Grundlagen von Software-Architektur

Stand: April 2020

Ziele und Inhalt

Sie lernen:

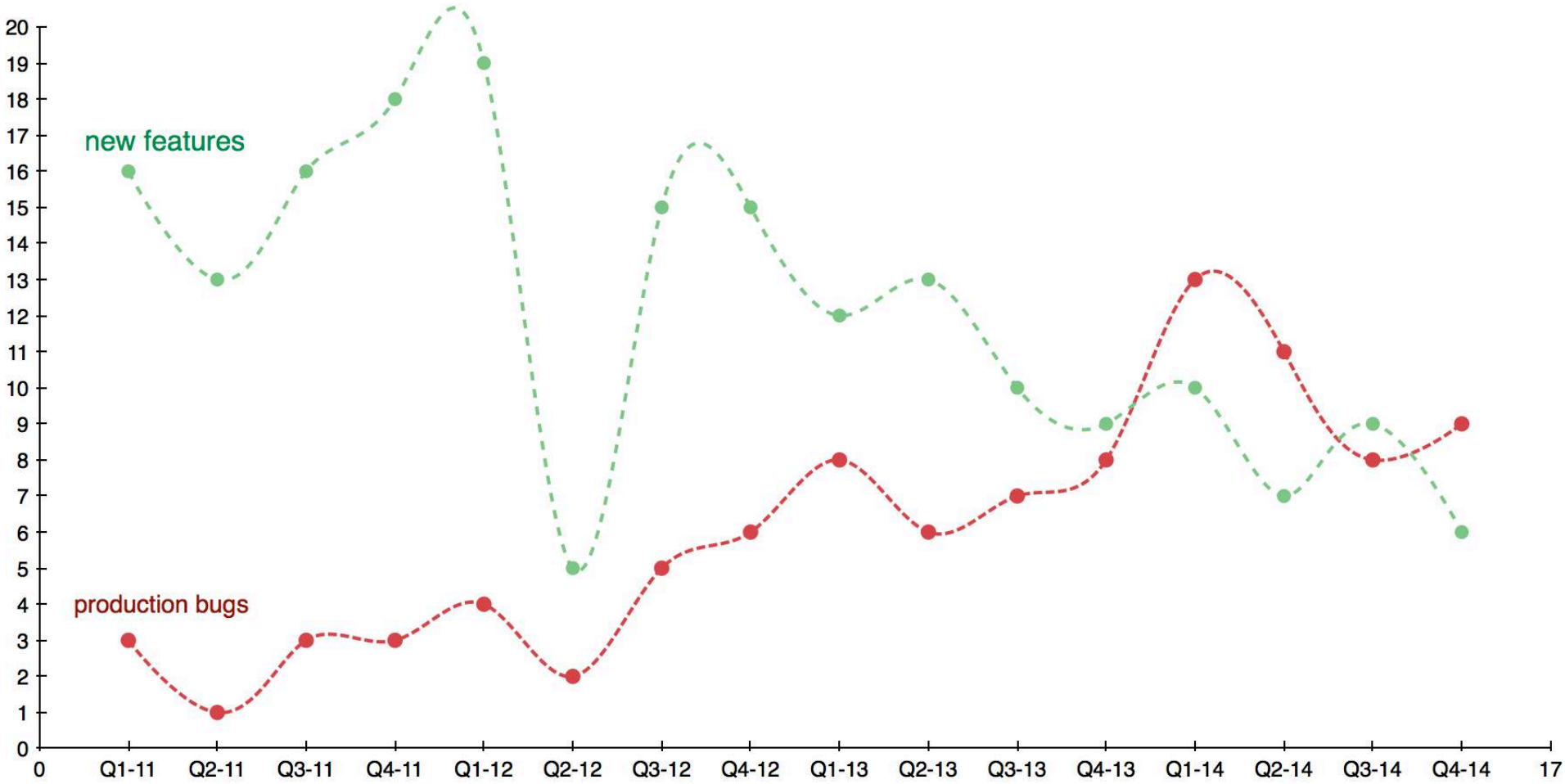
- Was ist Architektur?
- Einordnung in den Entwicklungsprozess
- Was leisten Architekten?
- Überblick über Sichten
- Überblick arc42

Sie diskutieren das Berufsbild des Software-Architekten in Ihrer Organisation

Lernziele gemäß iSAQB CPSA-F:

- **LZ 1-1: Definitionen von Softwarearchitektur diskutieren (R1)**
- **LZ 1-2: Ziele von Softwarearchitektur verstehen und herausstellen (R1)**
- LZ 1-3: Softwarearchitektur in Software-Lebenszyklus einordnen (R2)
- **LZ 1-4: Aufgaben und Verantwortung von Softwarearchitekten verstehen (R1)**
- **LZ 1-5: Rolle von Softwarearchitekten in Beziehung zu anderen Stakeholdern setzen (R1)**
- **LZ 1-6: Zusammenhang zwischen Entwicklungsvorgehen und Softwarearchitektur erläutern können (R1)**
- LZ 1-9: Zuständigkeit von Softwarearchitekten in organisatorischen Kontext einordnen (R3)
- LZ 1-10: Typen von IT-Systemen unterscheiden (R3)

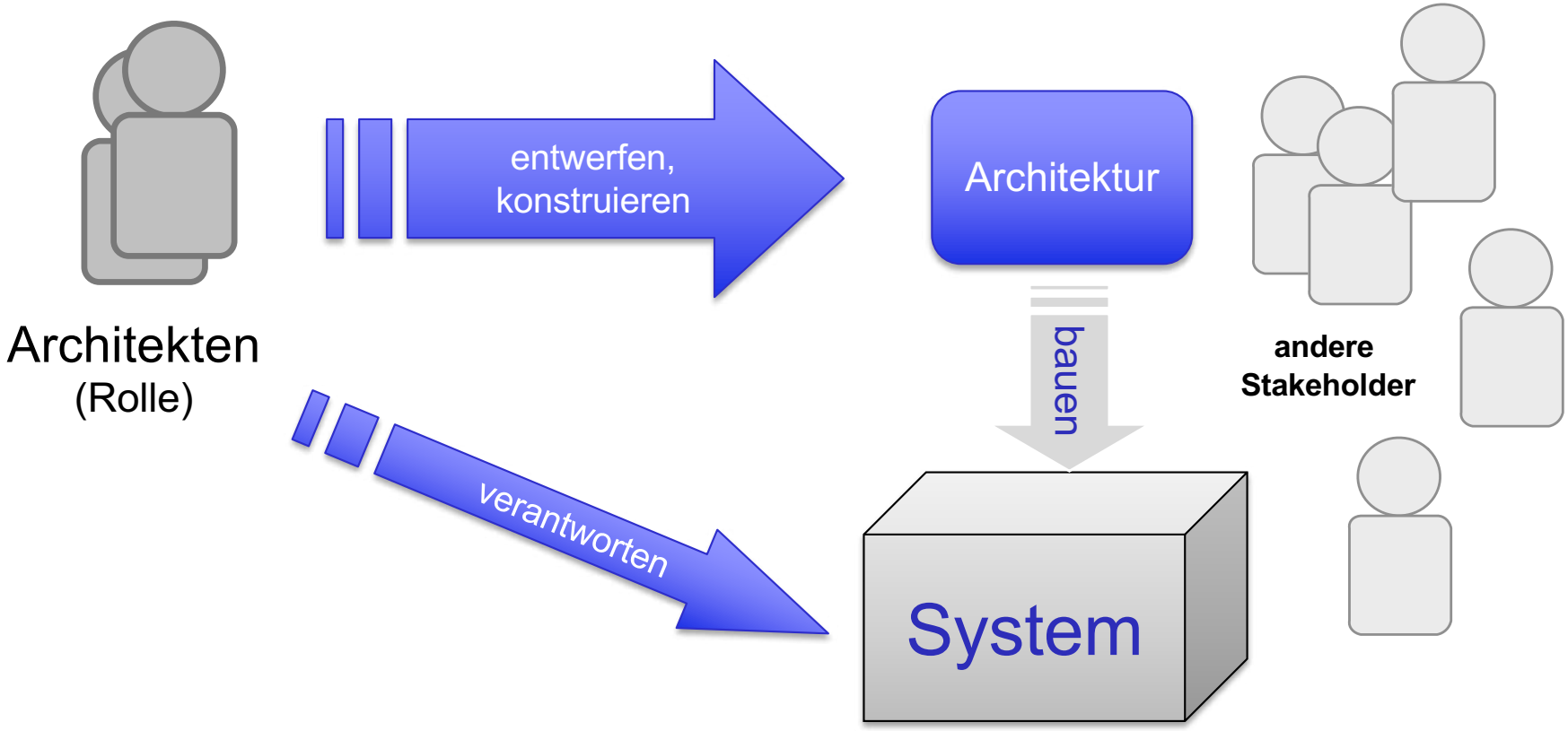
Darum Softwarearchitektur



Softwarearchitekten

- Sorgen für „**gute**“ Systeme:
  - im Sinne der maßgeblichen Stakeholder
  - verfolgen (auch) langfristige (Qualitäts) Ziele
  - sichern Konsistenz, Homogenität („konzeptionelle Integrität“)!
- Fokus auf
  - Erstellung, Evolution, Weiterentwicklung (Wirk)Betrieb

Architektur und System

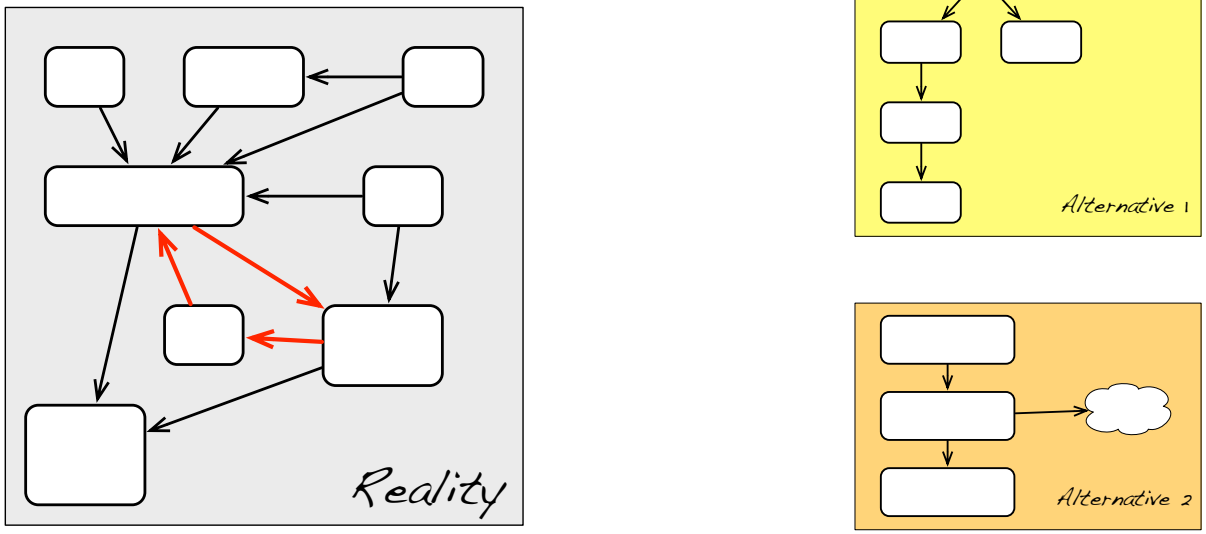


▲ die Rollen ▲ die Aktivitäten ▲ das Ergebnis

Softwarearchitektur

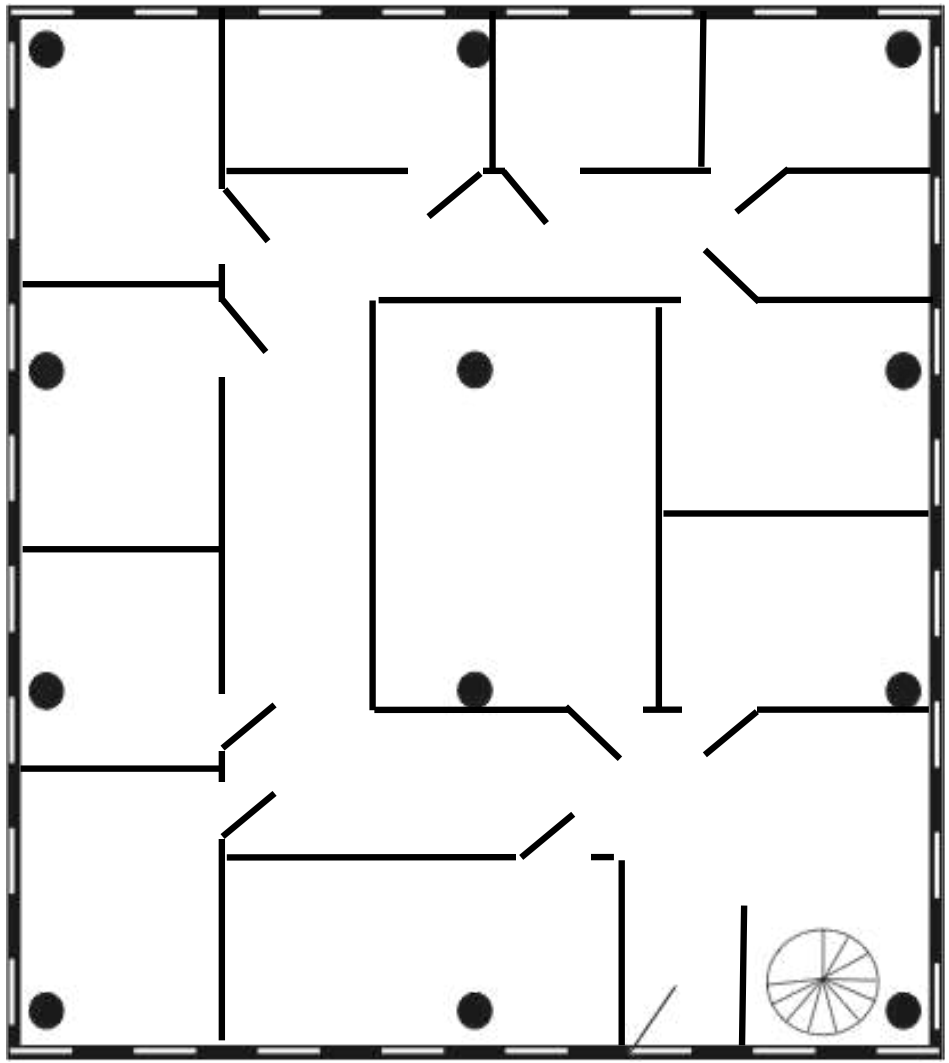
- Aufbau/Organisation eines Systems
  - Strukturen, Komponenten, Beziehungen, Konzepte, Entscheidungen etc.
- Sichert Erfüllbarkeit der **Qualitätsanforderungen**
  - Wartbar, erweiterbar, verständlich
  - Performant, sicher, robust
- Festlegungen
- Freiheitsgrade

Zeitliche Nutzung:
1. Soll („Plan“)
2. Ist („Doku“)



Das Ergebnis: Die Architektur!

- Genügend Festlegungen für Stabilität
- Genügend Freiheitsgrade für kreative Evolution



Es gibt viele Definitionen von Softwarearchitektur

„... the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time“

Garlan & Perry

„... the logical and physical structure of a system, forged by all strategic and tactical design decisions applied during development”

Grady Booch

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

Eoin Woods

„... a framework for change“

Tom DeMarco

„... die Struktur des Systems (oder die Strukturen). Sie besteht aus Komponenten, den nach außen sichtbaren Eigenschaften dieser Komponenten und den Beziehungen untereinander.

Bass, Clements, Kazman

Ein Auszug aus Definitionen der Literatur:
https://resources.sei.cmu.edu/asset_files/FactSheet/2010_010_001_513810.pdf

Definition Softwarearchitektur: IEEE-1471

„Die grundsätzliche Organisation eines Systems, wie sie sich in dessen Komponenten, deren Beziehung zueinander und zur Umgebung widerspiegelt, sowie die Prinzipien, die für seinen Entwurf und seine Evolution gelten.“

Gemeinsamkeiten der Definitionen

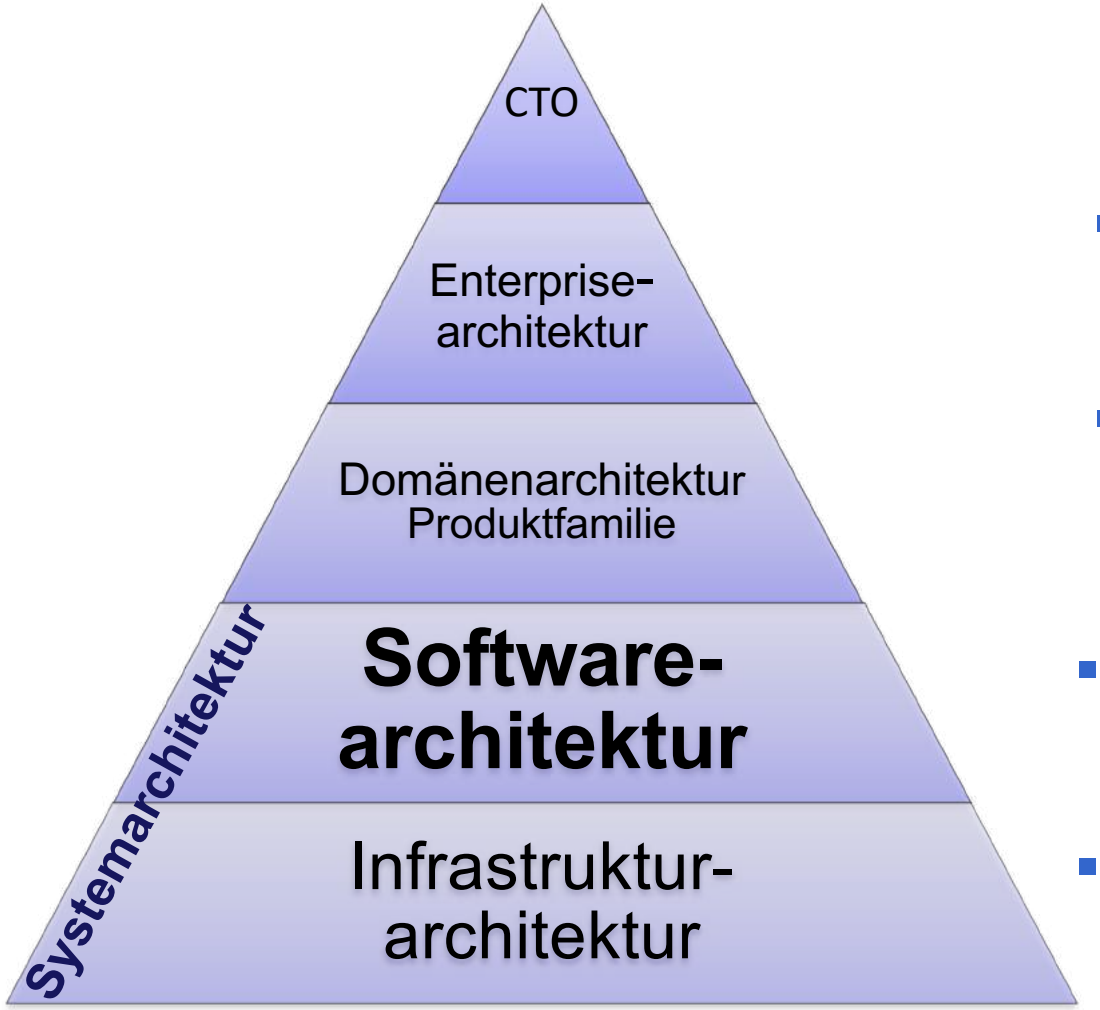
- Zerlegung des Problems in kleinere Einheiten (verständlich, beherrschbar)
- Struktur von Komponenten, deren Beziehungen zueinander und zur Umgebung
- Entwurfsentscheidungen
- Grundsätze und Konzepte (Prinzipien), die Entwicklung und Evolution des Systems bestimmen

Beachte Angemessenheit

- Architektur muss „passen“
- Angemessenheit hinsichtlich Problem (Teamgrösse, Komplexität, Kritikalität)

Ein Problem
viele mögliche Lösungen

Vorsicht: Viele Rollen ***architekt**



- Enterprisearchitektur
  - Struktur aller Anwendungen eines Unternehmens
- Domänenarchitektur
  - Alle Applikationen einer Abteilung oder eine Produktfamilie
- Softwarearchitektur:
  - Strukturen & Konzepte einzelner Software-Systeme
- Infrastruktur-Architektur:
  - Hardware, Betriebssysteme

Architekt als Multilinguist und Dolmetscher

Projektleitung, Management

Risikofaktoren, Budget

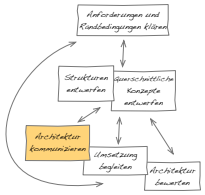
Fach- / Businessexperten

Anforderungen Einschränkungen

Komponenten, Schnittstellen, Konzepte

Entwickler

Requirements-Engineer



Vorgehen, Backlog

Product-Owner

Architekt (Rolle)

Komponenten, Risikofaktoren

QS/Test

Technische Infrastruktur, Hardware, Einschränkungen

Security, Datenschutz

Anforderungen

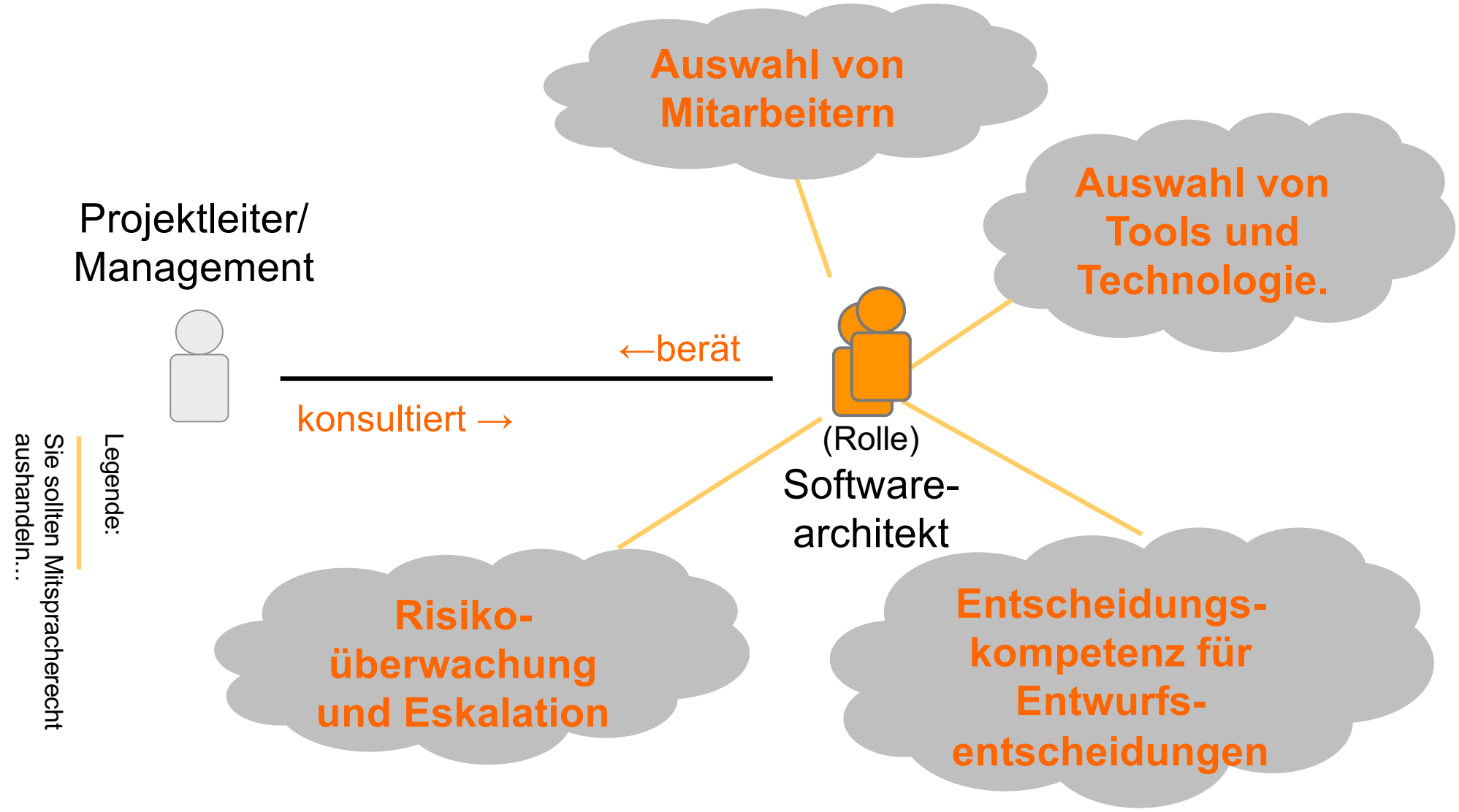
Corporate Security

Benutzer

<andere>

Betrieb, Hardwareentwicklung

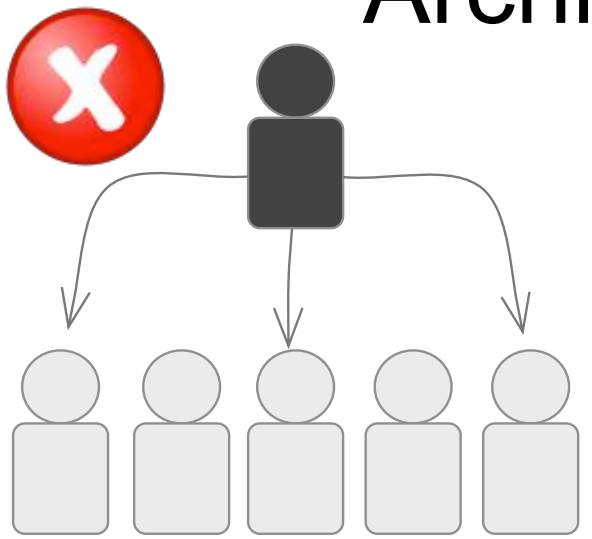
Management und Architekten



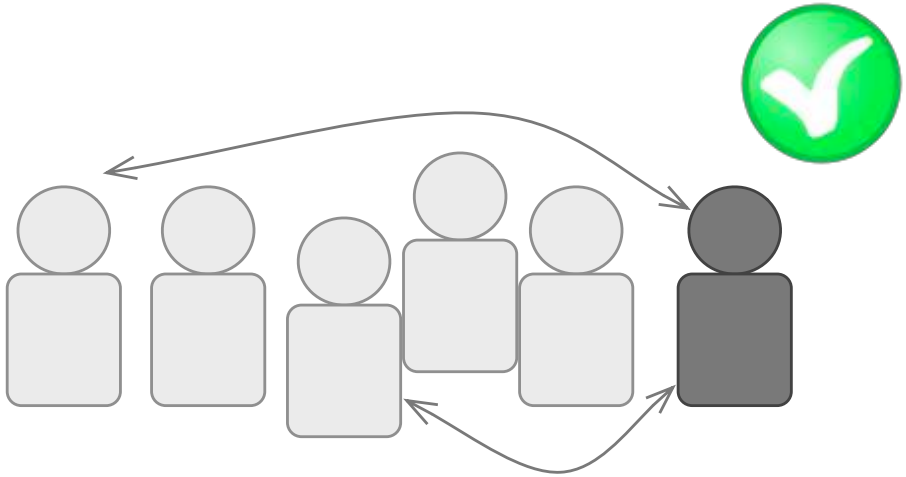
Management und Architektur ff.
(Conways Law...)

Team- oder Organisationsstruktur ist
(oft) kongruent zur Architektur

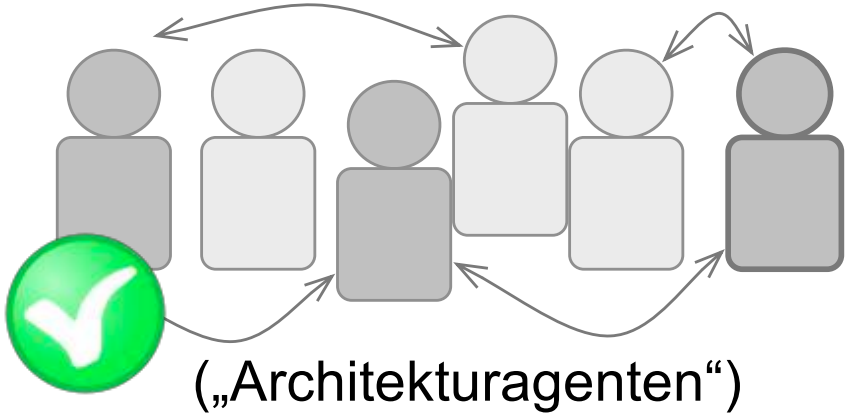
Architekten und Entwicklungsteam



Klassisch: Architekt bestimmt
von außerhalb des Teams

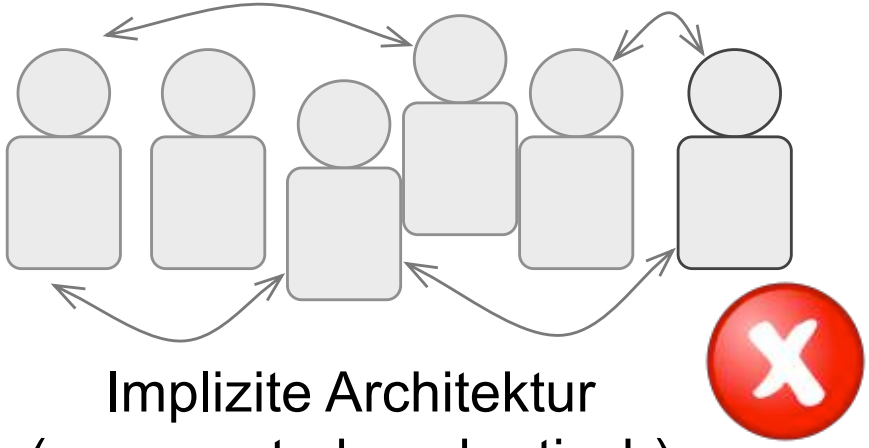


Architekt im Team



(„Architekturagenten“)
Architekturaufgaben im Team verteilt

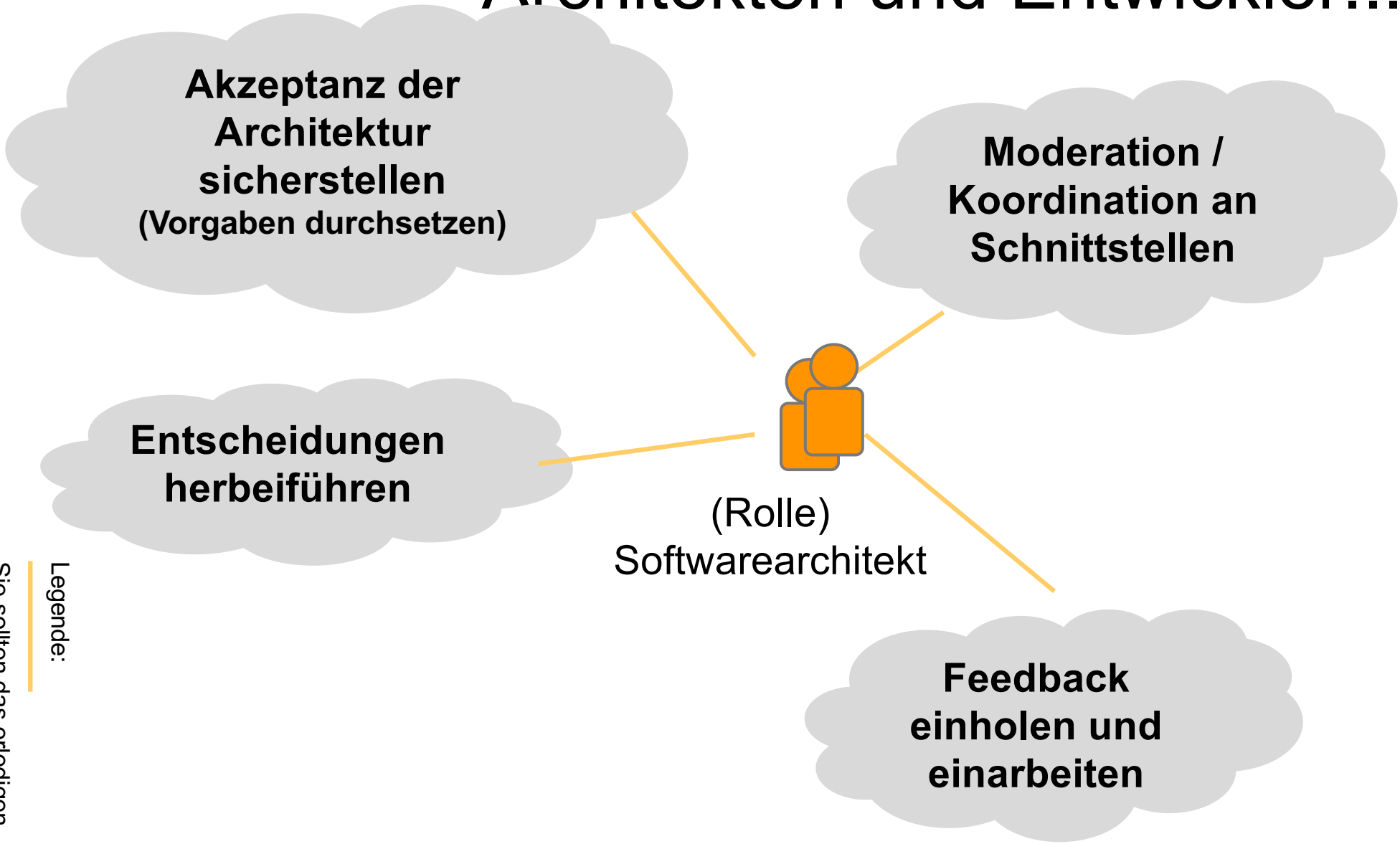
Architekturaufgaben im Team verteilt



Implizite Architektur
(emergent, demokratisch)

Nach Ideen von S. Toth: Vorgehensmuster für Software-Architektur, Hanser 2014

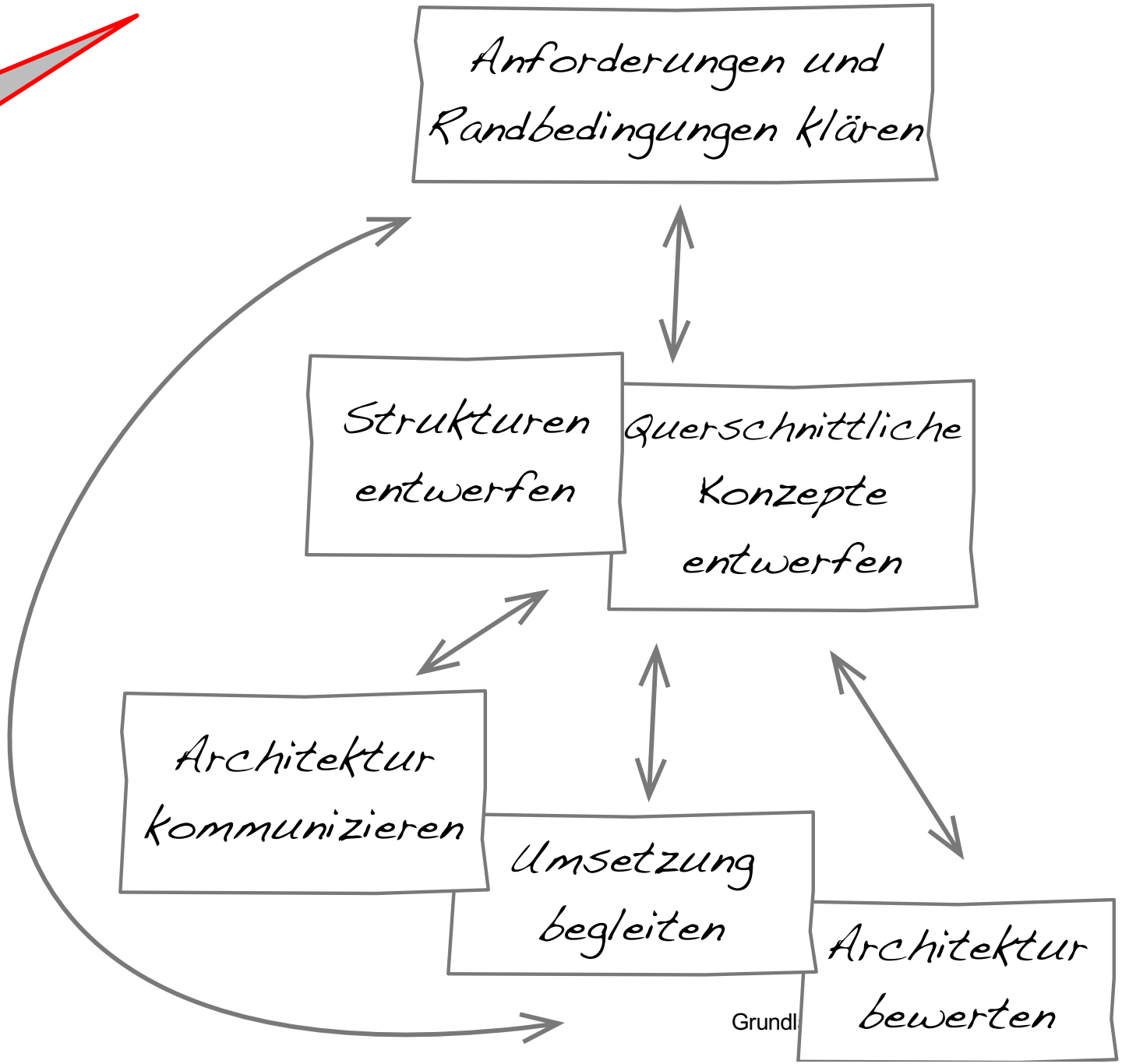
Architekten und Entwickler...



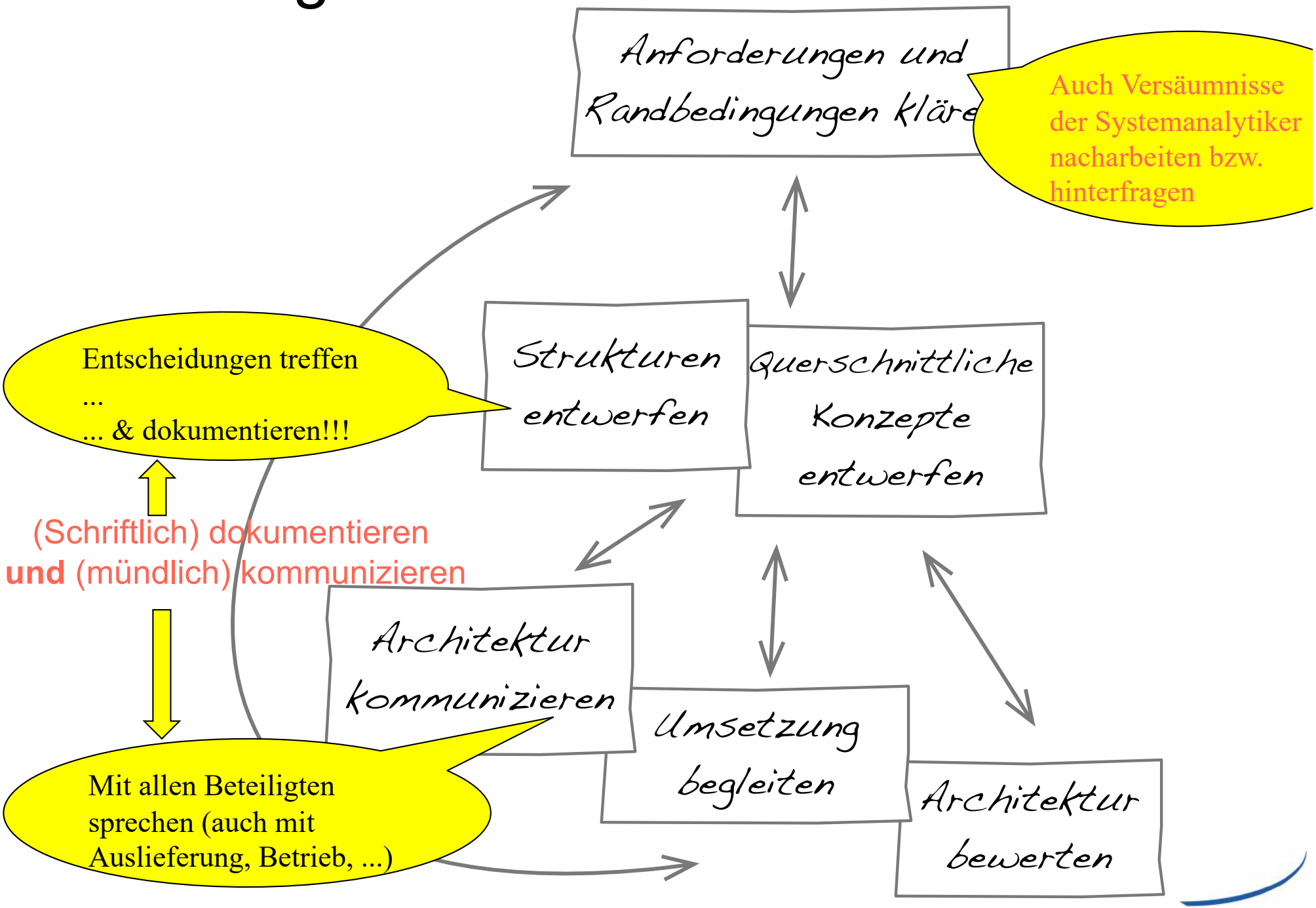
Legende:
Sie sollten das erledigen..

Tätigkeiten von Softwarearchitekten...

Reihenfolge
und Gewichtung
offen!



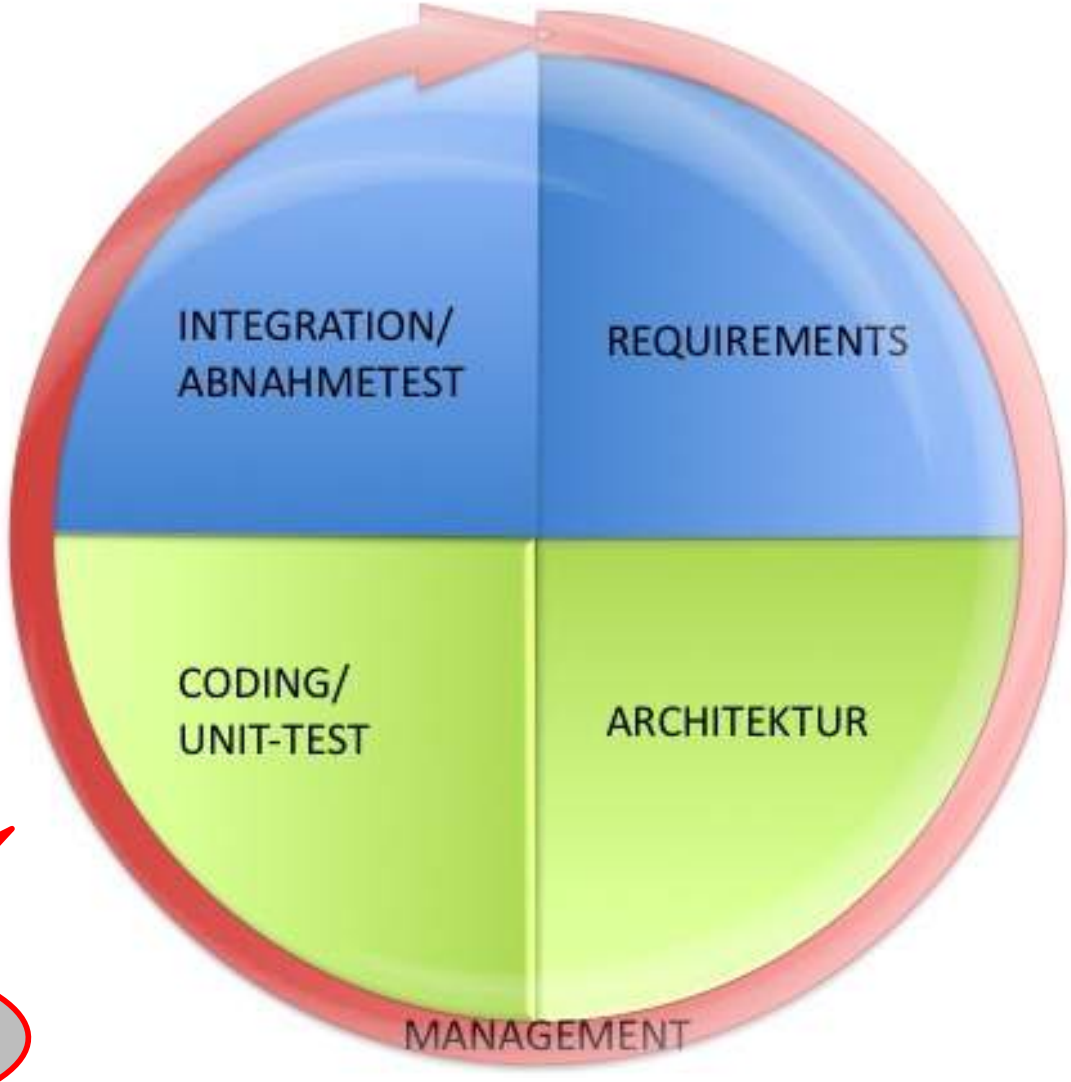
Tätigkeiten von Softwarearchitekten...



Architekten: Die Zehnkämpfer der IT

- entwerfen
  - Systemstrukturen
  - Schnittstellen
  - Komponenten
- entscheiden
  - technische Grundsatz- und Detailthemen
  - Entscheidungsfähigkeit unter Risiko
- dokumentieren
  - Strukturen & Entscheidungen
  - „passend“ für Stakeholder
  - arc42 liefert ein Template
- kommunizieren
  - argumentieren
  - überzeugen
  - „vermarkten“
- vereinfachen
  - Komplexität erkennen
  - Komplexität reduzieren
- implementieren
  - Bei Bedarf!
  - Prototypen
  - Referenzkomponenten
- beweisen / garantieren
  - Machbarkeit
  - Erfüllung von Anforderungen
- bewerten
  - Architekturen
  - Aufwände
  - Zulieferungen & Fertigteile
- beraten
  - Projektleiter
  - Entwicklungsteam
  - Management
- balancieren
  - konkurrierende Faktoren

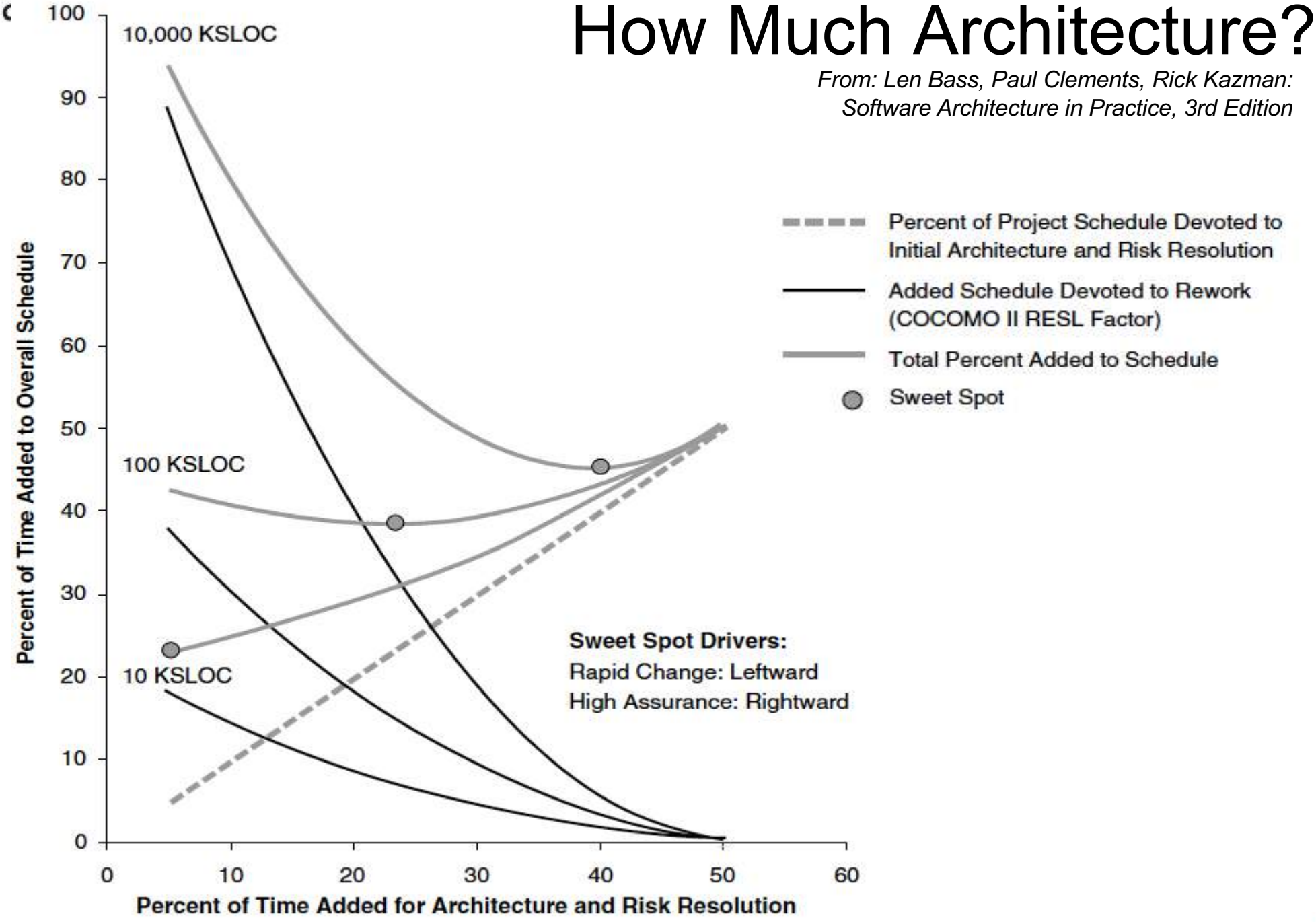
Gewichtung der Architektur in der Gesamtentwicklung



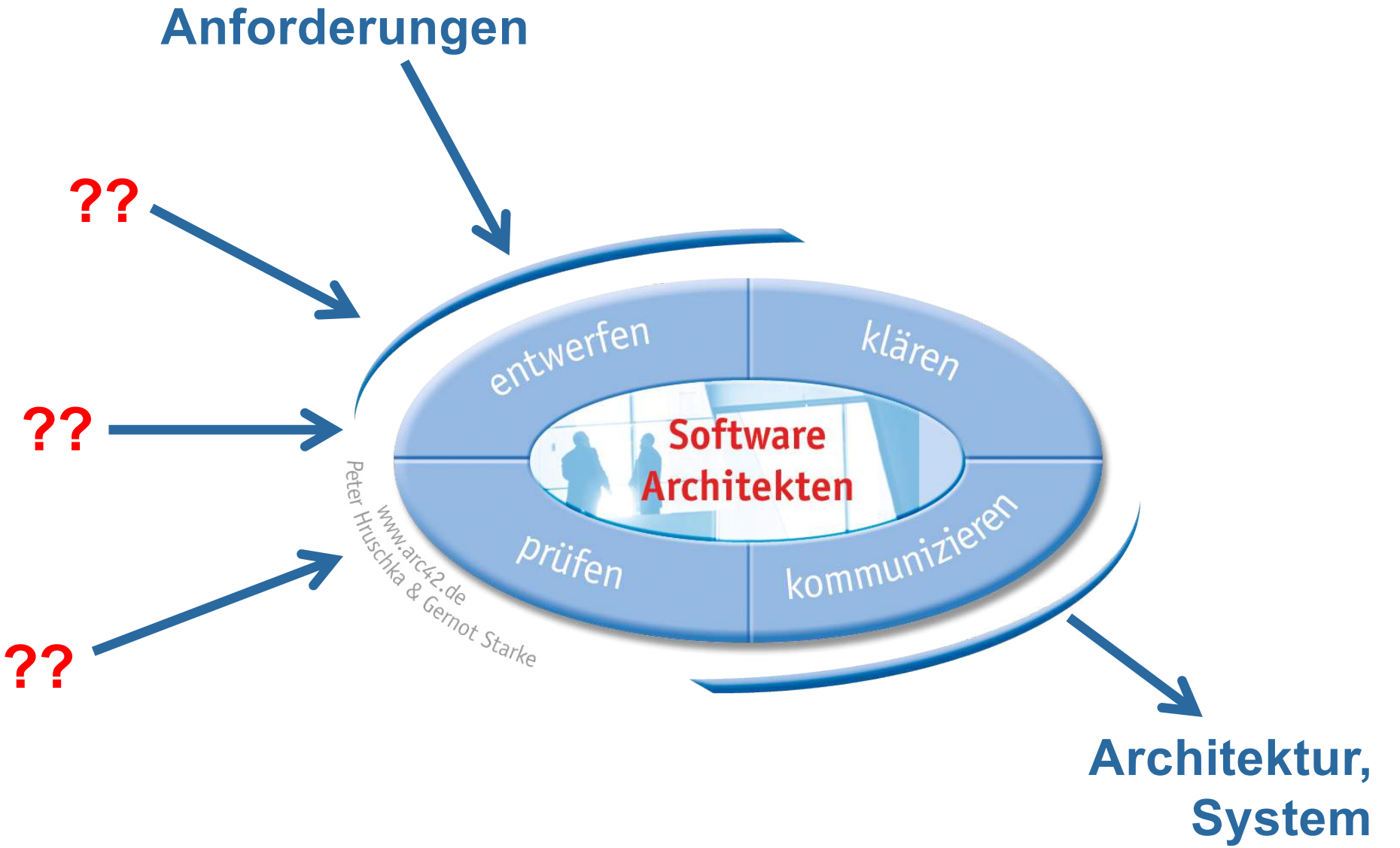
Keine zeitliche Reihenfolge!!

How Much Architecture?

From: Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice, 3rd Edition

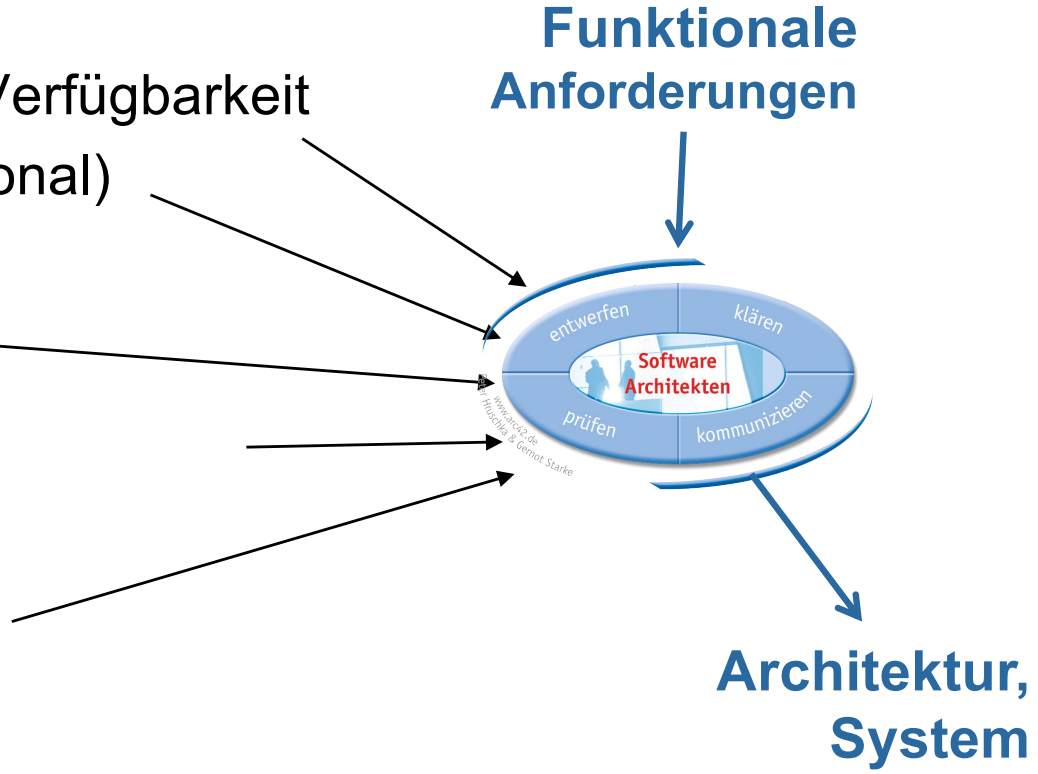


Architekten entwerfen...



... entwerfen (ff)

- Technologie und deren Verfügbarkeit
- Budget (Zeit, Geld, Personal)
- Organisatorische Randbedingungen
- Qualitätsanforderungen
- ...
- Gesetze, Verordnungen



Notwendig: Dinge ausprobieren
und Kompromisse schließen.

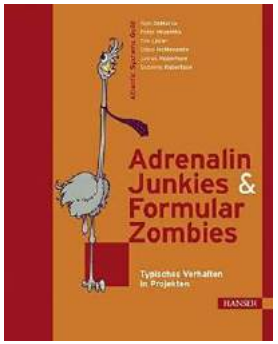
Architektur (Entwurf, Implementierung) ist **iterativer** Prozess!

Vorteile iterativ/inkrementellen Entwurfs

- Frühzeitige Rückmeldung:
  - **frühes Erkennen von Risiken**
  - **verlängert Zeit zur Abhilfe**
- Ermöglicht (frühzeitige) Reaktion auf Änderungen
- Auch: “Das Endspiel üben”

  > Ein Erfolgsmuster aus “Adrenalin Junkies & Formular Zombies, DeMarco et al., Hanser-Verlag

  > Regelmäßige Releases in kurzen Abständen!



Ein Problem - viele mögliche Lösungen

Entwerfen heißt: Entscheidungen treffen!

- Entwerfen erfordert Mut!
  (Analyse ist für Feiglinge: Bei Fehlern ist immer der Kunde Schuld, der die Forderung gestellt hat!)

- Es gibt KEINE STANDARDLÖSUNGEN für alle Projektarten, aber inzwischen einige erprobte Hilfsmittel:
  - Methodisches Vorgehen
  - Muster für Ergebnisgliederung
  - Architektur-, Entwurfs- und Implementierungsmuster

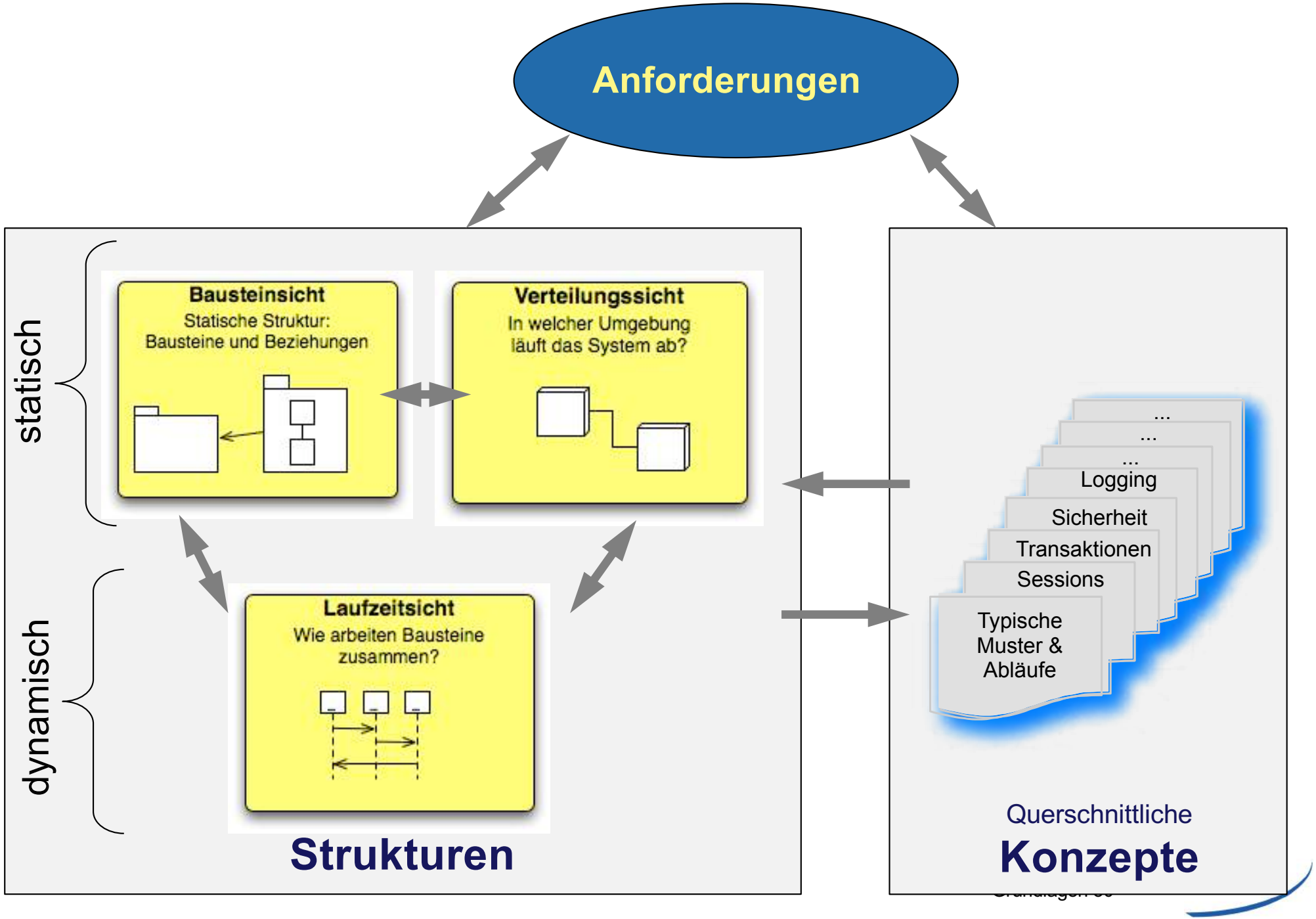
Schlechte Aussichten…

Desaster-Tipp

Fritz-5/Chessbase

- Wählen Sie eine **schlechte Perspektive** und bieten Sie keine Alternativen an
  (Dann erkennt garantiert NIEMAND eine Lösung!)

Architekturen entwickeln... ... mit verschiedenen Sichten



Templatebasierte Entwicklung/Doku

Anforderungen

Strukturen von Sourcecode

Entscheidungen

Template

- Muster für das Ergebnis, d.h. die Architektur
  - bewährte und praxisnahe Gliederung
    - beschleunigt schreiben und lesen von Dokumentation
    - vermeidet neu erfundene Räder
    - seit 2005 frei verfügbar, aktuell Version 7 (Februar 2017).
- Behälter (Repository) für alle architekturrelevanten Informationen
  - Basis zur Generierung Stakeholder-relevanter Dokumente
  - Frei verfügbar ☺ auf **arc42.de** und **arc42.org**
  - Dokumentation siehe http://docs.arc42.org
- Analog VOLERE*) für Requirements Engineering

*) www.systemsguild.com

Aufbau des arc42 Templates

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
….

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
….

7.Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
….

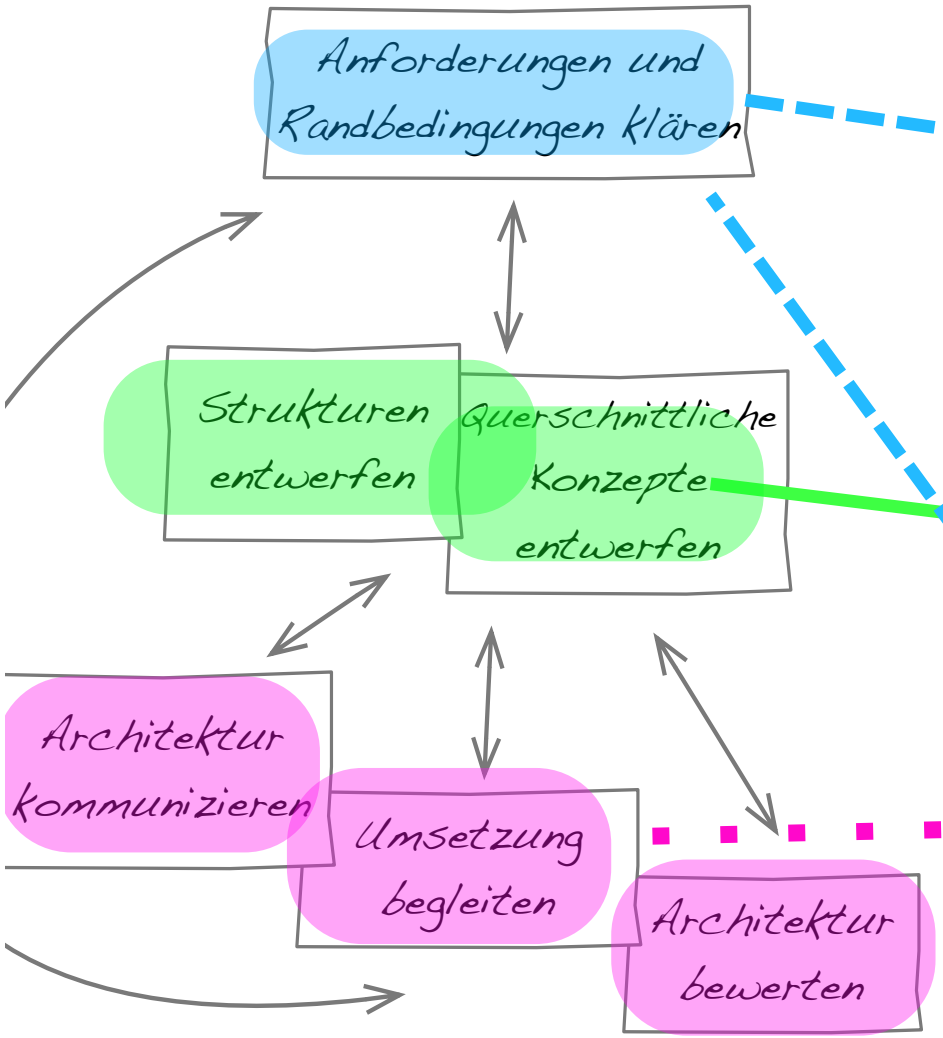
8. Querschnittliche Konzepte
8.1 Fachliche Struktur und Modelle
8.2 Architektur- und Entwurfsmuster
8.3 Unter-der-Haube
8.4 User Experience
….

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
….

10. Qualitätsanforderungen
10.1 Qualitätsbaum
10.2 Qualitätsszenarien

11. Risiken und technische Schulden

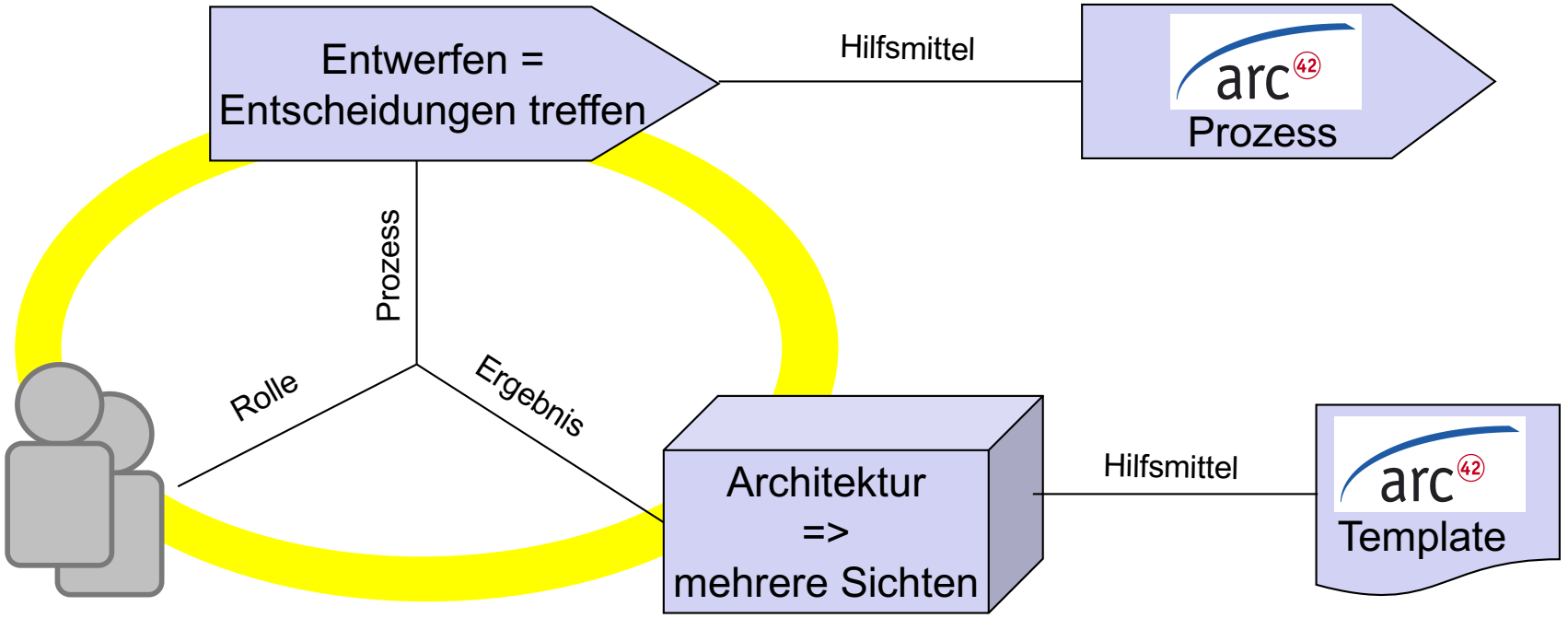
12. Glossar



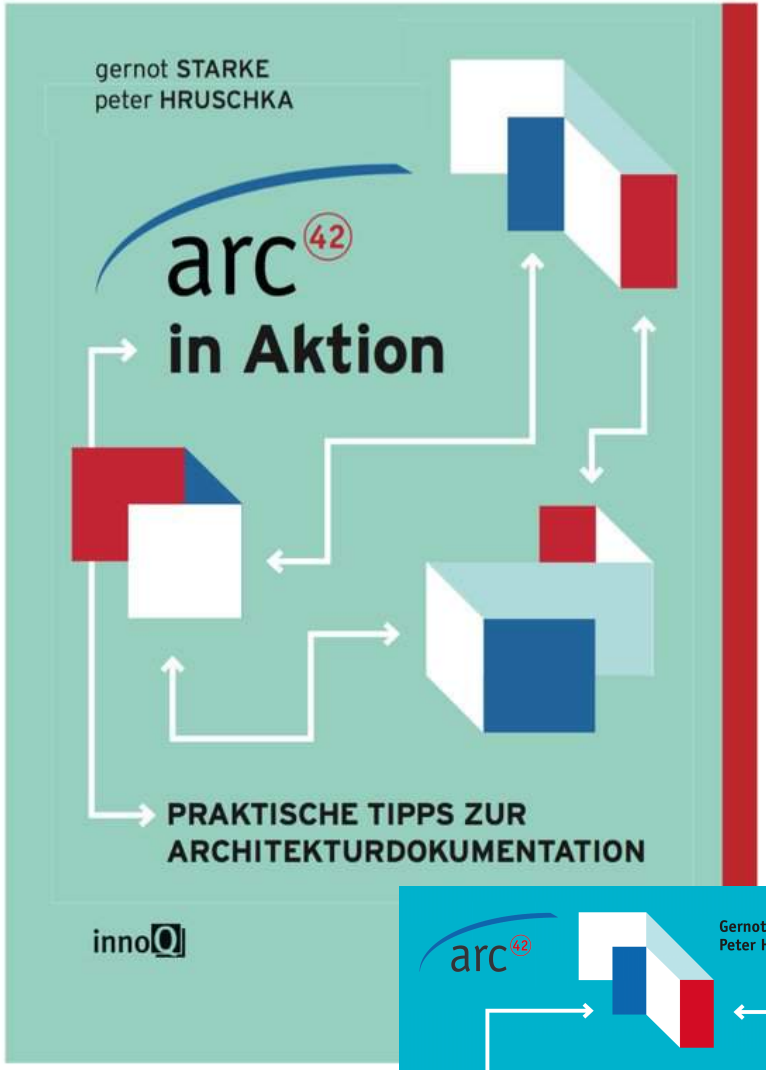
Softwarearchitektur

1. Einführung & Qualitätsziele
2. Randbedingungen
3. Kontextabgrenzung
4. Lösungsstrategie
5. Bausteinsicht
6. Laufzeitsicht
7. Verteilungssicht
8. Querschnittliche Konzepte
9. Entwurfsentscheidungen
10. Qualitätsszenarien
11. Risiken & technische Schulden
12. Glossar

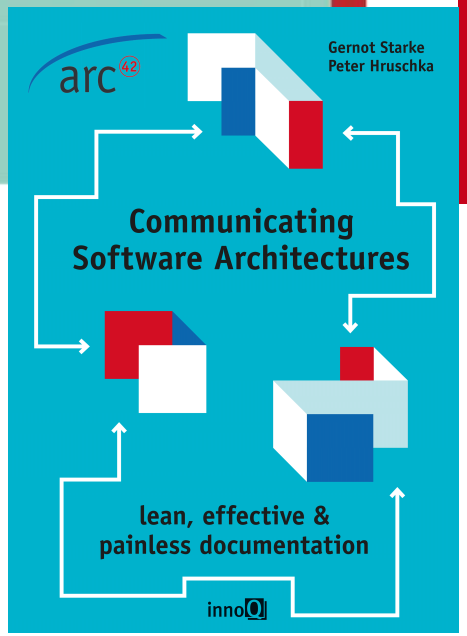
Zusammenfassung

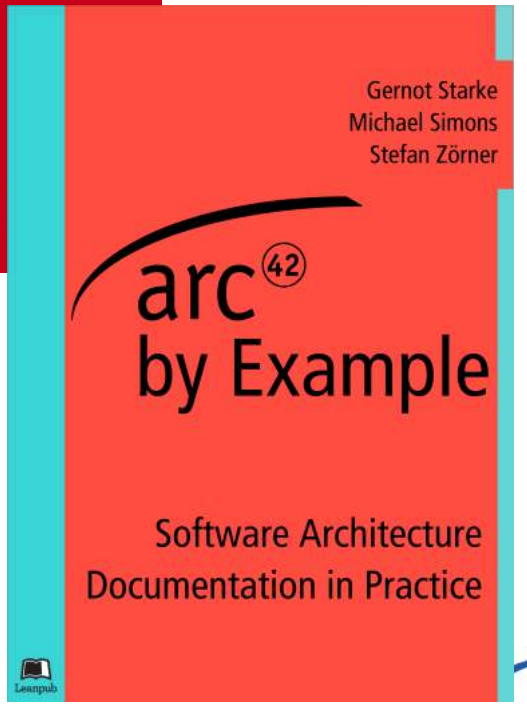


Architekten =
Rolle von
Zehnkämpfer
mit Verantwortung


gernot STARKE
peter HRUSCHKA
arc 42
in Aktion
PRAKTISCHE TIPPS ZUR
ARCHITEKTURDOKUMENTATION
inno Q


Knigge für
Softwarearchitekten
Reloaded
Peter Hruschka und
Gernot Starke
überarbeitete,
ergänzte Neuauflage


arc 42
Gernot Starke
Peter Hruschka
Communicating
Software Architectures
lean, effective &
painless documentation
inno Q


Gernot Starke
Michael Simons
Stefan Zörner
arc 42
by Example
Software Architecture
Documentation in Practice
Leanpub