

Verunfallte Softwarearchitektur.

Erfolgreiche Lösungen
höchstens per Zufall?



STEFAN ZÖRNER

Berlin Experts Days

Berlin, 04.04.2014



Verunfallte Softwarearchitektur.

Erfolgreiche Lösungen höchstens per Zufall?

Mitunter gelingt ein Entwicklungsvorhaben, und alle sind zufrieden. Oder es scheitert am Ende kläglich. Manchmal auch irgendwas dazwischen. Alles nur Zufall? Der Begriff "Zufällige Architektur" (engl. Accidental Architecture) ist als Anti-Pattern durchaus gebräuchlich, der Ausspruch "Historisch gewachsen" passt ebenfalls prima in diesen Kontext. Wie kann Softwarearchitektur zum Erfolg beitragen? Was genau macht eine gute Architektur aus? Wie erreicht oder erkennt man sie? Müssen am Ende alle glücklich sein? Oder sind Kompromisse sogar zwingend erforderlich? Dieser Vortrag ordnet Projektsituationen zwischen zufälliger und wirkungsvoller Softwarearchitektur ein. Er stellt bewährte Praktiken zum Kurs setzen vor und gibt konkrete Tipps rund um Entwurf und Bewertung für Euer eigenes Vorgehen. Werft Ballast ab und erhöht gleichzeitig die Wirksamkeit der Architekturarbeit in Eurem Projekt!



Stefan Zörner

- Softwareentwickler + -architekt bei embarc in Hamburg
- Vorher oose, IBM, Mummert + Partner, Bayer AG, ...

Schwerpunkte:

- Softwarearchitektur (Entwurf, Bewertung, Dokumentation)
- Java Technologien



✉ Stefan.Zoerner@embarc.de

🐦 [@StefanZoerner](https://twitter.com/StefanZoerner)

✂ [xing.to/szr](https://www.xing.com/profile/stefan_zoerner)

embarc
Software Consulting GmbH



Verunfallte Softwarearchitektur

embarc.de

3

Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen



Verunfallte Softwarearchitektur

embarc.de

4

Agenda

- 1 **Einstieg**
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen

1



Verunfallte Softwarearchitektur

embarc.de

5

Aufgabe (Teil 1)

Erstellt eine Karte für ein Architektur-Quartett!



- Wählt ein **System** aus, das Ihr gut kennt, z.B. Eurer aktuelles Projekt!
- Füllt für dieses System die **Vorlage** aus!
- Wenn Ihr Bezeichnungen und Stärken erfasst habt: Visualisiert das System:
 - Architekturüberblick oder
 - Avatar
- **Zeit: 5 Minuten**



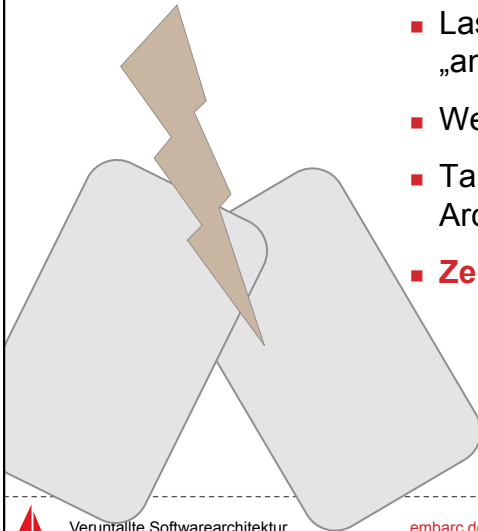
Verunfallte Softwarearchitektur

embarc.de

6

Aufgabe (Teil 2)

Tauscht Euch mit Eurem Nachbarn aus!



- Lasst die Karten gegeneinander „antreten“
- Welches System hat gewonnen?
- Tauscht Euch über die Architektur Eurer Systeme aus!
- **Zeit: 5 Minuten**



Verunfallte Softwarearchitektur

embarc.de

7

Äpfel mit Birnen vergleichen?



VS.



Verunfallte Softwarearchitektur

embarc.de

8

Agenda

- 1 Einstieg
- 2 **(Gute) Softwarearchitektur**
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen

2



Was ist Softwarearchitektur?

The image is a collage centered around the question 'Was ist Softwarearchitektur?'. It features several key elements:

- Books:** 'EFFECTIVE SOFTWARE ARCHITECTUREN' (blue cover), 'Software Architecture in Practice Second Edition' (yellow cover by Len Bass, Paul Clements, Rick Kazman), and 'Künige für Softwarearchitekten' (green cover).
- Diagrams:** A diagram showing '«subsystem» XBoard-Protokoll' and '«subsystem» Engine' connected by a downward arrow.
- Document:** A document titled 'Technisches Konzept: Persistenz' with a table of contents including 'Inhaltsverzeichnis', '1. Aufgabenstellung', '1.1 Persistenzanforderungen', '1.2 Architekturziele', '2. Lösung', '2.1 Kontext und Einflussfaktoren', '2.2 Optionen im Lösungsraum', '2.3 Obj-Mapping mit liberate', '2.4 Lösung für immer-nur-schach', '2.5 Transaktionen', '2.6 Quellen für weitere Informationen', '3. Anwendung', '3.1 Benötigte Bibliotheken', '3.2 Konfiguration in Eclipse', '3.3 Persistenz-Schritt für Schritt', '3.4 Beispielaufträge', '3.5 Automatische Tests mit Persistenz', '4. Anmerkungen', '4.1 Offene Punkte', '4.2 Nächste Schritte', and '4.3 Anregungen willkommen!'.
- Logos:** The 'UNIFIED MODELING LANGUAGE' logo with the letters 'UML'.
- Central Element:** A large, prominent red question mark.



Expertenmeinungen



“... not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change.”

(Grady Booch)

“Softwarearchitecture is about the important stuff, whatever that is.”

(Martin Fowler)

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

(Eoin Woods)



Auf den Punkt gebracht ...

Softwarearchitektur :=

Σ wichtige Entscheidungen

wichtige Entscheidungen :=

- fundamental
- im weiteren Verlauf nur schwer zu ändern



„Verunfallte“ Softwarearchitektur?

„Every interesting software-intensive system has an architecture. While some of these architectures are intentional, most appear to be accidental.“



aus
Grady Booch:
„The Accidental Architecture“
IEEE Software 2006



Verunfallte Softwarearchitektur

embarc.de

13

Was ist gute Softwarearchitektur?



Gute Softwarearchitektur

==

Entscheidungen wurden explizit getroffen.



Nein. Zufällige Architektur muss nicht schlecht sein.

Aber:

- Sie lässt sich schwer vermitteln.
- Sie lässt sich schwer bewerten.



Verunfallte Softwarearchitektur

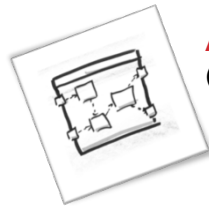
embarc.de

14

Was ist Architekturbewertung?

Architekturrelevante
Anforderungen

Architekturbewertung



Architektur / Entwurf
(Modelle, Entscheidungen)

Umsetzungsüberprüfung

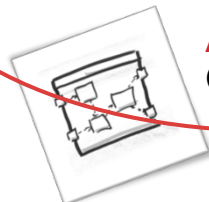
Umsetzung
(Lauffähiger Code)



Was ist Architekturbewertung?

Architekturrelevante
Anforderungen

Architekturbewertung



Architektur / Entwurf
(Modelle, Entscheidungen)

Umsetzungsüberprüfung

Umsetzung
(Lauffähiger Code)



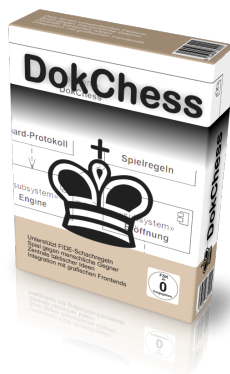
Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 **Qualitätsmerkmale meistern**
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen

3



Beispiel: Ein Schach-Programm



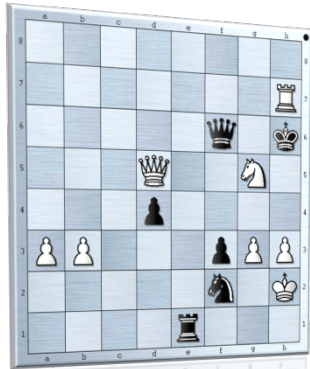
Zentrale Features

- Vollständige Umsetzung der offiziellen FIDE-Schachregeln
- Quelltext lädt zum Experimentieren und zum Erweitern ein
- Spiel gegen menschliche und Computergegner
- Beherrscht taktische Elemente wie Gabel und Spieß
- Anbindung von Eröffnungsbibliotheken



Konkrete Fragestellung

Wie stellt man die Spielsituation als Datenstruktur dar?



- Figuren auf dem Brett
- Spieler am Zug ...



Option 1: 2-dimensionales Array

```
public class ArrayStellung {
    private Figur[][] brett = new Figur[8][8];
    public ArrayStellung(String fen) {}
}
```



Expertenmeinung



“Bei Shredder ist die Effizienz klar im Vordergrund, die Wartbarkeit des Quelltextes muss da leider manchmal hinten anstehen.”

Stefan Meyer-Kahlen (Autor von Shredder)



Qualitätsziele



Die wichtigsten geforderten Qualitätsmerkmale für ein Softwaresystem heißen **Qualitätsziele** (oder Architekturziele).

Typischerweise werden als Qualitätsziele im Rahmen eines Architekturüberblicks die **Top-3** bis **Top-5** genannt.



Beispiel: Qualitätsziele DokChess

Wartbarkeit:

Alternative Algorithmen und Strategien, etwa zur Bewertung einer Schachstellung, können leicht implementiert werden, die Lösung integriert werden.

Effizienz:

Da die Engine in Seminaren und Vorträgen live demonstriert wird, erfolgt die Berechnung der Spielzüge

Attraktivität:

Die Engine spielt stark genug, um schwache Gegner sicher zu schlagen und Gelegenheitsspieler zumindest zu fordern.



Konkretisierung durch Szenarien

Ein Szenario ist ein **Beispiel** für die Verwendung des Systems, bei dem ein Qualitätsmerkmal die **Hauptrolle** spielt.

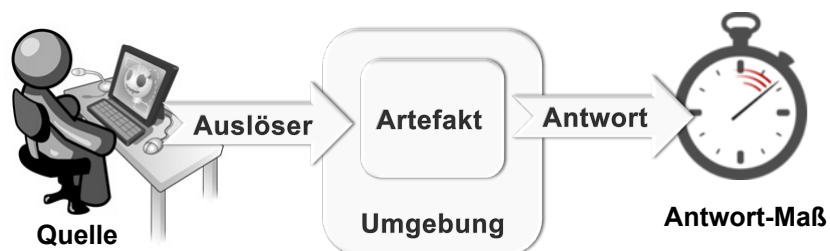
Ein Entwickler implementiert eine figurenzentrierte Bitboard-Repräsentation der Spielsituation. Der Aufwand dazu beträgt inklusive Austauschs der bestehenden Darstellung durch die neue maximal eine Woche.

Während einer Partie antwortet die Engine auf gegnerische Züge innerhalb von 5 Sekunden mit einem Zug.



Konkretisierung durch Szenarien

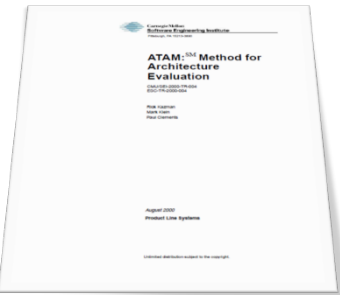
Typische Bestandteile eines Qualitätsszenarios:



In einer Partie ergibt sich für die Engine ein Matt in 3 Zügen. Die Engine zieht sicher zum Sieg.



Bezug zu Architekturbewertung



ATAM

Architecture tradeoff analysis method

- verbreitetste Methode zur Bewertung von Softwarearchitektur
- früh anwendbar
- szenarienbasiert

Qualitätsziele geben einen Hinweis, welche Kriterien auf Euren Quartettkarten wichtig sind.

Szenarien konkretisieren die Ziele und helfen dabei zu bewerten, wie gut sie erreicht werden (bzw. wurden).



Verunfallte Softwarearchitektur

embarc.de

Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen

4



Verunfallte Softwarearchitektur

embarc.de

30

Beispiel: Ein Schach-Programm

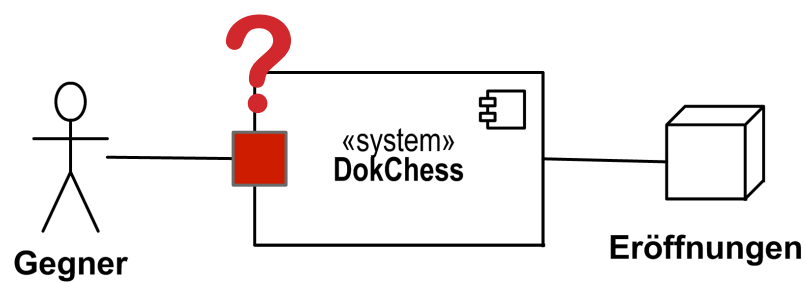


Zentrale Features

- Vollständige Umsetzung der offiziellen FIDE-Schachregeln
- Quelltext lädt zum Experimentieren und zum Erweitern ein
- Spiel gegen menschliche und Computergegner
- Beherrscht taktische Elemente wie Gabel und Spieß
- Anbindung von Eröffnungsbibliotheken



Systemkontext



Protokolle für Schachprogramme

- 2 Lösungen etabliert/dokumentiert
- beide ASCII-basiert, beide via stdin/stdout



Universal Chess Interface (UCI)

<http://wbec-ridderkerk.nl/html/UCIProtocol.html>

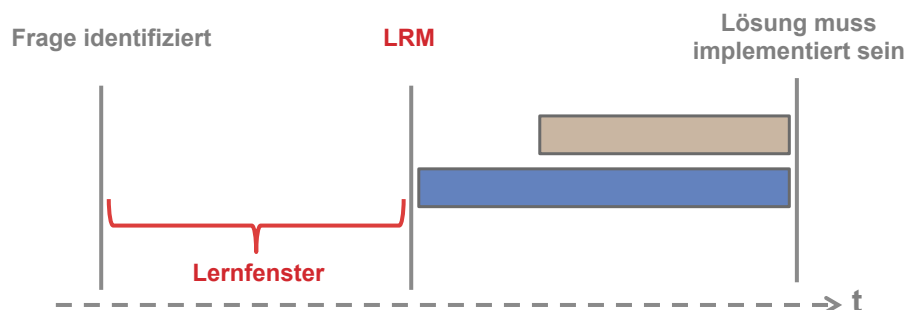
Chess Engine Communication Protocol („Xboard/WinBoard“)

<http://home.hccnet.nl/h.g.muller/engine-intf.html>



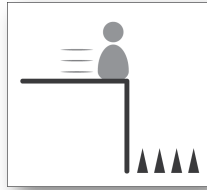
Wann eine Entscheidung treffen?

Ich habe eine Fragestellung, und 2 Optionen



Lese-Tipp

Vorgehensmuster „Der letzte vernünftige Moment“



„Wann sollte eine architekturelle Fragestellung idealerweise entschieden werden, um (1) das Risiko einer Fehlentscheidung zu minimieren und (2) eine optimale Entscheidung für das Projekt zu treffen?“

Vorgehensmuster für Softwarearchitektur.

Kombinierbare Praktiken in Zeiten von Agile und Lean

von Stefan Toth

Verlag: Hanser Fachbuch 2013

Sprache: Deutsch (240 Seiten)

ISBN-13: 978-3446436152



→ <http://www.swamuster.de>

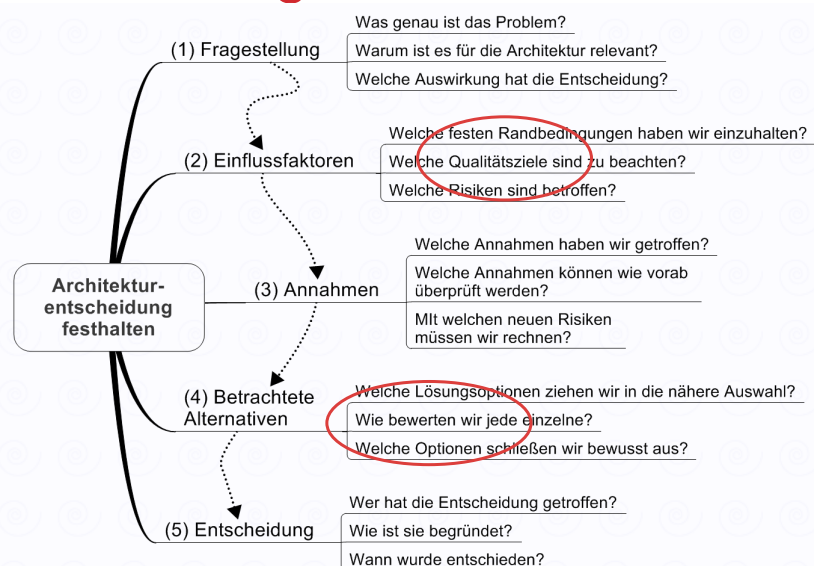


Verunfallte Softwarearchitektur

embarc.de

35

Entscheidungen festhalten



Verunfallte Softwarearchitektur

embarc.de

36

Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten**
- 6 Ein Selbsttest
- 7 Fazit und Weitere Informationen

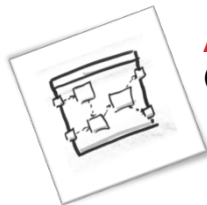
5



Was ist Architekturbewertung?

Architurrelevante
Anforderungen

Architekturbewertung



Architektur / Entwurf
(Modelle, Entscheidungen)

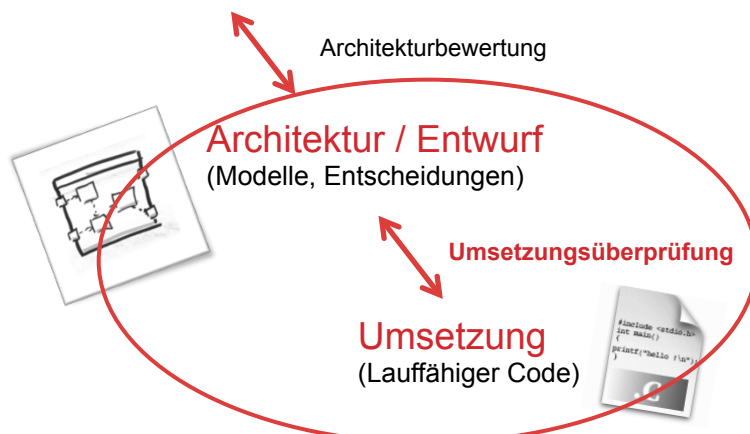
Architekturüberprüfung

Umsetzung
(Lauffähiger Code)



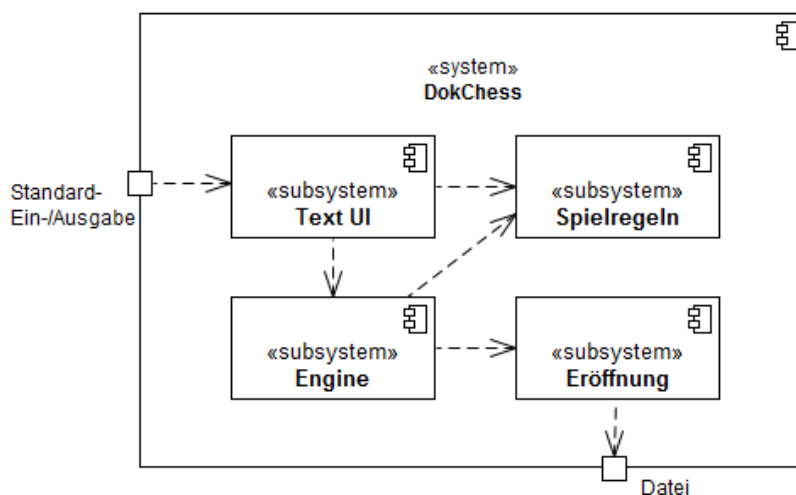
Was ist Umsetzungsüberprüfung?

Architurrelevante
Anforderungen



Beispiel

Zerlegung von DokChess nach Verantwortlichkeiten:



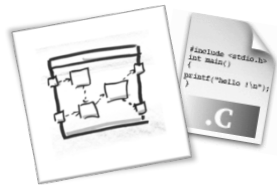
Spannende Fragen ...



Finde ich die Struktur so im Quelltext wieder?

Sind die Verantwortlichkeiten so wie gedacht?

Sind die Abhängigkeiten so, wie dargestellt?



Beispieltool: Sonargraph

Logical Architecture of 'Sonargraph-Demo_Step3' - Sorting elements by definition order, highlighting outgoing restrictions

Component	Common	Contact	Customer	Distribution...	Reque
Controller	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>
Data	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>
Domain	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>
Dsi	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>	<<unrestricted>>

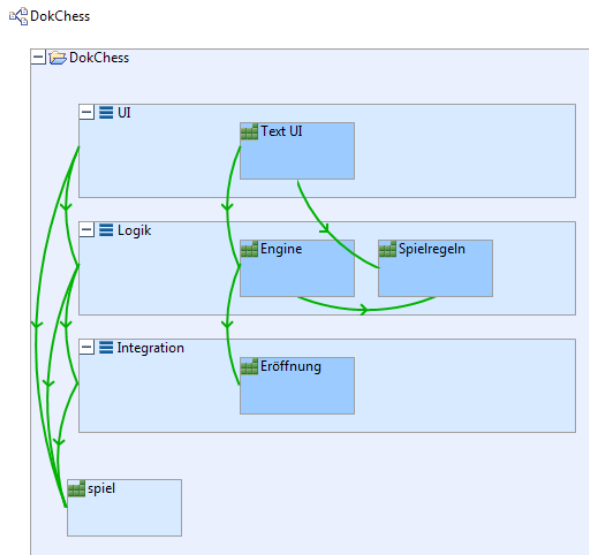
Assignment Patterns | Directly assigned type... | Indirectly assigned by... | Outgoing Dependencies

Outgoing Dependencies of Project 'CRM-Example Project'

Allowed Disallowed | Displayed Target Types...



Logische Architektur definieren



„Verstöße“ aufdecken

Architecture Violations (AV) tool interface showing detected violations.

Violating: 2 of 161 type dependencies (1.24 %) - Ignored: 0

From	To	Number	Architecture	Description
DokChess	DokChess		Logical	
UI	Integration	2		Not defined as allowed dependency
...dokchess.textui	...e.dokchess.eroeffnung	1		
Main	Eroeffnungsbibliothek	1		
...dokchess.textui	...ss.eroeffnung.polyglot	1		

Logical structure of system 'DokChess':

- DokChess
 - Text UI
 - de.dokchess.textui
 - Main
 - Eröffnung
 - de.dokchess.eroeffnung
 - polyglot
 - PolyglotOpeningBook
 - <types in 'eroeffnung'>
 - Eroeffnungsbibliothek

Red arrows highlight violations: `Main` depends on `Eroeffnungsbibliothek` (cross-layer), and `PolyglotOpeningBook` depends on `Eroeffnungsbibliothek` (cross-layer).



Expertenmeinungen



Je stärker die Wirklichkeit von unseren Wünschen abweicht, desto identischer ist sie mit sich.

(Michael Rumpf)

Architekturkonzepte sind edle Wünsche, solange sie nicht in harter Arbeit implementiert, getestet und verändert wurden.

(Stefan Toth)



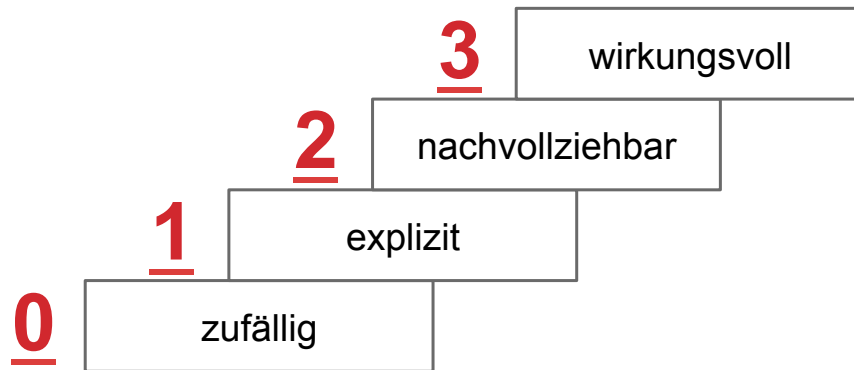
Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Ein Selbsttest**
- 7 Fazit und Weitere Informationen

6



Stufen der Softwarearchitektur



Verunfallte Softwarearchitektur

embarc.de

47

Kommunikation



Wie werden Fragen nach Architekturentscheidungen, -ansätzen, -konzepten beantwortet?

- „Historisch gewachsen ...“
- „Da musst Du Rene fragen ...“
- Architekturansätze sind im Team bekannt und können neuen Mitarbeitern nachvollziehbar erklärt werden.
- Es findet ein Austausch über die Architektur über Projekt-/Teamgrenzen hinweg statt.



Verunfallte Softwarearchitektur

embarc.de

48

Struktur



Wo finde ich die Struktur der Lösung?

- nur im Quelltext
- in Diagrammen / Modellen und konsistent dazu im Quelltext
- die Strukturen (z.B. Schichten, fachliche Zerlegung) sind nachvollziehbar an den Anforderungen ausgerichtet
- Teile/Komponenten werden wo sinnvoll über Projektgrenzen wiederverwendet



Übergreifende Konzepte



Wie wird mit übergreifenden Aspekten und Fragestellungen umgegangen?

- Tendenziell gibt es keine. Die Lösung ist inkonsistent.
- Es gibt einzelne Konzepte, die durchgängig eingehalten werden.
- Die Konzepte sind an den Qualitätszielen und architekturrelevanten Anforderungen ausgerichtet.
- Die Auswahl der Themen für einheitlichen Konzepte ist nachvollziehbar (inkl. bewusstes Weglassen)
- Lösungen für einheitliche Konzepte werden, wo sinnvoll, über Projektgrenzen geteilt.



Stufen der Softwarearchitektur

0 (zufällig)

Softwarearchitektur ist implizit, Erfolg Zufall.

1 (explizit)

Architekturansätze sind explizit vorhanden und benennbar
(Beispiele: Verwendung von Mustern, Entscheidungen für Frameworks)

2 (nachvollziehbar)

Die Lösung ist an Zielen und Anforderungen nachvollziehbar
ausgerichtet. Umsetzung und Architektur entsprechen sich.

3 (wirkungsvoll)

Die Lösung und das Vorgehen der Architekturarbeit sind reflektiert und
angemessen. Die Architektur wirkt über Projektgrenzen hinaus.



Agenda

- 1 Einstieg
- 2 (Gute) Softwarearchitektur
- 3 Qualitätsmerkmale meistern
- 4 Entscheidungen treffen und festhalten
- 5 Die Realität im Auge behalten
- 6 Drei Schritte
- 7 **Fazit und Weitere Informationen**

7



Äpfel mit Birnen vergleichen?



VS.



Verunfallte Softwarearchitektur

embarc.de

53

Äpfel mit Birnen vergleichen?



Toll Collect

VS.



Super Mario Bros.



Verunfallte Softwarearchitektur

embarc.de

54

Fazit

Ich kann auch hier unten gute Lösungen produzieren. Zufällig.

3

wirkungsvoll

Die Vorstellung erfolgreiche Lösungen nur dem Zufall zu überlassen, macht mir Angst.

explizit

Ich investiere lieber in Architekturarbeit, zumindest wenn Scheitern keine Option ist.



Drei Dinge

3

Qualitätsmerkmale meistern

Entscheidungen treffen
und festhalten

Die Realität im Auge behalten



„The Accidental Architecture“

„Accidental architectures are not evil things; indeed, they are inevitable in the growth of systems. It's only when we begin to turn these accidental architectures into intentional ones that we advance our understanding of software architecture.“



Grady Booch:
„The Accidental Architecture“
IEEE Software 2006



Verunfallte Softwarearchitektur

embarc.de

57

Softwarearchitekturen dokumentieren und kommunizieren



Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten

Autor: Stefan Zörner

Umfang: ca. 280 Seiten

Verlag: Carl Hanser Verlag, 2012

Sprache: Deutsch

ISBN: 978-3-446-42924-6

www.swadok.de



Verunfallte Softwarearchitektur

embarc.de

58

Blog-Serie bei Hanser Update

Arc42-Starschnitt | Hanser Update

update.hanser-fachbuch.de/tag/arc42-starschnitt/

HANSERupdate

Praxiswissen aus der IT

Home Online Marketing Webentwicklung Programmierung Softwareentwicklung IT-

arc42-Starschnitt: Gradle. Schnipsel Nr. 6: Grafisches Glossar

OK: Glossar klingt jetzt nicht sooo spannend. Das Inhaltsverzeichnis von arc42 enthält einige Abschnitte, wo auf den ersten Blick sonnenklar erscheint, was sie bergen. Das Glossar zählt wohl dazu. Langweilig? Ich finde nein. Für unser Vorhaben, Gradle nach arc42 gegliedert darzustellen, ist es wertvoll, und kommt eher zu spät. Und als Ergänzung zur "Massischen"

→ update.hanser-fachbuch.de/tag/arc42-starschnitt/

Essay "Beantwortung der Frage: Was ist Aufklärung?" nicht nur eben diesen Begriff. Er klärte in dem Zusammenhang auch noch einige weitere. Hier der Beginn des Aufsatzes ... Aufklärung ist der Ausgang des [...]

Verunfallte Softwarearchitektur

embarc.de

59

Unterstützung gesucht?

→ <http://www.embarc.de>

Workshops - embarc

www.embarc.de/leistungen/workshops/

embarc Software Consulting GmbH

Home Über uns **Unsere Leistung** Ihr Nutzen Blog Kontakt

Teams befähigen Workshops

Unsere Leistung... Teams befähigen

Gleichen Sie Ihr Wissen mit unserem ab und profitieren Sie von unserer Erfahrung! Wie fügt sich Softwarearchitektur in Ihr Vorgehen? Wie halten Sie Entscheidungen und Lösungsansätze nachvollziehbar fest? Wie bewerten oder dokumentieren Sie entstehende oder bereits bestehende Softwarelösungen? Wie adressieren Sie Unternehmensarchitektur pragmatisch aber gewinnbringend?

Idee: Fokussiert ein Thema bearbeiten und Können aufbauen

Dauer: 2-5 Tage

Transferleistung: 6/10 - Soweit möglich arbeiten wir an Ihrem eigenen Beispiel

Vielleicht interessant: In unseren Workshops lassen sich auch Credit Points für den ISAQB Certified Professional for Software Architecture (CPSA) sammeln

Beispielthemen:

- Architekturbewertung
- arc42

Verunfallte Softw

Vielen Dank.

Ich freue mich auf Eure Fragen!

✉ stefan.zoerner@embarc.de

t @StefanZoerner

✕ xing.to/szr



DOWNLOAD FOLIEN:
<http://www.embarc.de/blog/>

