3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Wiki ausgedruckt?

10 praxistaugliche Tipps für eine Architekturdokumentation

Stefan Zörner

oose Innovative Informatik GmbH

Wiki ausgedruckt?
10 praxistaugliche Tipps für eine Architekturdokumentation

Abstract

„Füllen Sie diese tolle Vorlage aus, dann vergessen Sie garantiert nichts. Verwenden Sie UML! Ist standardisiert, versteht jeder ...“Mit solch eines Beraters würdigen Hinweisen wird man in diesem Vortrag verschont. Stattdessen gibt es echtes Erfahrungswissen zum Festhalten einer Software-Architektur. Welche Ziele können damit verfolgt werden? Was gehört unbedingt rein? Welche Werkzeuge und Notationen haben sich bei Erstellung und Pflege bewährt? Und welche bei der Kommunikation im Team und an Dritte? Und vielleicht soll ja wirklich etwas ausgedruckt werden ...

Zielgruppe

Softwareentwickler und -architekten, ggf. auch Projekt- und Teamleiter
Voraussetzungen: Idealerweise eigene Erfahrung mit Softwareentwicklung im Team, Rolle beliebig

Agenda

1. Warum Softwarearchitekturen dokumentieren?
2. Die Aufgabe beschreiben
3. Die Lösung kommunizieren
4. Nicht an den Werkzeugen scheitern
5. Fazit und Weitere Informationen

Agenda

1

1 Warum Softwarearchitekturen dokumentieren?

2 Die Aufgabe beschreiben

3 Die Lösung kommunizieren

4 Nicht an den Werkzeugen scheitern

5 Fazit und Weitere Informationen

Fragen, die neue Mitarbeiter so stellen …

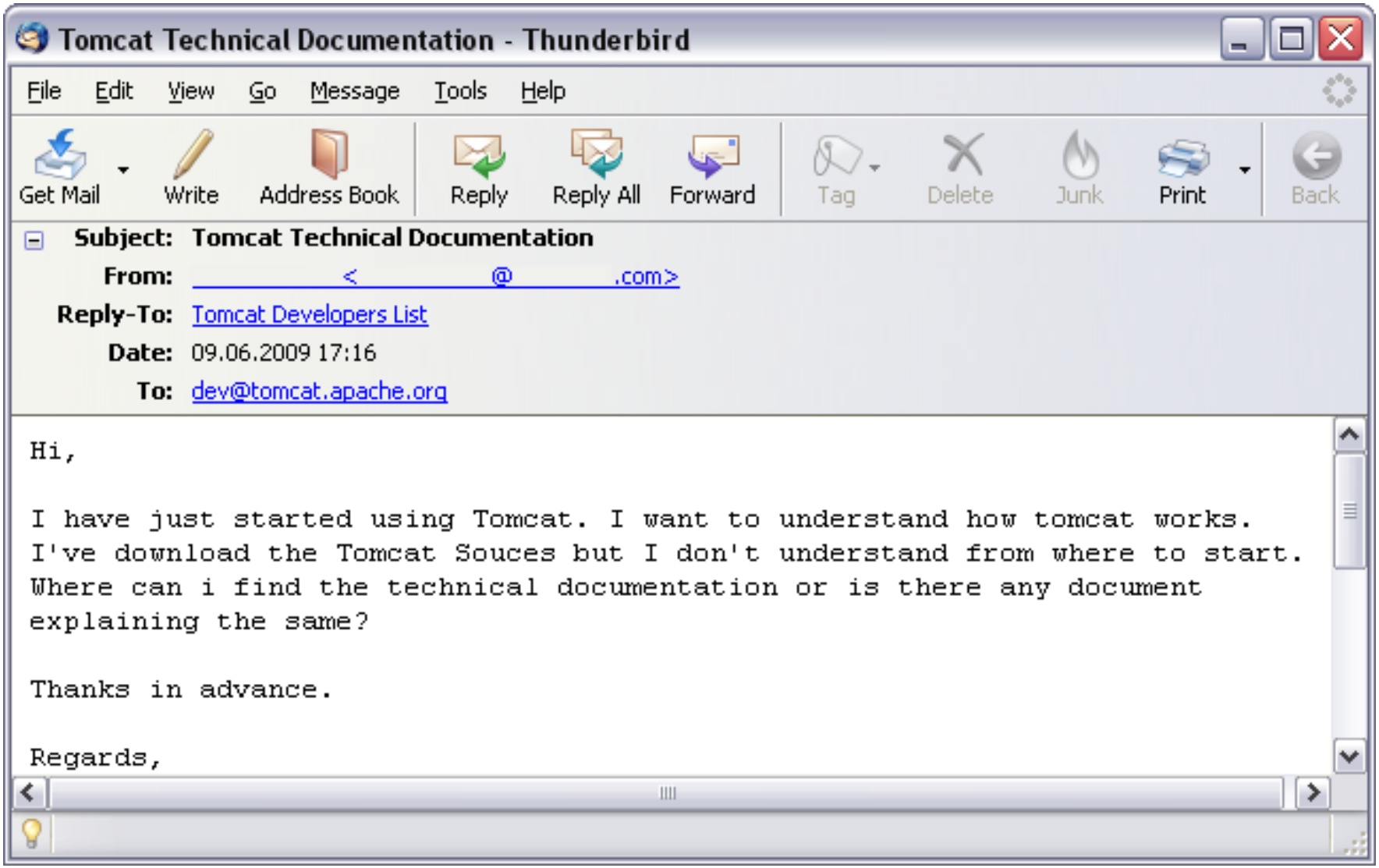
- Wo soll ich sitzen?
- Was brauche ich für Tools?
- Wie checke ich die Quelltexte aus, und wie baue ich die Software?
- Warum sind bei mir die Tests rot?
- …

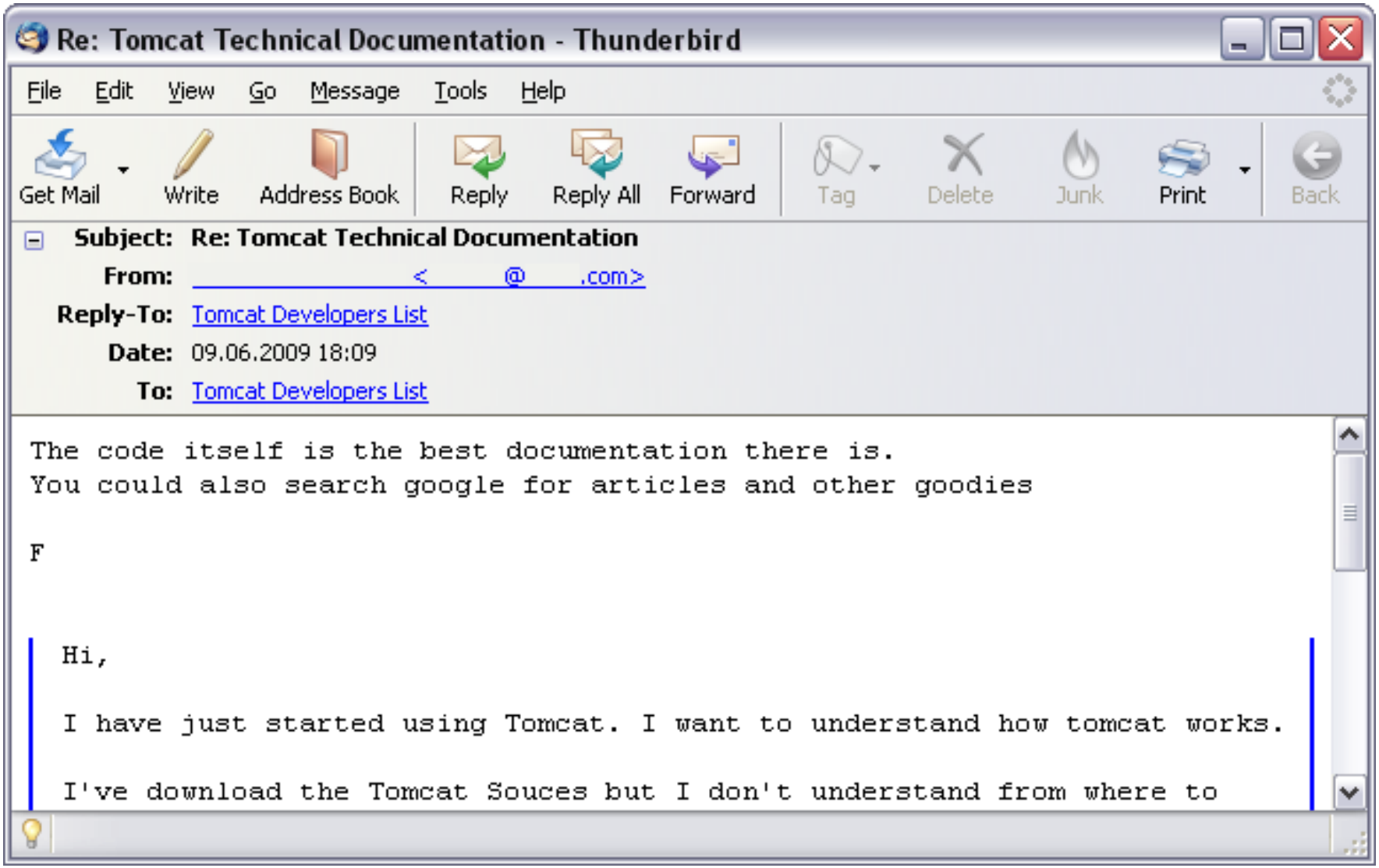
Fragen, die neue Mitarbeiter so stellen … (2)

- Ich finde mich nicht zurecht. Wie finde ich einen Einstieg?
- Diese Teile hier – wie arbeiten die zusammen? Habt Ihr das irgendwo aufgemalt?
- Ich soll hier neue Funktionalität hinzufügen, wie stelle ich das an?
- Ich habe hier etwas Ähnliches gefunden, kann ich das wiederverwenden?
- …

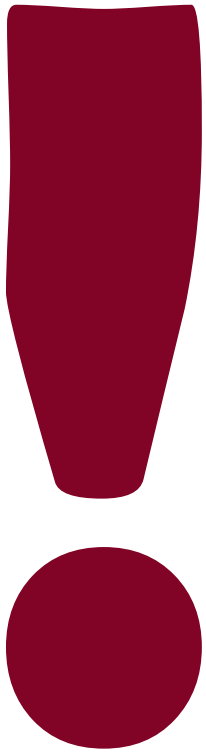
Fragen, die neue Mitarbeiter so stellen … (3)

- Diese Software, an der wir hier arbeiten, was macht die überhaupt?
- Warum benutzen wir eigentlich noch Java 1.4?
- Wieso habt Ihr das so gemacht? Ist das nicht viel zu kompliziert?
- Würde man das nicht eigentlich so machen?
- …


Tomcat Technical Documentation - Thunderbird
File Edit View Go Message Tools Help
Get Mail
Write
Address Book
Reply
Reply All
Forward
Tag
Delete
Junk
Print
Back
Subject: Tomcat Technical Documentation
From: < @ .com>
Reply-To: Tomcat Developers List
Date: 09.06.2009 17:16
To: dev@tomcat.apache.org
Hi,
I have just started using Tomcat. I want to understand how tomcat works.
I've download the Tomcat Souces but I don't understand from where to start.
Where can i find the technical documentation or is there any document
explaining the same?
Thanks in advance.
Regards,


Re: Tomcat Technical Documentation - Thunderbird
File Edit View Go Message Tools Help
Get Mail
Write
Address Book
Reply
Reply All
Forward
Tag
Delete
Junk
Print
Back
Subject: Re: Tomcat Technical Documentation
From: < @ .com>
Reply-To: Tomcat Developers List
Date: 09.06.2009 18:09
To: Tomcat Developers List
The code itself is the best documentation there is.
You could also search google for articles and other goodies
F
Hi,
I have just started using Tomcat. I want to understand how tomcat works.
I've download the Tomcat Souces but I don't understand from where to

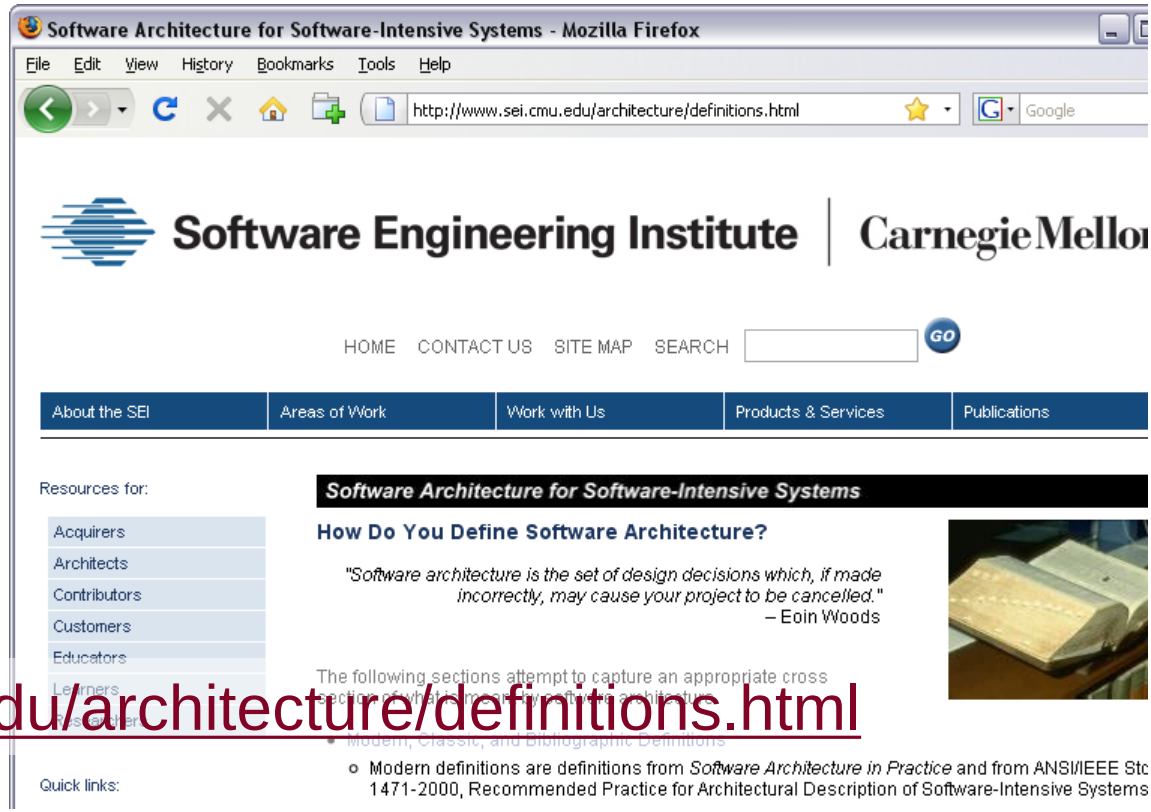
Antworten, die neue Mitarbeiter daraufhin erhalten …



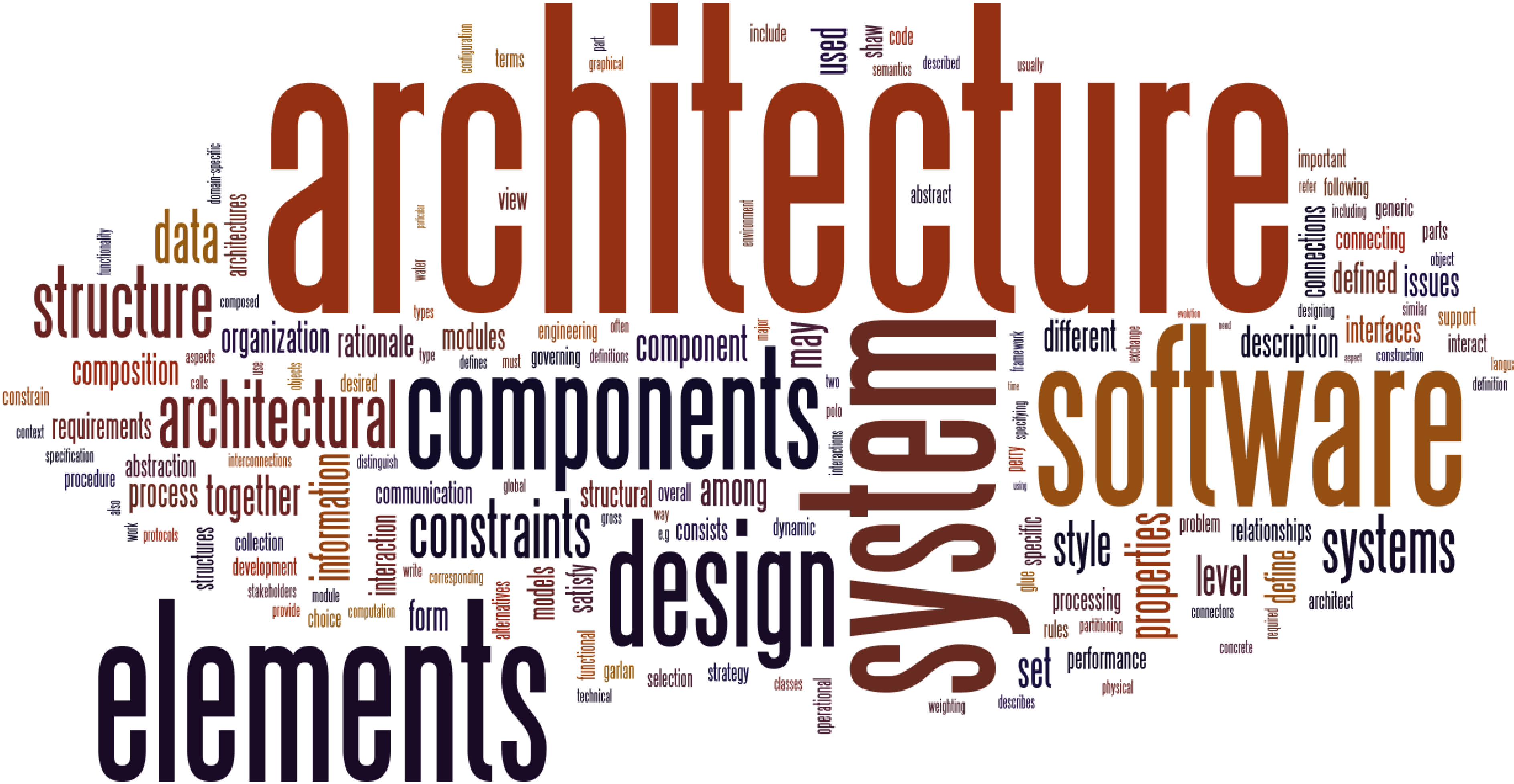
- Steht alles im Wiki.
- Das haben wir nicht dokumentiert – wir gehen agil vor.
- Das war schon so, als ich neu war.
- Das ist historisch gewachsen.

Was ist Softwarearchitektur?

- Es gibt nicht die eine allgemein akzeptierte Definition für Softwarearchitektur
- Das Software Engineering Institute (SEI) sammelt sogar Definitionen:



➔ http://www.sei.cmu.edu/architecture/definitions.html

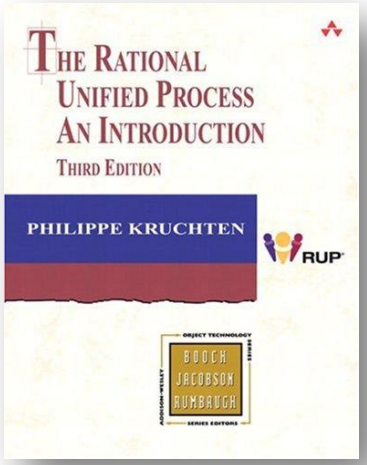

architecture
software
system
elements
components
design
architectural
structure
constraints
component
data
organization
rationale
modules
systems
style
properties
level
define
information
interaction
composition
requirements
abstraction
process
together
among
structural
description
different
interfaces
connections
defined
issues
connecting
generic
parts
important
following
used
include
code
terms
view
abstract
may
set
performance
processing
relationships
architect
communication
collection
development
form
models
satisfy
consists
dynamic
strategy
selection
garlan
functional
alternatives
choice
structures
procedure
constrain
architectures
engineering
governing
definitions
desired
interconnections
distinguish
usually
described
semantics
shaw
graphical
configuration
problem
connectors
concrete
rules
partitioning
physical
describes
weighting
operational
classes
technical
corresponding
computation
provide
stakeholders
module
protocols
write
global
overall
gross
perry
glue
specific
support
interact
languages
definition
construction
similar
object
including
refer
designing
framework
exchange
specifying
interactions
polo
two
environment
water
types
composed
aspects
calls
use
objects
context
specification
also
work
way
type
defines
must
often
major
functionality
domain-specific

Eine (!) konkrete Definition

Architektur := $\sum$ wichtige Entscheidungen

Softwarearchitektur umfasst die Summe verschiedener wichtiger Entscheidungen über

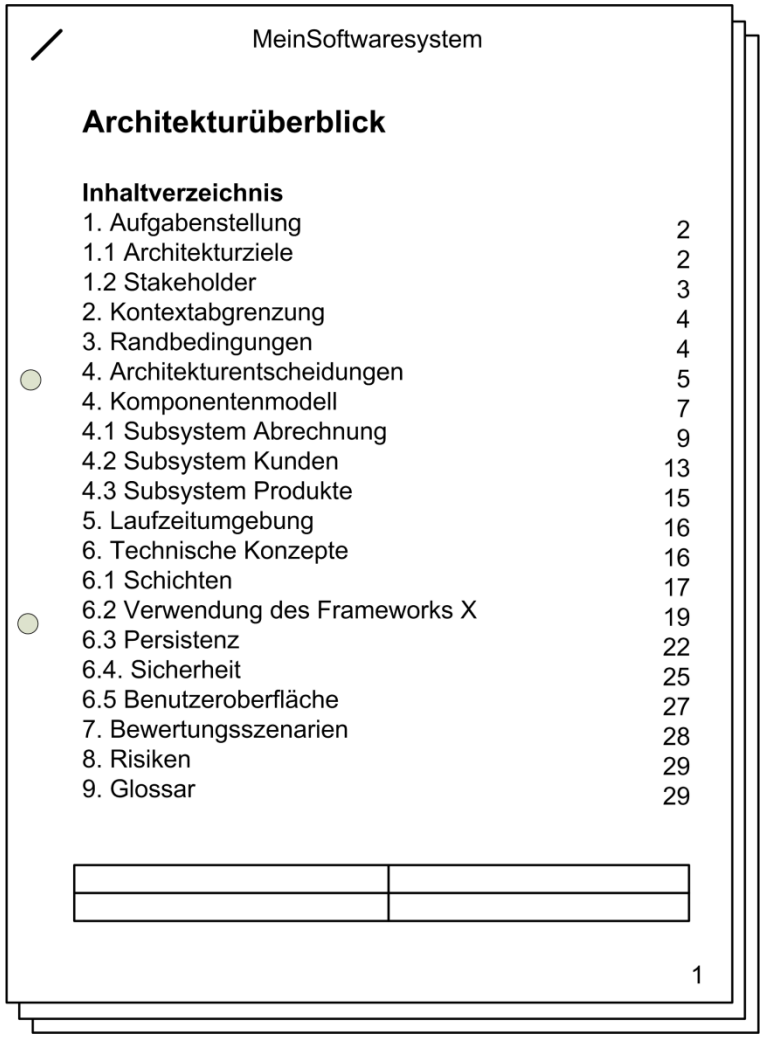
- die Auswahl von Strukturelementen und deren Schnittstellen, aus denen das System zusammengesetzt ist
- das Verhalten und Zusammenspiel dieser Elemente
- den hierarchischen Aufbau von Subsystemen
- den zugrunde liegenden Architekturstil
- …



G. Booch, P. Krutchen, K. Bittner and R. Reitman.

The Rational Unified Process — An Introduction. 1999.

Idealbild: Architekturüberblick auf < 30 Seiten



MeinSoftwaresystem

Architekturüberblick

Inhaltverzeichnis

1

Implementierung des Idealbildes?

arc42 -- Vorschlag für ein Template (Gernot Starke, Peter Hruschka)

➔ http://arc42.de/

Praxistipp #1 („Zielsetzung klären“)

Vor dem Anfertigen jeglicher Dokumentation stehen die Fragen „Für wen?“und „Weshalb?“.

- Wer sind unsere Zielgruppen?
- Welchen Zweck verfolgen wir mit der Dokumentation?

Drei Ziele von Architekturdokumentation

- Beim Entwurf der Architektur unterstützen
- Die Umsetzung und Weiterentwicklung des Systems leiten
- Die Architektur nachvollziehbar und bewertbar machen

Agenda

2

1 Warum Softwarearchitekturen dokumentieren?

2 Die Aufgabe beschreiben

3 Die Lösung kommunizieren

4 Nicht an den Werkzeugen scheitern

5 Fazit und Weitere Informationen

Firefox

Apache ActiveMQ ™ -- Index

activemq.apache.org

Google

ActiveMQ™

The Apache Software Foundation
http://www.apache.org/

Download ActiveMQ 5.6.0 Today!

Apache ActiveMQ ™ is the most popular and powerful open source messaging and Integration Patterns server.

Apache ActiveMQ is fast, supports many Cross Language Clients and Protocols, comes with easy to use Enterprise Integration Patterns and many advanced features while fully supporting JMS 1.1 and J2EE 1.4. Apache ActiveMQ is released under the Apache 2.0 License

Grab yourself a Download, try our Getting Started Guide, surf our FAQ or start Contributing and join us on our Discussion Forums.

Features

- Supports a variety of Cross Language Clients and Protocols from Java, C, C++, C#, Ruby, Perl, Python, PHP
  - OpenWire for high performance clients in Java, C, C++, C#
  - Stomp support so that clients can be written easily in C, Ruby, Perl, Python, PHP, ActionScript/Flash, Smalltalk to talk to ActiveMQ as well as any other popular Message Broker

Overview

- Index
- News
- New Features
- Getting Started
- FAQ
- Articles
- Books
- Download
- License

Search

Search

Sub Projects

- Apollo
- CMS
- NMS
- Camel

Community

- Support
- Contributing
- Discussion Forums
- Mailing Lists

Architekturziele als Produktkarton

- Was entwickeln wir eigentlich?
- Was ist das zentrale Verkaufs- oder Nutzungsargument ("Claim", "Slogan")
- Wem nützt es?
- Was sind die wesentlichen Features des Systems?
- Wie unterscheidet es sich von Produkten der Mitbewerber, oder der Vorgängerversion?

Speziell für die Architektur

- Welche Qualitätsmerkmale (= Ziele) sind besonders wichtig?
- Welche Randbedingungen sind zentral?

„DokChess“ – Ziele und Features

- DokChess ist eine voll funktionsfähige Schachengine
- Sie dient als einfach zugängliches und zugleich ungemein attraktives Fallbeispiel für Architekturentwurf , -bewertung und -dokumentation.
- Der verständliche Aufbau lädt zum Experimentieren und zum Erweitern der Engine ein
- Ziel ist nicht die höchstmögliche Spielstärke – dennoch gelingen den Gelegenheitsspieler ansprechende Partien

Wesentliche Features

- Vollständige Implementierung der FIDE-Schachregeln
- Unterstützt das Spiel gegen menschliche Gegner und andere Schachengines
- Beherrschung zentraler taktischer Ideen, beispielsweise Gabel und Spieß
- Integration mit modernen graphischen Schach-Frontends

Praxistipp #2 („Produktkarton“)

Jede Architekturbeschreibung, ganz gleich ob Dokument oder Präsentation, beginnt mit der Aufgabenstellung.

- Das Anfertigen eines virtuellen Produktkartons mit Slogan ist dabei sehr hilfreich.

Systemkontext

"Systemkontext erstellen bedeutet Umfeldanalyse. Das hat Robinson Crusoe auch als erstes gemacht.

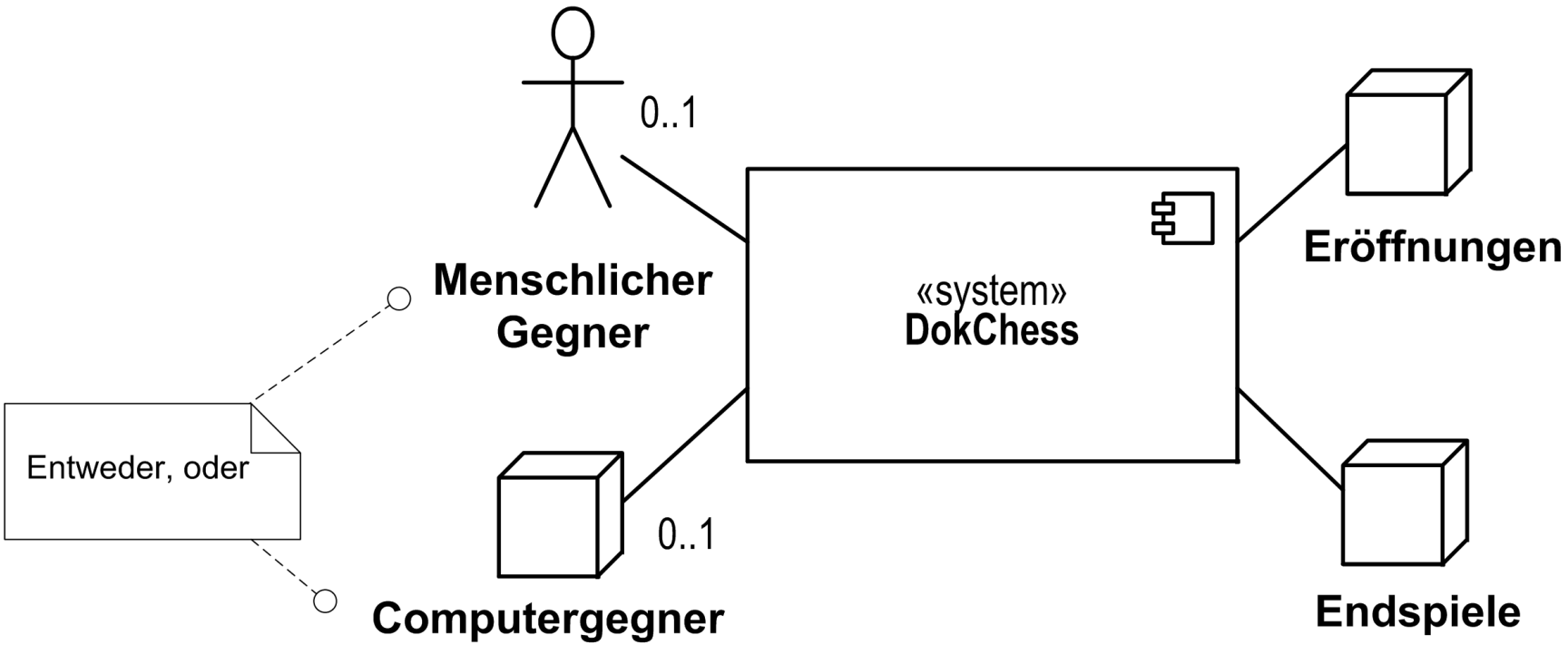
(Axel Scheithauer)

Kontext – Software agiert nicht allein …

- Stets gibt es Beteiligte außerhalb des Systems:
  - Anwendergruppen, die Funktionalität nutzen und erwarten
  - Fremdsysteme, die zur Ausführung erforderlich sind

„Die **Kontextsicht** zeigt das Umfeld, d.h. alle außerhalb des eigenen Systems liegenden Akteure, mit denen direkt kommuniziert wird.“

Systemkontext „DokChess“

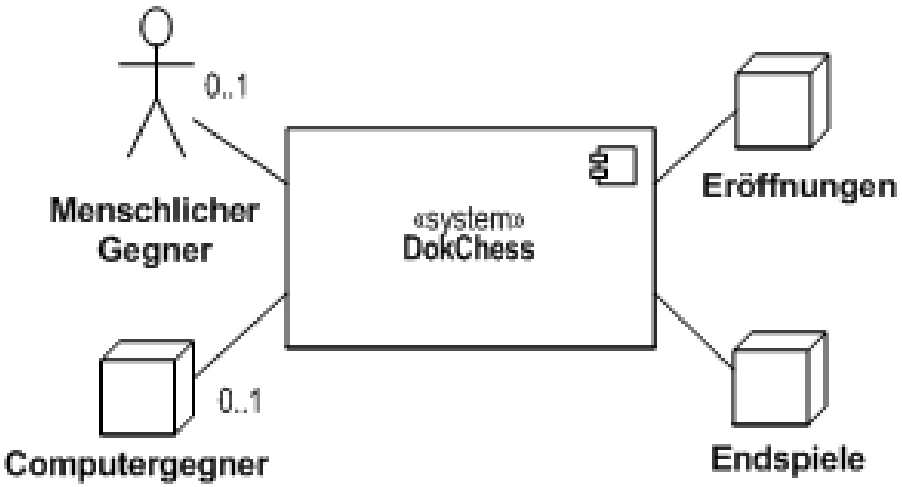


Praxistipp #3 („ Kontextabgrenzung“)

Ein guter Einstieg in ein System grenzt es zunächst von der Umgebung ab.

- Verantwortlichkeitsprinzip im Großen
- Mit wem interagieren wir, und warm?

Systemkontext und Architekturziele



Was ist drum herum?

Was steckt drin?

Agenda

3

1 Warum Softwarearchitekturen dokumentieren?

2 Die Aufgabe beschreiben

3 Die Lösung kommunizieren

4 Nicht an den Werkzeugen scheitern

5 Fazit und Weitere Informationen

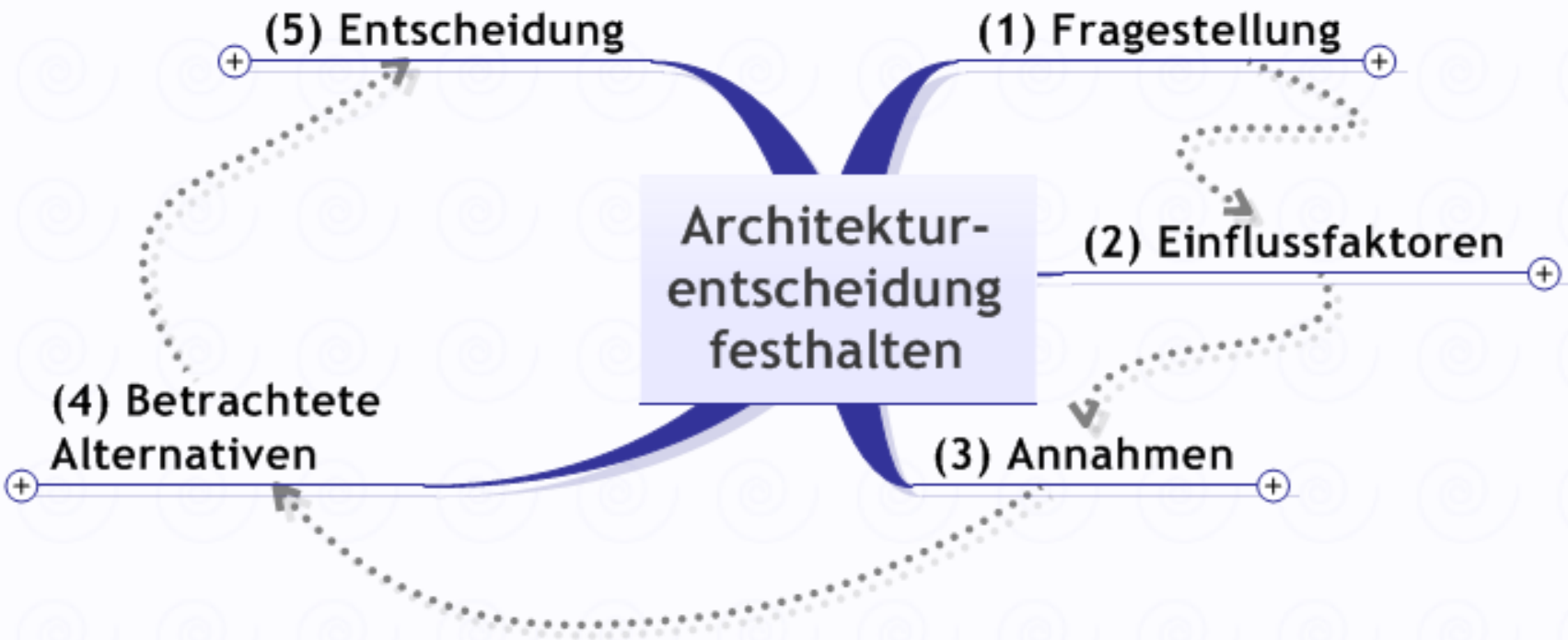
Was ist Softwarearchitektur?

"Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled."

(Eoin Woods)

- Architekturentscheidungen sind diejenigen, die sich im weiteren Verlauf nur sehr schwer revidieren lassen.
- Konsequenzen: höhere Kosten, Zeitverlust, ggf. scheitert das Vorhaben

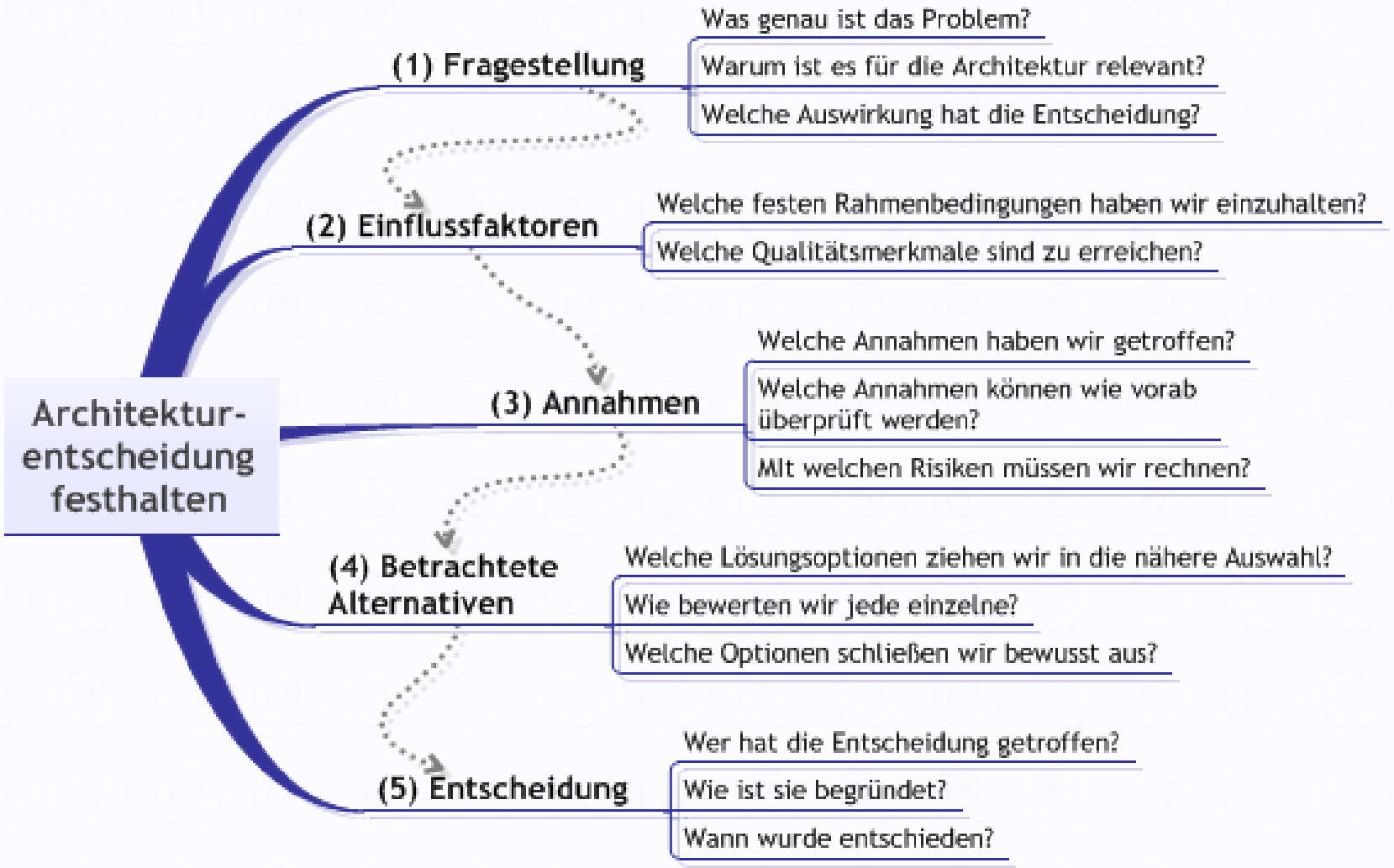
Entscheidungen treffen und festhalten. Ein Werkzeug



Praxistipp #4 („ Entscheidungen“)

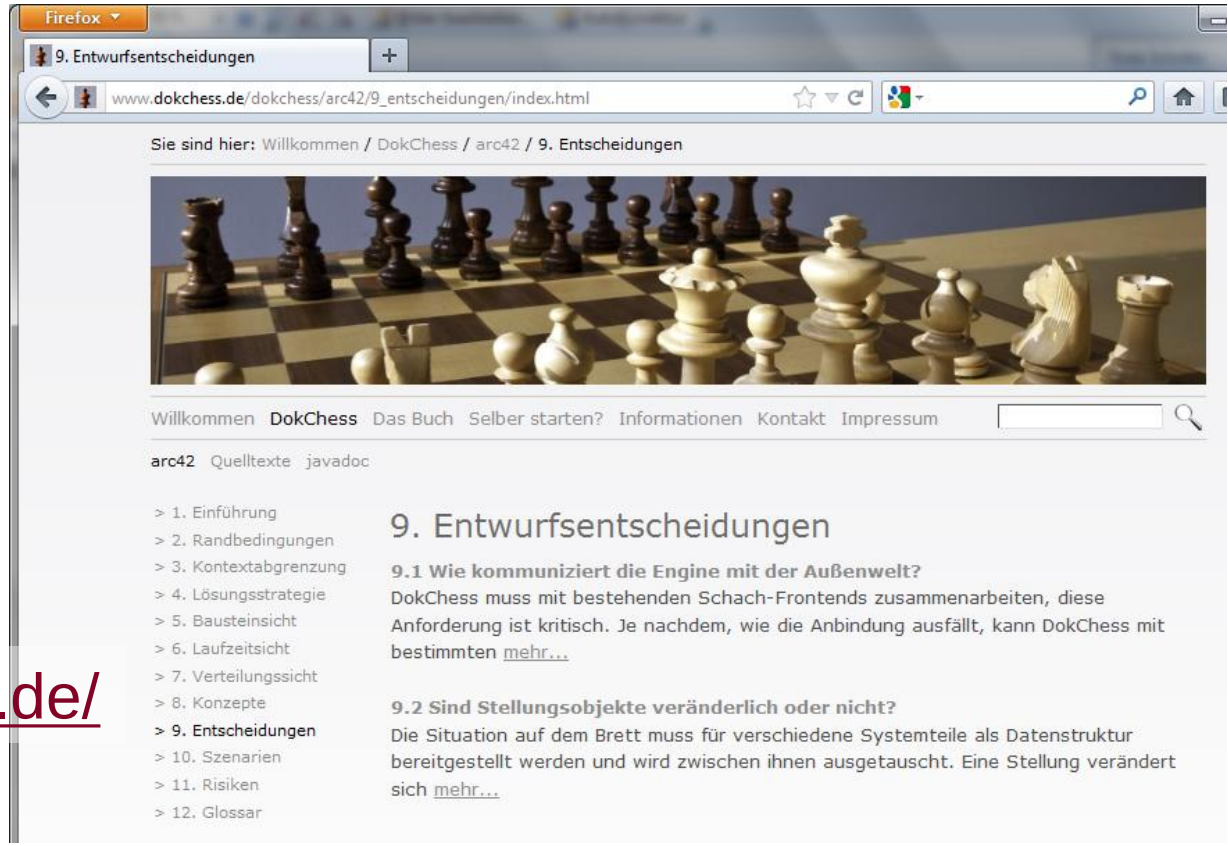
Zentrale Entscheidungen nachvollziehbar festhalten.

- Schlüssel, um bessere Antworten zu haben als „Historisch gewachsen“


Architektur-entscheidung festhalten
(1) Fragestellung
Was genau ist das Problem?
Warum ist es für die Architektur relevant?
Welche Auswirkung hat die Entscheidung?
(2) Einflussfaktoren
Welche festen Rahmenbedingungen haben wir einzuhalten?
Welche Qualitätsmerkmale sind zu erreichen?
(3) Annahmen
Welche Annahmen haben wir getroffen?
Welche Annahmen können wie vorab überprüft werden?
Mit welchen Risiken müssen wir rechnen?
(4) Betrachtete Alternativen
Welche Lösungsoptionen ziehen wir in die nähere Auswahl?
Wie bewerten wir jede einzelne?
Welche Optionen schließen wir bewusst aus?
(5) Entscheidung
Wer hat die Entscheidung getroffen?
Wie ist sie begründet?
Wann wurde entschieden?

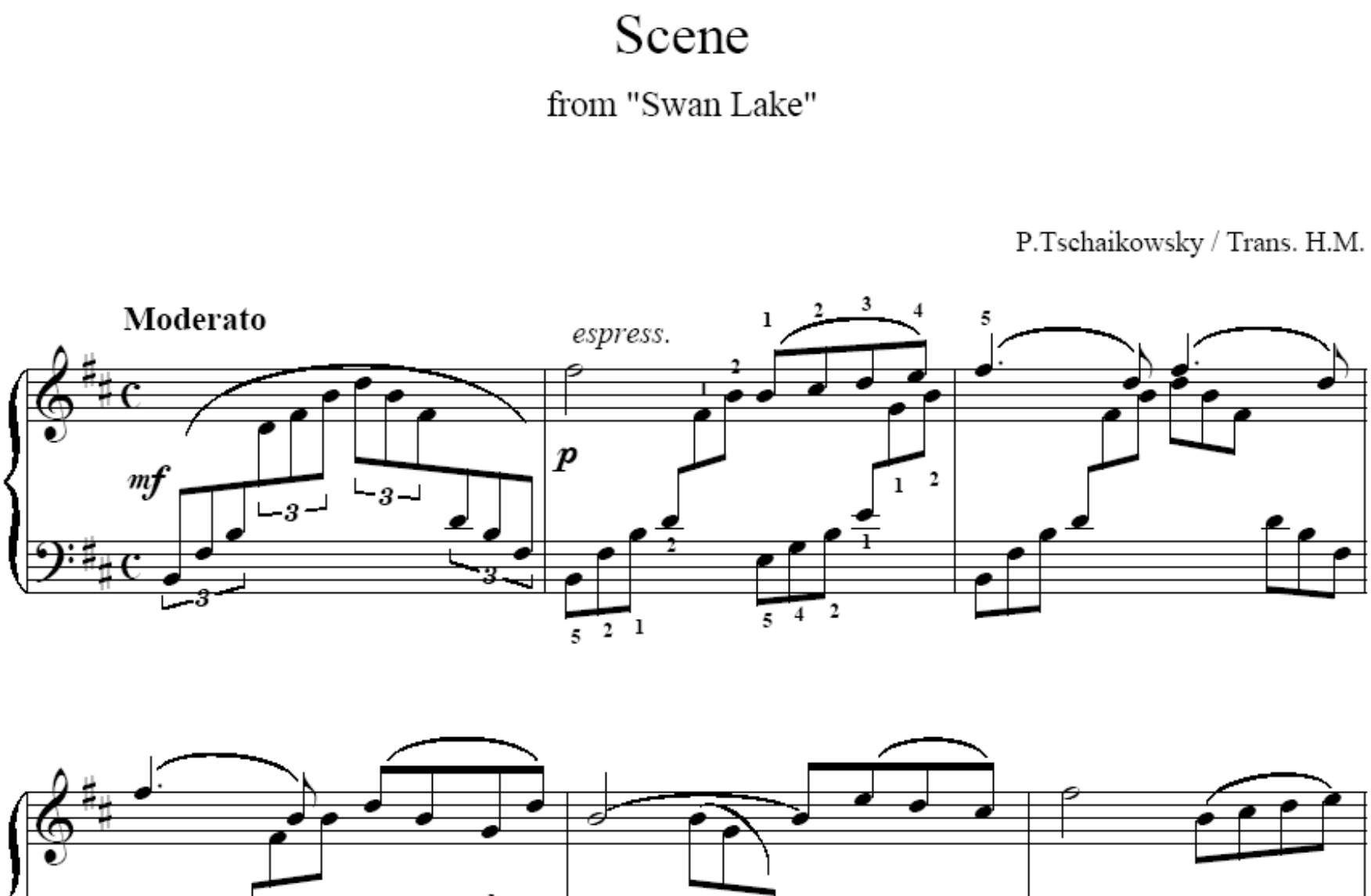
Zwei zentrale Entscheidungen in DokChess

- Wie kommuniziert die Engine mit der Außenwelt?
- Sind Stellungsobjekte veränderlich oder nicht?



http://www.dokchess.de/

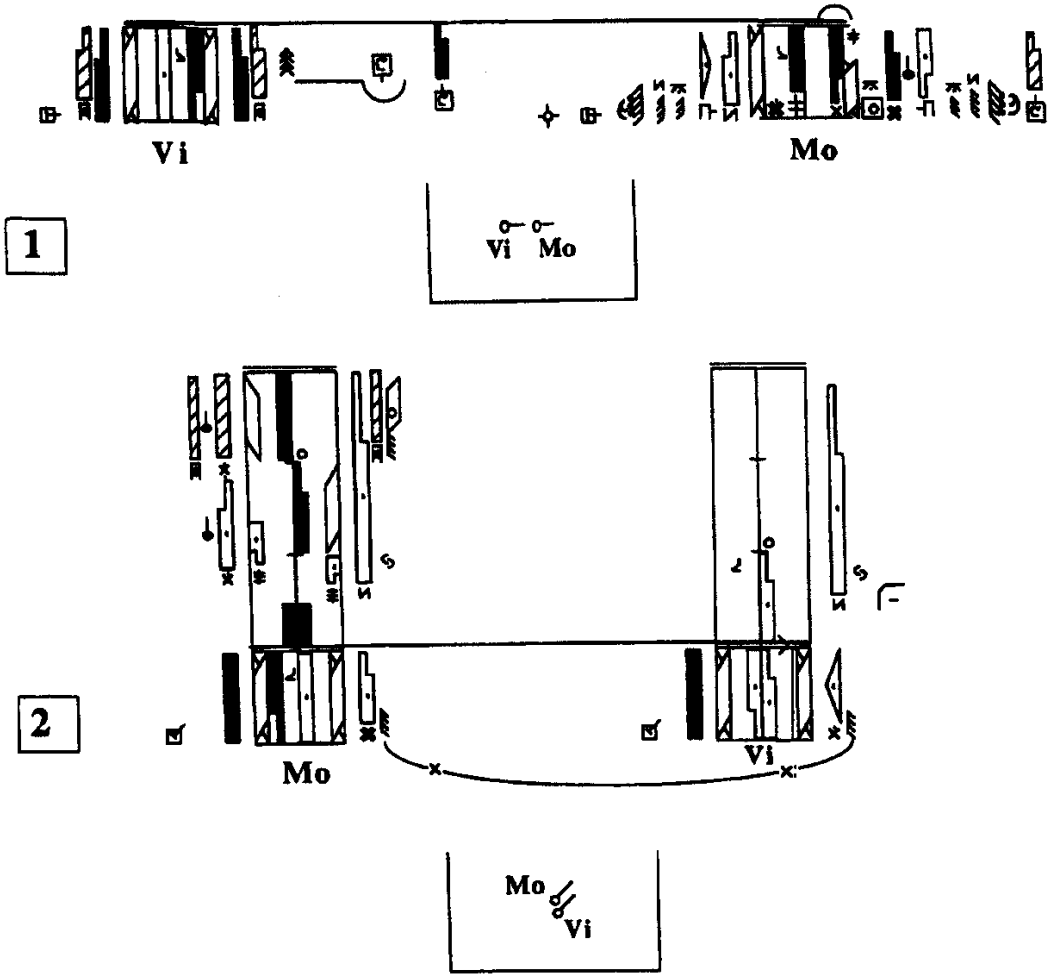
Schwanensee (1877)



Beispiel Tanznotation

Fig 1

Fig 2



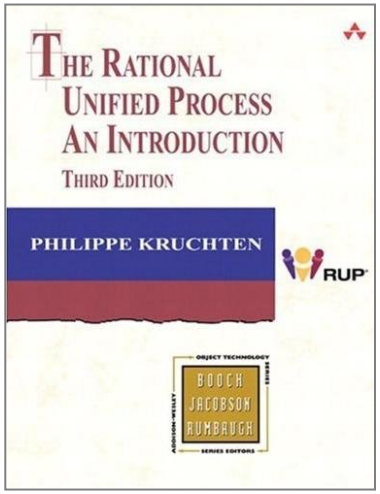
Sichten (Views) auf Softwarearchitektur

- Es ist sinnvoll, bestimmte Aspekte einer Software mit Bilder statt textuell zu beschreiben
- Ein einzelnes Bild reicht in der Regel nicht aus

  ➔ Unterschiedliche Sichten für unterschiedliche Aspekte

Beispiel: Rational Unified Process (P. Kruchten)

- 4 + 1 Views:
  - Logical View
  - Development View
  - Process View
  - Physical View
  - Scenarios

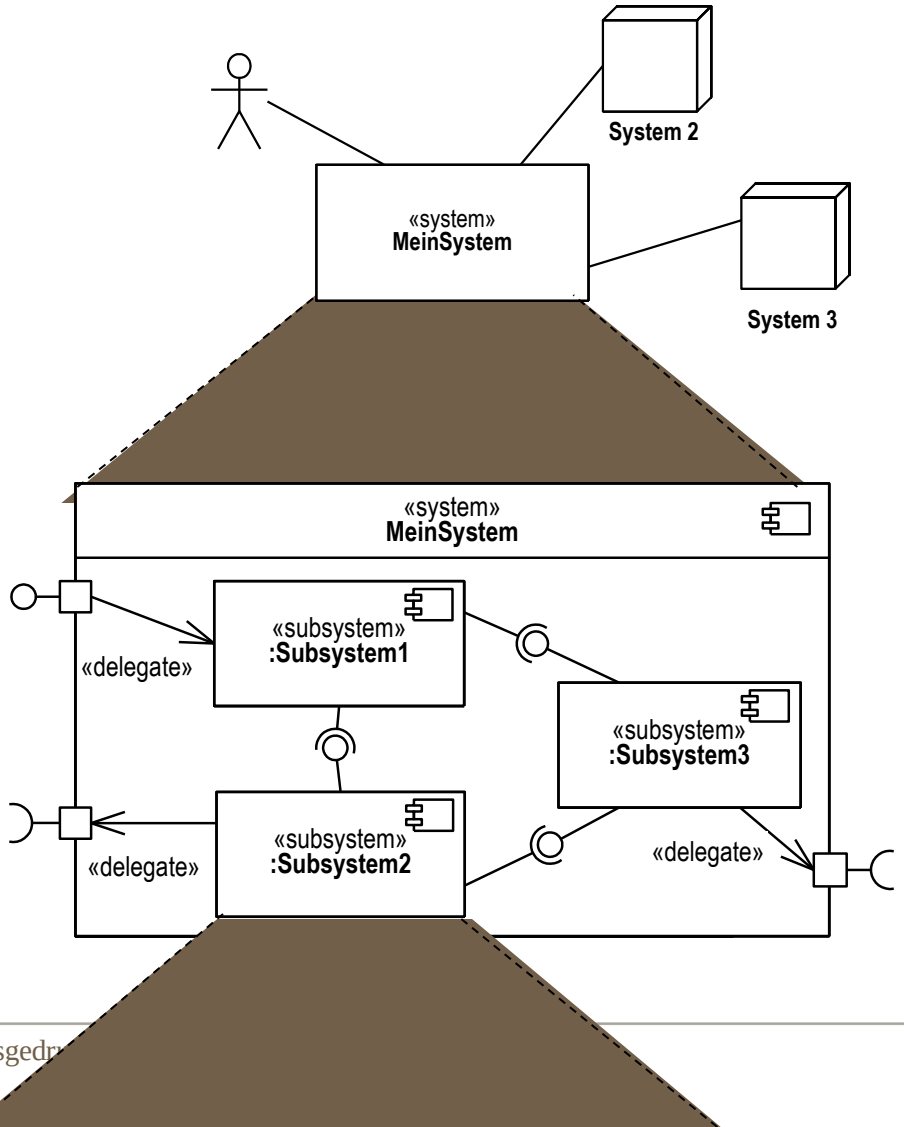


Alternativer Vorschlag für Sichten

Sichten nach arc42

- Kontextsicht
- Bausteinsicht (= Struktur)
- Laufzeitsicht (= Verhalten, Dynamik)
- Verteilungssicht (= Deployment auf die Zielumgebung)

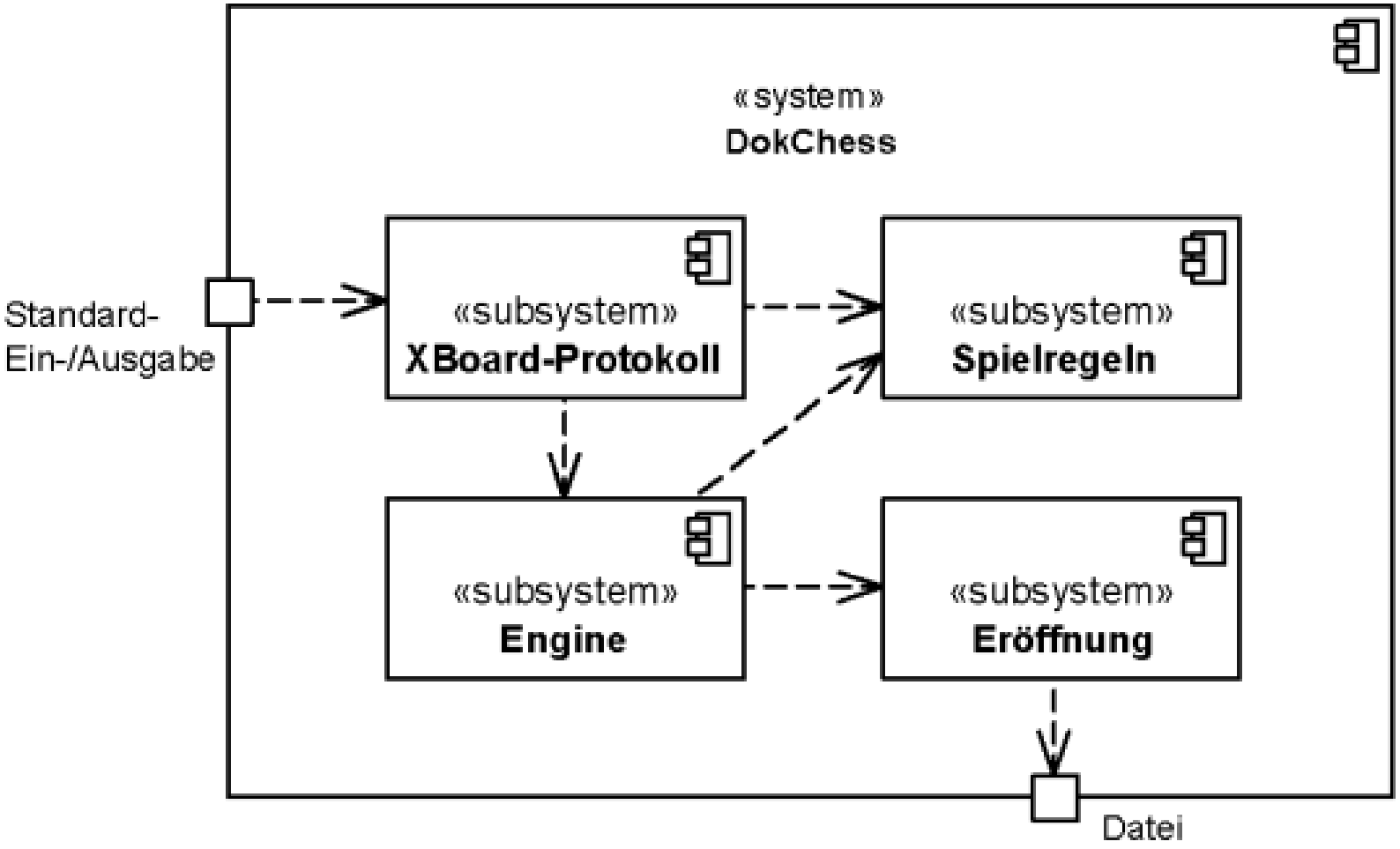
Zusammenspiel Kontextsicht / Bausteinsicht



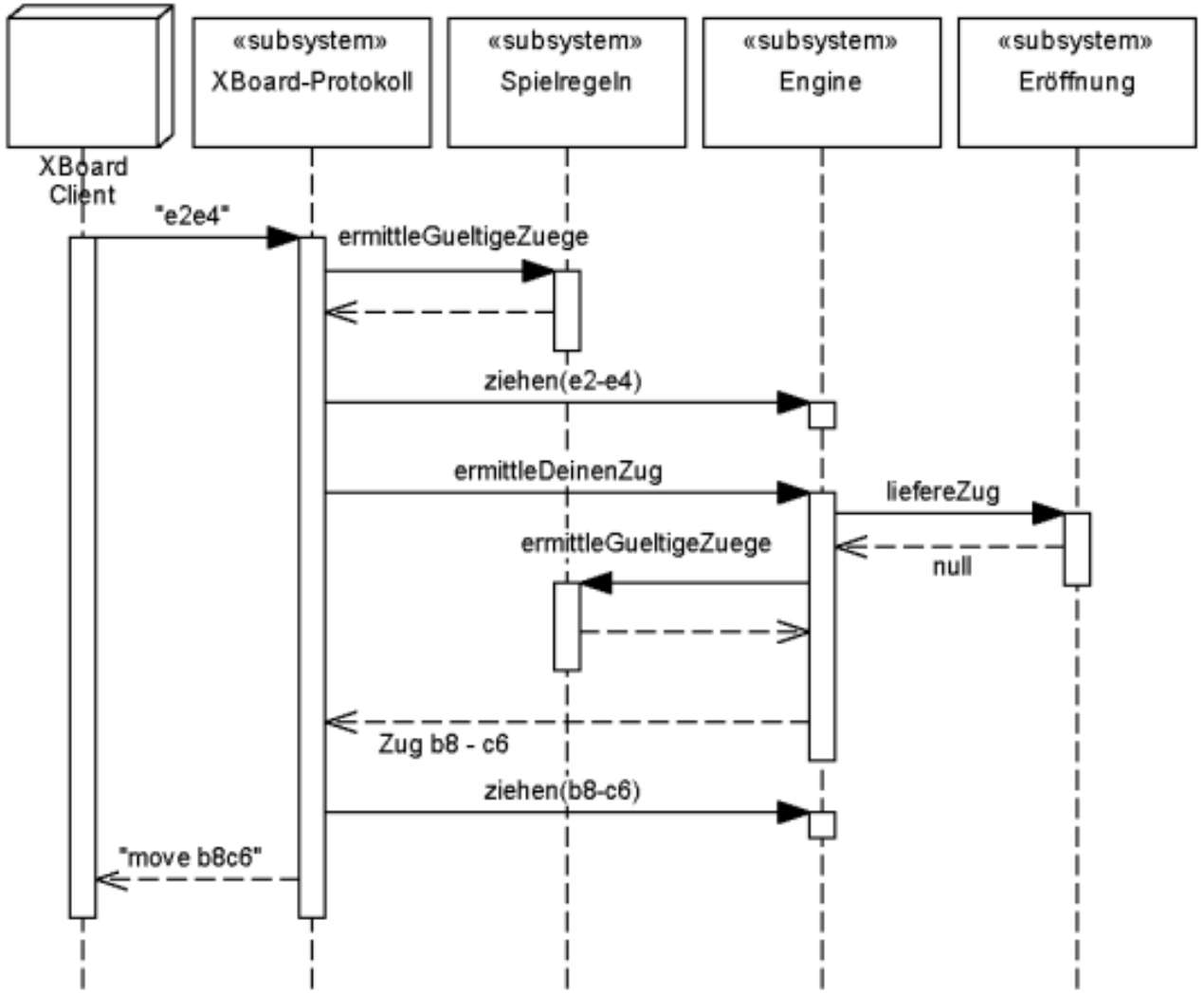
Blackbox

Whitebox

Beispiel: DokChess, Bausteinsicht Ebene 1



Laufzeitsicht: Ablauf bei der Ermittlung eines Zuges



Praxistipp #5 („Sichten“)

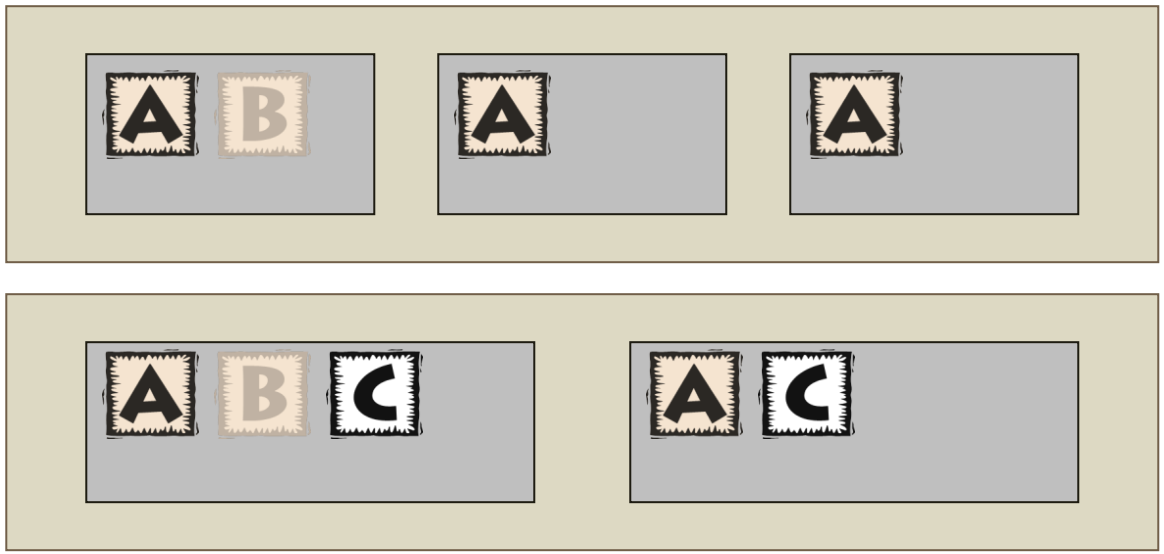
5.

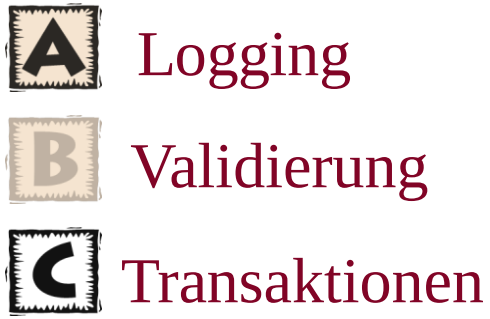
Unterschiedliche Sichten auf die Softwarearchitektur anfertigen, je nach Bedarf.

- Ausgangspunkt dabei ist die Zerlegung in Teile.

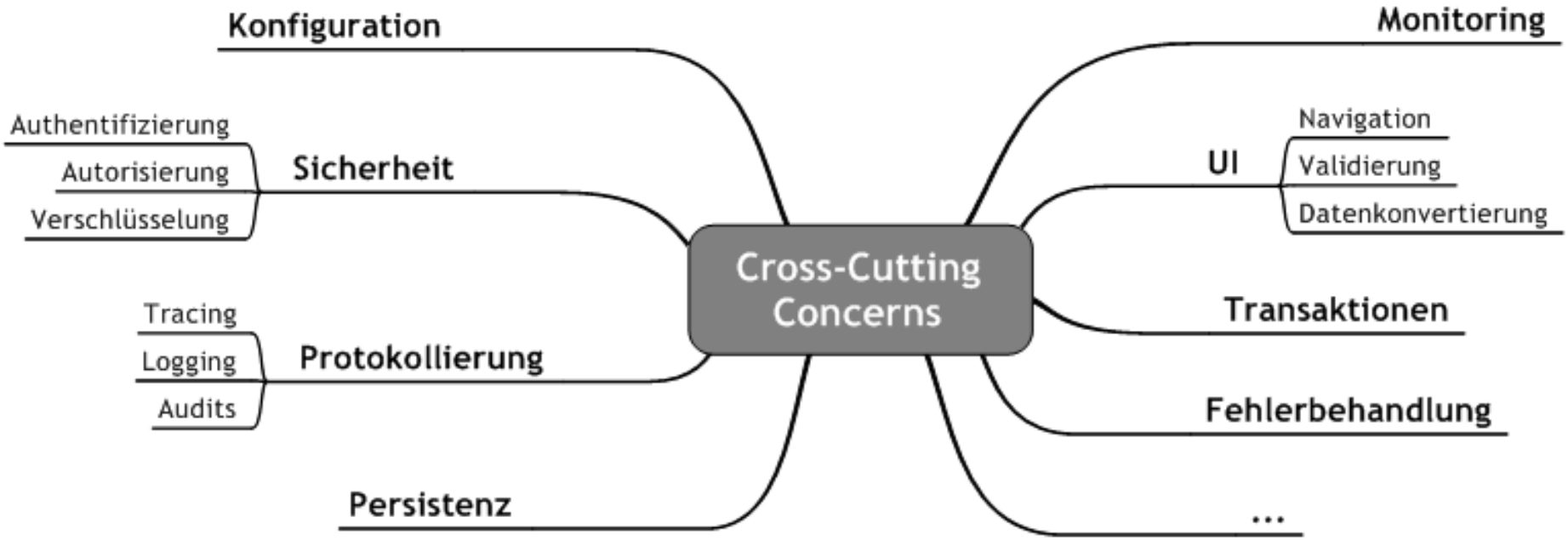
Grenzen der Modularisierung

- Es gibt in der Regel Aspekte, die sich nicht eindeutig einem Modul zuordnen lassen, sondern mehrere (ggf. alle) betreffen.
- Solche **Cross-Cutting concerns** (dt. übergreifende Belange, Querschnittsfunktionalität) finden sich horizontal und/oder vertikal (über mehrere Schichten) im Gesamtsystem, z.B. …





Beispiele für mögliche querschnittliche Belange



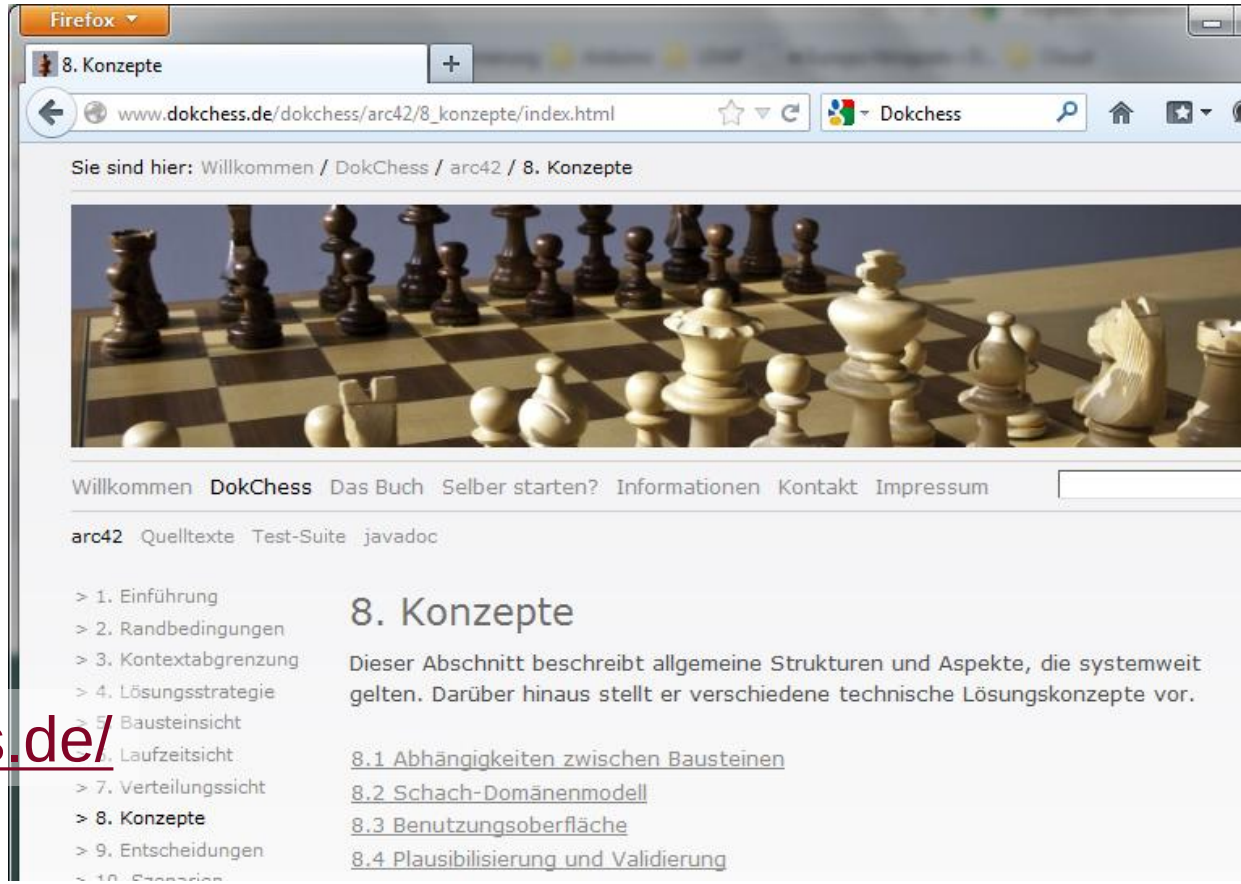
Praxistipp #6 („Übergreifende Themen“)

Gleiche Dinge auch gleich lösen. Zentrale, übergreifende Themen einmal bearbeiten, bewerten und dann geeignet festhalten und kommunizieren.

- Vermeidet Redundanzen.
- Begünstigt konsistente Lösungen.

Übergreifende Themen in DokChess

- Zusammenspiel von Bausteinen, Fehlerbehandlung, Testen Logging, …



http://www.dokchess.de/

Agenda

4

1 Warum Softwarearchitekturen dokumentieren?

2 Die Aufgabe beschreiben

3 Die Lösung kommunizieren

4 Nicht an den Werkzeugen scheitern

5 Fazit und Weitere Informationen

„Dokumentation“ als Fremdwort

Do|ku|men|ta|ti|on […zion] [lat.] die; -, -en:
1. a) Zusammenstellung u. Ordnung von Dokumenten und Materialien jeder Art, durch die das Benutzen und Auswerten ermöglicht oder erleichtert wird …

Vom pragmatischen Umgang mit Gliederungen

Architekturdokumentation != ein Template ausfüllen

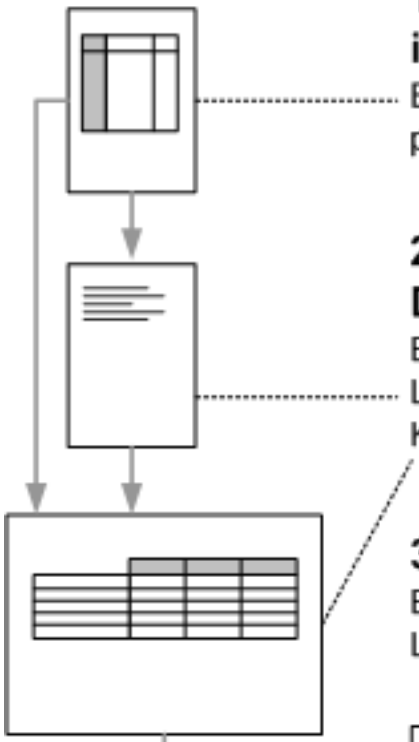
- Gliederungsvorschläge (z.B. arc42) motivieren Inhalte
- können Orientierung geben, wie Sie vorgehen

Für Arbeitsergebnisse entscheiden Sie sich explizit

- Welche Inhalte erstellen Sie?
- Für wen? (Zielgruppe)
- In welcher Tiefe? (detailliert? Überblick?)

Die Zielgruppen sind entscheidend für die Zutaten die Sie erstellen, und auch die Tiefe.

Ansatz



1. Zielgruppen der Dokumentation identifizieren und deren Intention beschreiben

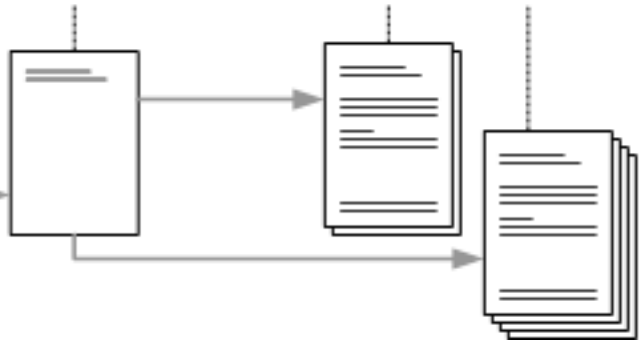
Ergebnis: Tabelle mit Zielgruppen, deren Interessen und primären Zielen

2. Anhand der Zielgruppen für Dokumentationsmittel entscheiden

Ergebnisse:
Liste mit Dokumentationsmitteln
Kreuztabelle mit Zielgruppen und Dokumentationsmitteln

3. Dokumente identifizieren, später ableiten

Ergebnisse:
Liste mit Dokumenten, Dokumente (später, im weiteren Verlauf)

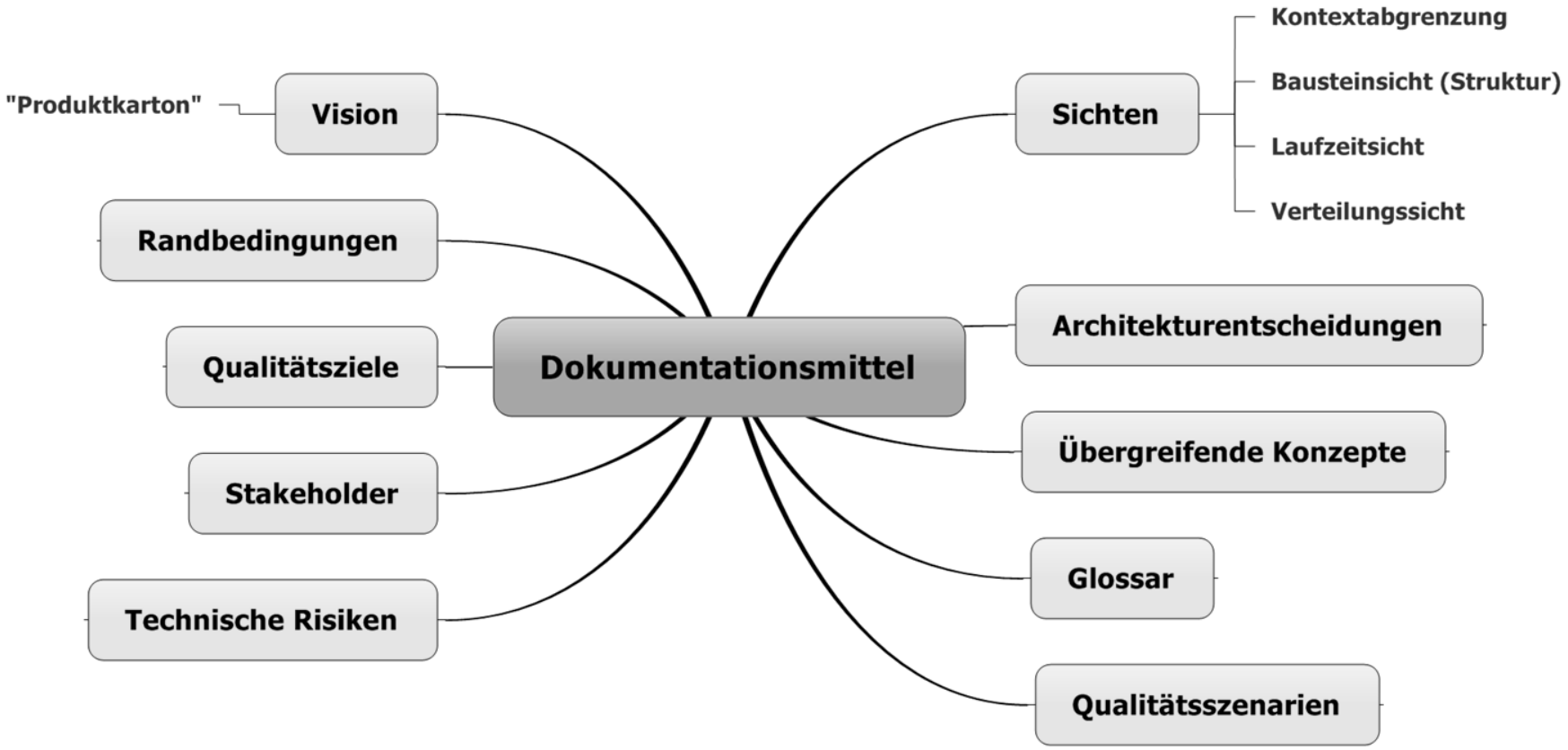


Zuordnung von Zielgruppen zu Inhalten

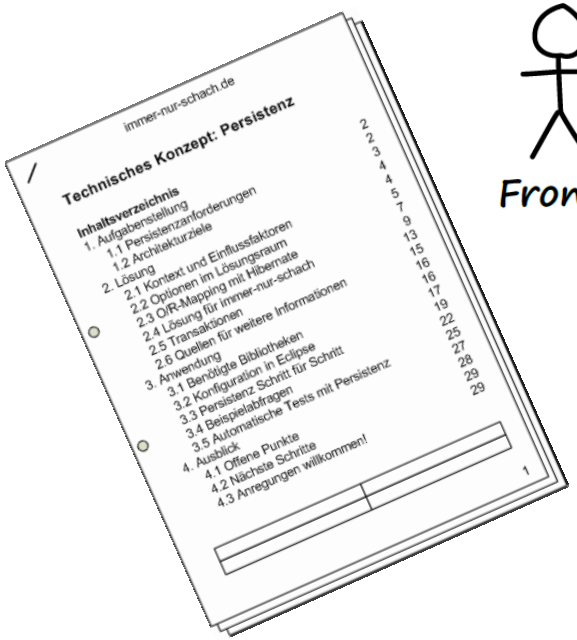
	Inhalt 1	Inhalt 2	Inhalt 3	Inhalt 4	…
Auftraggeber	Überblick	Überblick			
Architekturteam	Überblick	detailliert			
Entwickler		Überblick	detailliert	detailliert	
Tester			Überblick		
…					

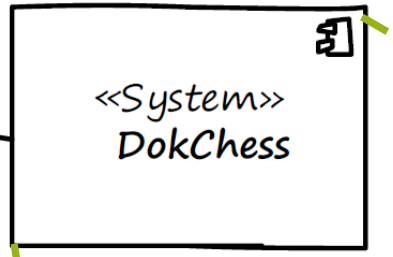
Beispielzielgruppen!

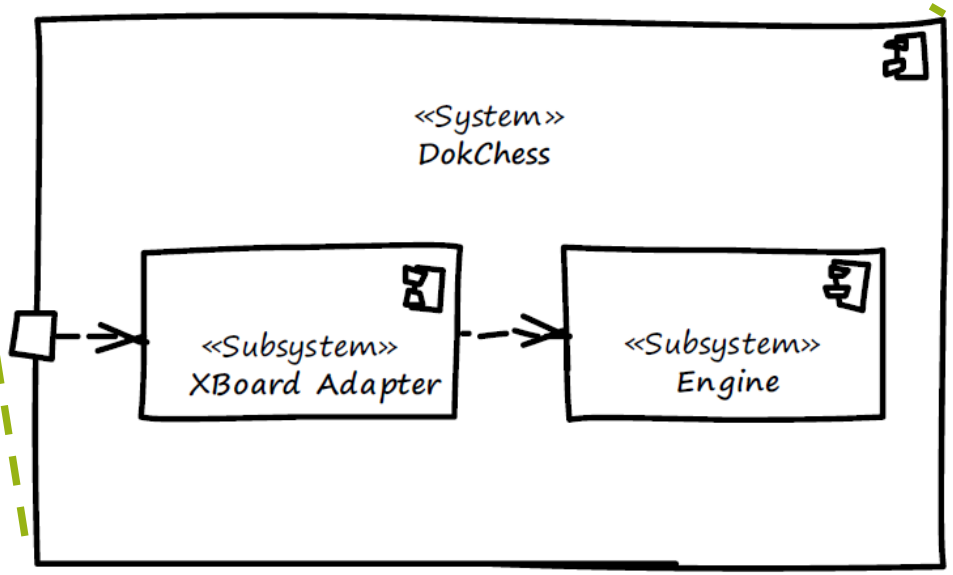
Beispiele für Inhalte („Zutaten“)

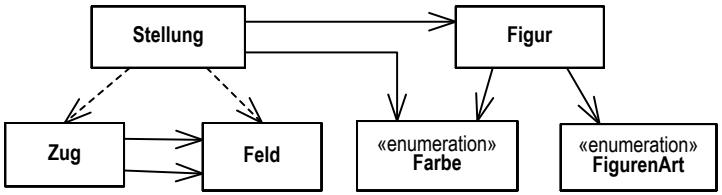


Beispiele für Zutaten









Praxistipp #7 („Fundgrube statt Template“)

Wirklich wirkungsvolle Architekturdokumentation ermöglicht das zielgruppengerechte Rekombinieren von Inhalten.

Die Werkzeugfrage

Tools unterstützen in Architekturdokumentation beim …

- Erstellen (insbesondere im Team) und Pflegen
- Verwalten der Ergebnisse (Fundgrube, Repository)
- Kommunizieren der Inhalte an unterschiedliche Zielgruppen

Konkrete Werkzeuge haben ihre Stärken in einem, maximal 2 der Punkte.

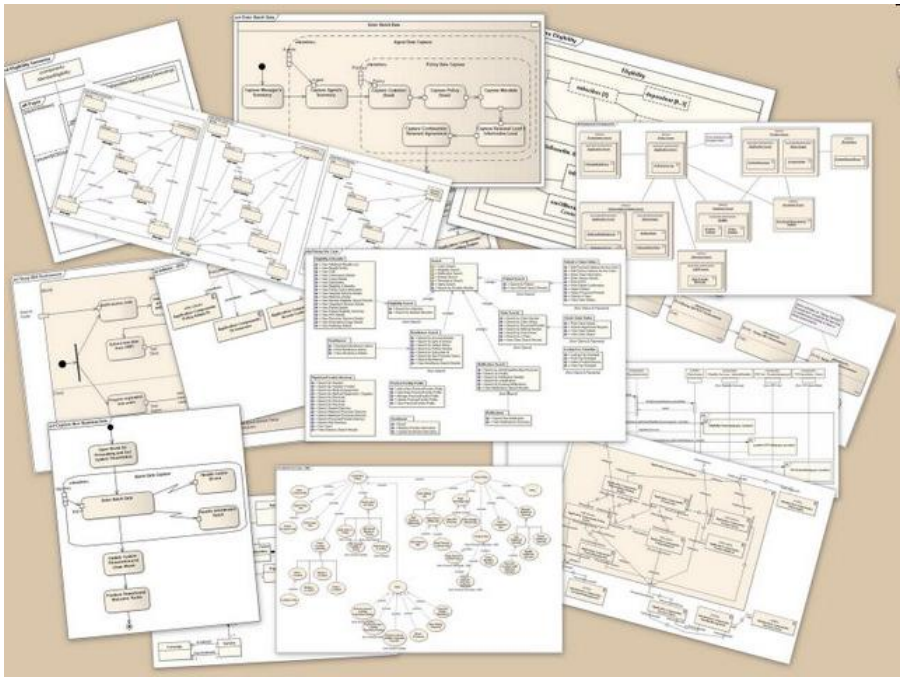
Werkzeugfrage angehen wie eine Architekturentscheidung

- Ziele? Randbedingungen? Risiken?
- Alternativen (Wiki? UML-Tool? …)
- Bewertung z.B. anhand Qualitätsszenarien

UML = Unified Modeling Language

- etablierte, standardisierte Notation im Bereich Software-Engineering
- Primäre Disziplinen:
  - Analyse
  - Entwurf / Architektur
- umfangreich, 14 Diagrammtypen

➔ http://www.uml.org/



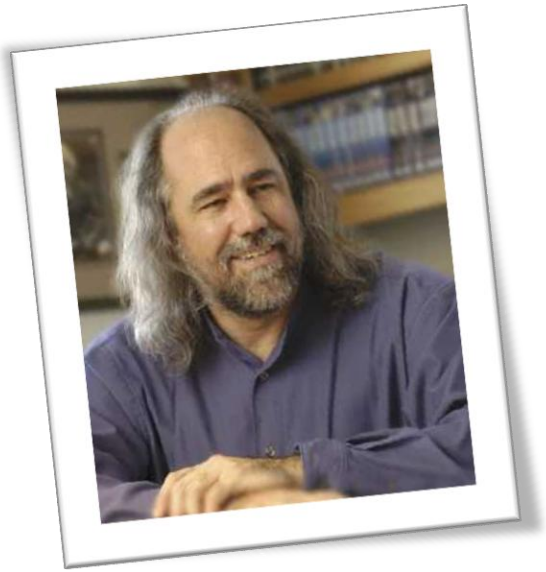
Simplified Chinese

Praxistipp #8 („Simplified UML“)

Verwende nur wenige unterschiedliche Modellelemente in Deinen UML-Diagrammen, die aber korrekt.

Idee hinter „Simplified UML”

- Leser ohne UML-Kenntnisse wird nicht von einer Symbolflut erschlagen
- Leser mit UML-Kenntnissen finden sich auch zurecht
- Sie profitieren trotzdem noch von wichtigen UML-Vorteilen

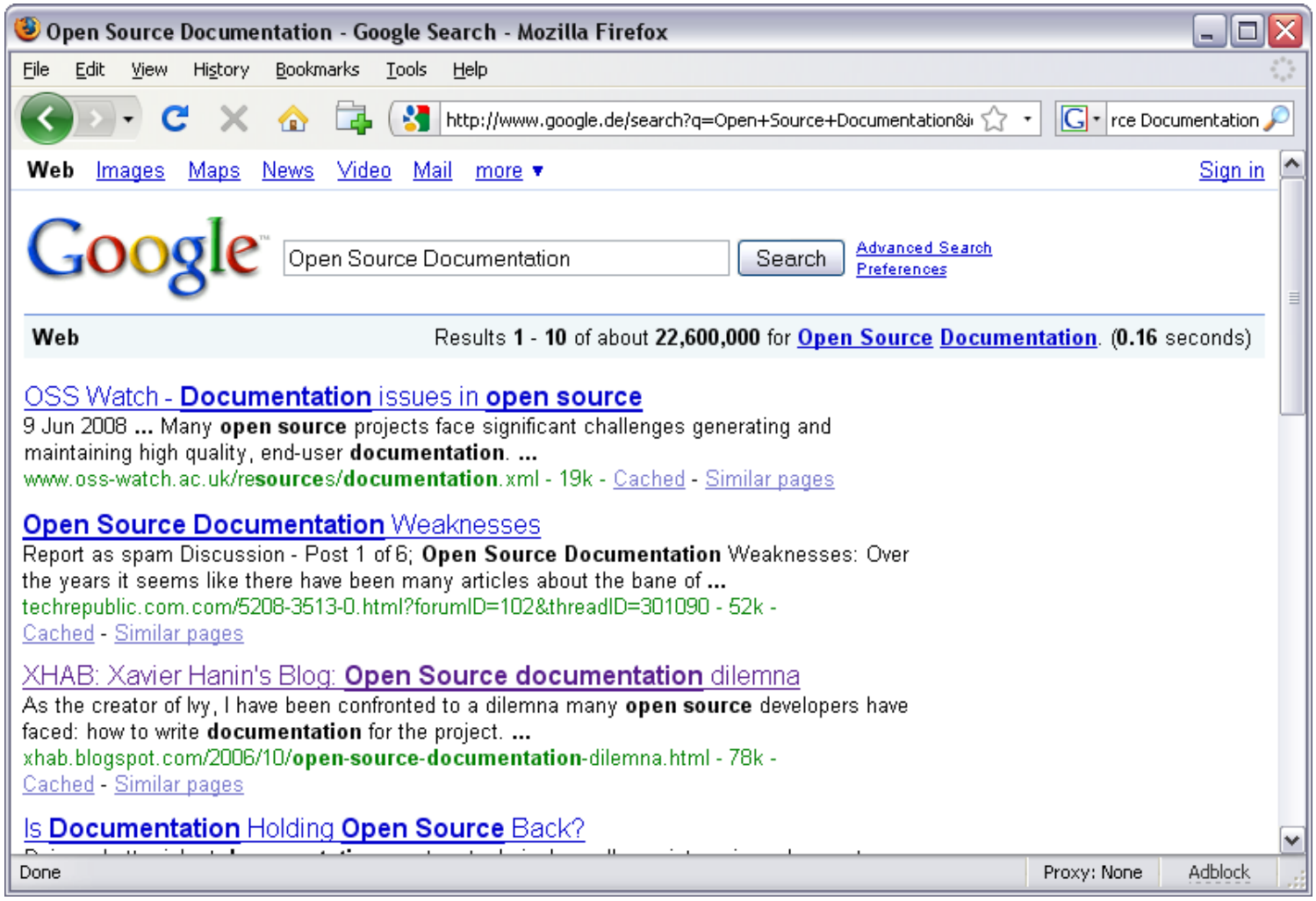


Rückendeckung:

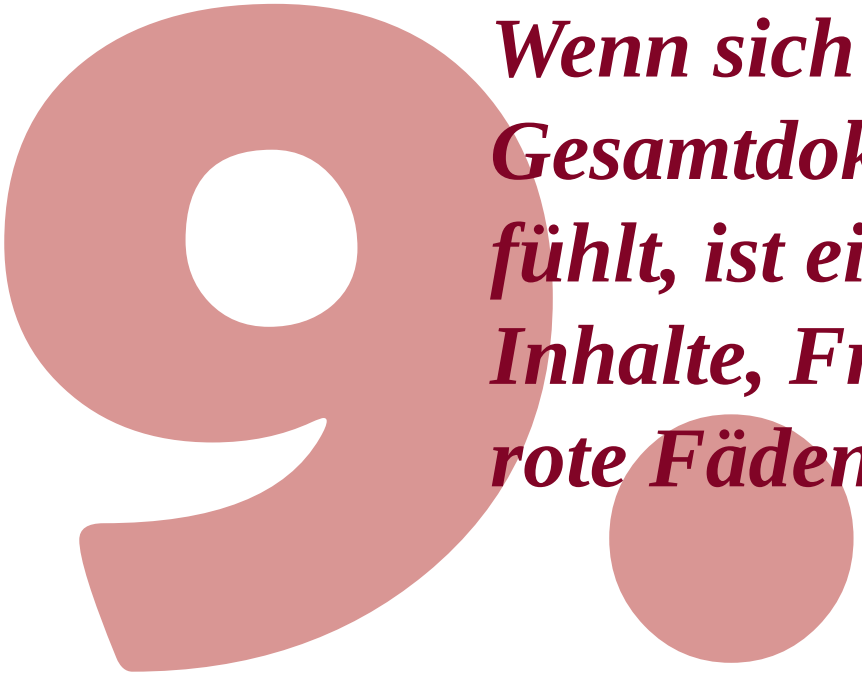
"One needs about 20% of the UML to attend to 80% of most modeling problems. So, there is value in spending energy on what you can remove from the UML rather than what you can add."

(Grady Booch 2011, im persönlichen E-Mail-Austausch)

Why Open Source Sucks …



Praxistipp #9 („Doctator“)



Wenn sich keiner für die Gesamtdokumentation verantwortlich fühlt, ist ein Wiki Garant für veraltete Inhalte, Fragmentierung, fehlende rote Fäden.

- Vorbeugen:
  Schaffen einer Rolle, der die Verantwortung für das Ganze übertragen wird

Niels van Kampenhout, ApacheCon 2009

Rechte und Pflichten des Doctators

- Überblick über die Dokumentation haben und schaffen
- Festlegung der Dokumente, die zu einem Release gehört
- Integration in die Werkzeugkette (Build, Versionierung)
- Unterstützung des Teams (Tools, Stil)
- Mitsprache bei der internen "Abnahme" von Ergebnissen
- Überwachung der Aktualität und Konsistenz der Inhalte

Agenda

5

1 Warum Softwarearchitekturen dokumentieren?

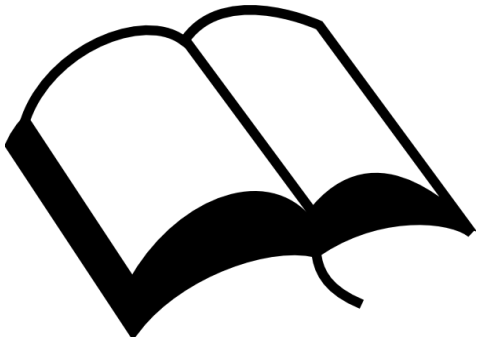
2 Die Aufgabe beschreiben

3 Die Lösung kommunizieren

4 Nicht an den Werkzeugen scheitern

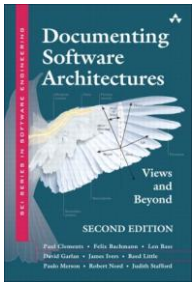
5 Fazit und Weitere Informationen

Sieben Regeln für gute Dokumentation



1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
   3. a) Erkläre Deine Notation
4. Verwende eine Standardstrukturierung
5. Halte Begründungen für Entscheidungen fest
6. Halte Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

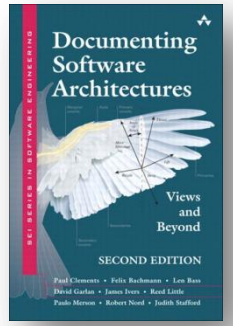
„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010



Mapping unserer bisherigen Tipps auf die 7 Regeln

	1	2	3	4	5	6	7
#1 („Zielsetzung klären“)						X	
#2 („Produktkarton“)	X						
#3 („Kontextabgrenzung“)			X				
#4 („Entscheidungen“)				X	X		
#5 („Sichten“)		X					
#6 („Übergreifende Konzepte“)		X	X				
#7 („Fundgrube“)	X	X					
#8 („Simplified UML“)			X				
#9 („Doctator“)							X

Reviews



Regel 7: „Überprüfe Dokumentation auf Gebrauchstauglichkeit."

(aus „Sieben Regeln für gute Dokumentation“)

Ein Review ist die Begutachtung von Ergebnissen (oder Zwischenergebnissen) durch Personen, die nicht an der Erstellung beteiligt waren. Das können Teammitglieder innerhalb des Vorhabens sein, oder auch Außenstehende.

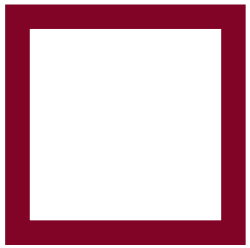
Ziele beim Review

Grundsätzlich können Sie überprüfen, ob eine Dokumentation …

- konform zu bestimmten Standards ist.
- die Arbeit mit der Architektur unterstützt (Gebrauchstauglichkeit).
- eine Bewertung der Architektur unterstützt (Analysierbarkeit, Nachvollziehbarkeit).

> Die Bewertung der Architektur selbst (also eine inhaltliche Überprüfung) ist zugegebenermaßen wichtiger und interessanter. Ein Review der Dokumentation ist im Vorfeld einer Architekturbewertung aber oft sehr wertvoll.

Mögliche Schritte für ein Review

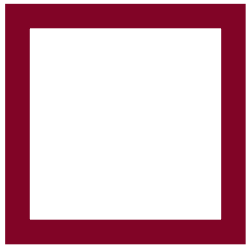


Planung
Review-Ziel(e) und zu prüfende Dokumentation (Umfang) festlegen
Gutachter und andere Beteiligte identifizieren
Termine planen und kommunizieren

Kickoff
Review-Ziele motivieren und kommunizieren
Zu prüfende Dokumentation an Gutachter verteilen
Fragenkataloge und Checklisten bereitstellen

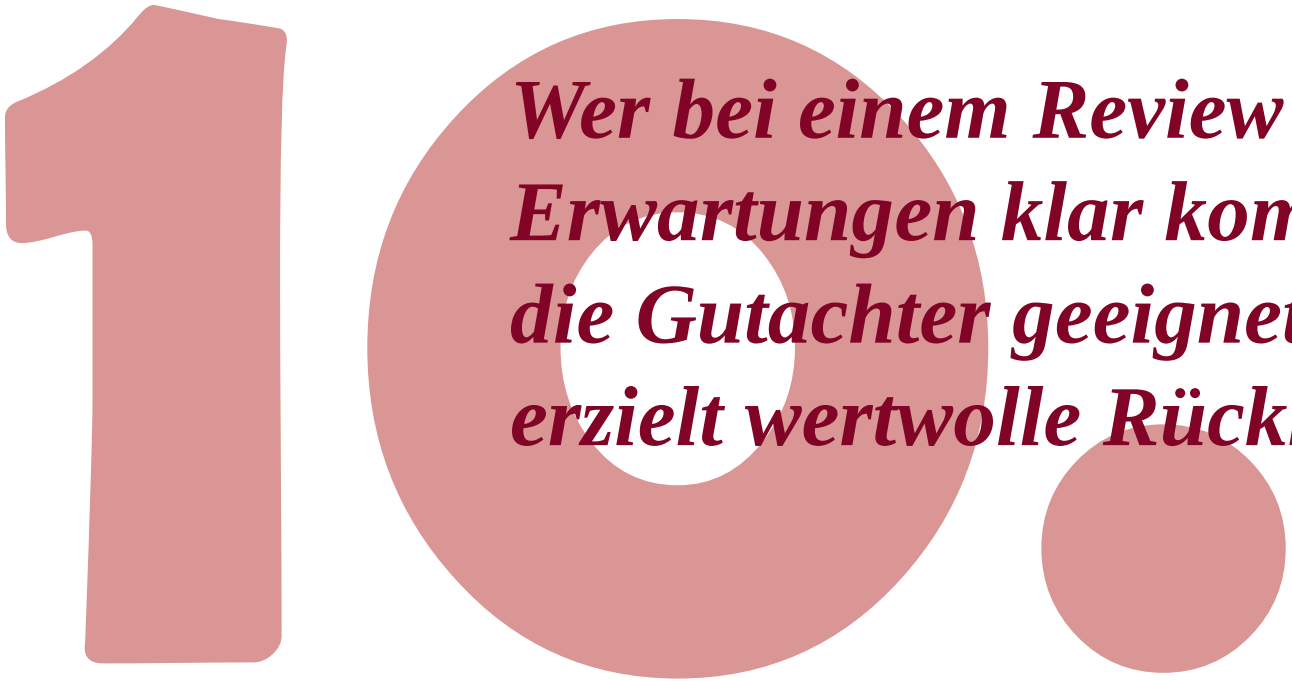
Individuelle Vorbereitung
Gutachter prüfen die Dokumentation
Gutachter halten individuell Stärken, Schwächen, ... fest

Review-Sitzung
Gemeinsam Anmerkungen der Gutachter durchsprechen
Ergebnisse zusammenführen und festhalten
Maßnahmen zur Beseitigung von Mängeln vereinbaren



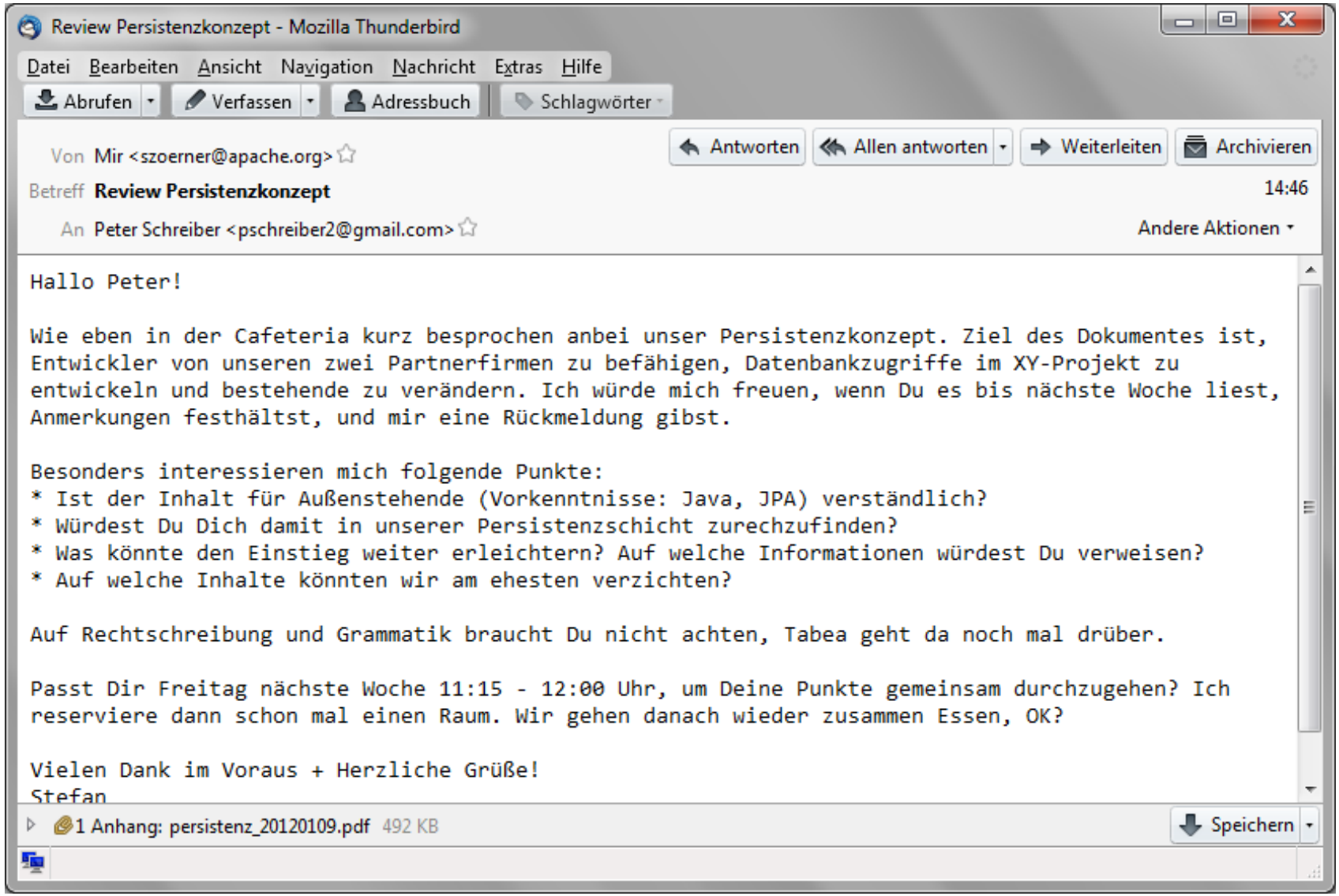
Nachbereitung
Anmerkungen einarbeiten (durch den Autor)

Praxistipp #10 („Reviews planen“)

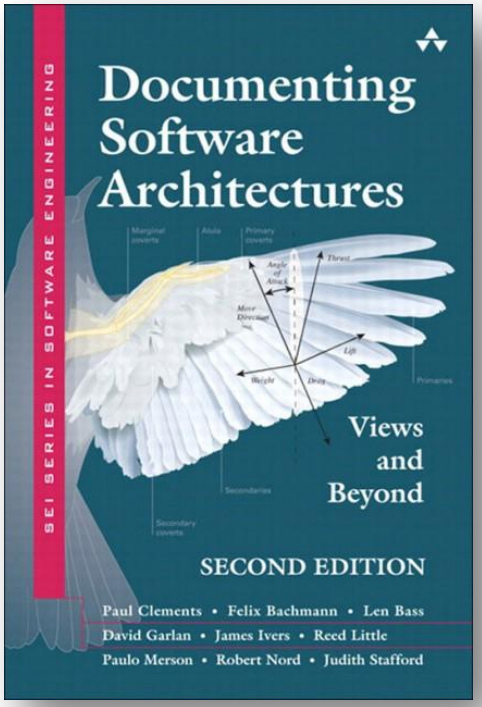


Wer bei einem Review die Erwartungen klar kommuniziert, und die Gutachter geeignet unterstützt, erzielt wertvolle Rückmeldungen.

- Checklisten, Fragenkataloge


Review Persistenzkonzept - Mozilla Thunderbird
Datei Bearbeiten Ansicht Navigation Nachricht Extras Hilfe
Abrufen Verfassen Adressbuch Schlagwörter
Von Mir <szoerner@apache.org>
Antworten Allen antworten Weiterleiten Archivieren
Betreff Review Persistenzkonzept
14:46
An Peter Schreiber <pschreiber2@gmail.com>
Andere Aktionen
Hallo Peter!
Wie eben in der Cafeteria kurz besprochen anbei unser Persistenzkonzept. Ziel des Dokumentes ist, Entwickler von unseren zwei Partnerfirmen zu befähigen, Datenbankzugriffe im XY-Projekt zu entwickeln und bestehende zu verändern. Ich würde mich freuen, wenn Du es bis nächste Woche liest, Anmerkungen festhältst, und mir eine Rückmeldung gibst.
Besonders interessieren mich folgende Punkte:
* Ist der Inhalt für Außenstehende (Vorkenntnisse: Java, JPA) verständlich?
* Würdest Du Dich damit in unserer Persistenzschicht zurechzufinden?
* Was könnte den Einstieg weiter erleichtern? Auf welche Informationen würdest Du verweisen?
* Auf welche Inhalte könnten wir am ehesten verzichten?
Auf Rechtschreibung und Grammatik braucht Du nicht achten, Tabea geht da noch mal drüber.
Passt Dir Freitag nächste Woche 11:15 - 12:00 Uhr, um Deine Punkte gemeinsam durchzugehen? Ich reserviere dann schon mal einen Raum. Wir gehen danach wieder zusammen Essen, OK?
Vielen Dank im Voraus + Herzliche Grüße!
Stefan
1 Anhang: persistenz_20120109.pdf 492 KB
Speichern

Literatur zum Thema (1)



Documenting Software Architectures:
Views and Beyond

Len Bass, Paul Clements, et al.
Addison Wesley, 2. Auflage Oktober 2010
Sprache: English (608 Seiten)
ISBN-10: 0321552687
ISBN-13: 978-0321552686

Literatur zum Thema (2)

Softwarearchitekturen dokumentieren:

Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten

Stefan Zörner

Hanser Fachbuch, Mai 2012

Sprache: Deutsch (ca. 280 Seiten)

ISBN-13: 978-3446429246

Geleitwort von Gernot Starke

Beispiel für einen Architekturüberblick

➔ http://dokchess.de/

Entscheidungen treffen + festhalten

Probekapitel auf

➔ http://www.swadok.de/

3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Ich freue mich auf Eure Fragen!

Stefan Zörner

oose Innovative Informatik GmbH :: sz@oose.de

oose.
Innovative Informatik