21 Peer-to-Peer-Architekturen

Ludger Bischofs, Wilhelm Hasselbring, Timo Warns

Peer-to-Peer-Architekturen sind Architekturen, die durch die Kommunikation gleichgestellter Peers gekennzeichnet sind. Im Gegensatz zu Client-Server-Architekturen, bei denen Clients und Server unterschiedliche Funktionalität besitzen, verfügen alle Peers eines P2P-Systems prinzipiell über die gleiche Funktionalität. Dies ermöglicht die Realisierung von Systemen mit besseren Skalierungseigenschaften, als es mit Client-Server-Architekturen möglich ist.

Genau wie das Client-Server-Kommunikationsmodell ist Peer-to-Peer (P2P) ein universelles Kommunikationsmodell, das in zahlreichen Anwendungsbereichen eingesetzt werden kann und eingesetzt wird. Aktuelle Anwendungsbereiche sind u. a. Dateitauschbörsen (z. B. Napster, `http://www.napster.com/`), Instant Messaging (z. B. ICQ, `http://www.icq.com/`), Internettelefonie (z. B. Skype, `http://www.skype.com/`), kollaboratives Arbeiten (z. B. Groove, `http://www.groove.net/`) und verteiltes Rechnen (z. B. OurGrid, `http://www.ourgrid.org/`). Wesentlicher Aspekt dieser Systeme ist die Nutzung von Ressourcen, die dezentral in einem System verteilt sind, wie z. B. Rechenleistung, Speicher und Netzwerkbandbreite.

Der Entwurf einer P2P-Architektur erfordert eine Abwägung zwischen verschiedenen Qualitätseigenschaften, über die das resultierende System verfügen soll. Dieses Kapitel diskutiert unabhängig von einer konkreten Applikation grundlegende Ausprägungen von P2P-Architekturen und illustriert diese Ausprägungen an konkreten Architekturen, die in verschiedenen Systemen zum Einsatz kommen.

21.1 Definitionen

Definitionen und Beschreibungen rund um den Begriff Peer-to-Peer sind in der Literatur sehr heterogen und zum Teil widersprüchlich. Eine häufig zitierte Definition von Shirky stellt P2P als eine Klasse von Internetapplikationen dar, die Ressourcen dezentral nutzen [Shi00]. Steinmetz und Wehrle verstehen unter P2P-Systemen sich selbst organisierende Systeme gleichgestellter, autonomer, vernetzter Einheiten (sogenannte Peers), die ohne zentrale Koordination mit dem

Ziel der gegenseitigen Nutzung von Ressourcen arbeiten [SW04]. Schollmeier versteht den Begriff des P2P-Netzwerks als eine verteilte Netzwerkarchitektur und grenzt diese gegenüber der Client-Server-Architektur ab [Sch01c].

Um eine differenzierte Betrachtungsweise zu ermöglichen und Missverständnissen vorzubeugen, werden nun einige P2P-Begriffe für dieses Kapitel definiert.

Definition (Peer-to-Peer) *Peer-to-Peer (P2P) ist ein Kommunikationsmodell gleichgestellter Einheiten (Peers), die direkt oder indirekt miteinander kommunizieren.*

Die Gleichstellung von Peers besteht darin, dass alle Peers die gleichen Dienste anbieten und in Anspruch nehmen können, d. h., dass ein Peer sowohl Client- als auch Serverfunktionalität übernehmen kann. Ein Peer wird daher auch als *Servent* bezeichnet [Sch01c], abgeleitet von den Begriffen Server und Client. In vielen Fällen wird ein solcher Servent durch eine Software-Applikation realisiert.

Definition (Peer-to-Peer-Applikation) *Eine Peer-to-Peer-Applikation ist ein Software-Programm, das die Funktionalität eines einzelnen Peers implementiert.*

Ein Peer muss natürlich nicht zwingend als Software realisiert sein; auch Umsetzungen in Hardware sind denkbar. Unabhängig davon werden Peers in einem P2P-Netzwerk organisiert, das die einzelnen Peers miteinander verbindet, um deren Kommunikation zu ermöglichen.

Definition (Peer-to-Peer-Netzwerk) *Ein Peer-to-Peer-Netzwerk ist ein Netz aus miteinander verbundenen Peers.*

Zwei Peers gelten dabei als miteinander verbunden, wenn sie untereinander Informationen austauschen können. P2P-Netzwerke sind meist logische Netzwerke, die auf einem physikalischen Netzwerk aufbauen und eine eigene vom physikalischen Netzwerk unabhängige Topologie aufweisen. Formal werden P2P-Netzwerke als Graphen mit Peers als Knoten und Verbindungen als Kanten modelliert. Sind zwei Peers direkt miteinander verbunden (gibt es also eine Kante zwischen beiden Peers im Graphen des P2P-Netzwerks), werden sie *Nachbarn* genannt. Ein Peer kann nur mit seinen Nachbarn direkt kommunizieren. Mit anderen Peers muss indirekt kommuniziert werden, d. h., dass die jeweiligen Peers, die auf einem Pfad zwischen zwei kommunizierenden Peers liegen, bei der Kommunikation vermitteln. Für eine solche vermittelnde Kommunikation und für den Aufbau, die Wartung und die Nutzung des P2P-Netzwerks werden P2P-Protokolle benötigt.

Definition (Peer-to-Peer-Protokoll) *Ein Peer-to-Peer-Protokoll beschreibt die Regeln und Konventionen, nach denen die Kommunikation in einem Peer-to-Peer-Netzwerk erfolgt.*

Eng verwandt zum P2P-Netzwerk ist der Begriff des P2P-Systems. P2P-Systeme nutzen P2P-Netzwerke zur Umsetzung der an sie gestellten Anforderungen. In Analogie zu Rechnernetzen und verteilten Systemen [Tan96] ist der wesentliche Unterschied, dass P2P-Systeme versuchen, die Existenz mehrerer Peers transparent zu halten.

Definition (Peer-to-Peer-System) *Ein Peer-to-Peer-System ist ein verteiltes System aus Peers, die ein P2P-Netzwerk nutzen.*

Wie allen Systemen liegt auch einem P2P-System eine Architektur zugrunde.

Definition (Peer-to-Peer-Architektur) *Eine Peer-to-Peer-Architektur ist die Architektur eines Peer-to-Peer-Systems.*

Insbesondere umfasst eine P2P-Architektur nicht nur die Architektur der einzelnen Applikation, sondern auch Aspekte des Netzwerks und seiner Topologie. Im Folgenden wird auf Peer-to-Peer-Architekturen als Schwerpunkt dieses Kapitels detaillierter eingegangen.

21.2 Klassifikation von P2P-Architekturen

Das P2P-Kommunikationsmodell kann durch verschiedene Architekturen realisiert werden. Diese Architekturen werden in erster Linie danach klassifiziert, wie strikt sie die Idee des Modells umsetzen. Reine P2P-Architekturen gelten als strenge Umsetzung, weil dabei alle Knoten als Servents auftreten. Auf der einen Seite können mit solchen Architekturen mögliche Einschränkungen von Client-Server-Architekturen aufgehoben werden, wie z. B. zu geringer verfügbarer Speicherplatz. Auf der anderen Seite unterliegen reine P2P-Architekturen aber anderen Problemen, die in Client-Server-Systemen eine geringere Rolle spielen. Zum Beispiel erfordert die Suche nach Ressourcen in reinen P2P-Architekturen ein höheres Nachrichtenaufkommen als in Client-Server-Architekturen. Dort muss evtl. nur ein einzelner Server befragt werden, während in der reinen P2P-Architektur meist mehrere oder sogar alle Peers befragt werden müssen. Die Performance kann dadurch im Vergleich zu Client-Server-Systemen beeinträchtigt werden. Derartige Probleme werden in hybriden Architekturen umgangen, indem P2P- und Client-Server-Konzepte gemischt werden.

21.2.1 Reine P2P-Architekturen

Reine P2P-Architekturen setzen das P2P-Kommunikationsmodell streng um und verzichten dementsprechend komplett auf zentrale Server. Alle Peers treten als Servents auf, sodass Dienste auf viele Knoten verteilt werden können und solche

Architekturen als dezentral gelten. Reine P2P-Architekturen lassen sich danach unterscheiden, ob sie strukturierte oder unstrukturierte Netzwerktopologien benutzen.

In *strukturierten P2P-Architekturen* unterliegt die Netzwerktopologie wohldefinierten Beschränkungen, sodass die Topologie einer logischen Struktur folgt. Verschiedene Ausprägungen sind denkbar, wie z. B. baumartige Strukturen (siehe Abbildung 21.1), die die Evolution des Netzwerks einschränken und damit Vorhersagen über das Verhalten des Systems erlauben. Strukturierte Topologien

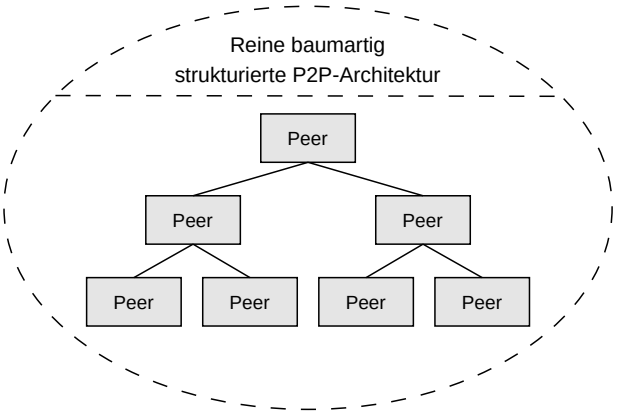

Abbildung 21.1: Reine baumartig strukturierte P2P-Architektur

erfordern jedoch einen höheren Aufwand bei der Pflege der Topologie. Zum Beispiel muss ein passender Platz in der Topologie für neu hinzukommende Peers gefunden werden. In der Regel müssen dafür mehrere bereits verbundene Peers befragt werden, bevor die passenden Nachbarn gefunden werden. Trennt sich ein Peer vom Netzwerk, sind u. U. komplexe Umformungen der Topologie notwendig bis die gewünschte Struktur wieder hergestellt ist. In einem System, bei dem Peers häufig neu hinzukommen bzw. die Verbindung trennen, kann dies die Performance signifikant beeinträchtigen.

Bei *unstrukturierten Architekturen* wird die Topologie nicht beschränkt (siehe Abbildung 21.2). Daher benötigt das P2P-Netzwerk nur eine minimale Pflege, wenn Peers dem System beitreten bzw. es verlassen. Ein hinzukommender Peer muss nur einen anderen, bereits verbundenen Peer finden, der unmittelbar Nachbar des neuen Peers werden kann.

Die Evolution eines unstrukturierten P2P-Netzwerks ist kaum vorhersagbar. Aussagen über Eigenschaften der Systeme sind damit schwierig, da diese häufig von Eigenschaften der Topologie abhängen. Oftmals haben unstrukturierte Systeme allerdings die Tendenz, sogenannte *Kleine-Welt-Netzwerke* zu bilden [Mil67, WS98a], auch wenn dies nicht explizit vorgesehen ist. Solche Netzwerke

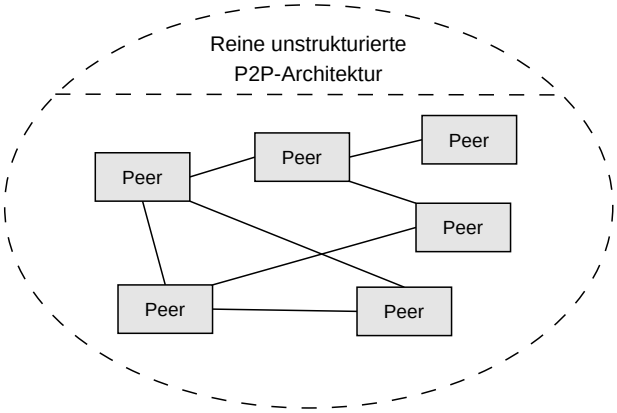

Abbildung 21.2: Reine unstrukturierte P2P-Architektur

sind stark geclustert, haben relativ kurze Weglängen zwischen einzelnen Peers und erlauben so eine effiziente Kommunikation.

Der Begriff der Strukturiertheit in P2P-Kontexten wird in der Literatur inkonsistent verwendet. In diesem Kapitel wird der Begriff für logische Strukturen von Netzwerktopologien benutzt. Er wird aber auch für Architekturen benutzt, bei denen die Platzierung von Daten nicht beliebig ist, sondern gewissen Regeln und Konventionen folgt, um z. B. Suchoperationen zu optimieren.

21.2.2 Hybride P2P-Architekturen

Hybride P2P-Architekturen setzen als Mischung von reinen P2P- und Client-Server-Architekturen gezielt zentrale Knoten ein, um Anforderungen an das System zu erfüllen, die nur schwer mit reinen P2P-Architekturen zu erfüllen sind. Hybride P2P-Architekturen lassen sich in zentralisierte und Super-Peer-Architekturen unterscheiden.

In *zentralisierten P2P-Architekturen* werden Knoten mit reiner Serverfunktionalität eingesetzt (siehe Abbildung 21.3), die exklusiv solche Dienste übernehmen, deren Umsetzung in einer reinen P2P-Architektur problematisch ist. Zum Beispiel setzte Napster aus Effizienzgründen zentrale Server ein, um die Suche nach Dateien ausreichend performant zu halten (siehe Abschnitt 21.4.1). Im Unterschied zu Client-Server-Architekturen werden nur einige Dienste exklusiv von Servern übernommen. Andere Dienste werden nach dem P2P-Kommunikationsmodell angeboten.

Super-Peer-Architekturen, die auch als hierarchische Architekturen bezeichnet werden, bieten einen Mittelweg zwischen dezentralen und zentralisierten Ansätzen. Ein Super-Peer ist ein Peer, der sowohl als Server für eine Menge von

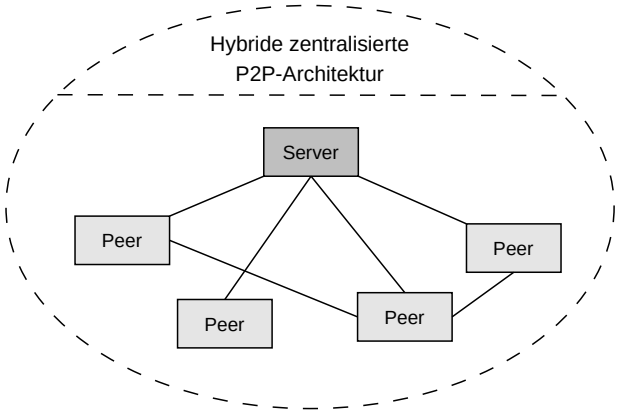

Abbildung 21.3: Hybride zentralisierte P2P-Architektur

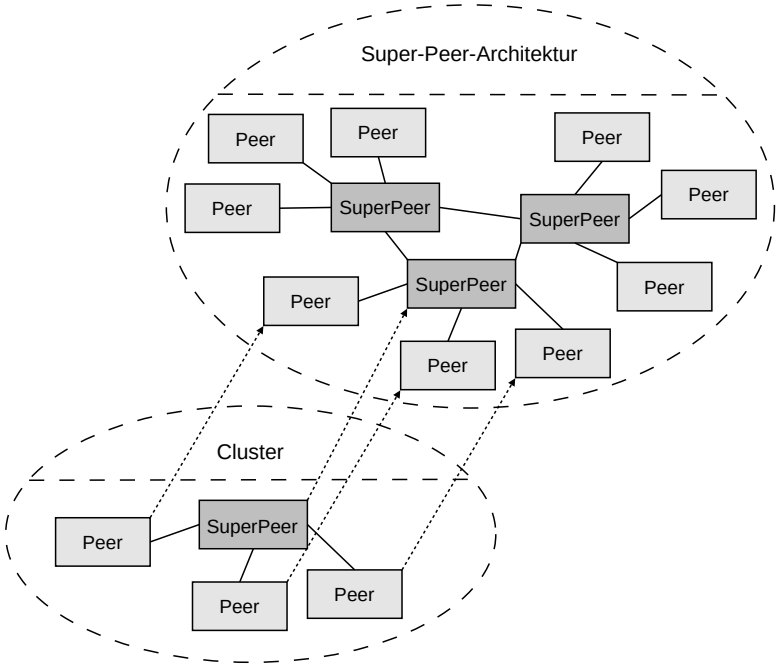

Abbildung 21.4: Hybride Super-Peer-Architektur

normalen Peers als auch als Servent für eine Menge von anderen Super-Peers auftritt (siehe Abbildung 21.4). Normale Peers sind Servents, die jeweils nur mit einem Super-Peer verbunden sind. Dieser tritt ihnen gegenüber als zentraler Server auf. Ein Super-Peer mit seinen zugehörigen Peers wird *Cluster* genannt. Da ein Super-Peer für seine Peers ein »Single Point of Failure« ist, wird er häufig redundant gehalten. In einem Cluster gibt es dann mehrere Super-Peers, die sich gleichartig verhalten und zusammen einen logischen Super-Peer bilden. Bei Ausfall eines Super-Peers bleiben seine Dienste durch die Redundanz der Super-Peers im Cluster verfügbar.

21.3 Schichten einer P2P-Applikation

Bei Peers handelt es sich um gleichgestellte Einheiten, die typischerweise sowohl Client- als auch Serverfunktionalität bereitstellen. Ein Nutzer kann die Funktionalität eines P2P-Systems, wie z. B. die Suche nach einer Datei, über die Benutzungsoberfläche einer P2P-Applikation verwenden. Die Funktionalität wird durch P2P-Dienste umgesetzt, die sich in allgemeine, anwendungs- und domänenspezifische Dienste unterscheiden lassen. Die Dienste nutzen das zugrunde liegende P2P-Netzwerk für die Kommunikation zwischen Peers, das die gegenseitige Verwendung von Ressourcen ermöglicht. Daraus ergibt sich im Allgemeinen eine Schichtenarchitektur einer P2P-Applikation, wie sie in Abbildung 21.5 dargestellt ist.

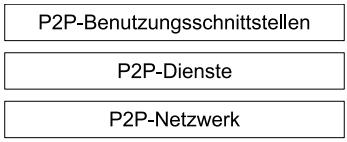

Abbildung 21.5: Typische Schichten einer P2P-Applikation

21.3.1 Benutzungsschnittstellen

Die Schicht der Benutzungsschnittstellen umfasst Schnittstellen einer P2P-Applikation zu ihrer Umgebung, von der sie benutzt wird. Eine Applikation bietet z. B. anderen Programmen oder Benutzern Schnittstellen an, um ihre Dienste in Anspruch nehmen zu können. Meist handelt es sich bei Peers um Applikationen, die direkt von Anwendern bedient werden, sodass grafische Benutzungsoberflächen angeboten werden.

Die Benutzungsschnittstellen abstrahieren von den Diensten der zugrunde liegenden P2P-Dienstschicht und bieten integrierte Schnittstellen, um funktionale Anforderungen der Applikationsdomäne zu erfüllen. Zum Beispiel verfügen *Instant-Messenger*-Applikationen über P2P-Dienste zum Versand von Nachrichten und sogenannte *Awareness*-Dienste, die über den Status der Benutzer wachen. Eine grafische Benutzungsoberfläche in der Schicht der Benutzungsschnittstellen abstrahiert von diesen Diensten und stellt deren Funktionalität grafisch dar, indem sie z. B. den Status der Benutzer visualisiert.

21.3.2 P2P-Dienste

Die Schicht der P2P-Dienste liegt unter den Benutzungsschnittstellen und bietet sowohl anwendungs- und domänenspezifische Dienste als auch allgemeine Dienste wie Indexierung und Suche. Ein anwendungsspezifischer Dienst kann beispielsweise das Auffinden eines Benutzers im P2P-Netzwerk oder das Video-Streaming zwischen zwei Peers sein, während ein allgemeiner Dienst das Auffinden eines Peers realisieren könnte. Eine klare Zuordnung bzw. Abgrenzung der Dienste ist dabei nicht immer möglich.

In Super-Peer-Architekturen besitzen Super-Peers üblicherweise zusätzliche Dienste gegenüber normalen Peers. Dies kann z. B. ein Index über die zur Verfügung stehenden Dateien auf den Peers des Clusters sein. Die Architektur eines Super-Peers und die eines normalen Peers können sich deshalb in Teilbereichen unterscheiden. In reinen P2P-Systemen dagegen bietet jeder Peer üblicherweise die gleichen Dienste, weshalb die beteiligten Peers meist eine sehr ähnliche oder identische Architektur aufweisen.

21.3.3 P2P-Netzwerk

Bei einem P2P-Netzwerk handelt es sich um ein logisches Netzwerk, das auf einem physikalischen Netzwerk aufsetzt und die Verbindung und Kommunikation zwischen den Peers sicherstellt. P2P-Netzwerke bauen auf einem zugrunde liegenden physikalischen Netzwerk, wie dem Internet, eine eigene Topologie auf, die nicht der Topologie des physikalischen Netzwerks entsprechen muss. Zum Beispiel sind Peers, die Nachbarn im logischen Netzwerk sind, im Allgemeinen keine Nachbarn im physikalischen Netzwerk.

Das P2P-Netzwerk wird durch die beteiligten Peers gebildet, die das verwendete Kommunikationsprotokoll kennen müssen, um Nachrichten mit anderen Peers austauschen zu können. Diese Netzwerkschicht stellt den P2P-Diensten Primitive, wie z. B. virtuelle Pipes und Push- und Pull-Mechanismen, zur Verfügung, um die Kommunikation mit anderen Peers zu ermöglichen. In der Umsetzung sind Funktionalitäten wie z. B. das Routen von Nachrichten oder die Verschlüsselung des Kommunikationskanals enthalten. Eine klare Abgrenzung zur darüberliegenden P2P-Dienstschicht ist allerdings nicht immer möglich. Prin-

zipiell beinhaltet die P2P-Netzwerkschicht Low-Level-Funktionalität, die zum Betrieb des P2P-Netzwerks notwendig ist.

Ein logisches Netzwerk kann auf einem beliebigen physikalischen Netzwerk aufbauen oder auch verschiedene physikalische Netzwerke integrieren. Meist werden P2P-Architekturen aber für Internetapplikationen entwickelt, die auf TCP/IP-Netzwerken basieren.

Das physikalische Netzwerk muss Funktionalitäten bereitstellen, die benachbarten Peers die Kommunikation ermöglichen. Zum Beispiel können *send*- und *receive*-Primitive zum Senden und Empfangen von Nachrichten angeboten werden. Um auch Nicht-Nachbarn zu erreichen, können darauf aufbauend Funktionalitäten zum Routing von Nachrichten im virtuellen P2P-Netzwerk umgesetzt werden, soweit sie nicht schon vom physikalischen Netzwerk bereitgestellt werden. Falls das physikalische Netzwerk ausreichend Funktionalität zum Betrieb eines P2P-Systems bietet, kann auf ein eigenes virtuelles Netzwerk verzichtet werden. Die Topologie des virtuellen Netzwerks entspricht dann der des physikalischen.

21.4 Beispiele von P2P-Architekturen

In diesem Abschnitt werden die dynamischen Aspekte einiger bekannter P2P-Architekturen beschrieben und als Beispiele für unterschiedliche Arten von P2P-Architekturen herangezogen. Napster ist ein hybrides zentralisiertes P2P-System, während Freenet dezentral und weitgehend unstrukturiert aufgebaut ist und daher zu den reinen P2P-Systemen zählt. Als reine, dezentrale, aber strukturierte Architektur wird Chord vorgestellt. FastTrack und JXTA stellen Beispiele für den Einsatz von Super-Peer-Architekturen dar. Die Qualität dieser Architekturen wird auch daran deutlich, dass sie bereits für verschiedene Systeme Verwendung gefunden haben.

21.4.1 Napster

Napster (`http://www.napster.com/`) ist eine Dateitauschbörse, die Ende der 90er Jahre die P2P-Konzepte auch bei einer breiten Internetöffentlichkeit bekannt gemacht hat. Bis heute gilt es mit einer Anzahl von registrierten Benutzern im zweistelligen Millionenbereich als eines der erfolgreichsten Systeme dieser Art. Aufgrund rechtlicher Probleme bezüglich der Verwertungsrechte von Dateien wurde Napster 2002 abgeschaltet, aber mit neuem Geschäftsmodell als Napster 2.0 im Oktober 2003 wieder gestartet. Die folgenden Beschreibungen beziehen sich auf die ursprüngliche Version von 1998.

Napster wurde nach einer hybriden zentralisierten Architektur entworfen. Client-Server-Konzepte werden für die Suche nach Dateien und für ein simples Sicherheitsmodell verwendet, während der eigentliche Dateiaustausch nach dem

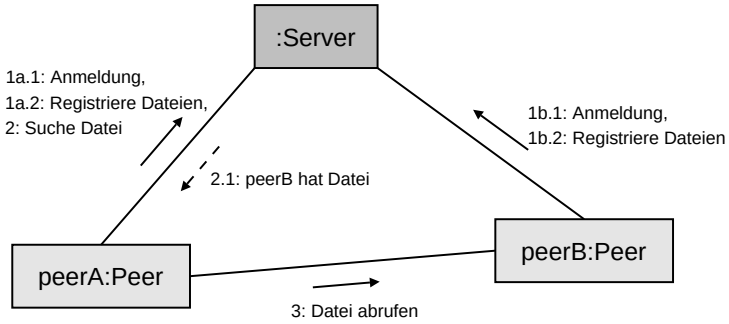

Abbildung 21.6: Kommunikation bei Napster

P2P-Modell stattfindet. Diese Architektur ermöglicht eine effiziente und vollständige Suche nach Dateien, entlastet aber gegenüber den vorher üblichen Verfahren zum Dateiaustausch, wie z. B. per FTP oder WWW, die zentralen Server, da sie nicht mehr am eigentlichen Dateiaustausch beteiligt sind. Der prinzipielle Kommunikationsablauf ist in Abbildung 21.6 zu sehen. Um am System teilnehmen zu können, müssen sich Peers beim zentralen Server anmelden und diesem ihre freigegebenen Dateien mitteilen. Das Sicherheitsmodell beschränkt sich auf die Überprüfung eines Passworts zu einem registrierten Benutzernamen. Der Server indexiert alle Dateien aller Peers, sodass er mit dieser globalen Sicht Suchanfragen effizient beantworten kann. Auf ihre Suchanfragen erhalten Peers die Adressen von den Peers, die im Besitz der Datei sind. Der suchende Peer kontaktiert diese Peers direkt, um die gewünschte Datei herunterzuladen. Das ursprüngliche Kommunikationsprotokoll ist proprietär und wurde nie veröffentlicht. Es wurden allerdings Spezifikationen des P2P-Protokolls publiziert, die durch Reengineering entstanden sind [Sch01b] und die Entwicklung verschiedener freier Applikationen für das System ermöglichten.

21.4.2 Freenet

Freenet [CSWH01] ist ein verteiltes kooperatives Dateisystem, das einer reinen, weitgehend unstrukturierten P2P-Architektur entspricht. Es wurde 1999 von Ian Clarke für die Veröffentlichung, Replikation und Abfrage von Dateien entwickelt und hatte als wichtigstes Entwurfsziel die Anonymität von Anbietern und Lesern der Dateien. Obwohl die Netzwerktopologie unstrukturiert ist, ergeben sich mithilfe von adaptiven Routing-Strategien kurze Pfade zwischen Peers zur effizienten Kommunikation.

Dateien werden bei Freenet durch eindeutige Schlüssel identifiziert, die mittels Hashfunktionen berechnet werden. Es werden mehrere Arten von Schlüsseln unterschieden. Sie dienen u. a. der Bildung hierarchischer Namensräume, der

Umsetzung eines rudimentären Update-Mechanismus und der Aufteilung großer Dateien auf mehrere kleinere. Die einfachste Form von Schlüsseln dient der Identifikation von Dateien. Bei Veröffentlichung einer Datei gibt der Benutzer einen kurzen beschreibenden Text an. Dieser wird als Schlüssel benutzt, um die Datei symmetrisch zu verschlüsseln. Aus der Beschreibung wird außerdem ein privater und ein öffentlicher Schlüssel generiert. Mit dem privaten Schlüssel wird die Datei signiert, um rudimentäre Integritätsprüfungen zu ermöglichen. Der eigentliche Identifikationsschlüssel wird als Hashwert vom öffentlichen Schlüssel berechnet. Um den Abruf der Datei zu ermöglichen, wird häufig nur der beschreibende Text veröffentlicht, da der Identifikationsschlüssel daraus berechnet werden kann.

Alle Peers nutzen einen Teil des auf ihren Rechnern zur Verfügung stehenden Speicherplatzes für einen *Datastore*, in dem Dateien mit ihren Identifikationsschlüsseln gespeichert werden. Alle Dateien im System werden nur in Datastores gehalten. Die Menge aller Datastores aller Peers bildet den Gesamtspeicher von Freenet. Benutzer können für ihren Peer konfigurieren, wie viel Platz sie zur Verfügung stellen wollen. Beim Veröffentlichen und Abfragen werden Dateien automatisch in den Datastores dupliziert. Ist dort nicht mehr genügend Platz vorhanden, wird nach einer »Least Recently Used«-Strategie Platz geschaffen. Dateien, die für einen längeren Zeitraum nicht angefragt werden, können daher ganz aus dem System verschwinden. Freenet gibt keine Garantien für die dauerhafte Persistenz.

Durch den Einsatz von adaptiven Routing-Strategien erreicht Freenet gute Leistungseigenschaften, obwohl es eine unstrukturierte P2P-Architektur benutzt. Alle Peers verfügen über eine dynamische Routing-Tabelle, in der Adressen anderer Peers mit Schlüsseln von Dateien, über die diese vermutlich verfügen, gespeichert werden. Anfragen werden zu Peers weitergeleitet, die bereits über Dateien mit »ähnlichen« Schlüsseln verfügen (siehe unten). In Verbindung mit umfangreichem Caching und dynamischer Anpassung der Routing-Tabellen kommt es zu einer Cluster-Bildung der Peers mit kurzen Kommunikationswegen. Solche Topologien werden auch *Kleine-Welt-Netzwerke* [Mil67, WS98a] genannt und zeichnen sich durch effiziente Kommunikation aus.

Um Dateien abfragen zu können, benötigen Benutzer den Identifikationsschlüssel der Datei, der z. B. aus dem Beschreibungstext berechnet werden kann. Es wird dann eine Suchanfrage an den eigenen Peer gestellt, der prüft, ob die gewünschte Datei im eigenen Datastore liegt. Falls sie dort gefunden wird, wird sie an den Benutzer zurückgeliefert. Anderenfalls wird die Routing-Tabelle nach dem Dateischlüssel durchsucht, der dem der gesuchten Datei am ähnlichsten ist. Diesem Schlüssel ist in der Tabelle ein Peer zugeordnet, dem dann eine Suchanfrage geschickt wird. Dieser Peer verfährt analog mit der Suchanfrage, bis die gewünschte Datei gefunden wird. Es entsteht eine »Anfragekette« vom ursprünglich anfragenden Peer zu einem Peer, der die Datei gespeichert hat. Entlang dieser Kette wird die Datei zurückgeliefert und bei jedem beteiligten Peer der Kette im Datastore dupliziert. Außerdem erweitern alle Peers der Kette

ihre Routing-Tabelle um einen Eintrag mit dem neuen Dateischlüssel. Anfragen werden mit einem quasi-eindeutigen Schlüssel versehen. Alle Nachrichten einer Anfrage haben den gleichen Schlüssel, sodass Peers Zyklen in der Kette erkennen und vermeiden können. Falls ein Peer einem anderen Peer keine Suchanfrage schicken kann, weil dieser z. B. nicht erreichbar ist, versucht er es mit dem Peer, der dem zweitähnlichsten Schlüssel zugeordnet ist usw. Falls keine Peers mehr in der Routing-Tabelle gefunden werden, wird eine Fehlermeldung zum vorhergehenden Peer der Kette geschickt, damit dieser den nächsten Peer seiner Routing-Tabelle befragen kann. Suchanfragen sind mit einer Lebenszeit in Form einer *Hops-To-Live*-Angabe versehen. Wenn diese Grenze erreicht wird, ohne dass die Datei gefunden wurde, wird eine Fehlermeldung entlang der Kette zurückgegeben. Die Veröffentlichung von Dateien erfolgt analog zur Abfrage.

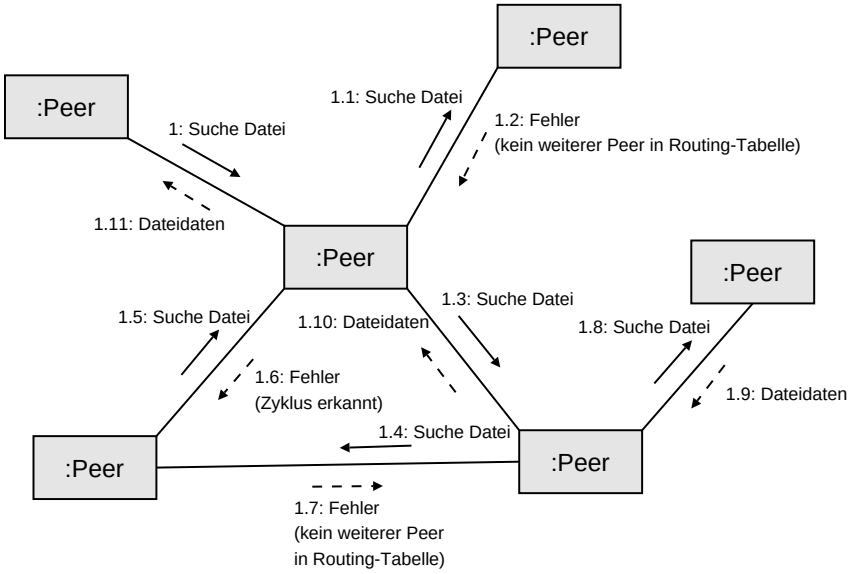

Abbildung 21.7: Kommunikation bei Freenet nach [CSWH01]

Zunächst wird der Identifikationsschlüssel der Datei berechnet. Um Konflikte auszuschließen, startet der Peer eine Anfrage für diesen Schlüssel, sodass eine Peer-Kette gebildet wird. Falls keine Datei für diesen Schlüssel gefunden wird, wird die Datei entlang der Kette verbreitet. Die jeweiligen Peers passen dabei ihre Routing-Tabelle mit dem neuen Dateischlüssel an. Falls eine bereits existierende Datei mit dem gleichen Schlüssel gefunden wird, muss der Benutzer einen anderen Schlüssel generieren, indem er z. B. den Beschreibungstext ändert.

Peers, die am System teilnehmen wollen, müssen zunächst einen bereits verbundenen Peer finden. Dies kann z. B. über Listen von verfügbaren Peers

geschehen, die über andere Kommunikationswege wie dem WWW verbreitet werden. Hat ein neuer Peer einen bereits verbundenen gefunden, wird ein eigenes Protokoll genutzt, das den neuen Peer bei einer Kette von anderen Peers bekannt macht.

Das wichtigste Entwurfsziel bei Freenet ist die Wahrung der Anonymität von Lesern und Anbietern von Dateien. Peers, die eine Suchanfrage von einem anderen Peer erhalten, können nicht erkennen, ob die Anfrage bei diesem Peer initiiert wurde oder ob es sich um eine weitergeleitete Anfrage handelt. Eine gefundene Datei wird entlang der Peer-Kette zurückgegeben, wobei jeweils nur benachbarte Glieder der Kette miteinander kommunizieren. Diese können nicht entscheiden, ob ihr Vorgänger bzw. Nachfolger Endglied der Kette ist, sodass die Anonymität dahingehend gewährleistet ist, dass kein Peer entscheiden kann, wer Anbieter oder Leser einer Datei ist. Zusätzlich werden die Dateien im Datastore der Peers verschlüsselt mit ihren Identifikationsschlüsseln gespeichert. Ein Benutzer kann daher nicht erkennen, was für Dateien sich auf seinem Peer befinden. Dafür müsste er vom Identifikationsschlüssel auf den Beschreibungstext der Datei schließen, um sie zu entschlüsseln.

21.4.3 Chord

Chord [SMK+01] ist eine reine strukturierte P2P-Architektur für Internetapplikationen. Es ermöglicht das effiziente Auffinden von Ressourcen und verfügt über gute Skalierungseigenschaften. Es ist die Basis für verschiedene andere P2P-Projekte, wie dem »*Cooperative File System*« [DKK+01].

Ressourcen und Peers werden bei Chord durch eindeutige ganzzahlige Schlüssel aus einem Intervall von 0 bis $2^m - 1$ identifiziert. Die Berechnung erfolgt ähnlich zu Freenet mithilfe einer Hashfunktion. Anschaulich werden die Peer-Schlüssel auf einem Kreis, dem *Chord-Ring*, angeordnet. Eine Ressource wird auf dem Peer abgelegt, dessen Schlüssel in Bezug auf den Kreis größer oder gleich dem der Ressource ist und dabei den minimalen Abstand hat. In Abbildung 21.8 wird ein Kreis mit $m = 6$ gezeigt. Zum Beispiel wird die Ressource mit dem Schlüssel 10 auf dem Peer mit dem Schlüssel 14 abgelegt, da 14 der nächstgrößere Schlüssel aller Peers ist. Der Charakter der Hashfunktion sorgt im Allgemeinen für eine gleichmäßige Verteilung der Ressourcen auf die Peers.

Jeder Chord-Peer verfügt über eine Art Routing-Tabelle, die *finger table* genannt wird. In dieser werden m IP-Adressen und Ports anderer Peers gespeichert. Der i-te Eintrag in der Tabelle eines Peers mit dem Schlüssel k verweist auf den Peer, dessen Schlüssel größer oder gleich $k + 2^{i-1}$ ist und den geringsten Abstand dazu hat. Daraus ergibt sich eine Struktur der P2P-Architektur, die abgesehen vom Betreten und Verlassen von Peers statisch ist. Für die Suche nach einer Ressource muss der Benutzer deren Schlüssel kennen. Ein Peer, der eine Suchanfrage erhält, leitet sie an den Peer weiter, dessen Schlüssel den kleinsten Abstand zu dem der Ressource aufweist, aber kleiner ist. Zuvor prüft der Peer, ob

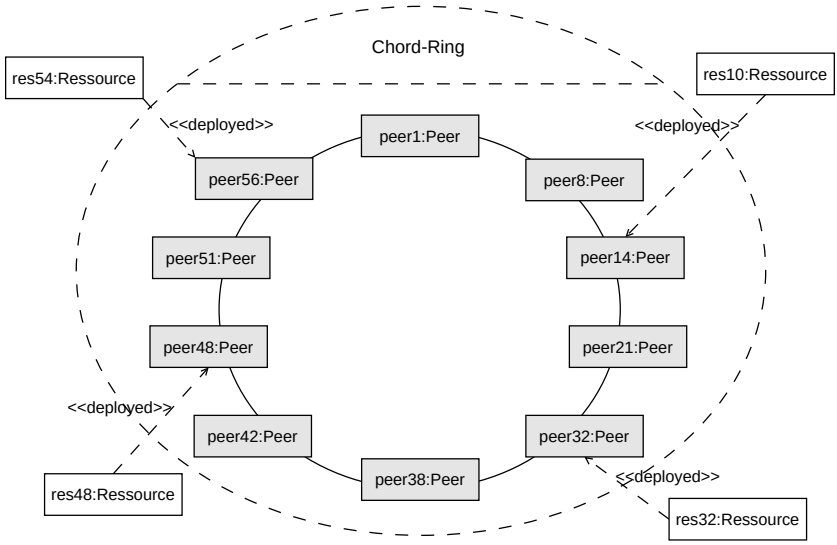

Abbildung 21.8: Der Chord-Ring nach [SMK+01]

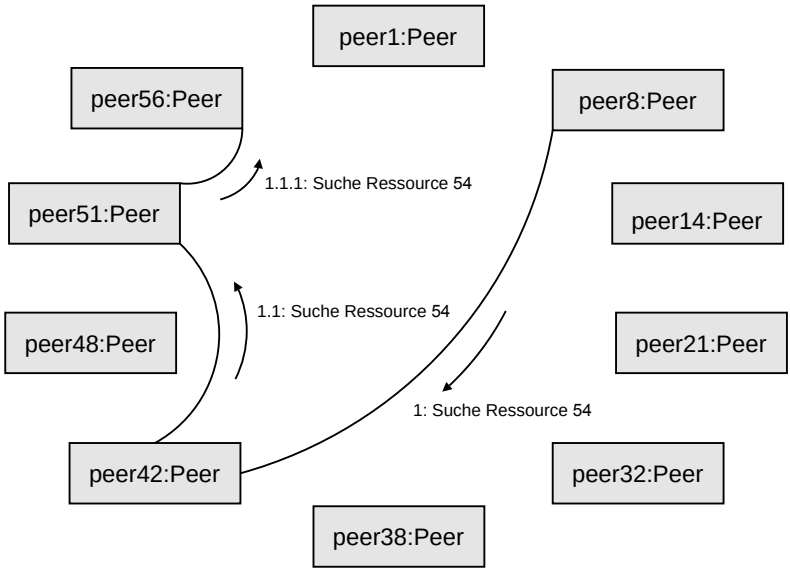

Abbildung 21.9: Kommunikation bei Chord nach [SMK+01]

nicht sein direkter Nachfolger die Ressource besitzt. Jede Weiterleitung verringert den Abstand zur Ressource um eine Potenz, sodass die Suche eine Komplexität von O(log N) bei N Peers hat. Beim Beispiel in Abbildung 21.9 wird bei Peer 8 eine Suche nach Ressource 54 initiiert. Peer 8 leitet entsprechend seiner Routing-Tabelle die Anfrage zu Peer 42. Von dort aus gelangt sie zu Peer 51. Dieser erkennt, dass sein direkter Nachfolger für die Ressource zuständig ist und leitet die Anfrage an den Nachfolger weiter.

Peers, die am System teilnehmen wollen, müssen zunächst einen bereits verbundenen Peer finden. Dies kann z. B. über Listen von verfügbaren Peers geschehen, die über andere Kommunikationswege wie dem WWW verbreitet werden. Hat ein neuer Peer einen bereits verbundenen gefunden, befragt er diesen nach seinem Nachfolger auf dem Kreis. Daraufhin meldet er sich bei diesem Nachfolger als neuer Vorgänger an und erhält evtl. einige Ressourcenschlüssel, für die er dann zuständig wird. Die Routing-Tabellen aller Peers werden periodisch aktualisiert. Peers befragen ihre Nachfolger nach neu hinzugekommenen Peers, die evtl. nun die Rolle des Nachfolgers übernehmen. Außerdem werden die anderen Routing-Einträge gepflegt, um die logarithmische Suchkomplexität zu erhalten.

21.4.4 FastTrack

FastTrack ist ein auf einer Super-Peer-Architektur basierendes P2P-Protokoll, das u. a. von der Dateitauschbörse KaZaA (`http://www.kazaa.com/`) eingesetzt wird. Die Anzahl der Nutzer von auf FastTrack aufbauenden P2P-Systemen lag bereits über der von Napster. FastTrack basiert auf dem Gnutella-

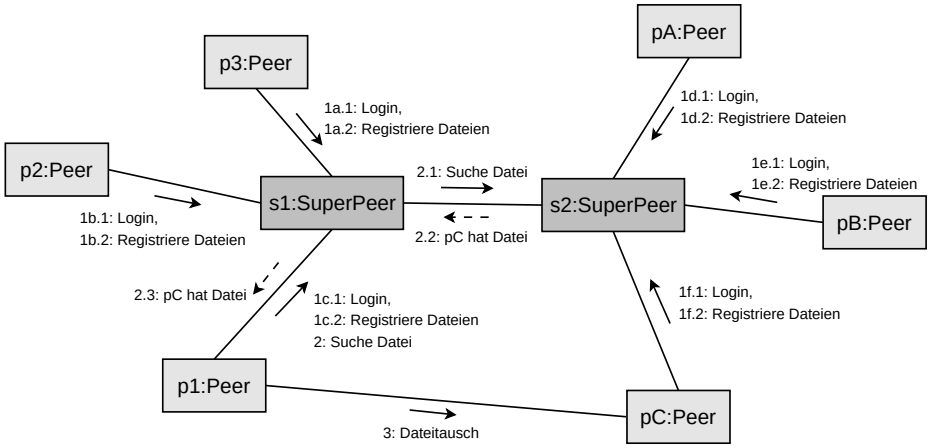

Abbildung 21.10: Kommunikation bei FastTrack

Protokoll, erweitert dieses jedoch um Super-Peers, um die Skalierbarkeit des Systems zu erhöhen. Leistungsfähige Computer mit schneller Netzanbindung werden automatisch zu Super-Peers und werden damit zeitweilig zu Index-Servern für langsamere Peers. Dies ist möglich, weil jede P2P-Applikation auch Super-Peer-Funktionalität besitzt und bei Bedarf Gebrauch davon machen kann.

Um die initiale Verbindung zum Netzwerk herstellen zu können, verfügen die Peers über eine Liste fest codierter IP-Adressen von Super-Peers. Ein Peer versucht, diese Super-Peers zu kontaktieren, um von dem ersten erreichten Super-Peer eine Liste aktuell aktiver Super-Peers anzufordern. Der Peer wählt einen Super-Peer aus dieser Liste aus, um sich bei ihm anzumelden und eine Liste der zum Austausch vorhandenen Dateien zu senden. Die Anmeldung am Super-Peer und die Registrierung der Dateien ist in Abbildung 21.10 als Schritt 1 zu erkennen. Suchanfragen werden ebenfalls an die Super-Peers gestellt, die mit weiteren Super-Peers kommunizieren, um die Anfrage zu beantworten. In der Abbildung ist dargestellt, wie Peer p1 in Schritt 2 eine Suchanfrage an seinen Super-Peer s1 sendet. Der Super-Peer s1 leitet die Suchanfrage an Super-Peer s2 weiter, der eine Antwort an s1 mit entsprechenden Suchtreffern zurücksendet. Daraufhin teilt s1 dem anfragenden Peer p1 mit, dass der Peer pC die gesuchte Datei besitzt. Im dritten Schritt erfolgt der Austausch der Datei direkt zwischen Peer p1 und Peer pC.

21.4.5 JXTA

JXTA (`http://www.jxta.org/`) beschreibt eine Reihe von Protokollen, die es Knoten in einem Netzwerk erlauben, auf eine P2P-orientierte Art und Weise miteinander zu kommunizieren und zu kollaborieren. Bei den Knoten kann es sich z. B. um P2P-Applikationen auf Mobiltelefonen, kabellosen PDAs, PCs oder auch Servern handeln. Prinzipiell bilden alle JXTA-Peers unabhängig von der konkreten P2P-Applikation ein logisches P2P-Netzwerk, in dem jeder Peer mit anderen Peers kommunizieren kann, auch wenn sich einige der Peers hinter Firewalls befinden oder unterschiedliche Netzwerkprotokolle verwenden. JXTA integriert damit verschiedene P2P-Applikationen und physikalische Netzwerke.

Wesentliche Ziele des JXTA-Projektes sind *Interoperabilität*, *Plattformunabhängigkeit* und *Erreichbarkeit*. Die Interoperabilität wird insbesondere über unterschiedliche P2P-Systeme und P2P-Gemeinschaften hergestellt. Plattformunabhängigkeit wird dadurch gewährleistet, dass verschiedene Sprachen, Systeme und Netzwerke unterstützt werden. Das Ziel der Erreichbarkeit besteht darin, Verbindungen zu jedem Gerät mit einem »digitalen Herzschlag« zu ermöglichen.

Aufgrund der Zielsetzung bietet JXTA zahlreiche Standardfunktionalitäten für P2P-Systeme, wie das Auffinden von Peers und Ressourcen im Netzwerk und die Überbrückung von Firewalls. Das Projekt definiert derzeit sechs Protokolle, die die Ausgangsbasis für die Entwicklung eigener P2P-Systeme darstellen. Das *Peer Discovery Protocol* beschreibt, wie Ressourcen gefunden bzw. veröffent-

licht werden. Das *Peer Resolver Protocol* bestimmt den Nachrichtenaustausch zwischen Peers. Statusinformationen von anderen Peers können über das *Peer Information Protocol* eingeholt werden. Das *Rendezvous Protocol* legt fest, wie Nachrichten an andere Peers propagiert werden, und das *Pipe Binding Protocol* beschreibt, wie Pipes (virtuelle Kommunikationskanäle) zwischen Peers angelegt werden. Das Routing zwischen Peers wird durch das *Endpoint Routing Protocol* beschrieben.

Zu Beginn des JXTA-Projektes wurden mehrere P2P-Software-Architekturen untersucht, um eine allgemeine Schichtenstruktur auf konzeptioneller Ebene herauszuarbeiten (siehe Abbildung 21.11). Eine typische P2P-Software kann grob in drei Schichten unterteilt werden. Die unterste Schicht bildet der Kern, der grundlegende Aufgaben wie die Kommunikation und Verwaltung von Peers übernimmt. Die mittlere Schicht beinhaltet Services, die starken Gebrauch von der untersten Schicht machen und höhere Dienste wie Indexierung, Suche und File-Sharing anbieten. Die oberste Schicht ist die Applikationsschicht, die dem Nutzer die eigentliche Funktionalität wie Telefonie, Dateiaustausch usw. über eine entsprechende Benutzungsoberfläche zur Verfügung stellt.

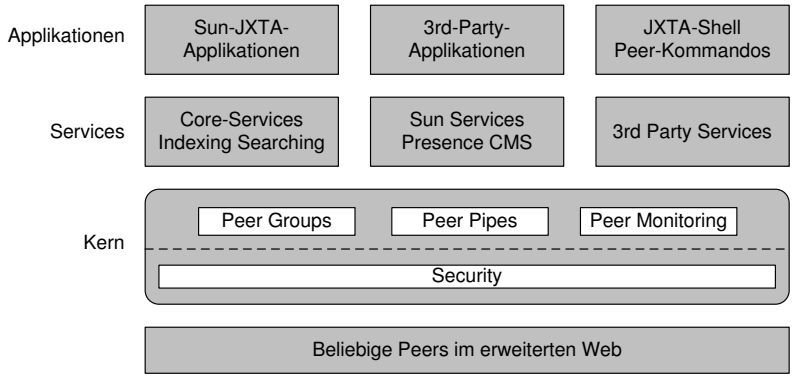

Abbildung 21.11: JXTA-Architektur [Sun05]

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 – Reengineering 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