

## 22 Peer-to-Peer-Architekturen

Ludger Bischofs, Wilhelm Hasselbring, Timo Warns

Das Konzept der Kommunikation zwischen gleichberechtigten Partnern ist nicht neu, erlebt aber zurzeit unter dem Begriff »Peer-to-Peer« (P2P) eine Art Renaissance. Der Auslöser dafür war vor allem das P2P-System Napster (<http://www.napster.com>), das als Tauschbörse für Audiodateien einen großen Bekanntheitsgrad erreichte.

Aktuelle Anwendungsbereiche von P2P-Systemen sind u.a. gemeinsamer Dateizugriff, Instant Messaging (z.B. ICQ, <http://www.icq.com/>), Telefonie (z.B. Skype, <http://www.skype.com/>), verteilte Suche, kollaboratives Arbeiten (z.B. Groove, <http://www.groove.net/>) und verteiltes Rechnen (z.B. Seti@home, <http://setiathome.ssl.berkeley.edu/>). Wesentlicher Aspekt dieser Systeme ist die Nutzung dezentraler Ressourcen, wie z.B. Rechenleistung, Speicher und Netzwerkbandbreite.

P2P-Systeme werden in letzter Zeit verstärkt auch in weiteren Domänen eingesetzt. Bei Osiris (Open Service Infrastructure for Reliable and Integrated Process Support) erfolgt die Ausführung von Prozessen (Workflows) in einer P2P-Architektur, in der Metadaten zwischen den Peers repliziert werden [SWSS03]. Zukünftige Einsatzgebiete von P2P-Systemen liegen u.a. im Bereich digitaler Bibliotheken [BHSS04, WBS04], in der verteilten Software-Entwicklung [BH04a] und der integrierten Gesundheitsversorgung [BHN<sup>+</sup>04].

### 22.1 Definitionen

Es existieren zurzeit verschiedene Definitionen und Beschreibungen rund um den Begriff Peer-to-Peer. Eine häufig zitierte Definition von Shirky fasst P2P als eine Klasse von Internetapplikationen auf, die Ressourcen dezentral nutzen [Shi00]. Steinmetz und Wehrle verstehen unter P2P-Systemen sich selbst organisierende Systeme gleichberechtigter, autonomer, vernetzter Einheiten (so genannte Peers), die ohne zentrale Koordination mit dem Ziel der gegenseitigen Nutzung von Ressourcen operieren [SW04]. Schollmeier versteht den Begriff des P2P-Netzwerks als eine verteilte Netzwerkarchitektur und grenzt diese gegenüber der Client-Server-Architektur ab [Sch01b].

Aufgrund des nicht einheitlichen Verständnisses der betrachteten Begriffe werden in diesem Abschnitt einige P2P-Begriffe definiert, um eine differenzierte Betrachtungsweise zu ermöglichen und Missverständnissen vorzubeugen.

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

Die Gleichstellung von Peers wird dadurch realisiert, dass ein Peer sowohl Client- als auch Server-Funktionalität übernehmen kann. Ein Peer wird deshalb auch als Servent bezeichnet [Sch01b], was von den Begriffen Server und Client abgeleitet wurde. Ein als Servent bezeichneter Peer umfasst sowohl Client- als auch Server-Funktionalität. In vielen Fällen handelt es sich bei einem Peer um eine Software-Applikation, die als Servent agiert.

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

Ein Peer muss allerdings nicht zwingend als Software realisiert sein. Es sind z.B. auch Umsetzungen in Hardware denkbar. Die Verbindung der Peers wird durch ein Peer-to-Peer-Netzwerk realisiert.

**Definition (Peer-to-Peer-Netzwerk)** *In einem Peer-to-Peer-Netzwerk werden Peers im Sinne des P2P-Kommunikationsmodells vernetzt.*

Netzwerke sind Strukturen und Systeme, die sich formal als Graphen modellieren lassen. Daher werden Peers häufig auch als Knoten bezeichnet. Bei P2P-Netzwerken handelt es sich meist um virtuelle Netzwerke, die auf einem physikalischen Netzwerk aufbauen. Wenn Peers direkt über das Netzwerk verbunden sind, werden sie *Nachbarn* genannt. Um die Kommunikation innerhalb eines P2P-Netzwerks zu ermöglichen, wird ein P2P-Protokoll benötigt.

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

Das P2P-Netzwerk wird von P2P-Systemen zur Umsetzung ihrer Anforderungen genutzt.

**Definition (Peer-to-Peer-System)** *Ein Peer-to-Peer-System ist ein verteiltes System, dem das Peer-to-Peer-Kommunikationsmodell zugrunde liegt.*

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

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

Insbesondere umfasst dies nicht nur die Architektur der einzelnen Applikation, sondern auch Aspekte der Netzwerktopologie. Peer-to-Peer-Architekturen sind der Schwerpunkt dieses Kapitels und werden deshalb in den folgenden Abschnitten detailliert behandelt.

## 22.2 Klassifikation von P2P-Architekturen

Das P2P-Kommunikationsmodell kann durch verschiedene Architekturen umgesetzt werden. Diese Architekturen werden in erster Linie danach klassifiziert, in welchem Maße sie die Idee des Modells realisieren. Reine P2P-Architekturen gelten als strenge Umsetzung, da dabei alle Knoten als Servents auftreten. Mit solchen Ansätzen können viele Beschränkungen von Client-Server-Architekturen aufgehoben werden, z.B. durch Verbesserung der Skalierbarkeit. Im Allgemeinen kommt es aber zu einer höheren Komplexität bei der Umsetzung bestimmter Funktionalitäten, wie z.B. der Sicherheit. Es steht kein zentraler Server für die Authentifizierung und Autorisation zur Verfügung, so dass sensitive Daten verteilt gehalten werden müssen. Außerdem ist die Einhaltung bestimmter nicht-funktionaler Eigenschaften schwierig. 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 Performanz kann dadurch im Vergleich zu Client-Server-Systemen beeinträchtigt werden. Derartige Probleme werden in hybriden Architekturen dadurch umgangen, dass P2P- und Client-Server-Konzepte gemischt werden.

### 22.2.1 Reine P2P-Architekturen

*Reine P2P-Architekturen* (engl. *pure p2p architectures*) setzen das P2P-Kommunikationsmodell streng um und verzichten dementsprechend komplett auf zentrale Server. Alle Peers treten als Servents auf, um Dienste auf viele Knoten verteilen zu können, so dass 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 bestimmten Beschränkungen. Verschiedene Ausprägungen sind denkbar, wie z.B. baumartige Strukturen (siehe Abbildung 22.1). Aufgrund der Beschränkungen findet die Evolution des Netzwerks innerhalb bestimmter Grenzen statt, so dass Vorhersagen über das Verhalten des Systems gemacht werden können. Dies erfordert jedoch einen höheren Aufwand bei der Pflege der Topologie. Der

Begriff der Strukturiertheit in P2P-Kontexten wird in der Literatur inkonsistent verwendet. Häufig werden damit auch Architekturen bezeichnet, bei denen die Platzierung von Daten nicht beliebig ist, sondern gewissen Regeln folgt, um z.B. Suchoperationen zu optimieren.

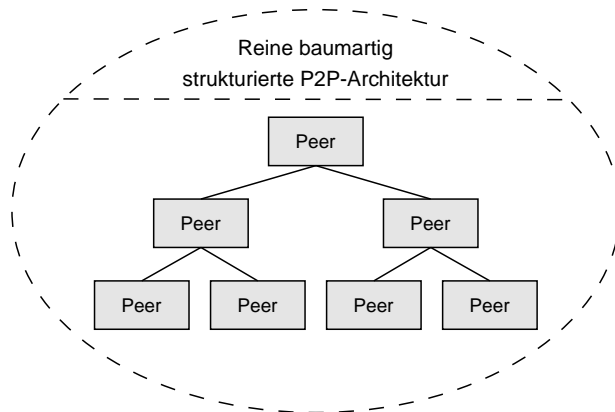


Abbildung 22.1: Reine baumartig strukturierte P2P-Architektur

Ein Peer, der einem solchen System beitreten möchte, muss seinen Platz in der Topologie finden. 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 notwendig, bis die gewünschte Struktur wieder hergestellt ist. In einem System, bei dem Peers häufig neu hinzukommen bzw. die Verbindung trennen, kann dies die Performanz wesentlich beeinträchtigen.

Bei *unstrukturierten Architekturen* wird die Topologie nicht beschränkt (siehe Abbildung 22.2). 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, um sich dem Netzwerk anzuschließen. Dieser Peer kann dann Verbindungen zu beliebigen anderen Peers vermitteln, die zu Nachbarn des neuen Peers werden.

Ein Peer kann nur mit seinen Nachbarn direkt kommunizieren. Mit anderen Peers muss indirekt kommuniziert werden, d.h., dass die jeweiligen Nachbarn bei der Kommunikation vermitteln. Direkte Kommunikation wird meist mit synchronen Verfahren umgesetzt, während indirekte Kommunikation über asynchronen Nachrichtenaustausch stattfindet.

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. Einige unstrukturierte Systeme haben allerdings die Tendenz, so genannte *Kleine-Welt-Netzwerke* zu bilden [Mil67,

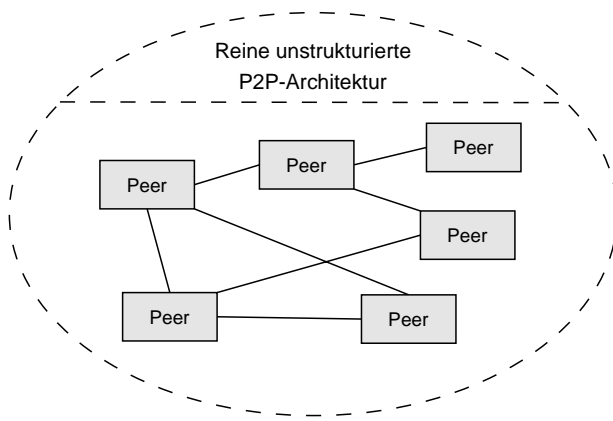


Abbildung 22.2: Reine unstrukturierte P2P-Architektur

WS98a]. Solche Netzwerke sind stark geclustert, haben relativ kurze Weglängen zwischen einzelnen Peers und erlauben so eine effiziente Kommunikation.

### 22.2.2 Hybride P2P-Architekturen

*Hybride P2P-Architekturen* setzen als Mischung von reinen P2P- und Client-Server-Architekturen gezielt zentrale Knoten ein. Die konkrete Ausprägung wird dabei vom Grad der Zentralisierung bestimmt. Als übliche Ansätze findet man zentralisierte und Super-Peer-Architekturen.

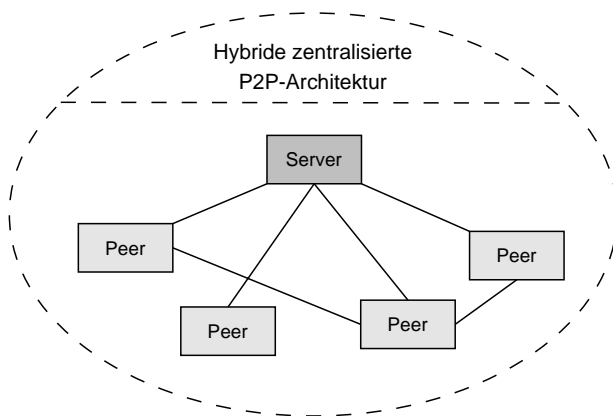
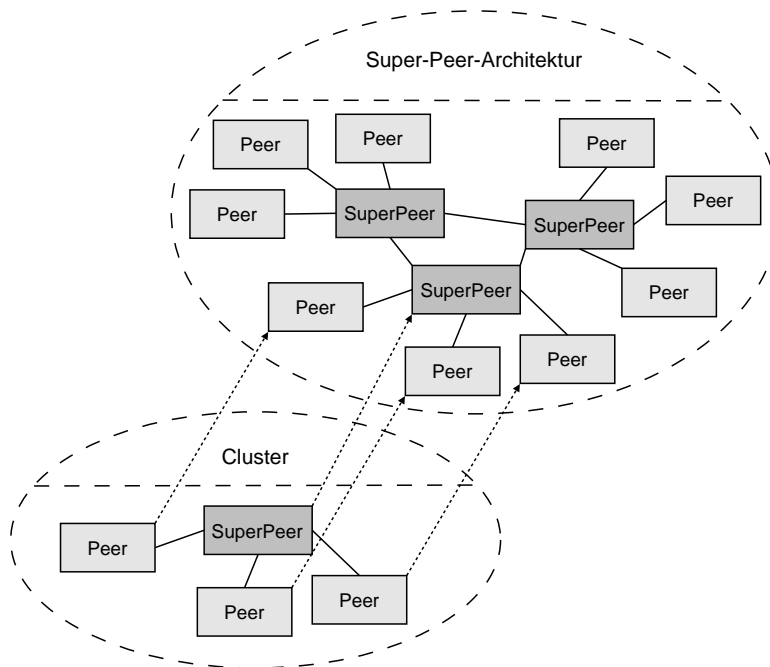


Abbildung 22.3: Hybride zentralisierte P2P-Architektur

Bei *zentralisierten P2P-Architekturen* übernehmen Knoten mit reiner Server-Funktionalität exklusiv bestimmte Dienste (siehe Abbildung 22.3). Da solche Server-Knoten u.a. die Verfügbarkeit und Zuverlässigkeit der entsprechenden Dienste im Gesamtsystem bestimmen, gelten sie als »Single Points of Failure«. Dies wird insbesondere dann in Kauf genommen, wenn der Dienst nur schwierig auf dezentrale Weise umgesetzt werden kann. Zum Beispiel haben dezentrale Algorithmen häufig eine höhere Komplexität, so dass aus Effizienzgründen auf zentrale Knoten ausgewichen wird. So setzte Napster zentrale Server ein, um die Suche zu optimieren (siehe Abschnitt 22.4.1). Im Unterschied zu Client-Server-Architekturen werden nur einige Dienste exklusiv von Servern übernommen. Andere Dienste werden nach dem P2P-Kommunikationsmodell angeboten.



**Abbildung 22.4:** Hybride Super-Peer-Architektur

*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 normalen Peers als auch als Servent für eine Menge von anderen Super-Peers auftritt (siehe Abbildung 22.4). 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 »virtuellen« Super-Peer bilden. Bei Ausfall eines Super-Peers bleiben seine Dienste durch Redundanz verfügbar.

## 22.3 Schichten einer P2P-Applikation

Bei Peers handelt es sich um gleichberechtigte Einheiten, die typischerweise sowohl Client- als auch Server-Funktionalität 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 nach anwendungs-, domänenspezifischen und allgemeinen Diensten 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 eine Schichtenarchitektur der Applikationen, wie sie in Abbildung 22.5 dargestellt ist.

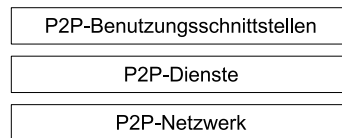


Abbildung 22.5: Typische Schichten einer P2P-Applikation

### 22.3.1 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, so dass grafische Benutzungsoberflächen angeboten werden.

Die Benutzungsschnittstellen abstrahieren von den Diensten der zugrunde liegenden P2P-Dienstschiicht und bieten integrierte Schnittstellen, um funktionale Anforderungen der Applikationsdomäne zu erfüllen. Zum Beispiel verfügen *Instant-Messenger*-Applikationen über Dienste zum Versand von Nachrichten und so genannte *Awareness*-Dienste, die über den Status der Benutzer wachen. Eine grafische Benutzungsoberfläche abstrahiert von diesen Diensten und stellt deren Funktionalität grafisch dar, indem sie z.B. Eingabefenster für Nachrichten an Benutzer, die online sind, bereitstellt.

### 22.3.2 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.

### 22.3.3 P2P-Netzwerk

Bei einem P2P-Netzwerk handelt es sich um ein virtuelles Netzwerk, das auf einem physikalischen Netzwerk aufsetzt und die Verbindung und Kommunikation zwischen den Peers sicherstellt. Der Ausdruck »virtuell« hat sich für Netzwerke bestimmter P2P-Systeme eingebürgert, die innerhalb eines zugrunde liegenden »physikalischen« Netzwerks, wie dem Internet, eine eigene Topologie aufbauen. Die Topologie des virtuellen Netzwerks muss nicht der Topologie des physikalischen Netzwerks entsprechen. Zum Beispiel können Peers, die Nachbarn im virtuellen Netzwerk sind, im physikalischen Netzwerk weit auseinander liegen. Der Ausdruck »physikalisch« hat sich für das zugrunde liegende Netzwerk eingebürgert, um es vom virtuellen Netzwerk abzugrenzen. Damit ist allerdings nicht gemeint, dass nur Hardware Teil der Schicht wäre.

Das P2P-Netzwerk wird durch die beteiligten Peers gebildet, die das verwendete Kommunikationsprotokoll kennen müssen, um Nachrichten mit anderen Peers austauschen zu können. Die Netzwerkschicht stellt den P2P-Diensten Primitive, wie z.B. virtuelle Pipes oder 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. Prinzipiell beinhaltet die P2P-Netzwerkschicht Low-Level-Funktionalität, die zum Betrieb des P2P-Netzwerks notwendig ist.

Virtuelle Netzwerke können verschiedenste Arten von physikalischen Netzwerken nutzen, solange die Kommunikation zwischen Nachbarn möglich ist.



Meist werden P2P-Architekturen aber für Internetapplikationen entwickelt, so dass Netzwerke nach dem TCP/IP-Modell verwendet werden.

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 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.

## 22.4 Beispiele von P2P-Systemen

In diesem Abschnitt werden bekannte P2P-Systeme beschrieben und als Beispiele für unterschiedliche P2P-Architekturstile herangezogen. Napster ist ein hybrides 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. JXTA nimmt eine Sonderstellung ein, da es kein P2P-System im eigentlichen Sinne ist, sondern vielmehr eine Basis für die Entwicklung eigener P2P-Systeme bietet.

### 22.4.1 Napster

Napster (<http://www.napster.com/>) ist ein File-Sharing-System, das Ende der 90er Jahre die P2P-Konzepte auch bei einer breiteren Öffentlichkeit bekannt gemacht hat. Bis heute gilt es mit ca. 38 Millionen registrierten Benutzern als eines der erfolgreichsten Systeme dieser Art. Aufgrund rechtlicher Probleme bezüglich der Verwertungsrechte von Dateien wurde Napster 2002 abgeschaltet. Obwohl es mit neuem Geschäftsmodell als Napster 2.0 im Oktober 2003 wieder gestartet wurde, konnte es bisher nicht die ursprüngliche Popularität zurückgewinnen. 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. Der eigentliche Dateiaustausch findet direkt zwischen den Knoten in P2P-Manier statt. Diese Architektur ermöglicht eine effiziente Suche nach Dateien, bei der alle passenden verfügbaren Dateien gefunden werden. Gegenüber den vorher üblichen Verfahren zum Dateiaustausch, wie z.B. per FTP oder WWW, werden die zentralen Server entlastet, da sie nicht mehr am eigentlichen Dateiaustausch beteiligt sind. Der prinzipielle Kommunikationsablauf ist in Abbildung 22.6 zu sehen. Um am System teilnehmen zu können,

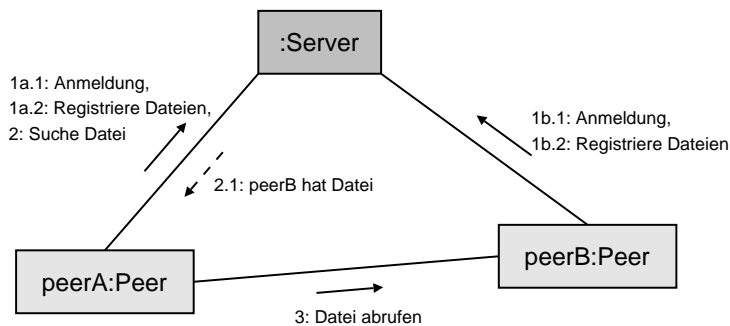


Abbildung 22.6: Kommunikation bei Napster

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, so dass 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 offene Spezifikationen publiziert, die durch Reengineering entstanden sind [Sch01a] und die Entwicklung verschiedener freier Applikationen ermöglichen.

### 22.4.2 Freenet

Freenet [CSWH01] ist eine Art verteiltes kooperatives Dateisystem, das nach einer reinen, weitgehend unstrukturierten P2P-Architektur entwickelt wurde. Es wurde 1999 von Ian Clarke für die Veröffentlichung, Replikation und Abfrage von Dateien entworfen. Das wichtigste Entwurfsziel ist die Anonymität von Anbietern und Lesern. Mit Hilfe von adaptiven Routing-Strategien wird die Netzwerktopologie optimiert, so dass sich kurze Kommunikationswege bilden.

Dateien werden bei Freenet durch eindeutige Schlüssel identifiziert, die mit Hilfe von 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 minimale 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 Client 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 Client 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 im Datastore dupliziert. Außerdem erweitern alle Peers 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, so dass 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.

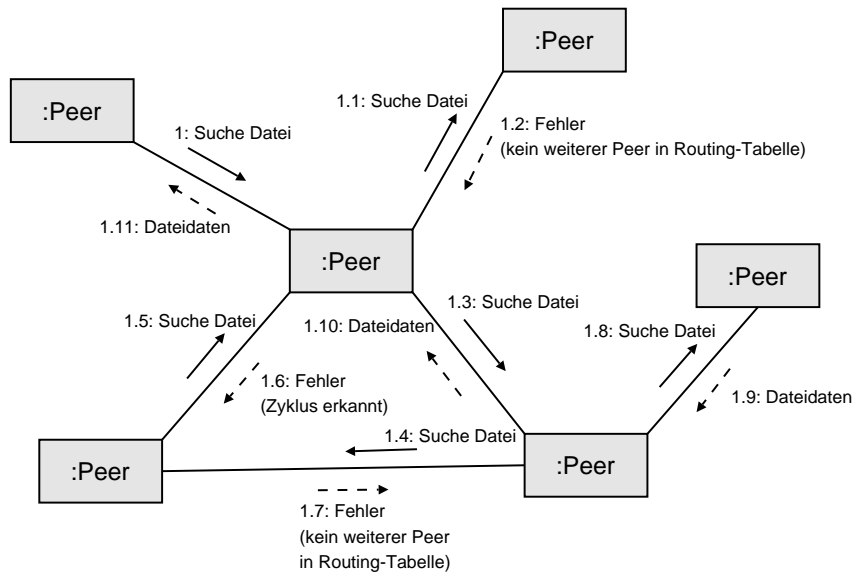


Abbildung 22.7: Kommunikation bei Freenet nach [CSWH01]

Die Veröffentlichung von Dateien erfolgt analog zur Abfrage. Zunächst wird der Identifikationsschlüssel der Datei berechnet. Um Konflikte auszuschließen, startet der Client eine Anfrage für diesen Schlüssel, so dass 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 besonderes 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. Clients, 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, so dass die Anonymität gewährleistet ist. Im Datastore der Clients werden verschlüsselte Dateien mit ihren Identifikationsschlüsseln gespeichert. Ein Benutzer kann nicht erkennen, was für Dateien sich auf seinem Client befinden. Dafür müsste er vom Identifikationsschlüssel auf den Beschreibungstext der Datei schließen, um sie zu entschlüsseln.

### 22.4.3 Chord

Chord [SMK<sup>+</sup>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<sup>+</sup>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 mit Hilfe 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 22.8 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. Die anderen Ressourcen werden entsprechend verteilt. Der Charakter der Hashfunktion sorgt im Allgemeinen für eine Gleichverteilung der Menge aller 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 nicht sein direkter Nachfolger die Ressource besitzt. Jede Weiterleitung verringert den Abstand zur Ressource um eine Potenz, so dass die Suche eine Komplexität von  $O(\log N)$  bei  $N$  Peers hat. Beim Beispiel in Abbildung 22.9 wird bei Peer 8 eine Suche nach Ressource 54 initiiert. Peer 8 leitet entsprechend seiner Routing-Tabelle die Anfrage zu Peer 42. Von dort

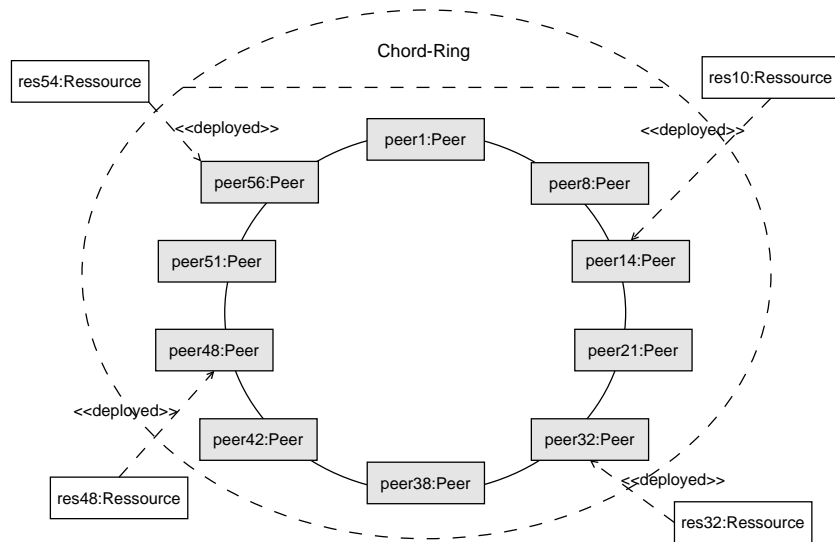


Abbildung 22.8: Der Chord-Ring nach [SMK<sup>+</sup>01]

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.

#### 22.4.4 FastTrack

FastTrack ist ein P2P-Protokoll, das u.a. von den File-Sharing-Programmen KaZaA (<http://www.kazaa.com/>) und Grokster (<http://www.grokster.com/>) eingesetzt wird. Die auf FastTrack basierenden File-Sharing-Netzwerke können bereits eine höhere Nutzerzahl als Napster aufweisen. Bekannte Eigenschaften von FastTrack sind die Fähigkeiten, unterbrochene Downloads fortzusetzen und Dateifragmente gleichzeitig von mehreren Peers herunterzuladen.

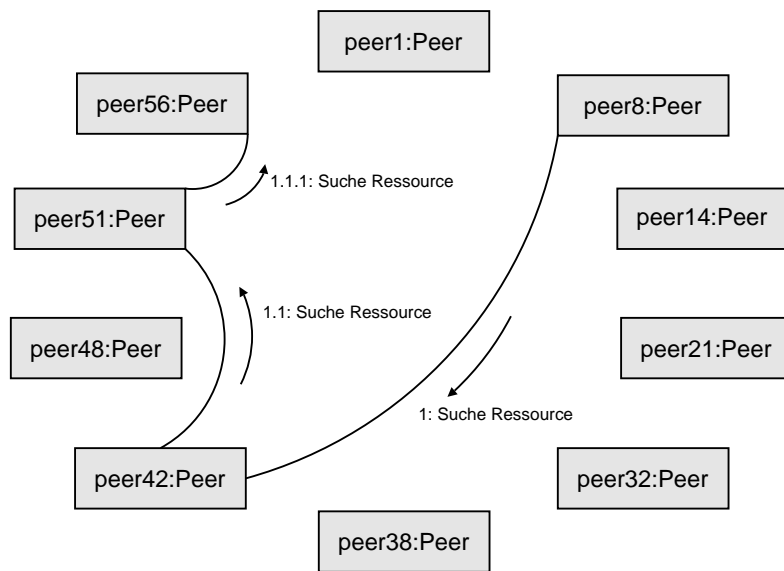


Abbildung 22.9: Kommunikation bei Chord nach [SMK<sup>+</sup>01]

FastTrack basiert auf dem Gnutella-Protokoll und erweitert dieses durch Super-Peers, um die Skalierbarkeit zu verbessern. Die Super-Peer-Funktionalität ist Teil der Client-Programme. Leistungsfähige Computer mit schneller Netz-anbindung werden automatisch zu Super-Peers und fungieren somit zeitweilig als Index-Server für langsamere Peers. Somit liegt Systemen, die auf FastTrack basieren, eine Super-Peer-Architektur zugrunde.

Um die initiale Verbindung zum Netzwerk herstellen zu können, verfügen die Peers über eine Liste fest kodierter IP-Adressen von Super-Peers. Ein Peer versucht, diese Super-Peers zu kontaktieren, um von dem ersten gefundenen Super-Peer eine Liste aktuell aktiver Super-Peers anzufordern. Der Peer wählt einen Super-Peer aus dieser Liste aus, um ihm eine Liste der zum Austausch vorhandenen Dateien zu senden. Suchanfragen werden ebenfalls an diesen Super-Peer gesendet, der mit weiteren Super-Peers kommuniziert, um die Anfrage zu bearbeiten (siehe Abbildung 22.10). Der Transfer der Dateien erfolgt dann direkt von Peer zu Peer.

Um den Datei-Download von mehreren Quellen zu ermöglichen, verwendet FastTrack den Hashalgorithmus UUHash, der es erlaubt, eine Prüfsumme über große Dateien in kurzer Zeit zu erstellen. Auf diese Weise können identische Dateien schnell identifiziert werden, um gezielt Fragmente dieser Dateien parallel herunterzuladen. Eine Schwachstelle dieses Algorithmus ist, dass zugunsten der Geschwindigkeit nur Teile der Dateien »gehasht« werden, weshalb das Einstellen korrupter bzw. gefälschter Dateien zu fehlerhaften Downloads führt, da unter-

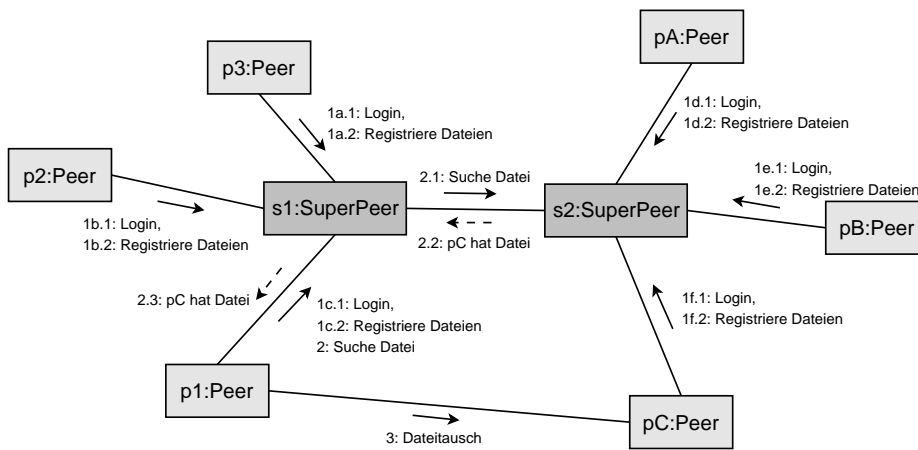


Abbildung 22.10: Kommunikation bei FastTrack

schiedliche Dateien als identisch angesehen werden können.

### 22.4.5 JXTA

JXTA (<http://www.jxta.org/>) beschreibt eine Reihe von Protokollen, die es Geräten in einem Netzwerk erlauben, auf eine P2P-orientierte Art und Weise miteinander zu kommunizieren und zu kollaborieren. Bei den Geräten kann es sich u.a. um Mobiltelefone, kabellose PDAs, PCs oder auch Server handeln.

JXTA-Peers bilden ein virtuelles P2P-Netzwerk, in dem jeder Peer mit anderen Peers und Ressourcen kommunizieren kann. Dies ist möglich, auch wenn sich einige der Peers und Ressourcen hinter Firewalls befinden oder unterschiedliche Netzwerkprotokolle verwenden.

Die Zielsetzung des JXTA-Projektes ist es zum einen, *Interoperabilität* über unterschiedliche P2P-Systeme und P2P-Gemeinschaften herzustellen. Ein weiterer wesentlicher Punkt ist die Gewährleistung von *Plattformunabhängigkeit*, indem verschiedene Sprachen, Systeme und Netzwerke unterstützt werden. Als dritter Punkt ist die *Erreichbarkeit* zu nennen, da jedem Gerät mit einem »digitalen Herzschlag« eine Verbindung ermöglicht werden soll.

Aufgrund der Zielsetzung bietet JXTA zahlreiche Standardfunktionalitäten für P2P-Systeme, wie das Auffinden von Peers und Ressourcen im Netzwerk und die Überbrückung von Firewalls. Es besteht die Möglichkeit, Dateien über das Netzwerk mit beliebigen anderen Nutzern auszutauschen. Eigene Gruppen von Peers und Geräten können über unterschiedliche Netzwerke erstellt werden. Eine sichere Kommunikation mit Peers kann über öffentliche Netzwerke erfolgen.



## JXTA-Architektur

Zu Beginn des JXTA-Projektes wurden mehrere P2P-Software-Architekturen untersucht und eine allgemeine Schichtenstruktur auf konzeptioneller Ebene herausgearbeitet (siehe Abbildung 22.11). 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.

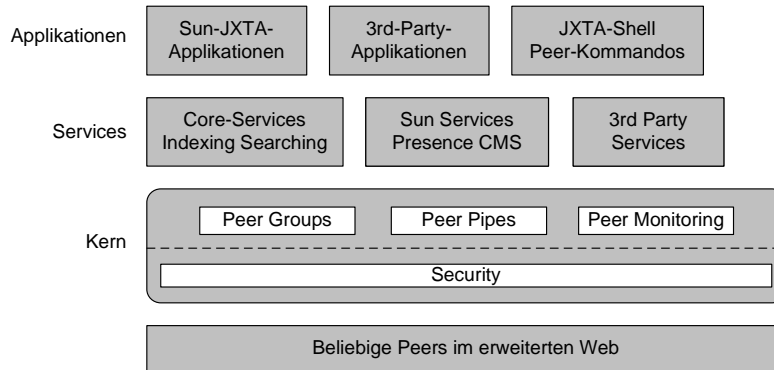


Abbildung 22.11: JXTA-Architektur [Sun05d]

## JXTA-Protokolle

Das Projekt JXTA definiert derzeit sechs Protokolle, die die Ausgangsbasis für die Entwicklung eigener P2P-Systeme darstellen.

Das *Peer Discovery Protocol* beschreibt, wie Ressourcen gefunden bzw. veröffentlicht 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.

# Literatur

- [ABB<sup>+</sup>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<sup>+</sup>96] Abowd, G.; Bass, L.; Clements, P.; Kazman, R.; Northrop, L.; Zaremski, A.: *Recommended Best Industrial Practice for Software Architecture Evaluation*. Technischer Bericht CMU/SEI-96-TR-025, Software Engineering Institute, Carnegie Mellon University, 1996
- [ABK<sup>+</sup>02] Astesiano, E.; Bidoit, M.; Krieg-Brückner, B.; Mosses, P. D.; Sannella, D.; Tarlecki, A.: Casl: The Common Algebraic Specification Language. In: *Theoretical Computer Science* 286 (2002), Nr. 2, S. 153–196
- [Ack96] Ackermann, P.: *Developing Object-Oriented Multimedia Software – Based on MET++ Application Framework*. dpunkt.verlag, 1996
- [ACM01] *Computing Curricula 2001 Computer Science*. Technischer Bericht, The Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery, Dezember 2001
- [ACM03] Alur, D.; Crupi, J.; Malks, D.: *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall, 2. Aufl., 2003
- [AFG<sup>+</sup>00] Andrade, L. F.; Fiadeiro, J. L.; Gouveia, J.; Lopes, A.; Wermelinger, M.: Patterns for Coordination. In: *Coordination Languages and Models*, Springer-Verlag, 2000, Bd. 1906 von *Lecture Notes in Computer Science*, S. 317–322
- [AG96] Allen, R.; Garlan, D.: *The Wright Architectural specification language*. Technischer Bericht CMU-CS-96-TBD, School of Computer Science, Carnegie Mellon University, 1996
- [AG97] Allen, R.; Garlan, D.: A Formal Basis for Architectural Connection. In: *ACM Transactions on Software Engineering and Methodology* (1997)
- [AGD97] Allen, R.; Garlan, D.; Douence, R.: Specifying Dynamism in Software Architectures. In: Leavens, G. T.; Sitaraman, M. (Hrsg.), *Proceedings of the Workshop on Foundations of Component-Based Software Engineering*, September 1997

- [AH90] Agrawal, H.; Horgan, J. R.: Dynamic program slicing. In: *Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation*, ACM Press, 1990, S. 246–256
- [AH01] de Alfaro, L.; Henzinger, T.: Interface Automata. In: *ESEC/FSE 01: Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2001, S. 109–120
- [Ale79] Alexander, C.: *The Timeless Way of Building*. Oxford University Press, 1979
- [All97] Allen, R.: *A Formal Approach to Software Architecture*. Dissertation, Carnegie Mellon School of Computer Science, Januar 1997
- [AMR96] André, E.; Müller, J.; Rist, T.: WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring. In: *EUROMEDIA'96*, 1996, S. 95–102
- [And04] Andresen, A.: *Komponentenbasierte Softwareentwicklung mit MDA, UML-2 und XML*, Bd. 2., neu bearbeitete Auflage. Hanser Verlag, 2004
- [App05] Apple, USA: QuickTime. 2005,  
URL <http://www.apple.com/quicktime/>
- [Arc05] ArcWay AG: ArcWay AG Home Page. Website, 2005,  
URL <http://www.arcway.com>
- [AUT] AUTOSAR-Konsortium: Automotive Open System Architecture.  
URL <http://www.autosar.org>
- [B] B-Homepage.  
URL <http://v1.fmnet.info/b/>
- [Bal97] Balzert, H.: *Lehrbuch der Softwaretechnik, Bd. 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, 1997
- [Bal00] Balzert, H.: *Lehrbuch der Software-Technik, Bd. 1: Software-Entwicklung*. Spektrum Akademischer Verlag, 2. Aufl., 2000
- [BB99] Bentsson, P.-O.; Bosch, J.: Haemo Dialysis Software Architecture Design Experiences. In: *Proceedings of the 21st International Conference on Software Engineering*, May 1999, S. 516–526
- [BBK+99] Balzert, H.; Behle, A.; Kelter, U.; Nagl, M.; Pauen, P.; Schäfer, W.; Six, H.; Voss, J.; Wadsack, J.; Weidauer, C.; Westfechtel, B.: Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme. September 1999
- [BBS03] Balsamo, S.; Bernardo, M.; Simeoni, M.: Performance evaluation at the software architecture level. In: Bernardo und Inveradi [BI03], S. 207–258

- [BC89] Beck, K.; Cunningham, W.: A laboratory for teaching object oriented thinking. In: *OOPSLA '89: Conference proceedings on Object-oriented programming systems, languages and applications*, New York, NY, USA: ACM Press, 1989, S. 1–6
- [BC00] Baldwin, C. Y.; Clark, K. B.: *The Power of Modularity*, Bd. 1 von *Design Rules*. MIT Press, 2000
- [BCD00] Bernardo, M.; Ciancarini, P.; Donatiello, L.: AEMPA: A Process Algebraic Description Language for the Performance Analysis of Software Architectures. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2000, S. 1–11
- [BCDW04] Bradbury, J.; Cordy, J.; Dingel, J.; Wermelinger, M.: A Survey of Self Management in Dynamic Software Architecture Specifications. In: *Proceedings of the ACM SIGSOFT International Workshop on Self-Managed Systems (WOSS'04)*, ACM Press, 2004
- [BCH<sup>+</sup>02] Baron, A.; Clarke, B.; Hertroys, P.; von Oosterom, N.; Hinley, D.: *Application Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2002
- [BCJ<sup>+</sup>02] Berry, C.; Carnell, J.; Juric, M.; Kunnumpurath, M.; Nashi, N.; Romanosky, S.: *J2EE Design Patterns Applied*. Wrox Press, 2002
- [BCK03] Bass, L.; Clements, P.; Kazman, R.: *Software Architecture in Practice*. SEI Series in Software Engineering, Addison-Wesley, 2. Aufl., 2003
- [BCR94] Basili, V. R.; Caldiera, G.; Rombach, H. D.: Goal Question Metric Paradigm. In: Marciniak, J. J. (Hrsg.), *Encyclopedia of Software Engineering*, John Wiley & Sons, Bd. 1, Kap. Goal Question Metric Paradigm, 1994, S. 528–532
- [BD00] Bruegge, B.; Dutoit, A.: *Object-Oriented Software Engineering – Conquering Complex and Changing Systems*. Prentice Hall, 2000
- [BDIS04] Balsamo, S.; DiMarco, A.; Inverardi, P.; Simeoni, M.: Model-Based Performance Prediction in Software Development: A Survey. In: *IEEE Transactions on Software Engineering* 30 (2004), Nr. 5, S. 295–310
- [BDM02] Bernardi, S.; Donatelli, S.; Merseguer, J.: From UML sequence diagrams and state-charts to analysable Petri net models. In: *Proceedings of WOSP2002*, 2002
- [BDW98] Briand, L. C.; Daly, J. W.; Wuest, J.: A Unified Framework for Cohesion Measurement in Object-Oriented Systems. In: *Empirical Software Engineering* 3 (1998), Nr. 1, S. 65–117
- [BE93] Beck, J.; Eichmann, D.: Program and Interface Slicing for Reverse Engineering. In: *Proceedings: 15th International Conference on Software Engineering*, IEEE Computer Society Press / ACM Press, 1993, S. 509–518
- [BEA] BEA: BEA Weblogic.

- URL <http://www.bea.com/products/weblogic/>
- [Bec97] Beck, K.: *Smalltalk – praxisnahe Gebrauchsmuster*. Markt und Technik, 1997
- [Bec03] Beck, K.: *Extreme Programming – die revolutionäre Methode für Softwareentwicklung in kleinen Teams*. Programmer's choice, Addison-Wesley, 2003
- [BEJV96] Binns, P.; Englehart, M.; Jackson, M.; Vestal, S.: Domain-Specific Software Architectures for Guidance, Navigation and Control. In: *International Journal of Software Engineering and Knowledge Engineering* 6 (1996), Nr. 2, S. 201–227
- [Ber98] Bernstein, P. A.: Repositories and Object Oriented Databases. In: *SIGMOD Record* 27 (1998), Nr. 1, S. 88–96
- [Béz05] Bézivin, J.: On the Unification Power of Models. In: *Software and Systems Modeling* 4 (2005), Nr. 2, S. 171–188
- [BFF<sup>+</sup>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<sup>+</sup>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 Nondeterminism, Priorities, Probabilities and Time. In: *Theoretical Computer Science* 202 (1998), S. 1–54
- [BG04] Bauer, A.; Günzel, H. (Hrsg.): *Data-Warehouse-Systeme – Architektur, Entwicklung, Anwendung*. dpunkt.verlag, 2. Aufl., 2004
- [BGM03] Balsamo, S.; Grosso, M.; Marzolla, M.: *Towards Simulation-Based Performance Modeling of UML Specifications*. Technischer Bericht CS-2003-2, Dep. di Informatica, Universtita Ca' Foscari Venezia, Italy, 2003
- [BGMT99] Bolch, G.; Greiner, S.; Meer, H.; Trivedi, K.: *Queueing Networks and Markov Chains*. John Wiley and Sons, 1999
- [BH04a] Bischofs, L.; Hasselbring, W.: A Hierarchical Super Peer Network for Distributed Software Development. In: de Lucia, A.; Gall, H. C.; Dustdar, S. (Hrsg.), *Proceedings of the Workshop on Cooperative Support for Distributed Software Engineering*

- Processes (CSSE 2004)*, Linz, Austria: Austrian Computer Society, September 2004, S. 99–106
- [BH04b] Buhl, H.-U.; Heinrich, B.: Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von 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
- [BHN+04] Bischofs, L.; Hasselbring, W.; Niemann, H.; Schuldt, H.; Wurz, M.: Verteilte Architekturen zur intra- und inter-institutionellen Integration von Patientendaten. In: Ammenwerth, E.; Gaus, W.; Haux, R.; Lovis, C.; Pfeiffer, K.; Tilg, B.; Wichmann, H. (Hrsg.), *Tagungsband der 49. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds2004)*, Innsbruck, Österreich: videel, September 2004, S. 87–89
- [BHSS04] Bischofs, L.; Hasselbring, W.; Schlegelmilch, J.; Steffens, U.: A Hierarchical Super Peer Network for Distributed Artifacts. In: Agosti, M.; Schek, H.-J.; Türker, C. (Hrsg.), *Pre-proceedings of the Sixth Thematic Workshop of the EU Network of Excellence DELOS*, Cagliari, Italy: Edizioni Libreria Progetto, Padova, 2004, S. 105–114
- [BHTV03] Baresi, L.; Heckel, R.; Thöne, S.; Varró, D.: Modeling and validation of service-oriented architectures: Application vs. style. In: *Proc. of the European Software Engineering Conference/Foundations of Software Engineering*, ACM Press, 2003, S. 68–77
- [BHTV04] Baresi, L.; Heckel, R.; Thöne, S.; Varró, D.: Style-Based Refinement of Dynamic Software Architectures. In: *Proc. of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA4)*, IEEE Computer Society Press, 2004, S. 155–164
- [BI03] Bernado, M.; Inveradi, P. (Hrsg.): *Formal Methods for Software Architecture*, Bd. 2804 von *Lecture Notes in Computer Science*, Springer-Verlag, 2003
- [Bie02] Bien, A.: *J2EE Patterns – Entwurfsmuster für die J2EE*. Programmer's choice, Addison-Wesley, 2002
- [Bir99] Birolini, A.: *Reliability Engineering: Theory and Practice*. Springer-Verlag, 3. Aufl., 1999
- [BK03a] Björkander, M.; Kobryn, C.: Architecting Systems with UML-2.0. 2003,  
URL <http://www.uml-forum.com/papers.htm>
- [BK03b] Block, M.; Konrad, S.: *Personalized and multimedia Webservice – Sightseeing4U*. Individuelle Projekte, Carl von Ossietzky

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

- [BKPS04] Böckle, G.; Knauber, P.; Pohl, K.; Schmid, K. (Hrsg.):  
*Software-Produktlinien – Methoden, Einführung und Praxis*.  
dpunkt.verlag, 2004
- [BKS04] Boll, S.; Krösche, J.; Scherp, A.: Personalized Mobile Multimedia  
meets Location-Based Services. In: Dadam, P.; Reichert, M. (Hrsg.),  
*Multimedia-Informationssysteme Workshop im Rahmen der 34.*  
*Jahrestagung der Gesellschaft für Informatik (Informatik 2004)*,  
Bd. 2, Ulm, Deutschland: GI, September 2004, Bd. 51 von *LNI*,  
S. 64–69
- [BLKW03] Boll, S.; Krösche, J.; Wegener, C.: Paper chase revisited – a real  
world game meets hypermedia. In: *Proc. der Intl. Conference on*  
*Hypertext (HT03)*, ACM, 2003, S. 126–127
- [BLBV04] Bengtsson, P.; Lassing, N.; Bosch, J.; van Vliet, H.:  
Architecture-level modifiability analysis (ALMA). In: *Journal of*  
*Systems & Software* 69 (2004), Nr. 1-2, S. 129–147
- [Blo01] Bloch, J.: *Effective Java*. Addison-Wesley, 2001
- [BM98] Baniassad, E. L. A.; Murphy, G. C.: Conceptual module querying  
for software reengineering. In: *Proceedings of the 20th*  
*international conference on Software engineering*, IEEE Computer  
Society Press, 1998, S. 64–73
- [BM04] Bertolino, A.; Mirandola, R.: CB-SPE Tool: Putting  
Component-Based Performance Engineering into Practice. In:  
Crnkovic, I.; Stafford, J. A.; Schmidt, H. W.; Wallnau, K. C.  
(Hrsg.), *CBSE 2004*, Springer-Verlag, 2004, Bd. 3054 von *Lecture*  
*Notes in Computer Science*, S. 233–248
- [BMM<sup>+</sup>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<sup>+</sup>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.; Mitbender, B. G.; Webster, W.: Program  
Understanding and the Concept Assignment Problem. In:  
*Communications of the ACM* 37 (1994), Nr. 5, S. 72–82
- [BN03] Bruns, K.; Neidhold, B.: *Audio-, Video- und*  
*Grafikprogrammierung*. München, Wien: Fachbuchverlag Leipzig  
im Carl Hanser Verlag, 2003
- [BO02] Bolusset, T.; Oquendo, F.: Formal refinement of dynamic software  
architectures based on rewriting logic. In: *Proc. RCS02, Int.*  
*Workshop on Refinement of Critical Systems*, 2002

- [Boe88] Boehm, B. W.: A Spiral Model of Software Development and Enhancement. In: *Computer* May 1988 (1988), S. 61–72
- [Bol03] Boll, S.: Vienna 4 U – What Web Services can do for personalized multimedia applications. In: *Seventh Multi-Conference on Systemics (SCI 2003), Cybernetics and Informatics*, Juli 2003, S. 220–225
- [Bor03] Borchers, B.: Verursacherbedingt verspätet – Das »fortschrittlichste Mautsystem der Welt« und die Realität. In: *C't – Magazin für Computer-Technik* (2003), Nr. 22
- [BOR04] Bastarrica, M. C.; Ochoa, S. F.; Rossel, P. O.: Integrated Notation for Software Architecture Specification. In: *Proc. of the XXIV International Conference of the SCCC*, 2004, S. 26–35
- [Bos00] Bosch, J.: *Design and Use of Software Architectures – Adopting and evolving a product-line approach*. Addison-Wesley, 2000
- [Bos02] Bosch, J.: Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. In: Chastek, G. J. (Hrsg.), *Proceedings of Software Product Lines, Second International Conference (SPLC2)*, Springer-Verlag, August 2002, Bd. 2379 von *Lecture Notes in Computer Science*
- [Bos04] Bosch, J.: Software Variability Management. Tutorial at 4th Working IEEE/IFIP Conference on Software Architecture (WICSA), Juni 2004
- [BP93] Burns, D. J.; Pitblado, R. M.: A Modified Hazop Methodology for Safety Critical System Assessment. In: Redmill, F.; Anderson, T. (Hrsg.), *Directions in Safety-Critical Systems*, Springer-Verlag, 1993, S. 232
- [BPM05] *BPML*. Technischer Bericht, Business Process Management Initiative, 2005,  
URL <http://www.bpml.org/BPML.htm>
- [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
- [Bra98] Braband, J.: RAMS-Management nach CENELEC. In: *Signal + Draht* 98 (1998), Nr. 12, S. 20–24
- [Bro00] Brown, A. W.: *Large-Scale Component-Based Development*. Prentice Hall, Englewood Cliffs, NJ, USA, 2000
- [Bro02] Broemmer, D.: *J2EE Best Practices – Java Design Patterns, Automation, and Performance*. John Wiley & Sons, 2002
- [BS95] Brodie, M.; Stonebraker, M.: *Migrating Legacy Systems – Gateways, Interfaces and The Incremental Approach*. San Francisco, CA, USA: Morgan Kaufmann, 1995



- [BS97] Bellin, D.; Simone, S. S.: *The CRC Card Book*. Addison-Wesley, 1997
- [BS01] Broy, M.; Stølen, K.: *Specification and Development of Interactive Systems. Focus on Streams, Interfaces and Refinement*. Springer-Verlag, 2001
- [BS02] Broy, M.; Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung – Eine kritische Einschätzung. In: *Informatik-Spektrum 25* (2002), Nr. 1, S. 3–11
- [BSB+04] Beneken, G.; Seifert, T.; Baehr, N.; Hanschke, I.; Rauch, O.: Referenzarchitekturen und MDA. In: Dadam, P.; Reichert, M. (Hrsg.), *INFORMATIK 2004 – Informatik verbindet, Bd. 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, Gesellschaft für Informatik, 2004, Bd. 51 von *GI-Edition Lecture Notes in Informatics*, S. 101–105
- [BSF02] Boger, M.; Sturm, T.; Fragemann, F.: Refactoring Browser for UML. In: Wells und Williams [WW02], S. 77–81
- [BST02] Bidoit, M.; Sannella, D.; Tarlecki, A.: Architectural Specifications in Casl. In: *Formal Aspects of Computing 13* (2002), S. 252–273
- [Buh98] Buhr, R. J. A.: Use Case Maps as Architectural Entities for Complex Systems. In: *IEEE Trans. Softw. Eng.* 24 (1998), Nr. 12, S. 1131–1155
- [BVH+03] Burnett, I.; Van de Walle, R.; Hill, K.; Bormans, J.; Pereira, F.: MPEG-21: Goals and Achievements. In: *IEEE MultiMedia* 10 (2003), Nr. 4, S. 60–70
- [CBB+03] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; Stafford, J.: *Documenting Software Architectures – Views and Beyond*. SEI Series in Software Engineering, Addison-Wesley, 2003
- [CC92] Chikofsky, E. J.; Cross, II, J. H.: Reverse Engineering and Design Recovery: A Taxonomy. In: *Software Reengineering*, IEEE Computer Society Press, 1992, S. 54–58
- [CC02] Carey, J.; Carlson, B.: *Framework Process Patterns – Lessons Learned Developing Application Frameworks*. Addison-Wesley, 2002
- [CCMT93] Canfora, G.; Cimitile, A.; Munro, M.; Taylor, T.: Extracting Abstract Data Types from C Programs: A Case Study. In: *Proceedings of the International Conference on Software Maintenance 1993*, IEEE Computer Society Press, September 1993, S. 200–209
- [CE00] Czarnecki, K.; Eisenecker, U. W.: *Generative Programming – Methods, Tools and Applications*. Addison-Wesley, 2000

- [CEN00] CENELEC (European Committee for Electro-technical Standardisation): CENELEC EN 50128: Railway Applications: Software for Railway Control and Protection Systems CENELEC, Brussels. 2000
- [Che76] Chen, P.: The Entity–Relationship Model — Towards a Unified View of Data. In: *ACM Transactions on Database Systems* 1 (1976), Nr. 1, S. 9–36
- [CHKT06] Conrad, S.; Hasselbring, W.; Koschel, A.; Tritsch, R.: *Enterprise Application Integration*. Elsevier Spektrum Akademischer Verlag, 2006
- [CKK02] Clements, P.; Kazman, R.; Klein, M.: *Handbook of Software Architecture Evaluation*. Addison-Wesley, 2002
- [Cle96] Clements, P. C.: A Survey of Architecture Description Languages. In: *IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design, Washington, DC, USA*, IEEE Computer Society Press, 1996, S. 16–25
- [CMR+97] Corradini, A.; Montanari, U.; Rossi, F.; Ehrig, H.; Heckel, R.; Löwe, M.: Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. In: Rozenberg, G. (Hrsg.), *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*, World Scientific, Kap. 3, 1997
- [CN02] Clements, P.; Northrop, L.: *Software Product Lines – Practices and Patterns*. SEI Series in Software Engineering, Addison-Wesley, 2002
- [COB] COBIT-Homepage. Technischer Bericht, IT Governance Institute, URL <http://www.isaca.org/cobit.htm>
- [Coc02] Cockburn, A.: *Agile Software-Entwicklung*. Addison-Wesley, 2002
- [Com] Apache Jakarta Commons.  
URL <http://jakarta.apache.org/commons/>
- [Con02] Conn, R.: Developing Software Engineers at the C-130J Software Factory. In: *IEEE Software* 19 (2002), Nr. 5, S. 25–29
- [Cop92] Coplien, J.: *Advanced C++ Styles and Idioms*. Addison-Wesley, 1992
- [Cor03] Cordy, J.: Comprehending Reality: Practical Challenges to Software Maintenance Automation. In: *Keynote address at IEEE 11th International Workshop on Program Comprehension, Portland*, IEEE Computer Society Press, 2003
- [CPT99] Canal, C.; Pimentel, E.; Troya, J. M.: Specification and refinement of dynamic software architectures. In: *Software Architecture*, Kluwer Academic Publishers, 1999, S. 107–126
- [CRW98] Clayton, R.; Rugaber, S.; Wills, L.: On the Knowledge Required to Understand a Program. In: *Proceedings of WCRE98*, IEEE Computer Society Press, Oktober 1998, S. 69–78

- [CSWH01] Clarke, I.; Sandberg, O.; Wiley, B.; Hong, T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Federrath, H. (Hrsg.), *Designing Privacy Enhancing Technologies – International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, Springer-Verlag, Juli 2001, Bd. 2009 von *Lecture Notes in Computer Science*, S. 46–66
- [CV95] Cimitile, A.; Visaggio, G.: Software Salvaging and the Call Dominance Tree. In: *Journal of Systems & Software* 28 (1995), S. 117–127
- [CW04] Correa, A. L.; Werner, C. M. L.: Applying Refactoring Techniques to UML/OCL Models. In: Baar, T.; Strohmeier, A.; Moreira, A. M. D.; Mellor, S. J. (Hrsg.), *UML*, Springer-Verlag, 2004, Nr. 3273 in *Lecture Notes in Computer Science*, S. 173–187
- [Daw01] Dawis, E. P.: Architecture of an SS7 Protocol Stack on a Broadband Switch Platform using Dualistic Petri Nets. In: *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2001, S. 323–326
- [DDK01] Dawis, E. P.; Dawis, J. E.; Koo, W. P.: Architecture of Computer-based Systems using Dualistic Petri Nets. In: *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, Oktober 2001
- [DDN00] Demeyer, S.; Ducasse, S.; Nierstrasz, O.: Finding refactorings via change metrics. In: *OOPSLA*, 2000, S. 166–177
- [DEM<sup>+</sup>99] Depke, R.; Engels, G.; Mehner, K.; Sauer, S.; Wagner, A.: Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen. In: *Informatik – Forschung und Entwicklung* 14 (1999), Nr. 2, S. 83–94
- [Der03] Dern, G.: *Management von IT-Architekturen – Informationssysteme im Fokus von Architekturplanung und -entwicklung*. Edition CIO, Wiesbaden: Vieweg Verlag, 2003
- [Deu02] van Deursen, A. (Hrsg.): *Proceedings / Ninth Working Conference on Reverse Engineering (WCRE02)*, IEEE Computer Society Press, 2002
- [DF98] Diaconescu, R.; Futatsugi, K.: *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*. World Scientific, 1998
- [DFI99] Diaconescu, R.; Futatsugi, K.; Iida, S.: Component-Based Algebraic Specification and Verification in CafeOBJ. In: *World Congress on Formal Methods (2)*, 1999, S. 1644–1663
- [DHM99] Duke, D. J.; Herman, I.; Marschall, M. S.: *PREMO: A Framework for Multimedia Middleware – Specification, Rationale, and Java Binding*. LNCS 1591, Springer-Verlag, 1999

- [Die03] Dietzsch, A.: Positionierung eines Unternehmensarchitektur-Ansatzes: Erfahrung der Schweizerischen Mobiliar im Architekturmanagement. In: *Enterprise Architecture und Enterprise Application Integration (EAI) – Proceedings des GI-Arbeitskreis Enterprise Architecture Frühjahrskonferenz 2003*, St. Gallen: Institut für Wirtschaftsinformatik IWI-HSG, 2003, S. 50–61
- [Die04] Dietrich, A.: Puff in der Landschaft. In: *NZZ-Folio* (2004), Nr. 5/04, S. 30–37
- [Dij70] Dijkstra, E. W.: Notes on Structured Programming. April 1970, URL <http://www.cs.utexas.edu/users/EWD/>
- [DIN68] DIN: *DIN 66200: Betrieb von Rechensystemen – Begriffe, Auftragsbeziehungen*. Technischer Bericht, Deutsches Institut für Normung e.V., 1968. Beuth Verlag Berlin
- [DIN90a] DIN: *DIN 25424: Fault Tree Analysis: Part 1 (Method and graphical symbols) and Part 2 (Manual: calculation procedures for the evaluation of a fault tree)*. Technischer Bericht, Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 1981/1990
- [DIN90b] DIN: *DIN V VDE 0801 –Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben*. Technischer Bericht, Deutsches Institut für Normung e.V., 1990
- [DIN91] DIN: *DIN 9126 – Information Technology, Software Product Evaluation, Quality, Characteristics and Guidelines for their Use*. Technischer Bericht, Deutsches Institut für Normung e.V., 1991
- [DK99] van Deursen, A.; Kuipers, T.: Identifying Objects Using Cluster and Concept Analysis. In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, Los Angeles, USA: ACM, Mai 1999, S. 246–255
- [DKK+01] Dabek, F.; Kaashoek, M. F.; Karger, D.; Morris, R.; Stoica, I.: Wide-area cooperative storage with CFS. In: *Proceedings of the eighteenth ACM symposium on Operating systems principles*, ACM Press, 2001, S. 202–215
- [DL04] Dyson, P.; Longshaw, A.: *Architecting Enterprise Solutions: Patterns for High-Capability Internet-Based Systems*. John Wiley & Sons, 2004
- [Dou99] Douglass, B. P.: *Doing Hard Time – Developing Real-time Systems with UML, Objects, Frameworks, and Patterns*. Object Technology Series, Addison-Wesley, 1999
- [DRD99] Ducasse, S.; Rieger, M.; Demeyer, S.: A Language Independent Approach for Detecting Duplicated Code. In: *ICSM*, 1999, S. 109–118
- [DS99] DeBaud, J.-M.; Schmid, K.: A Systematic Approach to Derive the Scope of Software Product Lines. In: *IEEE Computer Society*,

- Technical Council on Software Engineering; ACM, Special Interest Group on Software Engineering -SIGSOFT-: International Conference on Software Engineering*, ACM Press, 1999, S. 34–43
- [DW99a] Dröschel, W.; Wiemers, M.: *Das V-Modell 97 – Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. Oldenbourg-Verlag, 1999
- [DW99b] D’Souza, D. F.; Wills, A. C.: *Objects, Components, and Frameworks with UML – The Catalysis Approach*. Addison-Wesley, 1999
- [Dym02] Dymond, K. M.: *CMM Handbuch – Das Capability Maturity Model für Software*. Xpert.press, Springer-Verlag, 2002
- [EBK<sup>+</sup>05] Ehrig, H.; Braatz, B.; Klein, M.; Orejas, F.; Pérez, S.; Pino, E.: Object-Oriented Connector-Component Architectures. In: *Proc. Of FESCA, ETAPS satellite*, 2005
- [EFRV86] Estrin, G.; Fenchel, R. S.; Razouk, R. R.; Vernon, M. K.: SARA (system architects apprentice): modeling, analysis, and simulation support for design of concurrent systems. In: *IEEE Transactions on Software Engineering* 12 (1986), Nr. 2, S. 293–311
- [EJ02] Eden, A. H.; Jahnke, J. H.: Coordinating Software Evolution via Two-Tier Programming. In: *Proceedings of the 5th International Conference on Coordination Models and Languages*, Springer-Verlag, 2002, S. 149–157
- [EM90] Ehrig, H.; Mahr, B.: *Fundamentals of Algebraic Specification, Vol. 2: Module Specifications and Constraints*, Bd. 21 von *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1990
- [EM02] van Emden, E.; Moonen, L.: Java Quality Assurance by Detecting Code Smells. In: van Deursen [Deu02]
- [End93] Endler, M.: A model for distributed management of dynamic changes. In: *4th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, 1993
- [EOB<sup>+</sup>02] Ehrig, H.; Orejas, F.; Braatz, B.; Klein, M.; Piirainen, M.: A Generic Component Concept for System Modeling. In: *Proc. FASE ’02*, Springer-Verlag, 2002, Bd. 2306 von *Lecture Notes in Computer Science*
- [EPB<sup>+</sup>04] Ehrig, H.; Padberg, J.; Braatz, B.; Klein, M.; Orejas, F.; Pérez, S.; Pino, E.: A Generic Framework for Connector Architectures based on Components and Transformations. In: *Proc. FESCA’04, satellite of ETAPS’04, Barcelona, ENTCS*, 2004, Bd. 108, S. 53–67
- [ER03] Endres, A.; Rombach, D.: *A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories*. Addison Wesley, 2003

- [ESN03] Engels, G.; Sauer, S.; Neu, B.: Integrating Software Engineering and User-centred Design for Multimedia Software Developments. In: *In Proc. IEEE Symposia on Human-Centric Computing Languages and Environments – Symposium on Visual/Multimedia Software Engineering*, Auckland, New Zealand: IEEE Computer Society Press, Oktober 2003
- [Fak04] Fakultätentag Informatik: *Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten*. Technischer Bericht, November 2004,  
URL <http://www.ft-informatik.de>
- [FH84] Fjeldstadt, R.; Hamlen, W.: Application Program Maintenance Study: Report to Our Respondents. In: *Proc. IBM GUIDE Conference, no. 48*, April 1984
- [FH00] Fahmy, H.; Holt, R.: Using Graph Rewriting to Specify Software Architectural Transformations. In: *Proc. of Automated Software Engineering (ASE 2000)*, 2000, S. 187–196
- [FHK<sup>+</sup>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<sup>+</sup>95] Franks, G.; Hubbard, A.; Majumdar, S.; Petriu, D.; Rolia, J.; Woodside, M.: A toolset for Performance Engineering and Software Design of Client-Server Systems. In: *Performance Evaluation* 24 (1995), Nr. 1-2, S. 117–135
- [FI99] Futatsugi, K.; Iida, S.: Formal Architecture Description Techniques for Software Evolution. In: *Proc. Int. Workshop on the Principles of Software Evolution*, 1999
- [FK98] Frolund, S.; Koistinen, J.: *QML: A Language for Quality of Service Specification*. Technischer Bericht HPL-98-10, Hewlett-Packard Laboratories, 1998
- [FKB<sup>+</sup>05] Firus, V.; Koziolok, H.; Becker, S.; Reussner, R.; Hasselbring, W.: Empirische Bewertung von Performanz-Vorhersageverfahren für Software-Architekturen. In: *Tagungsband Software Engineering 2005 – Fachtagung des GI-Fachbereichs Softwaretechnik, Bd. P-64 der Reihe Lecture Notes in Informatics*, März 2005, S. 55–66
- [FM93] Fenelon, P.; McDermid, J. A.: An Integrated Toolset For Software Safety Analysis. In: *Journal of Systems & Software* 21 (1993), Nr. 3, S. 279–290
- [FM97] Fiadeiro, J. L.; Maibaum, T.: Categorical semantics of parallel program design. In: *Science of Computer Programming* 28 (1997), S. 111–138

- [FMC05] FMC Consortium: Fundamental Modeling Concepts Home Page. Website, 2005,  
URL <http://f-m-c.org/>
- [FMNP94] Fenelon, P.; McDermid, J.; Nicholson, M.; Pumfrey, D. J.: Towards Integrated Safety Analysis and Design. In: *ACM SIGAPP Applied Computing Review* 2 (1994), Nr. 1, S. 21–32
- [Foe03] Foegen, M.: Architektur und Architekturmanagement. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 57–65
- [Fou] Four J's: Four J's Development Tools.  
URL <http://www.4js.com>
- [Fow97] Fowler, M.: *Analysis Patterns*. Addison-Wesley, 1997
- [Fow00] Fowler, M.: *Refactoring – Wie Sie das Design vorhandener Software verbessern*. Addison-Wesley, 2000
- [Fow03] Fowler, M.: *Patterns für Enterprise-Application-Architekturen*. Bonn: mitp, 2003
- [FP97] Fenton, N. E.; Pfleeger, S. H.: *Software Metrics – a rigorous and practical approach*. International Thomson Computer Press, 1997
- [FPR00] Fontoura, M.; Pree, W.; Rumpe, B.: UML-F: A Modeling Language for Object-Oriented Frameworks. In: *ECOOP*, Springer-Verlag, Bd. 1850 von Lecture Notes in Computer Sciences, 2000, S. 63–82
- [FPR02] Fontoura, M.; Pree, W.; Rumpe, B.: *The UML Profile for Framework Architectures*. Object Technology Series, Addison-Wesley, 2002
- [Fra99] Franks, G.: *Performance Analysis of Distributed Server Systems*. Dissertation, Carleton University, Ottawa, 1999
- [FRF+02] Fowler, M.; Rice, D.; Foemmel, M.; Hieatt, E.; Mee, R.; Stafford, R. (Hrsg.): *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002
- [FSJ99a] Fayad, M. E.; Schmidt, D. C.; Johnson, R. E. (Hrsg.): *Building Application Frameworks – Object-Oriented Foundations of Framework Design*. Wiley computer publishing, John Wiley & Sons, 1999
- [FSJ99b] Fayad, M. E.; Schmidt, D. C.; Johnson, R. E. (Hrsg.): *Implementing Application Frameworks: Object-Oriented Frameworks at Work*. Wiley computer publishing, John Wiley & Sons, 1999
- [FV03] Faust, D.; Verhoef, C.: Software Product Line Migration and Deployment. In: *Software – Practice and Experience* 33 (2003), S. 933–955
- [Gam92] Gamma, E.: *Objektorientierte Software-Entwicklung am Beispiel von ET++ – Design-Muster, Klassenbibliothek, Werkzeuge*. Springer-Verlag, 1992

- [Gar03] Garlan, D.: Formal modeling and analysis of software architecture: Components, connectors, and events. In: Bernado und Inveradi [BI03], S. 1–24
- [GBS01] van Gurp, J.; Bosch, J.; Svahnberg, M.: On the Notion of Variability in Software Product Lines. In: Kazman, R.; Kruchten, P.; Verhoef, C.; van Vliet, H. (Hrsg.), *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01)*, IEEE Computer Society Press, 2001, S. 45–54
- [GFd98] Griss, M.; Favaro, J.; d'Alessandro, M.: Integrating Feature Modeling with the RSEB. In: *Proceedings of the Fifth International Conference on Software Reuse*, Vancouver, BC, Canada, 1998, S. 76–85
- [GH94] Gilmore, S.; Hillston, J.: The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling. In: *Proceedings of the Seventh International Conference for Modelling Techniques and Tools for Performance Evaluation*, 1994, S. 353–368
- [GHJV04] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. Programmer's Choice, Addison-Wesley, 2004
- [GHOS96] Gray, J.; Helland, P.; O'Neil, P.; Shasha, D.: The Dangers of Replication and a Solution. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, ACM Press, 1996, S. 173–182
- [GHR<sup>+</sup>02] Graham, P.; Hulzinga, S.; Rudd, C.; van Dijk, A.; von Winden, R.; Bekenkamp, P.; Doeff, I.; Hinley, D.; Leigh, C.; Stringfellow, I.; Wielinga, R.: *ICT Infrastructure Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2002
- [Gia95] Giannakopoulou, D.: *The TRACTA Approach for Behaviour Analysis of Concurrent Systems*. Technical report DoC 95/16, Department of Computing, Imperial College of Science, Technology and Medicine, September 1995
- [GKKS04] Gröne, B.; Knöpfel, A.; Kugel, R.; Schmidt, O.: *The Apache Modelling Project*. Institute Report 5, Hasso-Plattner-Institut für Softwaresystemtechnik, 2004, URL <http://apache.hpi.uni-potsdam.de>
- [GKP05] Grunske, L.; Kaiser, B.; Papadopoulos, Y.: Model-Driven Safety Evaluation with State-Event-Based Component Failure Annotations. In: *CBSE Component-based Software Engineering*, 2005, S. 33–48
- [GKR05] Grunske, L.; Kaiser, B.; Reussner, R.: Annotation of Component Specifications with Modular Analysis Models for Safety Properties.



- In: *Embedded Software Development with Components – An Overview on Current Research Trends*, Springer-Verlag, 2005, S. 737–748
- [Gla98a] Glass, R.: Maintenance: Less Is Not More. In: *IEEE Software* 15 (1998), Nr. 4, S. 67–68
- [Gla98b] Glass, R. L.: *Software Runaways. Lessons learned from Massive Software Project Failures*. Prentice Hall, 1998
- [Gla02a] Glass, R. L.: *Facts and Fallacies of Software Engineering*. Addison-Wesley, 2002
- [Gla02b] Glass, R. L.: The Naturalness of Object Orientation: Beating a Dead Horse? In: *IEEE Software* 19 (2002), Nr. 3, S. 103–104
- [Gla04] Glass, R. L.: Learning to Distinguish a Solution from a Problem. In: *IEEE Software* 21 (2004), Nr. 3, S. 111–112
- [GM02] Grassi, V.; Mirandola, R.: PRIMAmob-UML: A Methodology for Performance Analysis of Mobile Software Architectures. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2002, S. 262–274
- [GMW97] Garlan, D.; Monroe, R.; Wile, D.: Acme: An Architecture Description Interchange Language. In: *Proc. of CASCON'97*, 1997, S. 169–183
- [Gol04] Golm, M.: Offene Systemarchitekturen in der Automobilsoftware. In: *Objektspektrum* (2004), Nr. 5, S. 60–63
- [Gom04] Gomaa, H.: *Designing Software Product Lines with UML – From Use-Cases to Pattern-Based Software-Architectures*. Addison-Wesley, August 2004
- [Gov99] Govoni, D.: *Java Application Frameworks*. John Wiley & Sons, 1999
- [GP02] Gu, G.; Petriu, D.: XSLT Transformations from UML Models to LQN Performance Models. In: *ACM Proceedings of the International Workshop on Software and Performance*, 2002
- [Gra] GraTra: Graph Transformations & Graph Grammars.  
URL <http://www.gratra.org/>
- [Gru04] Grunske, L.: *Strukturorientierte Optimierung der Qualitätseigenschaften von softwareintensiven technischen Systemen im Architekturentwurf*. Dissertation, Universität Potsdam, 2004
- [GSCK04] Greenfield, J.; Short, K.; Cook, S.; Kent, S.: *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. John Wiley & Sons, 2004
- [GSMD03] van Gorp, P.; Stenten, H.; Mens, T.; Demeyer, S.: Towards automating source-consistent UML Refactorings. In: *Proceedings of UML 03*, 2003

- [Hab92] Habel, A.: *Hyperedge replacement: Grammars and Languages*, Bd. 643 von *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 1992
- [Hag03] Hagen, C.: Integrationsarchitektur der Credit Suisse. In: Aier, S.; Schönherr, M. (Hrsg.), *Enterprise Application Integration*, Berlin: GITO-Verlag, 2003, S. 61–81
- [Has97] Hasselbring, W.: Federated Integration of Replicated Information within Hospitals. In: *International Journal on Digital Libraries* 1 (1997), Nr. 3, S. 192–208
- [Has00] Hasselbring, W.: Information System Integration. In: *Communications of the ACM* 43 (2000), Nr. 6, S. 33–38
- [Has02] Hasselbring, W.: Component-Based Software Engineering. In: Chang, S. (Hrsg.), *Handbook of Software Engineering and Knowledge Engineering*, New Jersey: World Scientific Publishing, 2002, S. 289–305
- [HB85] Hutchens, D.; Basili, V.: System structure analysis: clustering with data bindings. In: *IEEE Transactions on Software Engineering* SE-11 (1985), Nr. 8, S. 749–757
- [Hei02] Heinrich, L. J.: *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*. München, Wien: Oldenbourg-Verlag, 7. Aufl., 2002
- [HFS04] Hein, A.; Fischer, T.; Stiel, S.: Fahrerassistenzsystem bei der Robert Bosch GmbH. In: Böckle et al. [BKPS04], S. 193–205
- [HH03] Hochstein, A.; Hunziker, A.: Serviceorientierte Referenzmodelle des IT-Managements. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 46–56
- [HHK<sup>+</sup>00] Hermanns, H.; Herzog, U.; Klehmet, U.; Mertsiotakis, V.; Siegle, M.: Compositional Performance Modelling with the TIPPool. In: *Performance Evaluation* 39 (2000), Nr. 1-4, S. 5–35
- [HHK02] Hermanns, H.; Herzog, U.; Katoen, J.: Process Algebra for Performance Evaluation. In: *Theoretical Computer Science* 274 (2002), Nr. 1-2, S. 43–87
- [HHKS97] Heuer-Hasenplatt, H.; Hollunder, B.; Kittlaus, H.-B.; Schumacher, N.: Bausteinorientierte Anwendungsentwicklung: Voraussetzungen, Anforderungen und Auswirkungen. In: *Objektspektrum*, Nr. 3, 1997, S. 40–51
- [Hil93] Hillston, J.: *PEPA – Performance Enhanced Process Algebra*. Technischer Bericht, Dept. of Computer Science, University of Edinburgh, 1993
- [HIM99] Hirsch, D.; Inverardi, P.; Montanari, U.: Modeling Software Architectures and Styles with Graph Grammars and Constraint Solving. In: *Proc. Working IFIP Conference on Software Architecture*, 1999

- [HK03] Hitz, M.; Kappel, G.: *UML Work: Von der Analyse zur Realisierung*. dpunkt.verlag, 2. Aufl., 2003
- [HKKI04] Higo, Y.; Kamiya, T.; Kusumoto, S.; Inoue, K.: Refactoring Support Based on Code Clone Analysis. In: Bomarius, F.; Iida, H. (Hrsg.), *PROFES*, Springer-Verlag, 2004, Nr. 3009 in *Lecture Notes in Computer Science*, S. 220–233
- [HM99] Hirsch, D.; Montanari, U.: Consistent transformations for software architecture styles of distributed systems. In: *Electronic Notes in Theoretical Computer Science* 28 (1999), S. 23–40
- [HN90] Harandi, M. T.; Ning, J. Q.: Knowledge-Based Program Analysis. In: *IEEE Software* 7 (1990), Nr. 1, S. 74–81
- [HNS00] Hofmeister, C.; Nord, R.; Soni, D.: *Applied Software Architecture*. Object technology series, Addison-Wesley, 2000
- [Hoa85] Hoare, C. A. R.: *Communicating Sequential Processes*. Prentice-Hall international series in computer science, Prentice Hall, 1985
- [HPR89] Horwitz, S.; Pfeiffer, P.; Reps, T.: Dependence analysis for pointer variables. In: *Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation*, ACM Press, 1989, S. 28–40,
- [HRB88] Horwitz, S.; Reps, T.; Binkley, D.: Interprocedural Slicing Using Dependence Graphs. In: *Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation*, 1988, S. 35–46
- [HRJ<sup>+</sup>04] Hasselbring, W.; Reussner, R.; Jaekel, H.; Schlegelmilch, J.; Teschke, T.; Kriehoff, S.: The Dublo Architecture Pattern for Smooth Migration of Business Information Systems: An experience report. In: *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)*, IEEE Computer Society Press, Mai 2004, S. 117–126
- [Hub04] Huber-Graul, M.: Software-Wartung als Herausforderung. 2004, URL <http://www.nzz.ch/2003/02/04/hy/page-article8NCG9.html>
- [Hüs94] Hüsener, T.: *Entwurf komplexer Echtzeitsysteme: State of the Art*. Nr. 11 in *Angewandte Informatik*, BI Wiss.-Verlag, 1994
- [HW03] Hohpe, B.; Woolf, G.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003
- [HW04] Heinrich, B.; Winter, R.: A Strategy Modelling Technique for Financial Services. In: *The European IS Profession in the Global Networking Environment, Proc. 12th European Conference on Information Systems*, 2004

- [HW05] Hafner, M.; Winter, R.: Vorgehensmodell für das Management der unternehmensweiten Applikationsarchitektur. In: Ferstl, O. K.; Sinz, E. J.; Eckert, S.; Isselhorst, T. (Hrsg.), *Wirtschaftsinformatik 2005: eEconomy – eGovernment – eSociety, Bamberg, 23.02.2005*, Heidelberg: Physica, 2005, S. 627–646
- [IBM] IBM: Informix 4GL product family.  
URL <http://www-3.ibm.com/software/data/informix/tools/4gl/>
- [IBM04a] IBM: QBIC Home Page. 2004,  
URL <http://wwwqbic.almaden.ibm.com/>
- [IBM04b] IBM: Rational Unified Process Evaluation V6.13, 2004  
URL <http://www-306.ibm.com/software/awdtools/>
- [IEC90] IEC: *Fault-Tree-Analysis (FTA)*. Standard IEC 61025, International Electrotechnical Commission, Genf, Schweiz, 1990
- [IEE90] IEEE: *Standard Glossary of Software Engineering Terminology*. Standard IEEE 610.12-1990, IEEE, 1990
- [IEE00] IEEE: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems – Description*. Standard IEEE 1471-2000, ANSI/IEEE, 2000
- [ISO98] ISO/IEC: *Reference Model for Open Distributed Processing – Part 1: Overview*. ISO Standard 10746-1, International Organization for Standardization, 1998
- [ISO99] ISO/IEC: *Interface Definition Language*. ISO Standard 14750, International Organization for Standardization, 1999
- [ISO01] ISO/IEC: *Software Engineering – Product Quality – Quality Model*. ISO Standard 9126-1, International Organization for Standardization, 2001
- [ite04] iteratec: iteratec e-Business Referenzarchitektur, Version 1.4. 2004,  
URL [http://www.iteratec.de/dokumente/iteratec\\_ebusiness\\_Referenzarchitektur.pdf](http://www.iteratec.de/dokumente/iteratec_ebusiness_Referenzarchitektur.pdf)
- [ITI] Official ITIL Webpages.  
URL <http://www.ogc.gov.uk/index.asp?id=2261>
- [ITU99] ITU: Message Sequence Charts, ITU-T Recommendation. 1999
- [JAD] JADE: Website,  
URL <http://jade.dautelle.com/>
- [Jah04] Jahnke, J. H.: Reverse engineering software architecture using rough clusters. In: *Processing IEEE Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS'04)*, IEEE Computer Society Press, 2004, S. 4–9
- [JBo04] JBoss Inc.: The JBoss Application Server. 2004,  
URL <http://www.jboss.org/products/jbossas>
- [JBu] JBuilder.  
URL <http://www.borland.com/jbuilder/>

- [JDB] JDBC: Website,  
URL <http://java.sun.com/products/jdbc/>
- [JF88] Johnson, R.; Foote, B.: Designing Reusable Classes. In: *Journal of Object-Oriented Programming* 1 (1988), Nr. 2, S. 22–35
- [JLMM04] Jablonski, S.; Lay, R.; Meiler, C.; Müller, S.: Process Based Data Logistics: a Solution for Clinical Integration Problems. In: *First International Workshop on Data Integration in the Life Sciences (DILS 2004), Leipzig, Germany*, Springer-Verlag, 2004, Nr. 2994 in Lecture Notes in Bioinformatics
- [JN99] Jacobson, E. E.; Nowack, P.: Frameworks and Patterns – Architectural Abstractions. In: Fayad et al. [FSJ99a], S. 29–54
- [JO93] Johnson, R. E.; Opdyke, W. F.: Refactoring and Aggregation. In: Nishio, S.; Yonezawa, A. (Hrsg.), *ISOTAS*, Springer-Verlag, 1993, Nr. 742 in Lecture Notes in Computer Science, S. 264–278
- [JPM03] Jablonski, S.; Petrov, I.; Meiler, C.: An Architectural Framework for Web Applications. In: *Proceedings of ICEIS 2003, 5th International Conference on Enterprise Information Systems, Angers, France*, 2003, Bd. 1, S. 285–293
- [JR01] Jacobi, C.; Rumpe, B.: Hierarchical XP. Improving XP for Large-Scale Projects in Analogy to Reorganization Processes. In: Succi, G.; Marchesi, M. (Hrsg.), *Extreme Programming Examined*, Addison-Wesley, 2001, S. 83–102
- [JRL00] Jazayeri, M.; Ran, A.; van der Linden, F. (Hrsg.): *Software Architecture for Product Families – Principles and Practices*. Addison-Wesley, 2000
- [JSZ97] Jahnke, J.; Schäfer, W.; Zuendorf, A.: Generic Fuzzy Reasoning Nets as a Basis for Reverse Engineering Relational Database Applications. In: Jazayeri, M.; Schauer, H. (Hrsg.), *ESEC/FSE '97*, Springer-Verlag, 1997, Bd. 1301 von *Lecture Notes in Computer Science*, S. 193–210
- [Kan92] Kant, K.: *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992
- [KBAW94] Kazman, R.; Bass, L.; Abowd, G.; Webb, M.: SAAM: A Method for Analyzing the Properties Software Architectures. In: *Proceedings of the 16th International Conference on Software Engineering, (Sorrento, Italy)*, 1994, S. 81–90
- [KBS] KBSt: Standards und Architekturen für eGovernment-Anwendungen.  
URL <http://www.kbst.bund.de/saga>
- [KCH<sup>+</sup>90] Kang, K. C.; Cohen, S. G.; Hess, J. A.; Novak, W. E.; Peterson, A. S.: *Feature-Oriented Domain Analysis (FODA) – Feasibility Study*. Technischer Bericht CMU/SEI-90-TR-21 ESD 90-TR-222,

- Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1990
- [Kel03] Keller, F.: *Über die Rolle von Architekturbeschreibungen im Software-Entwicklungsprozess*. Dissertation, Universität Potsdam, August 2003
- [Ker05] Kerievsky, J.: *Refactoring to Patterns*. Addison-Wesley signature series, Addison-Wesley, 2005
- [KGT06] Knöpfel, A.; Gröne, B.; Tabeling, P.: *Fundamental Modeling Concepts*. John Wiley & Sons, 2006
- [KJ04] Kirchner, M.; Jain, P.: *Patterns for resource management (Pattern-oriented Software Architecture, vol. 3)*. Wiley series in software design patterns, John Wiley & Sons, 2004
- [KKB<sup>+</sup>98] Kazman, R.; Klein, M.; Barbacci, M.; Longstaff, T.; Lipson, H.; Carriere, J.: The Architecture Tradeoff Analysis Method. In: *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 1998, S. 68–78
- [KKC00] Kazman, R.; Klein, M.; Clements, P.: *ATAM: Method for 136 Architecture Evaluation*. Technischer Bericht CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000
- [Kle75] Kleinrock, L.: *Queueing Systems, Volume 1: Theory*. A Wiley-Interscience publication, John Wiley & Sons, 1975
- [Kle99] Kleis, W.: *Konzepte zur verständlichen Beschreibung objektorientierter Frameworks*. Berichte aus der Informatik, Aachen: Shaker Verlag, 1999
- [KLM<sup>+</sup>97] Kiczales, G.; Lamping, J.; Menhdhekar, A.; Maeda, C.; Lopes, C.; Loingtier, J.-M.; Irwin, J.: Aspect-Oriented Programming. In: Akşit, M.; Matsuoka, S. (Hrsg.), *ECOOP'97 – Object-Oriented Programming, 11th European Conference*, Springer-Verlag, Bd. 1241 von *Lecture Notes in Computer Science*, 1997, S. 220–242
- [KLM03] Kaiser, B.; Liggesmeyer, P.; Mäckel, O.: A New Component Concept for Fault Trees. In: *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS'03)*, Adelaide, 2003
- [KMO<sup>+</sup>05] Kreowski, H.-J.; Montanari, U.; Orejas, F.; Rozenberg, G.; Taentzer, G. (Hrsg.): *Formal methods in software and systems modeling – essays dedicated to Hartmut Ehrig on the occasion of his 60th birthday*, Bd. 3393 von *Lecture Notes in Computer Science*. Springer-Verlag, 2005
- [KMRT02] Kienle, A.; Mambrey, P.; Reiband, N.; Tan, D.: Erfolgsfaktoren und Hindernisse bei Wissensmanagementlösungen. In: *GI Jahrestagung 2002*, Gesellschaft für Informatik, 2002, Bd. 19 von *GI-Edition Lecture Notes in Informatics*, S. 709–712

- [KMU03] Kramer, J.; Magee, J.; Uchitel, S.: Software architecture modeling & analysis: A rigorous approach. In: Bernado und Inveradi [BI03], S. 44–51
- [Kos00] Koschke, R.: *Atomic Architectural Component Recovery for Program Understanding and Evolution*. Dissertation, Universität Stuttgart, 2000
- [Koz04] Koziol, H.: *Empirische Bewertung von Performance-Analyseverfahren für Software-Architekturen*. Diplomarbeit, Universität Oldenburg, Fakultät II, Department für Informatik, Okt. 2004
- [KP00] King, P. J. B.; Pooley, R. J.: Derivation of Petri net performance models from UML specifications of communications software. Springer-Verlag, 2000, Bd. 2047 von *Lecture Notes in Computer Science*, S. 262–276
- [KPW04] King, R.; Popitsch, N.; Westermann, U.: METIS: a flexible database foundation for unified media management. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, ACM Press, 2004, S. 744–745
- [KR05] Krahn, H.; Rumpe, B.: *Evolution von Software-Architekturen*. Technischer Bericht Informatik-Bericht 2005-04, Technische Universität Braunschweig, Carl-Friedrich-Gauß-Fakultät für Mathematik und Informatik, Mai 2005
- [Krc90] Krcmar, H.: Bedeutung und Ziele von Informationssystemarchitekturen. In: *Wirtschaftsinformatik* 32 (1990), Nr. 5, S. 395–402
- [Kru95] Kruchten, P.: The 4+1 View Model of Architecture. In: *IEEE Software* 12 (1995), Nr. 6, S. 42–50
- [KTG<sup>+</sup>02] Keller, F.; Tabelaing, P.; Gröne, B.; Knöpfel, A.; Schmidt, O.; Kugel, R.; Apfelbacher, R.: Improving knowledge transfer at the architectural level: Concepts and notations. In: Arabnia, H. R.; Mun, Y. (Hrsg.), *Proceedings of the SERP'02, the International Conference on Software Engineering Research and Practice, Las Vegas*, CSREA Press, Juni 2002, S. 101–107
- [KW99] Kullbach, B.; Winter, A.: Querying as an Enabling Technology in Software Reengineering. In: Verhoef, C.; Nesi, P. (Hrsg.), *Proceedings of the 3rd Euromicro Conference on Software Maintenance and Reengineering*, Los Alamitos: IEEE Computer Society Press, 1999, S. 42–50
- [LBHB99] Lundberg, L.; Bosch, J.; Häggander, D.; Bengtsson, P.-O.: Quality Attributes in Software Architecture Design. In: *Proceedings of the IASTED 3rd International Conference on Software Engineering and Applications*, October 1999, S. 353–362

- [LBVB02] Lassing, N.; Bengtsson, P.; van Vliet, H.; Bosch, J.: Experiences with ALMA: architecture-level modifiability analysis. In: *Journal of Systems & Software* 61 (2002), Nr. 1, S. 47–57
- [Lea99] Lea, D.: *Concurrent Programming in Java, Design Principles and Patterns*. Addison-Wesley, 1999
- [Lev95] Leveson, N. G.: *SAFWARE: System Safety and Computers*. Addison-Wesley, 1995
- [LF99] Lopes, A.; Fiadeiro, J. L.: Using explicit state to describe architectures. In: *Proc. of Fundamental Approaches to Software Engineering*, Springer-Verlag, 1999, Bd. 1577 von *Lecture Notes in Computer Science*, S. 144–160
- [LH96] Larsen, L.; Harrold, M. J.: Slicing object-oriented software. In: *Proceedings of the 18th international conference on Software engineering*, IEEE Computer Society Press, 1996, S. 495–505
- [Lig00] Liggesmeyer, P.: *Qualitätssicherung softwareintensiver technischer Systeme*. Spektrum Akademischer Verlag, 2000
- [Lin03] Linthicum, D.: *Next Generation Application Integration: From Simple Information to Web Services*. Addison-Wesley, 2003
- [LN95] Landin, N.; Niklasson, A.: *Development of Object-Oriented Frameworks*. Diplomarbeit, Department of Communication Systems, Lund Institute of Technology, Lund University, Lund, Sweden, 1995
- [LRR03] Lichtenegger, R.; Rohloff, M.; Rosauer, B.: Beschreibung von Unternehmensarchitekturen: Sichten und Abhängigkeiten am Beispiel der IT-Infrastrukturarchitektur. In: Dittrich, K.; König, W.; Oberweis, A.; Rannenber, K.; Wahlster, W. (Hrsg.), *Informatik 2003*, Bonn: Köllen Druck+Verlag, 2003, Nr. P-35 in GI-Edition *Lecture Notes in Informatics*, S. 426–434
- [LS97] Lindig, C.; Snelting, G.: Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis. In: *Proceedings of the 1997 International Conference on Software Engineering*, ACM Press, 1997, S. 349–359
- [LSA04] Leser, F.; Scheibehenne, R.; Alt, R.: Ansatz zur Bestimmung des Architekturnutzens bei der Deutschen Telekom. In: Alt, R.; Österle, H. (Hrsg.), *Real-time Business. Lösungen, Bausteine und Potenziale des Business Networking*, Berlin et al.: Springer-Verlag, 2004, S. 233–253
- [LT87] Lynch, N.; Tuttle, M.: Hierarchical Correctness Proofs for Distributed Algorithms. In: *Proc. of the 6th Annual ACM Symposium on Principles of Distributed Computing*, 1987, S. 137–151
- [LT89] Lynch, N.; Tuttle, M.: An Introduction to Input/Output automata. In: *CWI quarterly* 2 (1989), Nr. 3, S. 219–246



- [LTP03] Lethbridge, T. C.; Tichelaar, S.; Ploederede, E.: The Dagstuhl Middle Metamodel: A Schema For Reverse Engineering. In: *Proceedings of the International Workshop on Meta-Models and Schemas for Reverse Engineering (ateM 2003)*, Springer-Verlag, 2003, Electronic Notes in Computer Science, S. 7–18
- [LV95] Luckham, D. C.; Vera, J.: An Event-Based Architecture Definition Language. In: *IEEE Transactions on Software Engineering* 21 (1995), Nr. 9, S. 717–734
- [LVB+93] Luckham, D. C.; Vera, J.; Bryan, D.; Augustin, L.; Belz, F.: Partial Orderings of Event Sets and Their Application to Prototyping Concurrent, Timed Systems. In: *Journal of Systems & Software* 21 (1993), Nr. 3, S. 253–265
- [LW97] Lutz, R. R.; Woodhouse, R. M.: Requirements Analysis Using Forward and Backward Search. In: *Annals of Software Engineering* 3 (1997), Nr. 1, S. 459–475
- [Mac05] Macromedia, Inc., USA: Macromedia – Flash MX 2004. 1995–2005,  
URL <http://www.macromedia.com/software/flash/>
- [MAD94] Menasce, D. A.; Almeida, V. A. F.; Dowdy, L. W.: *Capacity Planning and Performance Modeling – from mainframes to client-server systems*. Prentice Hall, 1994
- [MAD04] Menasce, D. A.; Almeida, V. A.; Dowdy, L. W.: *Performance by Design*. Prentice Hall, 2004
- [Mar02] Marinescu, F.: *EJB Design Patterns: Advanced Patterns, Processes, and Idioms*. John Wiley & Sons, 2002
- [Mat03] Mattern, T.: The New Architecture of Integration. In: *SAP Info* online Journal, 2003
- [Mau01] Mauri, G.: *Integrating Safety Analysis Techniques, Supporting Identification of Common Cause Failures*. Dissertation, Department of Computer Science, University of York, 2001
- [MB01] Mertens, P.; Bodendorf, F.: *Programmierte Einführung in die Betriebswirtschaftslehre – Institutionenlehre*. Wiesbaden: Gabler Verlag, 11. Aufl., 2001
- [MBC84] Marsan, M. A.; Balbo, G.; Conte, G.: A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems. In: *ACM Transactions on Computer Systems* (1984)
- [MBP+04] Moll, K.-R.; Broy, M.; Pizka, M.; Seifert, T.; Bergner, K.; Rausch, A.: Erfolgreiches Management von Software-Projekten. In: *Informatik-Spektrum* (2004)
- [MDEK95] Magee, J.; Dulay, N.; Eisenbach, S.; Kramer, J.: Specifying Distributed Software Architectures. In: Schäfer, W.; Botella, P. (Hrsg.), *Proc. 5th European Software Engineering Conf.*

- (ESEC 95), *Sitges, Spain*, Springer-Verlag, 1995, Nr. 989 in Lecture Notes in Computer Science, S. 137–153
- [MDJ02] Mens, T.; Demeyer, S.; Janssens, D.: Formalising Behaviour Preserving Program Transformations. In: *ICGT 2002*, 2002, Nr. 2505 in Lecture Notes in Computer Science
- [Mey90] Meyer, B.: Lessons from the design of the Eiffel libraries. In: *Communications of the ACM* 33 (1990), Nr. 9, S. 68–88
- [Mey97] Meyer, B.: *Object-Oriented Software Construction*. Prentice Hall, 2. Aufl., 1997
- [Mey99] Meyer, B.: The Unity of Software and the Power of Roundtrip Engineering. In: *Proceedings of the Technology of Object-Oriented Languages and Systems*, IEEE Computer Society Press, 1999, S. 2
- [MHR04] Matevska-Meyer, J.; Hasselbring, W.; Reussner, R.: Software Architecture Description supporting Component Deployment and System Runtime Reconfiguration. In: *Proceedings of Workshop on Component-Oriented Programming (WCOP 2004)*, Juni 2004
- [Mica] Microsoft Cooperation: Domain Specific Language (DSL) Tools. URL <http://lab.msdn.microsoft.com/vs2005/teamsystem/workshop/dsltools/>
- [Micb] Microsystems, S.: Java 2 Platform, Enterprise Edition (J2EE). URL <http://java.sun.com/j2ee/>
- [Mic05a] Microsoft Corporation, USA: About the Microsoft Foundation Classes. 2005, URL <http://msdn.microsoft.com/>
- [Mic05b] Microsoft Corporation, USA: Microsoft DirectShow 9.0. 2005, URL <http://msdn.microsoft.com/>
- [Mic05c] Microsoft Corporation, USA: .NET Development. 2005, URL <http://msdn.microsoft.com/>
- [Mil67] Milgram, S.: The Small World Problem. In: *Psychology Today* 61 (1967), S. 60–67
- [Mil80] Milner, R.: *A calculus of communicating systems*. Nr. 92 in Lecture Notes in Computer Science, Springer-Verlag, 1980
- [Mil89] Milner, R.: *Communication and Concurrency*. Prentice Hall, 1989
- [MJS<sup>+</sup>00] Müller, H. A.; Jahnke, J. H.; Smith, D.; Storey, M.; Tilley, S. R.; Wong, K.: Reverse Engineering: A Roadmap. In: Finkelstein, A. (Hrsg.), *The Future of Software Engineering*, ACM Press, 2000, S. 47–60
- [MKG97] Magee, J.; Kramer, J.; Giannakopoulou, D.: Analysing the behaviour of distributed software architectures: a case study. In: *Proceedings of the Sixth IEEE Computer Society Workshop on Distributed Computing Systems*, IEEE Computer Society, 1997
- [MM01] Maletic, J. I.; Marcus, A.: Supporting program comprehension using semantic and structural information. In: *Proceedings of the*

- [MM03] Miller, J.; Mukerji, J.: MDA Guide Version 1.0.1 (omg/2003-06-01). Juni 2003,  
URL <http://www.omg.org/docs/omg/03-06-01.pdf>
- [MMT70] Mesarovic, M. D.; Macko, D.; Takahara, Y.: *Theory of Hierarchical, Multilevel Systems*. New York, London: Academic Press, 1970
- [MNP<sup>+</sup>04] McCoy, D. W.; Natis, Y. V.; Pezzini, M.; Sinur, J.; Schulte, R. W.; Thompson, J.; Lhereux, B. J.; Kenney, F.; Haight, C.; Friedman, B.; Phifer, G.; James, G. A.; Blechar, M. J.; Vecchio, D.: Hype Cycle for Application Integration and Platform Middleware. 2004
- [MNS95] Murphy, G. C.; Notkin, D.; Sullivan, K.: Software reflexion models: bridging the gap between source and high-level models. In: *Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering*, ACM Press, 1995, S. 18–28
- [Mon98] Montenegro, S.: Formale Methoden in der Softwareentwicklung Heute und Morgen. In: *Elektronik Zeitschrift* 21 (1998)
- [MP94] McDermid, J. A.; Pumfrey, D. J.: A Development of Hazard Analysis to Aid Software Design. In: *Compass'94: 9th Annual Conference on Computer Assurance*, Gaithersburg, MD: National Institute of Standards and Technology, 1994, S. 17–26
- [MQ94] Moriconi, M.; Qian, X.: Correctness and Composition of Software Architectures. In: *Proceedings of ACM SIGSOFT'94: Symposium on Foundations of Software Engineering*, 1994, S. 164–174.
- [MQR95] Moriconi, M.; Qian, X.; Riemenschneider, R. A.: A formal approach to correct refinement of software architectures. In: *Proceedings of ACM SIGSOFT'94: Symposium on Foundations of Software Engineering*, 1995, Bd. 21(4), S. 356–372
- [MR97] Moriconi, M.; Riemenschneider, R.: *Introduction to Safl 1.0: A language for specifying software architecture hierarchies*. Technischer Bericht SRI-CSL-97-01, SRI International, 1997
- [MR04] Mortensen, K. H.; Rölke, H.: Petri net tool database. 2004,  
URL <http://www.daimi.au.dk/PetriNets>
- [MRR04] Meister, J.; Reussner, R.; Rohde, M.: Managing Product Line Variability by Patterns. In: Weske, M.; Liggesmeyer, P. (Hrsg.), *Proceedings of 5th Intl. Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World, Net.ObjectDays 2004, Erfurt, Germany*, Springer-Verlag, September 2004, Bd. 3263 von *Lecture Notes in Computer Science*, S. 153–168
- [MSUW04] Mellor, S. J.; Scott, K.; Uhl, A.; Weise, D.: *MDA Distilled*. Addison-Wesley, 2004

- [MT00] Medvidovic, N.; Taylor, R. N.: A Classification and Comparison Framework for Software Architecture Description Languages. In: *IEEE Transactions on Software Engineering* 26 (2000), Nr. 1, S. 70–93
- [MT04] Mens, T.; Tourwé, T.: A Survey of Software Refactoring. In: *IEEE Transactions on Software Engineering* 30 (2004), Nr. 2, S. 126–139
- [MW01] Mens, T.; Wermelinger, M.: *Proceedings of the Workshop on Formal Foundations of Software Evolution*. Technical Report UNL-DI-1-2001, Departamento de Informatica Faculdade de Ciencias e Tecnologia Universidade Nova de Lisboa, 2001
- [MW02] Mens, T.; Wermelinger, M.: Separation of concerns for software evolution. In: *Journal of software maintenance and evolution – research and practice* 14 (2002), Nr. 5, S. 311–315
- [MW04] Melchert, F.; Winter, R.: The Enabling Role of Information Technology for Business Performance Management. In: *Decision Support in an Uncertain and Complex World, Proc. of the 2004 IFIP International Conference on Decision Support Systems (DSS2004)*, 2004, S. 535–546
- [MWT95] Müller, H. A.; Wong, K.; Tilley, T.: Understanding Software Systems Using Reverse Engineering Technology. In: Alagar, V. S.; Missaoui, R. (Hrsg.), *Object-Oriented Technology for Database and Software Systems*, World Scientific, 1995, S. 240–252
- [MZK03] Meyer, M.; Zarnekow, R.; Kolbe, L. M.: IT-Governance – Begriffe, Status quo und Bedeutung. In: *Wirtschaftsinformatik* 45 (2003), Nr. 4, S. 445–448
- [Neu02] Neuhaus, U.: Service Level Agreements als Basis der Qualitätssicherung für einen IT-Betrieb. In: von Knop, J.; Haverkamp, W. (Hrsg.), *Zukunft der Netze*, Gesellschaft für Informatik, 2002, Nr. P-17 in GI-Edition Lecture Notes in Informatics, S. 309–316
- [NHW<sup>+</sup>02] Niemann, H.; Hasselbring, W.; Wendt, T.; Winter, A.; Meierhofer, M.: Kopplungsstrategien für Anwendungssysteme im Krankenhaus. In: *Wirtschaftsinformatik* 44 (2002), Nr. 5, S. 425–434
- [Nie02] Niere, J.: Fuzzy logic based interactive recovery of software design. In: *Proceedings of the 24th International Conference on Software Engineering (ICSE-02)*, ACM Press, Mai 2002, S. 727–728
- [NMM05] *Network-Integrated Multimedia Middleware*. Technischer Bericht, Computer Graphics Lab, Saarland Universität, Saarbrücken, 2005, URL <http://www.networkmultimedia.org/>
- [NNZ00] Nickel, U.; Niere, J.; Zundorf, A.: The FUJABA environment. In: *Proc. of 22nd International Conference on Software Engineering (ICSE-22)*, 2000, S. 742–745

- [NP90] Nosek, J. T.; Palvia, P.: Software maintenance management: changes in the last decade. In: *Journal of Software Maintenance 2* (1990), Nr. 3, S. 157–174
- [NW00] Noble, J.; Weir, C.: *Small Memory Software: Patterns for Systems with Limited Memory*. Addison-Wesley, 2000
- [OAS04] OASIS: Business Transaction Protocol (BTP), Version 1.0.9.1. 2004, URL [http://www.oasis-open.org/committees/document.php?document\\_id=6634](http://www.oasis-open.org/committees/document.php?document_id=6634)
- [ÖB03] Österle, H.; Blessing, D.: Business Engineering Model. In: Österle und Winter [ÖW03b], S. 65–86
- [ÖBH92] Österle, H.; Brenner, W.; Hilbers, K.: *Unternehmensführung und Informationssystem – Der Ansatz des St. Galler Informationssystem-Managements*. Informatik und Unternehmensführung, Stuttgart: Teubner, 1992
- [OMGa] OMG: *Meta Object 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 – Ressource Page*. Online-ressource, Object Management Group, URL <http://www.omg.org/oma/>
- [OMGd] OMG: *XML Metadata Interchange*. Technischer Bericht, Object Management Group, URL <http://www.omg.org/technology/documents/formal/xmi.htm>
- [OMG01] OMG: *General Ledger*. OMG Specification, Version 1.0 01-02-67, Object Management Group, 2001
- [OMG02] OMG: *Request for Proposals: MOF 2.0 Query / Views / Transformations*. Technischer Bericht, Object Management Group, 2002, URL <http://www.omg.org/>
- [OMG03a] OMG: *UML Profile for Schedulability, Performance and Time*. 2003, URL <http://www.omg.org/>
- [OMG03b] OMG: *Unified Modeling Language: Superstructure – version 2.0*. August 2003, URL <http://www.omg.org/uml/>
- [OMG03c] OMG: *The Unified Modeling Language, Version 1.5*. 2003, URL <http://www.omg.org/>

- [OMG04] OMG: *Request for Proposals: MOF Model to Text Transformation Language*. Technischer Bericht, Object Management Group, 2004, URL <http://www.omg.org/docs/ad/04-04-07.pdf>
- [Opd92] Opdyke, W.: *Refactoring Object-Oriented Frameworks*. Technischer Bericht UIUCDCS-R 92-1759, Univ. of Illinois at Urbana-Champaign, Dept. of Computer Science, 1992
- [Oqu04] Oquendo, F.:  $\pi$ -ADL: an Architecture Description Language based on the higher-order typed  $\pi$ -calculus for specifying dynamic and mobile software architectures. In: *Software Engineering Notes 29* (2004), Nr. 3, S. 1–14
- [Ora] Oracle Corporation: OCI. URL <http://otn.oracle.com/tech/oci>
- [Ora04] Oracle Corporation: *interMedia*. Technischer Bericht, 2004, URL <http://www.oracle.com/technology/products/intermedia/index.html>
- [Ora05] Oracle Corporation: *interMedia Documentation*. Technischer Bericht, 2005, URL <http://www.oracle.com/technology/documentation/intermedia.html>
- [OSC] Online Services Computer Interface. URL <http://www.osci.de>
- [Öst95] Österle, H.: *Business Engineering: Prozess- und Systementwicklung – Entwurfstechniken*, Bd. 1. Berlin et al.: Springer-Verlag, 2. Aufl., 1995
- [OT89] Ott, L. M.; Thuss, J. J.: The relationship between slices and module cohesion. In: *Proceedings of the 11th international conference on Software engineering*, ACM Press, 1989, S. 198–204
- [Ous94] Ousterhout, J.: *Tcl and the Tk Toolkit*. Addison-Wesley, 1994
- [ÖW03a] Österle, H.; Winter, R.: Business Engineering. In: *Business Engineering* [ÖW03b], S. 3–20
- [ÖW03b] Österle, H.; Winter, R. (Hrsg.): *Business Engineering – Auf dem Weg zum Unternehmen des Informationszeitalters*. Business Engineering, Springer-Verlag, 2. Aufl., 2003
- [Pad02a] Padberg, J.: Basic Ideas for Transformations of Specification Architectures. In: Heckel, R.; Mens, T.; M., W. (Hrsg.), *Proc. Workshop on Software Evolution through Transformations (SET 02), Satellite Event of ICGT'02, Oktober 2002*, Bd. 72 von *Electronic Notes in Theoretical Computer Science (ENTCS)*
- [Pad02b] Padberg, J.: Petri Net Modules. In: *Journal on Integrated Design and Process Technology* 6 (2002), Nr. 4, S. 121–137
- [PAMA00] Petriu, D.; Amer, H.; Majumdar, S.; Abdull-Fatah, I.: Using Analytic Models for Predicting Middleware Performance. In:

- [Par72] Parnas, D. L.: On the Criteria To Be Used in Decomposing Systems into Modules. In: *Communications of the ACM* 15 (1972), Nr. 12, S. 1053–1058
- [PAS98] Pree, W.; Althammer, E.; Sikora, H.: Framelets als handliche Architekturbausteine. In: *Softwaretechnik* 98, Paderborn, September 1998
- [PB04] Pletschen, W.; Böckmann, F.-J.: Infrastruktur-Management als Erfolgsfaktor. In: Dietrich, L.; Schirra, W. (Hrsg.), *IT im Unternehmen – Leistungssteigerung bei sinkenden Budgets*, Springer-Verlag, Xpert.press, 2004, S. 103–137
- [PBG04] Posch, T.; Birken, K.; Gerdorf, M.: *Basiswissen Softwarearchitektur – Verstehen, entwerfen, bewerten und dokumentieren*. dpunkt.verlag, 2004
- [Pet62] Petri, C. A.: *Kommunikation mit Automaten*. Dissertation, Technische Hochschule Darmstadt, 1962
- [Pet95] Petre, M.: Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. In: *Communications of the ACM* 38 (1995), Nr. 6, S. 33–44
- [PJC00] de Paula, V.; Justo, G.; Cunha, P.: Specifying and verifying reconfigurable software architectures. In: *International Symposium on Software Engineering for Parallel and Distributed Systems*, 2000, S. 21–31
- [PK99] Pree, W.; Koskimies, K.: Framelets – Small is Beautiful. In: Fayad et al. [FSJ99a], S. 411–413
- [PK00] Pree, W.; Koskimies, K.: Framelets – small and loosely coupled frameworks. In: *ACM Computing Surveys* 32 (2000), Nr. 1es, S. 6
- [PLV97] Posnak, E.; Lavender, R.; Vin, H.: An adaptive framework for developing multimedia software components. In: *Communications of the ACM* 40 (1997), Nr. 10, S. 43–47
- [PMSH01] Papadopoulos, Y.; McDermid, J. A.; Sasse, R.; Heiner, G.: Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. In: *Int. J. of Reliability Engineering and System Safety* 71 (2001), Nr. 3, S. 229–247
- [PNW] Petri Nets World.  
URL <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>
- [Poo99] Pooley, R.: Using UML to Derive Stochastic Process Algebra Models. In: *Proceedings of the 25th UK Performance Engineering Workshop*, 1999, S. 23–34

- [Pre94] Pree, W.: Meta Patterns — A Means for Capturing the Essentials of Reusable Object-Oriented Design. In: *Lecture Notes in Computer Science* 821 (1994)
- [Pre95] Pree, W.: *Design Patterns for Object-Oriented Software Development*. ACM Press Books, Addison-Wesley, 1995
- [Pre96a] Pree, W.: *Framework Patterns*. SIGS Books and Multimedia, 1996
- [Pre96b] Pree, W.: Frameworks – Past, present, future. In: *Object magazine – improving software quality through object development & reuse* 6 (1996), Nr. 3
- [Pre97a] Pree, W.: Component-Based Software Development – A New Paradigm in Software Engineering? In: *Software – Concepts and Tools* 18 (1997), Nr. 4, S. 169–174
- [Pre97b] Pree, W.: Essential Framework Design Patterns. In: *Object magazine – improving software quality through object development & reuse* 7 (1997), Nr. 1
- [Pre97c] Pree, W.: *Komponentenbasierte Softwareentwicklung mit Frameworks*. dpunkt.verlag, 1997
- [Pre97d] Pree, W.: Object-Oriented Design Patterns and Hot Spot Cards. In: *IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS97)*, Como, Italy, September 1997
- [Pre99] Pree, W.: Hot-Spot-Driven Framework Development. In: Fayad et al. [FSJ99a], S. 379–393
- [PS02] Petriu, D. C.; Shen, H.: Applying the UML Performance Profile: Graph Grammar-Based Derivation of LQN Models from UML Specifications. In: *TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, London, UK: Springer-Verlag, 2002, S. 159–177
- [Pum99] Pumfrey, D.: *The Principled Design of Computer System Safety Analyses*. Dissertation, Department of Computer Science, University of York, 1999
- [PVL96] Posnak, E.; Vin, H.; Lavender, R.: Presentation Processing Support for Adaptive Multimedia Applications. In: *Proc. of Multimedia Computing and Networking 1996 (MMCN96)*, Januar 1996, S. 234–245
- [PW92] Perry, D. E.; Wolf, A. L.: Foundations for the Study of Software Architecture. In: *Software Engineering Notes* 17 (1992), Nr. 4, S. 40–52
- [PW02] Petriu, D.; Woodside, C.: Software Performance Models from System Scenarios in Use Case Maps. In: *Proceedings of the 12th International Conference for Modelling Tools and Techniques for Computer and Comm. System Performance Evaluation*, 2002, S. 141–158



- [QFD] QFD: Quality Function Deployment. QM-Lexikon.  
URL <http://www.quality.de/lexikon/qfd.htm>
- [Qui94] Quilici, A.: A memory-based approach to recognizing programming places. In: *Communications of the ACM* 37 (1994), Nr. 5, S. 84–93
- [Ran00] Ran, A.: ARES Architectural Framework for Software Architecture. In: Jazayeri et al. [JRL00], S. 1–29
- [RBD+03] Rudd, C.; Bicket, D.; Diamon, P.; Bull, R.; Fröhlich, S.; Best, R.: *Software Asset Management*. ITIL IT Infrastructure Library, London: Office of Government Commerce (OGC), 2003
- [RBJ97] Roberts, D.; Brant, J.; Johnson, R.: A Refactoring Tool for Smalltalk. In: *Theory and Practice of Object Systems* 3 (1997), S. 253–263
- [RBSP02] Riebisch, M.; Böllert, K.; Streitferdt, D.; Phillipow, I.: Extending Feature Diagrams with UML Multiplicities. In: *Proceedings of Integrated Design and Process Technology*, Society for Design and Process Science, June 2002
- [RCM99] Rajala, N.; Campara, D.; Mansurov, M.: inSight – Reverse Engineer CASE Tool. In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE99)*, IEEE Computer Society Press / ACM Press, 1999, S. 630–633
- [RD03] Riva, C.; Del Rosso, C.: Experiences with Software Product Family Evolution. In: *Proceedings of the International Workshop on Principles of Software Evolution*, IEEE Computer Society Press, September 2003
- [RE99] Rosel, A.; Erni, K.: Experiences with the Semantic Graphics Framework. In: Fayad et al. [FSJ99b], Kap. 27, S. 629–657
- [Rea05] RealNetworks: *Helix Community*. Technischer Bericht, 2005, URL <https://helixcommunity.org/>
- [Rei86] Reisig, W.: *Petrinetze*. Springer-Verlag, 2. Aufl., 1986
- [RHMM04] Roshandel, R.; van der Hoek, A.; Mikic-Rakic, M.; Medvidovic, N.: Mae – A System Model and Environment for Managing Architectural Evolution. In: *ACM Transactions on Software Engineering and Methodology* 13 (2004), Nr. 2, S. 240–276
- [RHS05] Richter, J.-P.; Haller, H.; Schrey, P.: Serviceorientierte Architektur — Das aktuelle Schlagwort. In: *Informatik-Spektrum* 28 (2005), Nr. 6
- [Ric03] Richter, C.: *Entwurf und Realisierung eines webbasierten personalisierten multimedia Musik-Newsletters*. Individuelles Projekt, Carl von Ossietzky Universität Oldenburg, Juli 2003
- [Rie96] Riel, A. J.: *Object-Oriented Design Heuristics*. Addison-Wesley, 1996
- [Rie00] Riehle, D.: *Framework Design: A Role Modeling Approach*. Dissertation, Swiss Federal Institute of Technology Zurich, 2000
- [Ris00] Rising, L.: *Pattern Almanac 2000*. Addison-Wesley, 2000

- [RJ97] Roberts, D.; Johnson, R.: Evolving Frameworks – A Pattern Language for Developing Object-Oriented Frameworks. In: *Pattern Languages of Program Design 3*, Illinois, USA: Addison-Wesley, 1997
- [RJB98] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The Unified Modeling Language Reference Manual*. Addison Wesley, Dezember 1998
- [RL04] Rook, S.; Lippert, M.: *Refactorings in großen Softwareprojekten*. dpunkt.verlag, 2004
- [Rog97] Rogers, G. F.: *Framework-Based Software Development in C++*. Prentice Hall Series on Programming Tools and Methodologies, Prentice Hall, 1997
- [RR03] Ravichandran, T.; Rothenberger, M.: Software reuse strategies and component markets. In: *Communications of the ACM* 46 (2003), Nr. 8, S. 109–114
- [RS95] Rolia, J.; Sevcik, K.: The Method of Layers. In: *IEEE Transactions on Software Engineering* 21 (1995), Nr. 8, S. 682–688
- [RS99] Rout, T.; Sherwood, C.: Software Engineering Standards and the Development of Multimedia-Based Systems. In: *In Fourth IEEE International Symposium and Forum on Software Engineering Standards*, Mai 1999
- [RS02] Rumpe, B.; Schröder, A.: Quantitative Survey on Extreme Programming Projects. In: Wells und Williams [WW02], S. 43–46
- [RSP04] Riebisch, M.; Streitferdt, D.; Pashov, I.: Modeling Variability for Object-Oriented Product Lines. In: Buschmann, F.; Buchmann, A. P.; Cilia, M. (Hrsg.), *Object-Oriented Technology. ECOOP 2003 Workshop Reader*, Springer-Verlag, 2004, Bd. 3013 von *Lecture Notes in Computer Science*, S. 165–178
- [Rue02] Rueegg-Stuerm, J.: *Das neue St. Galler Management-Modell: Grundkategorien einer integrierten Managementlehre: der HSG-Ansatz*. Bern: Haupt, 2002
- [Rum96] Rumpe, B.: *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, 1996
- [Rum04] Rumpe, B.: *Modellierung mit UML – Sprache, Konzepte und Methodik*. Xpert.press, Springer-Verlag, 2004
- [Rum05] Rumpe, B.: *Agile Modellierung mit UML – Codegenerierung, Testfälle, Refactoring*. Xpert.press, Springer-Verlag, 2005
- [RWP04] Ross, K.; Westermann, G. U.; Popitsch, N.: METIS – A Flexible Database Solution for the Management of Multimedia Assets. In: *Proc. of the 10th International Workshop on Multimedia Information Systems (MIS 2004)*, August 2004
- [Saw95] Sawhney, M.: *Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystems für eine Universität*.

- Diplomarbeit, Universität Osnabrück, Fachbereich  
Wirtschaftswissenschaften, Osnabrück, Juni 1995
- [SB99] van Solingen, R.; Berghout, E.: *The Goal/Question/Metric Method, A Practical Method for Quality Improvement of Software Development*. McGraw-Hill, 1999
- [SB04] Scherp, A.; Boll, S.: Generic support for personalized mobile multimedia tourist applications. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia – Technische Demonstration*, New York, NY, USA: ACM Press, Oktober 2004, S. 178–179
- [SB05a] Scherp, A.; Boll, S.: A lightweight process model and development methodology for component frameworks. In: *Proceedings of the tenth International Workshop on Component-Oriented Programming*, Juli 2005
- [SB05b] Scherp, A.; Boll, S.: MM4U – A framework for creating personalized multimedia content. In: Srinivasan, U.; Nepal, S. (Hrsg.), *Managing Multimedia Semantics*, Hershey, PA, USA: IRM Press, Kap. 11, 2005
- [SB05c] Scherp, A.; Boll, S.: Paving the Last Mile for Multi-Channel Multimedia Presentation Generation. In: Chen, Y.-P. P. (Hrsg.), *Proceedings of the 11th Multimedia Modeling (MMM) Conference*, Melbourne, Australia: IEEE Computer Society, Januar 2005, S. 190–197
- [Sch97] Scheer, A.-W.: *Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse*. Springer-Verlag, 2. Aufl., 1997
- [Sch99a] Schmid, H. A.: Framework Design by Systematic Generalization. In: Fayad et al. [FSJ99a], Kap. 15, S. 353–378
- [Sch99b] Schwanitz, D.: *Alles was man wissen muss*. Eichborn Verlag, 1999
- [Sch01a] Scholl: Napster Messages. April 2001,  
URL <http://opennap.sourceforge.net/napster.txt>
- [Sch01b] Schollmeier, R.: A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In: *First International Conference on Peer-to-Peer Computing (P2P'01)*, Linköping, Schweden, August 2001, S. 101–102
- [Sch04] Schott, A.: Architekturzentrierte Software-Entwicklung – elitäre Technik-Disziplin oder ökonomische Notwendigkeit? In: Dadam, P.; Reichert, M. (Hrsg.), *INFORMATIK 2004 – Informatik verbindet, Bd. 2*, Gesellschaft für Informatik, 2004, Bd. 51 von *GI-Edition Lecture Notes in Informatics*, S. 116–120
- [SEH03] Sim, S. E.; Easterbrook, S.; Holt, R. C.: Using benchmarking to advance research: a challenge to software engineering. In: *ICSE '03: Proceedings of the 25th International Conference on Software*

- Engineering, Washington, DC, USA: IEEE Computer Society, 2003, S. 74–83
- [SEI05] SEI: *What are the Duties of a Chief Software Architect?* Technischer Bericht, Software Engineering Institute, Carnegie Mellon University, 2005,  
URL <http://www.sei.cmu.edu/architecture/>
- [SES02] Stroulia, E.; El-Ramly, M.; Sorenson, P.: From Legacy to Web through Interaction Modeling. In: *Proceedings of the International Conference on Software Maintenance (ICSM '02)*, Montreal, Kanada: IEEE Press, Oktober 2002, S. 320–329
- [SG95] Shaw, M.; Garlan, D.: Formulations and Formalisms in Software Architecture. In: van Leeuwen, J. (Hrsg.), *Computer Science Today: Recent Trends and Developments*, Springer-Verlag, Bd. 1000 von *Lecture Notes in Computer Science*, 1995, S. 307–323
- [SG96] Shaw, M.; Garlan, D.: *Software Architecture – Perspectives on an Emerging Discipline*. An Alan R. Apt book, Prentice Hall, 1996
- [SGM02] Szyperski, C.; Gruntz, D.; Murer, S.: *Component Software – Beyond Object-Oriented Programming*. Addison-Wesley Component Software Series, Addison-Wesley, 2. Aufl., 2002
- [SH99] Scheer, A.-W.; Hoffmann, M.: From Business Process Model to Application System – Developing an Information System with the House of Business Engineering (HOBE). In: *Advanced Information Systems Engineering, 11th International Conference CAiSE 1999, Heidelberg, Germany*, Springer-Verlag, 1999, Bd. 1626 von *Lecture Notes in Computer Science*, S. 2–9
- [Shi00] Shirky, C.: What Is P2P.. And What Isn't? 2000,  
URL <http://www.openp2p.com/>
- [Sie04] Siedersleben, J.: *Moderne Softwarearchitektur – Umsichtig planen, robust bauen mit Quasar*. dpunkt.verlag, 2004
- [Sih01] Sihling, M.: *Methodische Entwicklung und rollenbasierte Integration von Komponentenframeworks*. Dissertation, Technische Universität München – Institut für Informatik, 2001
- [SM95] Storey, M.-A. D.; Muller, H. A.: Manipulating and documenting software structures using SHriMP views. In: *International Conference on Software Maintenance*, IEEE Computer Society Press, Oktober 1995, S. 275–284
- [Smi90] Smith, C. U.: *Performance Engineering of Software Systems*. Addison-Wesley, 1990
- [Smi02] Smith, C. U.: *Performance Solutions: A Practical Guide To Creating Responsive, Scalable Software*. Addison-Wesley, 2002
- [SMK<sup>+</sup>01] Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. F.; Balakrishnan, H.: Chord – A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proceedings of the 2001 Conference on*

- Applications, Technologies, Architectures, and Protocols for Computer Communications*, ACM Press, 2001, S. 149–160
- [SMM02] Shokoufandeh, A.; Mancoridis, S.; Maycock, M.: Applying Spectral Methods to Software Clustering. In: van Deursen [Deu02], S. 3–12
- [SN95] Steinmetz, R.; Nahrstedt, K.: *Multimedia: Computing, Communications and Applications*. Prentice Hall, 1995
- [Sof04] Bericht: Software-Projekt für Finanzämter gescheitert. Juli 2004, URL <http://www.heise.de/newsticker/meldung/48843>
- [Som04] Sommerville, I.: *Software Engineering*. Addison-Wesley, 7. Aufl., 2004
- [SPL03] Seacord, R. C.; Plakosh, D.; Lewis, G. A.: *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Addison-Wesley, 2003
- [SS04] Streckenbach, M.; Snelting, G.: Refactoring Class Hierarchies with KABA. In: *Proceedings of OOPSLA 04*, 2004
- [SS05a] Schelp, J.; Schwinn, A.: Extending the Business Engineering Framework for Application Integration Purposes. In: *Applied Computing – Proceedings of the 2005 ACM Symposium on Applied Computing*, New York, NY, USA: ACM Press, 2005, Bd. 2, S. 1333–1337
- [SS05b] Schwinn, A.; Schelp, J.: Design Patterns for data integration. In: *Journal of Enterprise Information Management* 18 (2005), Nr. 4, S. 471–482
- [SSL01] Simon, F.; Steinbücker, F.; Lewerentz, C.: Metrics Based Refactoring. In: *Proceedings of European Conference Software Maintenance and Reengineering*, 2001, S. 157–169
- [SSN02] Sharma, R.; Stearns, B.; Ng, T.: *J2EE Connector Architecture and Enterprise Application Integration*. Addison-Wesley, 2002
- [SSRB00] Schmidt, D.; Stal, M.; Rohnert, H.; Buschmann, F.: *Patterns for Concurrent and Networked Objects (Pattern-oriented Software Architecture, vol. 2)*. Wiley series in software design patterns, John Wiley & Sons, 2000
- [Sta73] Stachowiak, H.: *Allgemeine Modelltheorie*. Wien: Springer-Verlag, 1973
- [Sta03] Starke, G.: Architektur und Flexibilität – Ein Widerspruch? In: *IT FOKUS* o. A. (2003), Nr. 3, S. 22–26
- [Ste00] Steinmetz, R.: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, 3. Aufl., 2000
- [Sun] Sun Java Center J2EE Patterns.  
URL <http://java.sun.com/blueprints/patterns/>
- [Sun04] Sun Microsystems: EJB 2.1 Specification Final Release 2.1. 2004, URL <http://java.sun.com/products/ejb/index.jsp>

- [Sun05a] Sun Microsystems: *Java Foundation Classes (JFC/Swing)*. Technischer Bericht, 1994–2005,  
URL <http://java.sun.com/products/jfc/>
- [Sun05b] Sun Microsystems: *Java Media Framework API (JMF)*. Technischer Bericht, 1994–2005,  
URL <http://java.sun.com/products/java-media/jmf/>
- [Sun05c] Sun Microsystems: *JMF 2.0 Documentation Downloads*. Technischer Bericht, 1994–2005,  
URL <http://java.sun.com/products/java-media/jmf/>
- [Sun05d] Sun Microsystems: *JXTA v2.3.x: Java Programmer's Guide*. Januar 2005
- [SW04] Steinmetz, R.; Wehrle, K.: Peer-to-Peer-Networking und -Computing. In: *Informatik-Spektrum* 27 (2004), Nr. 1, S. 51–54
- [SWSS03] Schuler, C.; Weber, R.; Schuldt, H.; Schek, H.-J.: Peer-to-Peer Process Execution with Osiris. In: Orłowska, M. E.; Weerawarana, S.; Papazoglou, M. P.; Yang, J. (Hrsg.), *Service-Oriented Computing (ICSOC 2003)*, Springer-Verlag, 2003, Bd. 2910 von *Lecture Notes in Computer Science*, S. 483–498
- [SWZ99] Schürr, A.; Winter, A.; Zündorf, A.: The PROGRES-Approach: Language and Environment. In: Ehrig, H.; Engels, G.; Kreowski, J.-J.; Rozenberg, G. (Hrsg.), *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages and Tools*, World Scientific, 1999
- [Tab00] Tabeling, P.: *Der Modellhierarchieansatz zur Beschreibung nebenläufiger, verteilter und transaktionsverarbeitender Systeme*. Berichte aus der Informatik, Shaker Verlag, 2000
- [Tab01] Tabeling, P.: Wissensorientierte Beschreibung großer Software-Systeme – ein Ansatz jenseits softwareorientierter Konzepte. In: *Knowtech Konferenzband*, Dresden: Knowtech Conference, 2001
- [Tab02] Tabeling, P.: Ein Metamodell zur architekturorientierten Beschreibung komplexer Systeme. In: *Proceedings of Modellierung 2002, GI-Lecture Notes in Informatics, Tutzing*, 2002
- [Tab04a] Tabeling, P.: Architectural Description with Integrated Data Consistency Models. In: *Proceedings of the IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2004
- [Tab04b] Tabeling, P.: Eine Entwicklungsplattform für die architekturorientierte Programmierung. In: *Lecture Notes in Informatics (LNI) – Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen*, 2004
- [Tab05] Tabeling, P.: *Software-Systeme und ihre Modellierung – Grundlagen, Methoden und Techniken*. Springer-Verlag, 2005

- [Tau03] Taubner, D.: Effizientes Software-Engineering: Vorgehen für wirtschaftliche Projekte. In: *IM – Die Fachzeitschrift für Information Management & Consulting* (2003), Nr. 18, S. 14–18
- [TB01] Tokuda, L.; Batory, D.: Evolving Object-Oriented Designs with Refactorings. In: *Journal of Automated Software Engineering* 8 (2001), S. 89–120
- [TDDN00] Tichelaar, S.; Ducasse, S.; Demeyer, S.; Nierstrasz, O.: A Meta-model for Language-Independent Refactoring. In: *Proceedings ISPSE, 2000*, IEEE Computer Society Press
- [TDDN01] Tichelaar, S.; Ducasse, S.; Demeyer, S.; Nierstrasz, O.: Refactoring UML models. In: *Proceedings of UML 01*, Springer-Verlag, 2001, Nr. 2185 in Lecture Notes in Computer Science
- [TG03] Tabeling, P.; Gröne, B.: Mappings between Object-oriented Technology and Architecture-based models. In: Al-Ani, B.; Arabnia, H. R.; Mun, Y. (Hrsg.), *Proceedings of the SERP'03, the international conference on software engineering research and practice, Las Vegas*, CSREA Press, Juni 2003, Bd. II, S. 568–574
- [TH00] Tzerpos, V.; Holt, R. C.: On the Stability of Software Clustering Algorithms. In: *International Workshop on Program Comprehension*, IEEE Computer Society Press, 2000
- [TJK<sup>+</sup>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<sup>+</sup>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:

- [Tro05] URL <http://www.borland.com/together/>  
Trolltech: *Trolltech – Qt Product Overview*. Technischer Bericht, 2005,
- [TX00] URL <http://www.trolltech.com/products/qt/>  
Tsai, J. J. P.; Xu, K.: A comparative study of formal verification techniques for software architecture specifications. In: *Annals of Software Engineering* 10 (2000), Nr. 1–4, S. 207–223
- [van04] *The Cuypers Multimedia Transformation Engine*. Technischer Bericht, CWI, 2004,  
URL <http://media.cwi.nl:8080/demo/>
- [VDI03] VDI: VDI-Richtlinie 3633. 2003,  
URL <http://www.vdi.de/vdi/vrp/richtliniendetails/index.php?ID=9509528>
- [VGRH96] Vesely, W. E.; Goldberg, F. F.; Roberts, N. H.; Haasl, D. F.: *Fault Tree Handbook*. U. S. Nuclear Regulatory Commission. 1996
- [vHW03] van der Aalst, W. M.; Hofstede, A. H. t.; Weske, M.: *Business Process Management: A Survey*. In: van der Aalst, W.; ter Hofstede, A.; Weske, M. (Hrsg.), *Business Process Management: International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, Springer-Verlag, 2003, Bd. 2678, S. 1–12
- [Vit03] Vitharana, P.: Risks and challenges of component-based software development. In: *Communications of the ACM* 46 (2003), Nr. 8, S. 67–72
- [VSW02] Völter, M.; Schmid, A.; Wolff, E.: *Server Component Patterns: Component Infrastructures illustrated with EJB*. John Wiley & Sons, 2002
- [W3C02] W3C: *Web Services Architecture*. 2002,  
URL <http://www.w3.org/>
- [W3C04] Scalable Vector Graphics (SVG) 1.2. 2004  
URL <http://www.w3.org/>
- [W3C05] Synchronized Multimedia Integration Language (SMIL 2.0), 2005,  
URL <http://www.w3.org/>
- [WAGL99] Wedekind, H.; Albrecht, J.; Günzel, H.; Lehner, W.: *Repositories for Data Warehouse Systems in a Middleware Environment*. In: *Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis, ISAS'99, Orlando (FL), 1999*, S. 298–305
- [WBF97] Wiggerts, T.; Bosma, H.; Fielt, E.: *Scenarios for the Identification of Objects in Legacy Systems*. In: Baxter, I. D.; Quilici, A.; Verhoef, C. (Hrsg.), *Proceedings of the 4th Working Conference on Reverse Engineering (WCRE '97)*, Amsterdam, Niederlande: IEEE Computer Society Press, Oktober 1997, S. 24–32



- [WBS04] Wurz, M.; Brettlecker, G.; Schuldt, H.: Data Stream Management and Digital Library Processes on Top of a Hyperdatabase and Grid Infrastructure. In: Agosti, M.; Schek, H.-J.; Türker, C. (Hrsg.), *DELOS Workshop: Digital Library Architectures*, 2004, S. 37–48
- [WC01] Westfechtel, B.; Conradi, R.: Software Architecture and Software Configuration Management. In: *SCM*, Springer-Verlag, 2001, Bd. 2649, S. 24–39
- [WE02] Weber, M.; Eisenführ, F.: *Rationales Entscheiden*. Springer-Verlag, 2002
- [Wei84] Weiser, M.: Program Slicing. In: *IEEE Transactions on Software Engineering* 10 (1984), Nr. 4, S. 352–357
- [Wen80] Wendt, S.: Modified Petri Nets as Flowcharts for Recursive Programs. In: *Software – Practice and Experience* 10 (1980), S. 935–942
- [Wen82a] Wendt, S.: Der Kommunikationsansatz in der Software-Technik. In: *Data Report* 17 (1982), Nr. 4
- [Wen82b] Wendt, S.: Einführung in die Begriffswelt allgemeiner Netzsysteme. In: *Regelungstechnik* 30 (1982), Nr. 1
- [Wen91] Wendt, S.: *Nichtphysikalische Grundlagen der Informationstechnik – Interpretierte Formalismen*. Springer-Verlag, 2. Aufl., 1991
- [WF98] Wijegunaratne, I.; Fernandez, G.: *Distributed Applications Engineering – Building new applications and managing legacy applications with distributed technologies*. Practitioner series, Springer-Verlag, 1998
- [WF99] Wermelinger, M.; Fiadeiro, J. L.: Algebraic Software Architecture Reconfiguration. In: *Proceedings ESEC/FSE'99*, Springer-Verlag, 1999, Bd. 1687 von *Lecture Notes in Computer Science*, S. 393–409
- [WF00] Wermelinger, M.; Fiadeiro, J. L.: Graph Transformation Approach to Software Architecture Reconfiguration. In: *Proc. of the Joint APPLIGRAPH and GETGRATS Workshop on Graph Transformation Systems*, 2000, S. 134–141
- [WF04] Wang, G.; Fung, C. K.: Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems. In: *Proc. 37th Hawaii International Conference on Systems Sciences (HICSS 2004)*, IEEE Computer Society Press, 2004, S. 90272a
- [WHH94] Weide, B. W.; Heym, W. D.; Hollingsworth, J. E.: *Reverse Engineering of Legacy Code is Intractable*. Technischer Bericht OSU-CISRC-10/94-TR55, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio, Oktober 1994
- [Wil96] Wills, L. M.: Using Attributed Flow Graph Parsing to Recognize Clichés in Programs. In: Cuny, J. E.; Ehrig, H.; Engels, G.;

- Rozenberg, G. (Hrsg.), *Proc. 5th Int. Workshop on Graph Grammars and their Application to Computer Science*, Springer-Verlag, 1996, Bd. 1073 von *Lecture Notes in Computer Science*, S. 170–184
- [Win03a] Winter, R.: An Architecture Model for Supporting Application Integration Decisions. In: *Proc. 11th European Conference on Information Systems*, 2003
- [Win03b] Winter, R.: Modelle, Techniken und Werkzeuge im Business Engineering. In: Österle und Winter [ÖW03b], S. 87–118
- [WJ90] Wirfs-Brock, R. J.; Johnson, R. E.: Surveying current research in object-oriented design. In: *Communications of the ACM* 33 (1990), Nr. 9, S. 104–124
- [WL99] Weiss, D. M.; Lai, C. T. R.: *Software Product-Line Engineering – A Family-Based Software Development Process*. Addison-Wesley, 1999
- [WLF01] Wermelinger, M.; Lopes, A.; Fiadeiro, J. L.: A graph based architectural (Re)configuration language. In: *Software Engineering Notes* 26 (2001), Nr. 5, S. 21–32
- [WM81] Wedekind, H.; Müller, T.: Stücklistenorganisation bei einer großen Variantenzahl. In: *Angewandte Informatik* 23 (1981), Nr. 9, S. 377–383
- [WMSR00] Walker, R. J.; Murphy, G. C.; Steinbok, J.; Robillard, M. P.: Efficient mapping of software system traces to architectural views. In: *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, IBM Press, 2000, S. 12
- [Woo03] Woods, D.: *Enterprise Services Architecture*. O'Reilly & Associates, 2003
- [WS98a] Watts, D. J.; Strogatz, S. H.: Collective dynamics of small-world networks. In: *Nature* 393 (1998), Nr. 6684, S. 440–442
- [WS98b] Williams, L. G.; Smith, C. U.: Performance evaluation of software architectures. In: *WOSP '98: Proceedings of the first international workshop on Software and performance*, New York, NY, USA: ACM Press, 1998, S. 164–177
- [WW02] Wells, D.; Williams, L. A. (Hrsg.): *Extreme Programming and Agile Methods*, Nr. 2418 in *Lecture Notes in Computer Science*, Springer-Verlag, 2002
- [WY96] Woods, S.; Yang, Q.: The program understanding problem: Analysis and a heuristic approach. In: *Proceedings of the 18th International Conference on Software Engineering*, IEEE Computer Society Press, 1996, S. 6–15
- [YC79] Yourdon, E.; Constantine, L. L.: *Structured Design – Fundamentals of a Discipline of Computer Program and System Design*. Prentice Hall, 1979

- [Z] The Z notation.  
URL <http://vl.zuser.org/>
- [Zac87] Zachman, J. A.: A Framework for Information Systems Architecture. In: *IBM Systems Journal* 26 (1987), Nr. 3, S. 276–292
- [ZB03] Zarnekow, R.; Brenner, W.: Auf dem Weg zu einem produkt- und dienstleistungsorientierten IT-Management. In: *HMD – Praxis der Wirtschaftsinformatik* 40 (2003), Nr. 232, S. 7–16
- [ZG04] Zhang, X.; Gupta, R.: Cost effective dynamic program slicing. In: *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation*, ACM Press, 2004, S. 94–106
- [ZLX04] Zhou, Y.; Lu, J.; Xu, H. L. B.: A comparative study of graph theory-based class cohesion measures. In: *Software Engineering Notes* 29 (2004), Nr. 2, S. 13
- [Zül05] Züllighoven, H. (Hrsg.): *Object-Oriented Construction Handbook – Developing Application-Oriented Software with the Tools and Materials Approach*. Morgan Kaufman Publ./dpunkt.verlag, 2005
- [ZWDZ04] Zimmermann, T.; Weisgerber, P.; Diehl, S.; Zeller, A.: Mining Version Histories to Guide Software Changes. In: *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society Press, 2004, S. 563–572
- [ZXZY02] Zhou, Y.; Xu, B.; Zhao, J.; Yang, H.: ICBMC: an improved cohesion measure for classes. In: *Proceedings International Conference on Software Maintenance*, IEEE Computer Society Press, 2002, S. 44–53
- [Zyl02] van Zyl, J.: A perspective on service based architecture: the evolutionary concept that assists technology providers in dealing with a changing environment. In: *SAICSIT '02: Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2002, S. 249
- [ZYXX02] Zhao, J.; Yang, H.; Xiang, L.; Xu, B.: Change impact analysis to support architectural evolution. In: *Journal of software maintenance and evolution – research and practice* 14 (2002), Nr. 5, S. 317–333