

Vorwort

Software-Architektur ist ein aktuelles und wichtiges Gebiet sowohl in der Industrie als auch in der Forschung. Neue Technologien, Software-Plattformen oder *Open-Source*-Software bewirken zunehmende Komplexität der zu entwickelnden Software-Systeme, sodass eine wohl durchdachte Architektur immer bedeutender wird. Es gibt wenige – hauptsächlich englischsprachige – Bücher zum Thema Software-Architektur, die diese Fragen in klarer und komprimierter Weise behandeln, aber unseres Wissens nach kein einziges, das sich auf Software-Architekturen für verteilte Systeme spezialisiert hat. Diese Lücke zu füllen, und Wissen aus der Forschung zum Anwender zu transportieren, stellt die Motivation dieses Buch dar.

Aber wie erhält man die beste Architektur für ein konkretes Software-Projekt? Gibt es überhaupt „die beste“ Architektur für ein Problem? Welche Bestandteile haben Software-Architekturen und was sind die *wirklich* wichtigen Dinge, die man als Software-Architekt bedenken muss? Ein erster wesentlicher Aspekt, den wir in diesem Buch behandeln, ist, welche Fragen im Zusammenhang mit Software-Architekturen wichtig sind.

Das Ziel der meisten Software-Projekte ist es, moderne und tragfähige Software zu entwickeln. Idealerweise sollte man mittels Komposition und Aggregation von Bausteinen in der Lage sein, sowohl Strukturen als auch Prozesse in Software zu gießen, und der Software-Architekt muss viele Entscheidungen bezüglich der Beschaffenheit zukünftiger Software treffen. Der zweite wesentliche Aspekt, den wir behandeln, ist, Merkmale und Konsequenzen der Technologien und Bausteine kritisch zu beleuchten, um Software-Architekten ein besseres Fundament für die zu treffenden Entscheidungen zu bieten.

Des Weiteren sind wir der Ansicht, dass Fragen nach benötigter Kommunikation und Koordination sich in Entscheidungen bezüglich der verwendeten Bausteine einer Software-Architektur manifestieren. Bausteine, die uns besonders wichtig erscheinen, wie Peer-to-Peer, Web-Services, Workflow, Push-Systeme, Event-basierte Systeme, Payment und andere, sind in den bisherigen Büchern viel zu kurz gekommen und werden von uns in der notwendigen Tiefe behandelt, was wiederum zur Folge hat, dass wir notwendigerweise einige Details weglassen mussten – die Experten in den jeweiligen Bereichen mögen uns verzeihen. Als Ausgleich gibt es zu jedem einzelnen Teilbereich vertiefendes Material, auf das wir verweisen. Unser Ziel ist es, die Zusammenhänge und Merkmale von Bausteinen und Technologien für moderne und zukunftsfähige Software in *einem* Buch zu behandeln und miteinander in Beziehung zu setzen, sodass Sie als Software-Architekt in einem kompakten Buch die wichtigsten Aspekte vorfinden.

Eine realistische Fallstudie zu Beginn des Buches erläutert den Bezugsrahmen für Software-Architekturen. Weitere grundlagenorientierte Kapitel vertiefen die nötigen Kenntnisse, um moderne Software-Architekturen zu entwerfen. Schließlich analysieren die „technisch“orientierten Kapitel die möglichen Umsetzungen des zuvor Behandelten.

Abschließend möchten wir uns noch bei einigen Menschen bedanken: Hermann Engesser (Springer-Verlag) für seinen Enthusiasmus in Bezug auf dieses Buchprojekt sowie seine organisatorische Unterstützung. Frau Beate Pannewig (Springer-Verlag) sei für ihre organisatorische Hilfe in der Planung und Umsetzung des Buches herzlich gedankt. Des Weiteren möchten wir uns bei unseren Kollegen für die vielen interessanten Diskussionen (in alphabetischer Reihenfolge) bedanken: Karl Aberer, Pascal Fenkam, Michael Fischer, Mehdi Jazayeri, Clemens Kerer, Engin Kirda, Roman Kurmanowysch, Joe Oberleitner, Martin Pinzger, Roman Schmidt und Gerald Reif.

Einen wesentlichen Beitrag zum musikalischen Wohlbefinden der Autoren während der Stunden des Schreibens haben die vielen Musiksender im Internet geleistet.

Unser besonderer Dank gebührt Anja, Birgit und Sabrina für ihre Geduld und ihre moralische Unterstützung, die wir für die (oftmals späten) Arbeitsstunden dringend benötigten, sowie ihre Toleranz einen großen Teil unserer Zeit diesem Buch widmen zu können.

Wien und Lausanne, Mai 2003

*Schahram Dustdar
Harald Gall
Manfred Hauswirth*

Inhalt

1 Was ist Software-Architektur?	1
1.1 Software-Architektur als Abstraktion	2
1.2 Software-Architektur als Bauplan.....	3
1.3 Software-Architektur-Terminologie	5
1.4 Was ist Software-Architektur?.....	7
1.5 Was ist eine „gute“ Software-Architektur?.....	8
1.6 Software-Architektur versus System-Architektur	10
Literatur	11
2 MobiTeam: Eine Design-Review-Fallstudie	13
2.1 Fallstudie: Design Review von Mobiltelefonie-Software	14
2.1.1 Der Design-Review-Prozess.....	17
2.1.2 Softwaretechnische Unterstützung verteilter Produktentwicklung.....	19
2.2 Architekturelle Konzepte und Abstraktionen.....	23
2.2.1 Ein erster Blick auf die Architektur von MobiTeam	25
2.2.2 Die Software-Architektur von MobiTeam.....	30
2.3 Abschließende Bemerkungen	36
Literatur	36
3 Architektureller Entwurf	39
3.1 Modellgeleitete Entwicklung	39
3.2 Was ist Software-Architektur?.....	43
3.2.1 Das Ziel eines architekturellen Entwurfs.....	45
3.2.2 Der Entwurf einer Produkt-Familie	46
3.2.3 Die Struktur komplexer Software-Systeme	46
3.3 Sichten auf ein Software-System	48
3.3.1 Konzeptuelle oder logische Sicht	49
3.3.2 Modul-Sicht.....	51
3.3.3 Prozess-Sicht	52
3.3.4 Physische Sicht.....	52
3.3.5 Szenarien	53
3.3.6 Anwendung des „4+1“-Sichten-Modells	53
3.4 Beschreibungen einer Software-Architektur	54
3.4.1 Die Elemente einer Software-Architektur	54
3.4.2 Architekturelle Beschreibungen	56

3.5 Acme – eine Architektur-Beschreibungssprache	57
3.5.1 Architekturelle Struktur.....	57
3.5.2 Eigenschaften von Architektur-Elementen.....	59
3.5.3 Design-Einschränkungen.....	60
3.5.4 Programm-Familien.....	62
3.5.5 Das Acme-Architektur-Design-Werkzeug	63
3.5.6 Resümee	64
3.6 Die Dokumentation einer Software-Architektur	64
Literatur	66
4 Architekturelle Prinzipien, Bausteine und Muster.....	69
4.1 Qualitätsattribute einer Software-Architektur.....	69
4.2 Architekturelle Stile	74
4.2.1 Datenzentrierte Software-Architekturen.....	76
4.2.2 Datenfluss-Architekturen	77
4.2.3 Virtuelle Maschinen-Architekturen.....	78
4.2.4 Call-and-Return-Architekturen.....	79
4.2.5 Unabhängige Komponenten-Architekturen.....	80
4.2.6 Heterogene architekturelle Stile	82
4.2.7 Architekturelle Stile in ihrer Anwendung	82
4.3 Architekturelle Stile und Qualitätsattribute	85
4.4 Architekturelle Stile und Muster.....	89
4.4.1 Proxy	91
4.4.2 Broker.....	94
4.4.3 Master-Slave.....	97
4.4.4 Client-Dispatcher-Server.....	100
4.4.5 Publisher-Subscriber	103
4.4.6 Model-View-Controller (MVC)	105
4.5 Abschließende Bemerkungen	109
Literatur	110
5 Web-Services.....	113
5.1 Einführung	113
5.2 Simple Object Access Protocol (SOAP).....	117
5.2.1 Aufbau einer SOAP-Nachricht.....	118
5.3 Web Service Description Language (WSDL).....	120
5.3.1 Struktur von WSDL-Dokumenten.....	121
5.3.2 Nachrichtenmuster.....	122
5.4 Universal Description, Discovery and Integration.....	125
5.4.1 Business-Registry	127
5.4.2 Business-Registrations	127
5.4.3 Service-Type-Registration.....	128
5.4.4 Geschäftsinformation – businessEntity	128
5.4.5 Service-Information – businessService	129
5.4.6 Binding-Information – bindingTemplate.....	129
5.4.7 Das tModel.....	130
5.4.8 Das Zusammenspiel von UDDI und SOAP.....	131

5.5 Komposition von Web-Services	135
5.5.1 Interaktionsmuster elementarer Web-Services	136
5.5.2 Interaktionsmuster zusammengesetzter Web-Services	138
5.6 Abschließende Bemerkungen	141
Literatur	141
6 Workflows und Koordination.....	143
6.1 Einführung	143
6.2 Begriffe	144
6.3 Architektur-Sichtweisen von Workflow-Systemen.....	145
6.4 Systembeispiele.....	147
6.4.1 Groove	147
6.4.2 Caramba.....	150
6.5 Web-Services-Workflows.....	152
6.5.1 Business Process Execution Language for Web-Services	154
6.6 Abschließende Bemerkungen	159
Literatur	159
7 Peer-to-Peer-Architekturen	161
7.1 Einführung	161
7.2 Einordnung des Peer-to-Peer-Ansatzes	163
7.2.1 Peer-to-Peer vs. Event-basiert	164
7.2.2 Peer-to-Peer vs. Push.....	164
7.2.3 Peer-to-Peer vs. Mobile Agents.....	165
7.2.4 Peer-to-Peer vs. Distributed Databases.....	166
7.2.5 Anwendungsbereiche	167
7.3 Das Peer-to-Peer-Modell.....	168
7.3.1 Charakteristische architekturelle Parameter	169
7.3.2 Weitere Kriterien	171
7.4 Beispiele für Peer-to-Peer-Architekturen.....	171
7.4.1 Napster.....	171
7.4.2 Gnutella	172
7.4.3 FastTrack	177
7.4.4 Freenet	177
7.4.5 P-Grid	183
7.4.6 Weitere Systeme	190
7.5 Anwendungsdomänen	193
7.6 Abschließende Bemerkungen	195
Literatur	195
8 Komponentenmodelle.....	199
8.1 Einführung	199
8.2 Komponententechnologie	199
8.2.1 Komponenten und Objekte	200
8.2.2 Verteilte Objektmodelle	200

8.3 Business-to-Business-Architekturen und Komponentenmodelle.....	204
8.4 Architektur von J2EE und .NET	205
Literatur	209
9 Security.....	211
9.1 Sicherheit im Software-Lebenszyklus	211
9.2 Kryptographische Grundlagen	212
9.2.1 Hash-Funktionen	212
9.2.2 Symmetrische und asymmetrische Verschlüsselung	213
9.2.3 Zertifikate und elektronische Unterschrift.....	214
9.2.4 Public Key Infrastructure	216
9.3 Sicherheitsaspekte.....	216
9.4 Architekturelle Aspekte	219
9.4.1 Abhörsichere, authentifizierte Datenübertragung.....	220
9.4.2 Secure Socket Layer.....	222
9.4.3 Firewalls	224
9.5 Abschließende Bemerkungen	226
Literatur	227
10 Push- und Event-Architekturen	229
10.1 Push-Systeme.....	229
10.1.1 Komponentenmodell für Push-Systeme	230
10.1.2 Ein Anwendungsszenario für Push-Systeme	237
10.1.3 Mögliche Anwendungsfälle.....	239
10.1.4 Abschließende Bemerkungen	240
10.2 Event-basierte Systeme	240
10.2.1 Grundlagen Event-basierter Systeme	241
10.2.2 Vergleich Event-basierter Systeme mit Push-Systemen.....	242
10.3 Abschließende Bemerkungen	243
Literatur	243
11 Electronic Payment	245
11.1 E-Payment-Szenario	246
11.2 Klassifikationsparameter.....	247
11.3 Zahlungsinstrumente	248
11.3.1 Kreditkarten.....	248
11.3.2 Elektronisches Geld.....	250
11.3.3 Elektronische Schecks	251
11.4 Sicherheitsanforderungen	252
11.5 Beispiel-Architekturen.....	253
11.5.1 Secure Electronic Transaction	254
11.5.2 Millicent	255
11.6 Rechtliche Aspekte	257
Literatur	258
Sachverzeichnis.....	259