis, G.; Morris, E.; Smith, D.: Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture. In: CSMR '06: Proceedings of the Conference on Software Maintenance and Reengineering, IEEE Computer Society Press, 2006, S. 15–23

[LMW07] Luhmann, T.; Meister, J.; Wulff, C.: Serviceorientierte Produktplattform für das Energiemanagement der Zukunft. In: Wirtschaftsinformatik 49 (2007), Nr. 5, S. 343–351

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

[Loh05] Lohse, M.: Network-Integrated Multimedia Middleware, Services, and Applications. Dissertation, Department of Computer Science, Saarland University, Juni 2005

[LRS05] Lohse, M.; Repplinger, M.; Slusallek, P.: Dynamic Media Routing in Multi-User Home Entertainment Systems. In: Proceedings of The Eleventh International Conference on Distributed Multimedia Systems (DMS), Knowledge Systems Institute, 2005, S. 271–276

[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

[LS02] Lohse, M.; Slusallek, P.: An Open Platform for Multimedia Entertainment Systems. In: EUROPRIX Scholars Conference at MindTrek Media Week; Tampere, Finland, 2002

[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

[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

[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

[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

[MB02] Mellor, S. J.; Balcer, M. J.: Executable UML: A Foundation for Model-Driven Architecture. Addison-Wesley, 2002

[MB06] Marks, E. A.; Bell, M.: Executive's guide to service-oriented architecture. Hoboken, NJ: John Wiley & Sons, 2006

[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)

[MC05a] Microsoft Corporation, U.: MFC Reference. 2005, URL http://msdn2.microsoft.com/en-us/library/d06h2x6e(VS.71).aspx

[MC05b] Microsoft Corporation, U.: Microsoft DirectShow 9.0. 2005, URL http://msdn.microsoft.com/en-us/library/ms783323.aspx

[MC05c] Microsoft Corporation, U.: .NET Development. 2005, URL http://msdn2.microsoft.com/en-us/netframework/

[McB07] McBride, M. R.: The Software Architect. In: Communications of the ACM 50 (2007), Nr. 5, S. 75–81

[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 International, 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

[MG06] Mens, T.; Gorp, P. V.: A Taxonomy of Model Transformation. In: Electronic Notes in Theoretical Computer Science 152 (2006), S. 125–142

[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

[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

[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

[MLM+06] MacKenzie, C. M.; Laskey, K.; McCabe, F.; Brown, P. F.; Metz, R.: OASIS Reference Model for Service Oriented Architecture V 1.0. August 2006, URL http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[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

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

[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

[Mon00] Monroe, R. T.: Capturing Software Architecture Design Expertise with Armani. Technischer Bericht CMU-CS-98-163, Carnegie Mellon University, School of Computer Science, September 2000. Version 2.3

[Mos05] Moses, T.: eXtensible Access Control Markup Language (XACML) Version 2.0. Technischer Bericht, OASIS, February 2005, URL http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[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

[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

[Mul04] Mullender, M.: Dealing with Concurrency: Designing Interaction Between Services and Their Agents. 2004, URL http://msdn2.microsoft.com/en-us/library/ms978508.aspx

[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, URL ftp://progftp.vub.ac.be/tech_report/2001/vub-prog-tr-01-03.pdf

[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

[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

[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] NMM: 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

[OAS05] OASIS: Web Service Implementation Methodology. 2005, URL http://www.oasis-open.org/committees/download.php/13420/fwsi-im-1.0-guidlines-doc-wd-publicReviewDraft.htm#_Toc105485380

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

[OLH08] Offermann, P.; Liebrecht, L.; Haarländer, N.: SOAM - Eine Methode zur Konzeption von betrieblichen Softwaresystemen entsprechend der SOA. In: ERP Management 4 (2008), Nr. 1, S. 32–35

[OMGa] OMG: Meta Object Facility (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 – Resource Page. Technischer Bericht, 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

[OMG97] OMG: Object Constraint Language. OMG Specification, Version 1.1 97-08-08, Object Management Group, 1997

[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/docs/ad/02-04-10.pdf

[OMG03] OMG: UML Profile for Schedulability, Performance and Time. 2003, URL http://www.omg.org/cgi-bin/doc?formal/2003-09-01

[OMG05] OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Final Adopted Specification ptc/05-11-01, Object Management Group, Needham, MA, November 2005, URL http://www.omg.org/cgi-bin/doc?ptc/2005-11-01

[OMG06] OMG: MOF 2.0 Core Specification (formal/2006-01-01). 2006, URL http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[OMG07] OMG: Unified Modeling Language: Superstructure – version 2.1.1. Februar 2007, URL http://www.omg.org/uml/

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

[Ope03] Opengroup: TOGAF Enterprise Edition Version 8.1. The Open Group, 2003, URL http://www.opengroup.org/architecture/togaf8-doc/arch/

[OSC] OSCI, L.: Online Services Computer Interface. URL http://www1.osci.de/sixcms/detail.php?id=1181

[OSHS07] Offermann, P.; Schröpfer, C.; Holschke, O.; Schönherr, M.: SOA: The Software-Architecture behind Service-Orientation. In: MDD, SOA und IT-Management Workshop, Oldenburg, GITO, 2007, S. 1–13

[Öst95] Österle, H.: Business Engineering in the Information Age – Heading for New Processes. New York: Springer-Verlag, 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

[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

[Pap03] Papazoglou, M. P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: WISE, IEEE Computer Society, 2003, S. 3–12

[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

[Par79] Parnas, D. L.: On the criteria to be used in decomposing systems into modules. In: (1979), S. 139–150

[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

[Pen03] Pender, T.: UML Bible. John Wiley & Sons, 2003

[PH90] Prahalad, C. K.; Hamel, G.: The Core Competence of the Corporation. In: Harvard Business Review 68 (1990), Nr. 3, S. 79–91

[PH06] Papazoglou, M. P.; van den Heuvel, W.-J.: Service-Oriented Design and Development Methodology. In: International Journal of Web Engineering and Technology (2006)

[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. 1, S. 6

[Ple] Pleuss, A.: Workshop on Model-Driven Development of Advanced User Interfaces, 2007. Web, URL `http://www.zmmi.de/mddaui2007/`

[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

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

[Por04] Porter, M. E.: Competitive Advantage: Creating and Sustaining Superior Performance. New York, NY, USA: Free Press, 2004

[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

[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/

[RFW+04] Raistrick, C.; Francis, P.; Wright, J.; Carter, C.; Wilkie, I.: Model Driven Architecture with Executable UML. Cambridge University Press, 2004

[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

[RHS05a] Richter, J.-P.; Haller, H.; Schrey, P.: Serviceorientierte Architektur. In: Informatik-Spektrum 28 (2005), Nr. 5, S. 413–416

[RHS05b] 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 Web-basierten 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

[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: 4th 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

[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

[RWR06] Ross, J. W.; Weill, P.; Robertson, D.: Enterprise Architecture as Strategy. Creating a Foundation for Business Execution. Boston, MA, USA: Harvard Business School Press, 2006

[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, 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, URL `http://research.microsoft.com/~cszypers/events/WCOP2005/`

[SB05b] Scherp, A.; Boll, S.: Context-driven smart authoring of multimedia content with xSMART. In: Proc. of the 13th annual ACM Int. Conf. on Multimedia; Hilton, Singapore, ACM Press, 2005, S. 802–803

[SB05c] 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

[SB05d] 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

[Sch99] Schmid, H. A.: Framework Design by Systematic Generalization. In: Fayad et al. [FSJ99a], Kap. 15, S. 353–378

[Sch01a] Schmitt, J. B.: Heterogeneous Network Quality of Service Systems. Norwell, MA, USA: Kluwer Academic Publishers, 2001

[Sch01b] Scholl: Napster Messages. April 2001, URL http://opennap.sourceforge.net/napster.txt

[Sch01c] 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

[Sch04a] Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Victoria, British Columbia: Trafford Publishing, 2. Aufl., 2004, URL http://www.enterprise-architecture.info/EA_Book_EAFrameworks.htm

[Sch04b] 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

[Sch07a] Scherp, A.: A Component Framework for Personalized Multimedia Applications. Dissertation, Carl von Ossietzky University of Oldenburg, School of Computing Science, Business Administration, Economics and Law, Department of Computing Science, 2007, URL http://ansgarscherp.net/dissertation/

[Sch07b] Schmedes, T.: Modellierung service-orientierter Architekturen in der Energieversorgung. In: Software Engineering 2007 – Beiträge zu den Workshops, GI, 2007, S. 187–194

[SDS01] Stojanovic, Z.; Dahanayake, A.; Sol, H.: A Methodology Framework for Component-Based System Development Support. In: International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, 2001

[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/arch_duties.html

[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

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

[SG04] Schmerl, B.; Garlan, D.: AcmeStudio: Supporting Style-Centered Architecture Development (Research Demonstration). In: ICSE '04: Proceedings of the 26th International Conference on Software Engineering, Washington, DC, USA: IEEE Computer Society, 2004, S. 704–705

[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/lpt/a/p2p/2000/11/24/shirky1-whatisp2p.html`

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

[Sie07] Siedersleben, J.: SOA revisited: Komponentenorientierung bei Systemlandschaften. In: Wirtschaftsinformatik 49 (2007), S. 110–117

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

[Sim93] Simon, H. A.: The Organization of Complex Systems, Hierarchy Theory: The Challenge of Complex Systems. In: Pattee, H. H. (Hrsg.), The International Library of Systems Theory and Philosophy, Braziller, G., 1993

[Sim95] Simonyi, C.: The Death of Computer Languages, The Birth of Intentional Programming. Technischer Bericht, Microsoft, 1995, URL `ftp://ftp.research.microsoft.com/pub/tr/tr-95-52.doc`

[SJB04] Satzinger, J. W.; Jackson, R. B.; Burd, S. D.: System Analysis and Design in a Changing World. Course Technology Press, 2004

[SK97] Sztipanovits, J.; Karsai, G.: Model-Integrated Computing. In: Computer 30 (1997), Nr. 4, S. 110–111

[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

[Som89] Sommerville, I.: Software Engineering. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989

[Som04] Sommerville, I.: Software Engineering. Pearson and 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

[SS07] Schelp, J.; Stutz, M.: A Balanced Scorecard Approach to Measure the Value of Enterprise Architecture. In: Journal of Enterprise Architecture 3 (2007), Nr. 4

[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

[ST07] Starke, G.; Tilkov, S. (Hrsg.): SOA-Expertenwissen. dpunkt.verlag, 2007

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

[Sta00] for Electro-technical Standardisation), C. E. C.: CENELEC EN 50128: Railway Applications: Software for Railway Control and Protection Systems CENELEC. Brussels, 2000

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

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

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

[Sun05] Sun, M.: JXTA v2.3.x: Java Programmer's Guide. 2005, URL `http://www.it.uom.gr/teaching/ParallelDistributedJava/software/jxta/JxtaProgGuide_v2.3.pdf`

[Sun06] Sun, D. N. S.: EJB 3.0 Specification Final Release 3.0. 2006, URL `http://java.sun.com/products/ejb/index.jsp`

[Sun08a] Sun, D. N. S.: Java 2 Platform, Enterprise Edition (J2EE). 1994 - 2008, URL `http://java.sun.com/javaee/index.jsp`

[Sun08b] Sun, D. N. S.: The AWT in 1.0 and 1.1. April 1994 - 2008, URL `http://java.sun.com/products/jdk/awt/`

[Sun08c] Sun, D. N. S.: Java Foundation Classes (JFC). Technischer Bericht, Sun Microsystems, 1994–2008, URL `http://java.sun.com/products/jfc/download.html`

[Sun08d] Sun, D. N. S.: Java Media Framework API (JMF). Technischer Bericht, 1994–2008, URL http://java.sun.com/javase/technologies/desktop/media/jmf/

[Sun08e] Sun, D. N. S.: JMF 2.0 Documentation Downloads. Technischer Bericht, 1994–2008, URL http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/specdownload.html

[SVEH07] Stahl, T.; Völter, M.; Efftinge, S.; Haase, A.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. dpunkt.verlag, 2. Aufl., 2007

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

[SZ99] Sharifi, H.; Zhang, Z.: A methodology for achieving agility in manufacturing organisations: An introduction. In: International Journal of Production Economics 62 (1999), Nr. 1-2, S. 7–22, URL http://dx.doi.org/10.1016/S0925-5273(98)00217-5

[TA07] Trolltech ASA, N., Oslo: Qt. 2007, URL http://trolltech.com/products/qt

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

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

[Tan96] Tanenbaum, A. S.: Computer Networks. Prentice Hall, 1996

[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

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

[THC04] Tang, A.; Han, J.; Chen, P.: A Comparative Analysis of Architecture Frameworks. Technical Report SUTIT-TR2004.01, Swinbourne University of Technology, Centre for Component Software and Enterprise Systems, Swinbourne, 2004 2004, URL http://www.it.swin.edu.au/centres/TechnicalReports/2004/SUTIT-TR2004.01.pdf

[Thi06] Thilloy, C.: Enterprise Service Oriented Methodology. 2006, URL http://www.multiforce.com/soa/methodology.html

[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/

[TTW05] Tilkov, S.; Tilly, M.; Wilms, H.: Lose Kopplung mit Web-Services einfach gemacht. In: Java Spektrum 5 (2005)

[TWE+04] Tuecke, S.; Welch, V.; Engert, D.; Pearlman, L.; Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, RFC 3820. Technischer Bericht, IETF, 2004

[UG07] Uslar, M.; Grüning, F.: Zur semantischen Interoperabilität in der Energiebranche: CIM IEC 61970. In: Wirtschaftsinformatik 49 (2007), Nr. 4, S. 295–303

[Uhl08] Uhl, A.: Model-Driven Development in the Enterprise. In: IEEE Software 25 (2008), Nr. 1, S. 46–49

[USL+05] Uslar, M.; Schmedes, T.; Luhmann, T.; Lucks, A.; Winkels, L.; Appelrath, H.-J.: Interaction of EMS related systems by using the CIM standard. In: Filho, W. L.; Gomez, J. M.; Rautenstrauch, C. (Hrsg.), ITEE 2005: Second International ICSC Symposium on Information Technologies in Enviromental Engineering, Shaker Verlag, Aachen, 2005, S. 596–610

[USLA05] Uslar, M.; Schmedes, T.; Luhmann, T.; Appelrath, H.-J.: Eine serviceorientierte Architektur für das dezentrale Energiemanagement. In: Tagungsband GI-Jahrestagung, Band 2, 2005, S. 622–626

[vCG+00] van Ossenbruggen, J. R.; Cornelissen, F. J.; Guerts, J. P. T. M.; Rutledge, L. W.; Hardman, H. L.: Cuypers: a semi-automatic hypermedia presentation system. Technischer Bericht INS-R0025, CWI, The Netherlands, Dezember 2000

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

[VF98] Vokurka, R. J.; Fliedner, G.: The journey toward agility. In: Industrial Management & Data Systems 98 (1998), Nr. 4, S. 165–171

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

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

[VS06] Völter, M.; Stahl, T.: Model-Driven Software Development. John Wiley & Sons, 2006

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

[W3C04] W3C: Web Services Glossary. Februar 2004, URL `http://www.w3.org/TR/ws-gloss/`

[W3C05a] W3C: Scalable Vector Graphics (SVG) Full 1.2 Specification. April 2005. Http://www.w3.org/TR/SVG12/

[W3C05b] W3C: Synchronized Multimedia Integration Language (SMIL 2.1). Dezember 2005, URL `http://www.w3.org/TR/SMIL2/`

[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

[War94] Ward, M. P.: Language-Oriented Programming. In: Software – Concepts and Tools 15 (1994), Nr. 4, S. 147–161

[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

[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

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

[WF06] Winter, R.; Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In: Society, I. C. (Hrsg.), EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006), Los Alamitos, CA, USA: IEEE Computer Society, 2006, S. 1–8 (CDROM)

[WFK+04] Welch, V.; Foster, I.; Kesselman, C.; Mulmo, O.; Pearlman, L.; Tuecke, S.; Gawor, J.; Meder, D.; Siebenlist, F.: X.509 Proxy Certificates for Dynamic Delegation. In: Proceedings of the 3rd Annual PKI R&D Workshop, April 2004

[WH03] Willkomm, J.; Humm, B.: i-Portal-Patterns – Lösungsmuster für wiederkehrende Anforderungen. In: INFORMATIK 2003 – Innovative Informatikanwendungen, Band 2, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2003, Bd. 35 von Lecture Notes in Informatics, S. 329–334

[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

[Wir71] Wirth, N.: Program development by stepwise refinement. In: Commun. ACM 14 (1971), Nr. 4, S. 221–227

[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

[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

[WS08] Winter, R.; Schelp, J.: Enterprise Architecture Governance: The Need for a Business-to-IT Approach. In: N.N. (Hrsg.), Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC2008), Mar 16-20, 2008, Fortaleza, Cears, Brazil, New York, NY, USA: ACM Press, 2008, S. (in Vorb.)

[WV01] Weill, P.; Vitale, M. R.: Place to Space: Migrating to eBusiness Models. Boston, MA: Harvard Business School Press, 2001

[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

[WZ07] Winter, A.; Ziemann, J.: Model-based Migration to Service-oriented Architectures – A Project Outline. In: Sneed, H. (Hrsg.), CSMR 2007, 11th European Conference on Software Maintenance and Reengineering, Workshops, März 2007, S. 107–110

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

[YSG99] Yusuf, Y. Y.; Sarhadi, M.; Gunasekaran, A.: Agile manufacturing: the drivers, concepts and attributes. In: International Journal of Production Economics 62 (1999), Nr. 1-2, S. 33–43, URL http://dx.doi.org/10.1016/S0925-5273(98)00219-9

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

[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

[ZLKW06] Ziemann, J.; Leyking, K.; Kahl, T.; Werth, D.: Enterprise Model driven Migration from Legacy to SOA. In: Gimnich, R.; Winter, A. (Hrsg.), Workshop Software-Reengineering und Services, 2006, Fachberichte Informatik, S. 18–27

[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. 13–13

[ZS00] Zhang, Z.; Sharifi, H.: A methodology for achieving agility in manufacturing organisations. In: International Journal of Operations & Production Management 20 (2000), Nr. 4, S. 496–512

[Zuk97] Zukowski, J.: Java AWT Reference. O'Reilly & Associates, 1997, URL http://www.oreilly.com/catalog/javawt/book/index.html

[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

[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