February 2007

Eine Strukturvorlage zur effektiven Dokumentation von Software- und IT Architekturen

Gernot Starke
gs@gernotstarke.de

Peter Hruschka
Atlantic Systems Guild, hruschka@b-agile.de

In: Oberweis, Andreas, u.a. (Hg.) 2007. *eOrganisation: Service-, Prozess-, Market-Engineering*; 8. Internationale Tagung Wirtschaftsinformatik 2007. Karlsruhe: Universitätsverlag Karlsruhe

ISBN: 978-3-86644-094-4 (Band 1)
ISBN: 978-3-86644-095-1 (Band 2)
ISBN: 978-3-86644-093-7 (set)

Eine Strukturvorlage zur effektiven Dokumentation von Software- und IT Architekturen

Dr. Gernot Starke

D-50858 Köln
gs@gernotstarke.de

Dr. Peter Hruschka

Atlantic Systems Guild
D-52080 Aachen
hruschka@b-agile.de

Abstract

IT-Entwicklungsprojekte verwenden heute immer noch ungebührlich viel Zeit zur Entwicklung projektspezifischer Strukturen für die Dokumentation von Software- und IT-Architekturen. Durch die Verwendung von *Strukturvorlagen* lässt sich einerseits dieser Aufwand erheblich reduzieren, andererseits die Qualität von Architekturdokumentation deutlich steigern. Der Beitrag stellt die arc42-Schablone zur Architekturdokumentation vor, die sich in vielen kommerziellen, industriellen und Open-Source Projekten bewährt hat.

1 IT-Praxis benötigt Strukturvorlagen

In vielen Projekten, in denen wir als Berater involviert werden, verwenden Mitarbeiter ungebührlich viel Zeit für die Entwicklung von Strukturen und Prozessen für die Dokumentation der jeweiligen Software-Architektur. Dabei besitzen gerade die Strukturen von Architekturdokumentation erhebliches Wiederverwendungspotenzial über die Grenzen einzelner Projekte hinweg: Theorie und Praxis sind sich über viele notwendige Merkmale und Bestandteile von Architekturdokumentation einig (siehe dazu insbesondere [Clements+02] sowie [Starke02]. Unter anderem gehören dazu:

- Verwendung verschiedener Sichten (etwa: Kontextsicht, statische & dynamische Sicht, Verteilungs-/Deploymentsicht), um die unterschiedlichen Strukturebenen von IT-Systemen adäquat repräsentieren zu können.
- Dokumentation wesentlicher Entwurfsentscheidungen, beispielsweise zu Ablaufsteuerung, Persistenz, Benutzungsoberfläche, Fehler-/Ausnahmebehandlung, Protokollierung, Tracing, Monitoring, Systemverteilung/Middleware und anderen.
- Dokumentation wesentlicher Architektur- und Entwurfsziele, um Architekturdokumentation autark von anderen Dokumenten verständlich zu halten (Kruchten hat in [Kru95] hierfür eigens die Use-Case-View propagiert).
- Weitmögliche Verwendung standardisierter Modellierungssprachen: Zur strukturellen Beschreibung von Softwaresystemen hat sich beispielsweise die vielen Unternehmen die UML (Unified Modeling Language) etabliert.

Weiterhin erhöhen bekannte Strukturen *per se* die Verständlichkeit und Wiederverwendbarkeit der darin dokumentierten Inhalte – was sich für Unternehmen als zusätzlicher Effizienzgewinn darstellt.
Aus diesem Grund sollten IT-Entwicklungsprojekte in hohem Maße vorhandene (und erprobte!) Strukturen von Architekturdokumenten (hier genannt Strukturvorlagen) wiederverwenden, anstatt solche Strukturen projektspezifisch neu zu entwerfen.

2 Eine Strukturvorlage für Software- und IT-Architekturen

Ein nach unserer Erfahrung geeignetes Vorbild einer solchen Strukturvorlage für Software- und IT-Architekturen ist das frei verfügbare arc42-Template. Es ist seit 2004 frei verfügbar ([arc42]) und steht unter der sehr flexiblen Creative-Commons Lizenz, die dieses Template für beliebigen Gebrauch freigibt. Seine Struktur resultiert aus der Erfahrung vieler Entwicklungsprojekte in unterschiedlichen Branchen. Das arc42-Template ist wahlweise in MS-Word Format, pdf oder als Mindmap erhältlich. Abbildung 1 zeigt die oberste Gliederungsebene als Mindmap, die für Textdokumente (je nach Umfang) durchaus der Kapitel- oder Dateistruktur entsprechen kann. Eine detaillierte Gliederung finden Sie im Anhang.

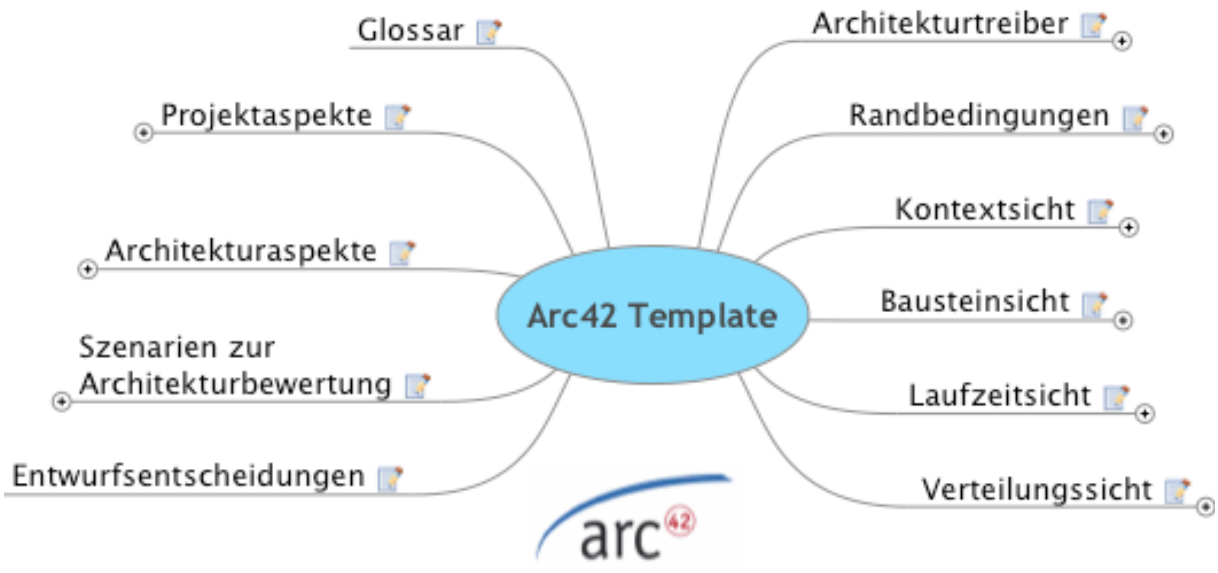

Abb. 1: Oberste Gliederungsebene des arc42 Strukturtemplates (Mindmap-Version)

Architekturdokumentation gemäß dieser Struktur soll als *Informationsrepository* interpretiert werden, nicht als einzelnes Dokument. Aus einem solchen Repository kann stakeholder- oder lesergerechte Information extrahiert werden, ohne die bei rein dokumentenbasierten Ansätzen übliche Redundanz. Als ideale Ergänzung hierfür erweisen sich mehrbenutzerfähige Modellierungswerkzeuge mit zentraler Datenhaltung (wie etwa Rational Rose, Rational Software Architect, MagicDraw oder Enterprise-Architect).

3 Der Kern: Architektursichten

Schon in den 70er-Jahren hat David Parnas gefordert, dass man mindestens zwei Strukturen für Software-Architekturen dokumentieren sollte: einerseits die vorhandenen Moduln und deren Zusammensetzung und andererseits eine dynamsiche Ablaufsicht, d.h. das Verhalten des Softwaresystems zur Laufzeit. Im Lauf der Jahre sind viele Vorschläge für die Anzahl der notwendigen Sichten und deren Wahl entstanden (Mehr dazu siehe unten im Abschnitt: Was sagen die Anderen).

Unsere Erfahrung zeigt, dass drei wesentliche Sichten in der Praxis genügen - unter vereinfachten Randbedingungen sogar noch weniger.

3.1 Bausteinsicht

Unvermeidbar ist die Bausteinsicht. Sie zeigt die (statischen) Bausteine des Systems und deren Zusammenhänge. Wir haben für arc42 das neutrale Wort „Baustein" als Oberbegriff für die Vielzahl von Namen gewählt, die im Projektalltag dafür verwendet werden, wie z.B. Module, Komponente, Klasse, Subsystem, Interface, Paket, Bibliothek, Framework, Schicht, Partition, Tier, Funktion, Makro, Operation, Tabelle, Datenbank oder Datei. Ein Baustein ist also im Endeffekt jedes kleine oder große Stück Quellcode, unabhängig davon mit welcher Technologie es entwickelt wird. Wir wollen für jeden Baustein in der Architekturdokumentation wissen, wie er heißt, welche Verantwortung er im Rahmen des Gesamtsystems übernimmt, welche Leistung er erbringt, über welche Schnittstellen er verfügt, u.v.a. mehr. Das arc42-Template stellt dafür sowohl Muster für seine Black-Box-Charakterisierung zur Verfügung, als auch ein Muster für die Beschreibung des Innenlebens in Form einer White-Box-Beschreibung. Die abstrakteste Form eines Bausteins ist das Gesamtsystem. Im logischen Kontextdiagramm wird es in Bezug zur Systemumgebung gesetzt.

3.2 Verteilungsicht

Mehr und mehr Systeme sind heute auf mehrere Standorte oder auf mehrere Rechner verteilt. Diese Dimension decken wir in der Verteilungssicht ab. Alles, worauf Software laufen kann, wird als Knoten in den Verteilungsdiagrammen dokumentiert. Neben diesen Knoten brauchen wir auch die "Kanäle", über die Informationen zwischen den Knoten ausgetauscht werden kann. Den Knoten werden dann die Bausteine, die darauf gespeichert werden oder laufen, zugeordnet. Zu den Kanälen hält man die Art und Klassifikation der Informationen fest, die darüber fließen. Sie brauchen eine derartige Sicht nicht zu erstellen, wenn Sie Software für einen einzigen Standardrechner schreiben, also kurz: wenn Verteilung für Sie keine Rolle spielt. Aber Vorsicht: selbst in einfacheren Fällen hat uns die Praxis gezeigt, dass Verteilungsdiagramme nützlich sind, insbesondere das Verteilungskontextdiagramm. In dieser Vogelperspektive dokumentieren Sie neben Ihrem System als einen Knoten wenigstens die Kanäle (oder Medien), über die Ihr System mit Nachsystemen Informationen austauscht. Insbesondere erleichtert die Verteilungssicht die Abstimmung zwischen Entwicklungs- und Betriebsorganisationen (Rechenzentren).

3.3 Laufzeitsicht

Als dritte Sicht schlagen wir vor, ausgewählte Laufzeitszenarien zu dokumentieren. Diese Laufzeitsicht veranschaulicht das gewünschte dynamische Verhalten Ihres Softwaresystems. Gehen Sie an diese Herausforderungen ähnlich wie Software-Tester heran: Wählen Sie zunächst Szenarien für den normalen Ablauf aus. Ergänzen Sie diese dann um Grenz- und Stress-Szenarien für wichtige Ausnahme- oder Sonderfälle. Laufzeitsichten sind ein Hilfsmittel, um dynamischen Verhalten zu finden oder zu bewerten. Beispielsweise erachten wir den Systemstart (Bootstrapping) oder wichtige Anwendungsfälle als geeignete Kandidaten für Laufzeitszenarien.
Für alle diese Sichten bietet beispielsweise die UML geeignete grafische Dokumentationsmittel. Eine ausführliche Darstellung, welche Ausdrucksmittel für welchen Zweck finden Sie in [ESA 05] oder unter [arc42].

4 Die Kapitel der arc42 Strukturvorlage

Lassen Sie uns einige wichtige Punkte aus der arc42-Strukturvorlage herausgreifen, ohne im Rahmen dieses Artikels vollständig auf die einzelnen Kapitel einzugehen (für weitergehende Erläuterungen und Beispiele siehe [arc42]).

- Formulieren Sie grundsätzlich explizite Architekturtreiber und Architekturziele (im Abschnitt „Architekturtreiber“)! Danach beurteilen Sie am Projektende, ob Ihre Architektur gut genug ist. Sie werden diese Ziele auch als Ausgangsbasis brauchen, wenn Sie Ihre Architekturen systematisch bewerten wollen.

- Legen Sie Randbedingungen, Einschränkungen und Annahmen schriftlich fest (im Abschnitt „Randbedingungen“), damit Sie wissen, wo Sie Freiheitsgrade für Entwurfsentscheidungen haben und wo nicht.

- Grenzen Sie Ihr System gegen die Systemumgebung systematisch ab und dokumentieren es in der „Kontextsicht“. Aus logischer Sicht hat das hoffentlich schon ein Requirements Engineer gemacht. Sie sollen die physischen Schnittstellen und die Kanäle bzw. die Medien von und zu Nachbarsystemen dokumentieren. An diesen Schnittstellen müssen Sie mit anderen Projekten oder Systemen verhandeln.

- An der „Bausteinsicht“ geht kein Weg vorbei. Sie entspricht in etwa dem Grundrissplan eines Hauses. Wenn Sie sonst gar nichts von Ihrer Architektur dokumentieren - diese Sicht muss (bis zum benötigten Niveau) auf jeden Fall dokumentiert sein.

- Entwurfsentscheidungen müssen über die Lebensdauer von Systemen nachvollziehbar bleiben. Deshalb sieht das arc42-Template dieses Kapitel vor, um alle wichtigen Entscheidungen, die Sie als Architekt im Laufe eines Projekts getroffen haben, rasch wieder auffindbar zu machen.

- Durch die szenariobasierte Architekturbewertung (beispielsweise ATAM, siehe [Clements02]) haben wir Mittel und Wege, über gezielt ausgewählte Szenarien bestimmte wünschenswerte Eigenschaften von Softwarearchitekturen, wie Flexibilität, Robustheit, Zuverlässigkeit, ... nachprüfen und bewerten zu können. Wenn Sie solche Bewertungen vornehmen, dann dokumentieren Sie die Grundlagen dafür im Abschnitt „Szenarien zur Architekturbewertung“.

- Das Kapitel der „Architekturaspekte“ ist in der detaillierten Gliederung (siehe Anhang) sehr ausführlich. Nicht alle Aspekte, die für Software- Architekturen relevant sind, lassen sich direkt in Baustein-, Laufzeit- oder Verteilungssichten darstellen. Zu diesen „Architekturaspekten“ zählen unter anderem Persistenz, Ablaufsteuerung, Fehler- & Ausnahmebehandlung, Monitoring, Migration und andere.

- Unter der Überschrift „Projektaspekte“ sind viele Punkte gesammelt, die die Schnittstellen zwischen Architekten und anderen Projektbeteiligten betreffen. So z. B. Risikolisten als Grundlage für die Diskussion mit Projektleitern oder Technologieexperten, Change Requests zur Diskussion mit Auftraggebern, Endanwendern und Requirements Engineers oder Auswirkungen zur rechtzeitigen Diskussion mit der Inbetriebnahme, den Organisatoren oder Migrationsspezialisten.

5 Praktische Erfahrungen

Wir haben in den letzten Jahren bei zahlreichen Kunden unterschiedlicher Branchen (etwa: Telekommunikation, Medizintechnik, Logistik, öffentliche Verwaltung) die arc42 Schablone in

Projekten verschiedener Größe und Ausrichtung erfolgreich eingesetzt. Auf Basis dieser Erfahrungen haben wir Struktur des Templates iterativ angepasst – jedoch seit 2004 keine signifikanten strukturellen Änderungen mehr vorgenommen. Die meisten Ergänzungen oder Erweiterungen betrafen den Abschnitt „Architekturaspekte“, zu dem unsere Kunden zahlreiche Ergänzungsvorschläge unterbreitet haben. Einige konkrete Erfahrungen zur Einführung:

- Die Einarbeitung in die Struktur des Templates bedeutet, wie die Einführung jeglicher Methodik oder Technologie, eine Lernkurve für die Beteiligten. Unserer Erfahrung nach lässt sich diese Lernkurve durch geschickte Wahl der betreffenden Pilotprojekte deutlich reduzieren: Projekte mit ohnehin strukturell durchdachter und ausformulierter Dokumentation sind nach unserer Erfahrung innerhalb kürzester Zeit in der Lage, die vorhandenen Dokumente in der arc42-Struktur abzubilden.

- Projekte können das arc42-Template als eine Art „Checkliste“ verwenden, um ihre eigene Architekturdokumentation auf Vollständigkeit hin zu prüfen. Dies kann etwa in Form halb- bis ganztägiger Architekturreviews geschehen.

- Mit der Einführung einer projektübergreifenden Strukturvorlage zur Dokumentation geht oftmals die Überarbeitung oder Optimierung der betroffenen Architekturprozesse einher: Mehrmals wurde durch die thematisch breit angelegte Struktur des arc42-Templates den betroffenen IT-Architekten erst die Breite ihres eigenen Aufgabenspektrums vor Augen geführt. Das wiederum führte für betroffene Unternehmen zu einer klar umrissenen Rollendefinition[1] der Architekten.

- In regelmäßigen Abständen von 3-6 Monaten sollten Architekten aus verschiedenen Projekten eines Unternehmens in kurzen Workshops ihre Erfahrungen mit dem Einsatz der Strukturvorlage gegenseitig abgleichen. Ein einfaches Mittel ist die *Architekturpräsentation* der jeweiligen Systeme, bei der in Form eines Vortrags von maximal 60 Minuten Dauer eine Kurzfassung jedes Kapitels der arc42-Dokumentation präsentiert wird. Insbesondere müssen dabei spezifische Erweiterungen und Ergänzungen an der Strukturvorlage abgestimmt werden.

[1] Arc42 postuliert folgende sechs Kernaktivitäten von IT-Architekten: 1.: Einflussfaktoren klären (d.h. Anforderungen und Randbedingungen prüfen), 2.: Systeme strukturieren, 3.: Architekturaspekte konzipieren, 4.: Umsetzung überwachen, 5.: Architekturen bewerten und 6.: Architekturen kommunizieren.

Insgesamt empfehlen wir Organisationen und Unternehmen, so schnell wie möglich eine einheitliche Struktur zur Dokumentation von IT-Architekturen einzuführen – der Produktivitäts- und Qualitätsgewinn in Projekten und Systemen wiegt die (in der Regel ohnehin nur kurze) Lernkurve schnell wieder auf.

6 Alternative Ansätze von Strukturvorlagen für IT-Architekturen

In der Literatur und im Internet gibt es einige weitere Ansätze von Strukturvorlagen zur Dokumentation von Software- und IT-Architekturen. Einige davon möchten wir kurz vorstellen.

Philippe Kruchten hat mit seinem Artikel über „4+1 Views" [Kruchten95] das Konzept der Architektursichten hoffähig gemacht. Dieses Modell hat für die zentralen Ideen in arc42 Pate gestanden, ebenso für die Ansätze aus [Starke05].

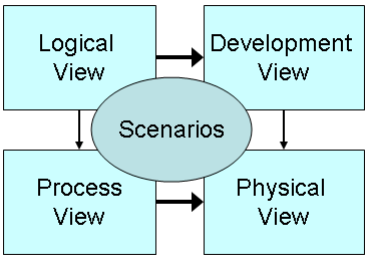

Abb. 3: 4 +1 Views von Philippe Kruchten

Als Bestandteil des Rational Unified Process (RUP) wird der 4+1 Ansatz heute in der Praxis eingesetzt, außerhalb den RUP unseres Wissens nach nicht. Die Konsistenz zwischen den einzelnen Sichten ist schwer herstellbar. Positiv an Kruchtens Ansatz ist der Einbezug der fachlichen Sicht, die sowohl in den Scenarios (in Form von Use-Cases) als auch im Logical View Eingang finden.

Zu den Urvätern von Architektursichten gehört der ehemalige IBM-Forscher John A. Zachmann (siehe [zifa]). Er schuf ab 1970 den nach ihm benannten Architekturframework [zifa]. Dieser Framework geht über die reine Beschreibung von IT-Systemen deutlich hinaus, und bezieht sich im Ursprung auf Unternehmen als Ganzes. Der Zachman-Framework postuliert 30 (!) Sichten. Das empfinden wir als deutlich zu viel für den alltäglichen Gebrauch, obwohl Zachmann sich insbesondere im Umfeld US-amerikanischer Großunternehmen verbreiten konnte.

In den USA werden viele (hautpsächlich militärische) Systeme nach dem C4ISR Architecture Framework (siehe [C4ISR]) entwickelt. Er teilt Architekturen in 5 Sichten: Funktionen, Informationen, Organisation, Infrastruktur und Netzwerk. Jede Sicht wird weiter in statische Modelle, so genannte Produkte, zerlegt. Es gibt dazu nur wenig frei zugängliche Dokumentation, jedoch basieren andere Ansätze wie TOGAF (siehe unten) und DODAF auf dem C4ISR-AF.

Deutlich weiter verbreitet ist das ISO Referenzmodell für Open Distributed Processing (RM-ODP, [rmodp]). Es geht nach einem objektorientierten Ansatz vor und enthält neben 5 Sichten (genannt Viewpoints: Enterprise, Information, Computational, Engineering, Technology) noch 8 unterschiedliche Transparencies. Diese beschreiben die übergreifenden Aspekte access, failure, location, migration, replication und transaction. Zu diesem Modell gibt es eine ganze Reihe ergänzender Dokumentation, jedoch kaum veröffentliche Beispieldokumentation. Unserer Erfahrung nach ist RM-ODP recht schwerfällig und eignet sich eher für langlaufende Großprojekte (jedoch würde einigen von denen auch ein schlankerer Dokumentationsansatz unserer Ansicht nach gut stehen :-).

Das Architekturframework der Open Group, genannt TOGAF (siehe [togaf]), kann man für 90 Tage kostenfrei evaluieren, die kommerzielle Nutzung ist teilweise eingeschränkt. Zur Dokumentation von Architekturen schlägt TOGAF ein *Enterprise Continuum* vor, eine Sammlung sämtlicher architekturrelevanten Ergebnisse. Diese sind untergliedert in Geschäfts-, Daten-, Applikations- und Technologiearchitektur. Die Dokumentation von TOGAF enthält kaum konkrete Hinweise, wie Architekturdokumentation praktisch aussehen soll.

Dana Bredemeyer, geistiger Vater des HP-internen Software-Architekturprozesses, schlägt auf [bredemeyer] vor, Architekturbeschreibungen mit den Architekturtreibern zu beginnen, unterteilt in Architekturvision, Architekturanforderungen, Annahmen, Einflussfaktoren sowie Schlüsselkonzepte und Architekturstil. Anschließend dienen auch bei Bredemeyer eine Reihe von Sichten der Beschreibung der detaillierten Architektur. Er gruppiert Sichten in folgende Matrix:

	Verhaltenssicht	Struktursicht
Konzeptionelle Architektur (abstrakt)	Trace einzelner "Kollaborationen", d.h. Szenarien	Architekturdiagramm, informelle Komponentenspezifikation
Logische Architektur (detailliert)	Kollaborationsdiagramme	Architekturdiagramm mit Schnittstellen, Schnittstellenspezifikationen
Ausführungs-	Kollaborationsdiagramme,	Architekturdiagramme, die

architektur (Prozess- und Verteilungssicht)	die Prozesse ausdrücken	aktive Komponenten zeigen

Tab. 1: Sichten nach Dana Bredemeyer [Bredemeyer]

Damit enthält sein Vorschlag einiges von den Dingen, die wir für Architekturbeschreibungen als notwendig erachten, leider fehlt in seinem Modell Platz für die übergreifenden Lösungsaspekte. Positiv an seinem Ansatz erachten wir, dass er die notwendigen Sichten (Bausteine, Laufzeit, Verteilung sowie Kontextsicht) immerhin mit seiner Begrifflichkeit abdecken kann - wenn uns auch die klare terminologische Trennung von Sichten und Architekturebenen fehlt.

7 Fazit

Die Verwendung einer Strukturvorlage zur Dokumentation (und letztlich Kommunikation) von IT- und Softwarearchitekturen erweist sich in der Praxis als förderlich für die Effektivität und Effizienz von Entwicklungsprojekten und auch der Qualität der betreffenden IT-Systeme. Die arc42-Vorlage bietet eine erprobte und praxistaugliche Struktur an, die sich in vielen Fällen bewährt hat. Diese Struktur lässt sich bei Bedarf unternehmens- oder auch systemspezifisch anpassen. Sie ist leicht erlern- und vermittelbar und von daher mit geringem Aufwand in Unternehmen und Organisationen einzuführen.

8 Danksagung

Die Autoren möchten sich bei den zahlreichen Anwendern und Reviewern des arc42 Templates für die konstruktiven Diskussionen sowie Verbesserungsvorschläge bedanken, die diese Strukturvorlage letztendlich erst praxistauglich gemacht haben.

9 Anhang: Detaillierte Gliederung des arc42-Architekturtemplates

Folgende Tabelle zeigt die Verfeinerung der arc42-Struktur auf die in der Praxis relevanten Gliederungsebenen.

Nr.	Titel	Erläuterung
1	Einleitung	Kurzbeschreibung des Systems, Formalia: Änderungshistorie, Versionsstand, Release-Notes
2 2.1 2.2 2.3	Architekturtreiber Architekturziele Aufgabenstellung Stakeholder	Dieses Kapitel ist ein Extrakt aus den Systemanforderungen, die im Idealfall als eigenständige Dokumentation (Requirements Model, Systemanalyse) vorliegen.
3 3.1 3.2 3.3	Randbedingungen Technische Randbed. Organisatorische Randbed. Konventionen	Dieses Kapitel beschreibt die Einschränkungen, denen Architekten beim Systementwurf unterliegen. Thematisch sind sie mit den Anforderungen verwandt, stammen in der Regel jedoch aus anderen Quellen.
4	Kontextsicht	Hier wird das System im Kontext aller Nachbarsysteme und externen Schnittstellen dokumentiert.
5 5.1 5.2 5.3	Bausteinsicht Gesamtsystem Ebene 2 Ebene 3	Hier wird die Bausteinstruktur des Systems beschrieben, wechselweise in White- und Blackboxbeschreibungen.
6 6.1 6.2 6.3	Laufzeitsicht Laufszenario 1 Laufszenario 2 Laufszenario 3	Hier wird das Zusammenspiel der Systembausteine zur Laufzeit dokumentiert. Dabei können spezifische Szenarien (etwa: Bootstrapping) gesondert herausgehoben werden.
7	Verteilungssicht	Hier wird die Verteilung des Systems auf Hardware dokumentiert – (Deployment-Sicht).
8	Entwurfsentscheidungen	
9	Szenarien zur Architekturbewertung	Grundlagen für die szenariobasierte Architekturbewertung.
10 10.1 10.2 etc.	Architekturaspekte: • Persistenz • Benutzungsoberfläche • Ergonomie • Ablaufsteuerung • Transaktionsbehandlung • Sessionbehandlung • Sicherheitsaspekte • Verteilung • Ausnahmebehandlung • Parallelisierung • Konfiguration • Internationalisierung • Migration • Monitoring • Protokollierung	Hier dokumentieren Sie sichtenübergreifende (i.d.R. technische) Aspekte. Die Aufzählung ist als Beispiel zu betrachten und i.d.R. system- oder unternehmensspezifisch zu ergänzen.
11	Projektaspekte	Informationen und Risiken von oder für das Management des Projektes.
12	Glossar	Projekt- oder systemspezifische Begriffe.

Tab. 2: Verfeinerte Gliederung des arc42 Architekturtemplates

Literaturverzeichnis

[arc42] Peter Hruschka, Gernot Starke: arc42 – Das Internet-Portal für Software-Architekten. Online: www.arc42.de

[Bredemeyer] Bredemeyer, Dana: Architecture Documentation Action Guide. Online: http://www.bredemeyer.com/architecture_documentation_action_guides.htm (Stand: Juli 2006).

[C4ISR] C4ISR Architecture Framework. Online: www.software.org/pub/afc/c4isr-dodaf.asp, Stand Juli 2006.

[Clements01] Clements, Paul, Rick Kazman & Mark Klein: Evaluating Software Architectures: Methods and case Studies. Addison Wesley, 2001.

[Clements03] Clements, Paul et. al: Documenting Software Architectures. Addision Wesley, 2003.

[Kruchten95] Kruchten, Philippe: The 4+1 View Model of Architecture. IEEE Software Vol. 12, Nr. 6, 1995.

[rmodp] Online: www. dstc.edu.au/Research/Projects/ODP/standards.html (Stand: Juli 2006).

[Starke05] Starke, Gernot: Effektive Software-Architekturen – Ein praktischer Leitfaden. Carl Hanser Verlag, München, 2. Auflage, 2005.

[togaf] The OpenGroup: Architecture Framework (Version 8.1). Nach kostenfreier Registrierung online erhältlich unter www.opengroup.org. (Stand Juli 2006)

[zifa] Zachman Institute for Framework Advancement. Online: www.zifa.com (Stand: Juli 2006)