Architektur von Informationssystemen

Hochschule für angewandte Wissenschaften

Sommersemester 2016

Nils Löwe / nils@loewe.io / @NilsLoewe

Raumänderung

Die Vorlesung findet jetzt in Raum 1060 statt!

Was ist Softwarearchitektur?

Geschichte und Trends

Sichten auf Architekturen

Qualiät und andere nichtfunktionale Anforderungen

Architekturmuster

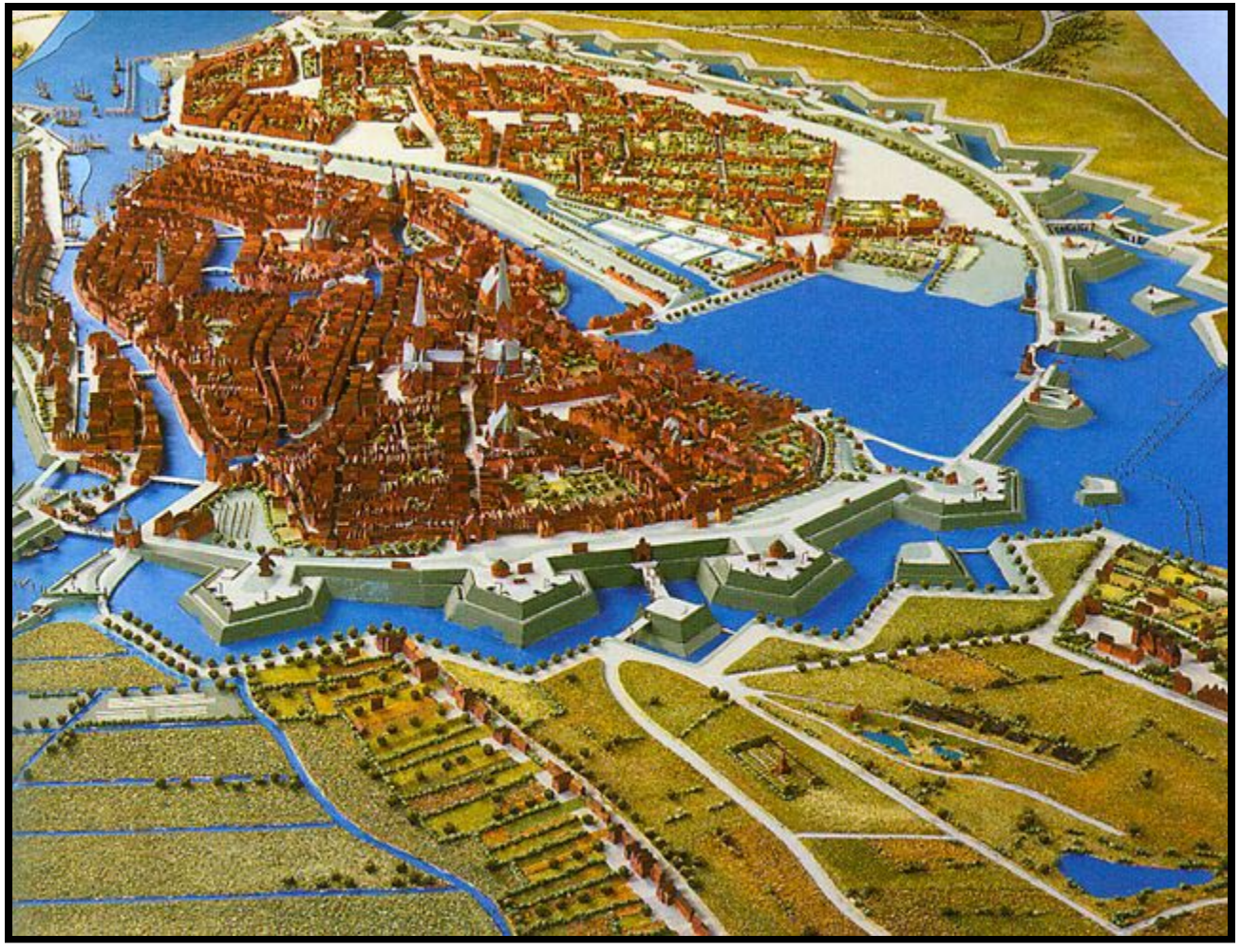
Dokumentation von Architekturen

Technologien und Frameworks

Wiederholung
Was ist Softwarearchitektur?

Was ist Architektur?

- Der Begriff stammt aus dem Mittelalter
- Oberbegriff für Baustil / Baukunst
- Ziel: Ordnung und Generalisierung struktureller Beziehungen in Produkten des Bauwesens
- Ziel: Erfahrung und Wissen verallgemeinern.



The software architecture of deployed software is determined by those aspects that are the hardest to change. *(Chris Verhoef)*

Architektur besteht aus Strukturen

- die Komponenten (Bausteine), aus denen ein System besteht
- die wesentlichen (extern sichtbaren) Eigenschaften dieser Komponenten
- die Beziehungen der Komponenten untereinander

Architektur basiert auf Entwurfsentscheidungen

- Entscheidungen zum Entwurf der Komponenten
- Entscheidung für eine bestimmte Technologie

Architektur besteht aus verschiedenen *Sichten*

- jede Sicht dokumentiert einzelne Aspekte des Gesamtsystems
- jede Sicht ist für bestimmte Stakeholder nützlich

Architektur ist Abstraktion

- Essenzielle Aufgabe von Architekten: Weglassen von nicht benötigten Informationen
- Informationen werden bewusst gefiltert um die Darstellung lesbar und verständlich zu halten

Architektur vs. Entwurf/Design?

- die Grenze ist fließend
- Design (oder Entwurf) bezeichnet den Prozess der Erstellung der Architektur

"Das Leben von Software-Architekten besteht aus einer langen und schnellen Abfolge suboptimaler Entwurfsentscheidungen, die meist im Dunkel getroffen werden."

(Phillipe Kruchten)

Architekten beraten

- Management und Auftraggeber bei der Projektplanung und -Organisation
- Auftraggeber und Analyseteams zu Machbarkeit, Kosten/Nutzen, Auswirkungen von Anforderungen, Realisierung und Betrieb
- Projektleiter bei der Organisaton und Steuerung des Implementierungsteams
- Projektleiter beim Managment (technischer) Risiken
- das Implementierungsteam bei der Umsetzung
- Hardware-Architekten und Systembetreiber hinsichtlich Hardware-Anforderungen
- die Qualitätssicherung über Kritikalität und Testbarkeit von Systembestandteilen

Architekten dokumentieren - angemessen

- an den Bedürfnissen der Adressaten orientieren
- pragmatisch arbeiten (manchmal reicht eine Skizze auf einem alten Umschlag)

Die Projekte sollen agil, flexibel und kurzfristig wandlungsfähig bleiben!

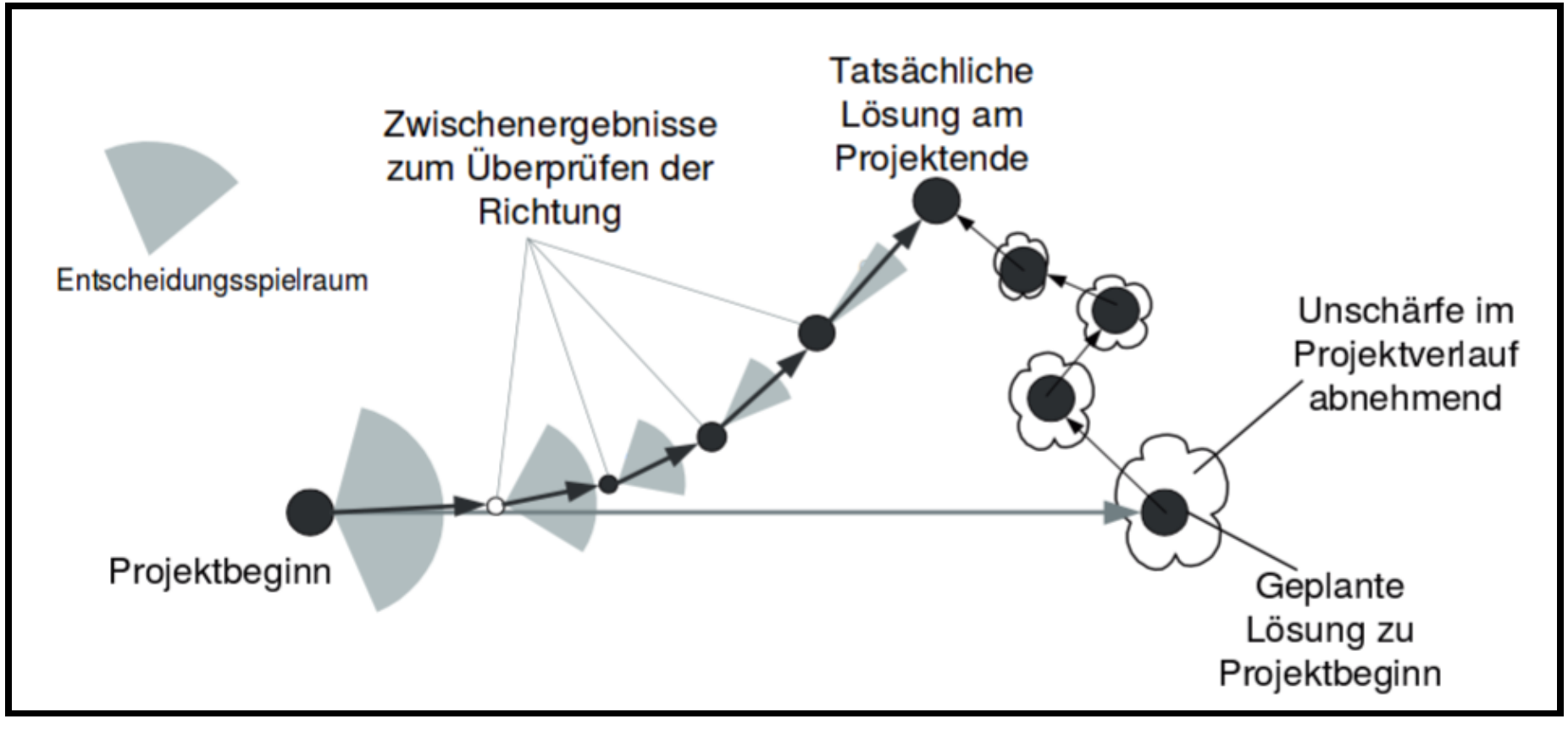
"Die zuverlässigste, preiswerteste und robusteste Komponente eines Systems ist diejenige, die erst gar nicht realisert werden muss!"

(Gernot Starke)

Architekten bewerten

An welchen Stellen des Systems sind nicht-funktionale Anforderungen (z.B. Performance) riskant oder kritisch?

Architekturen entstehen in Zyklen und Iterationen



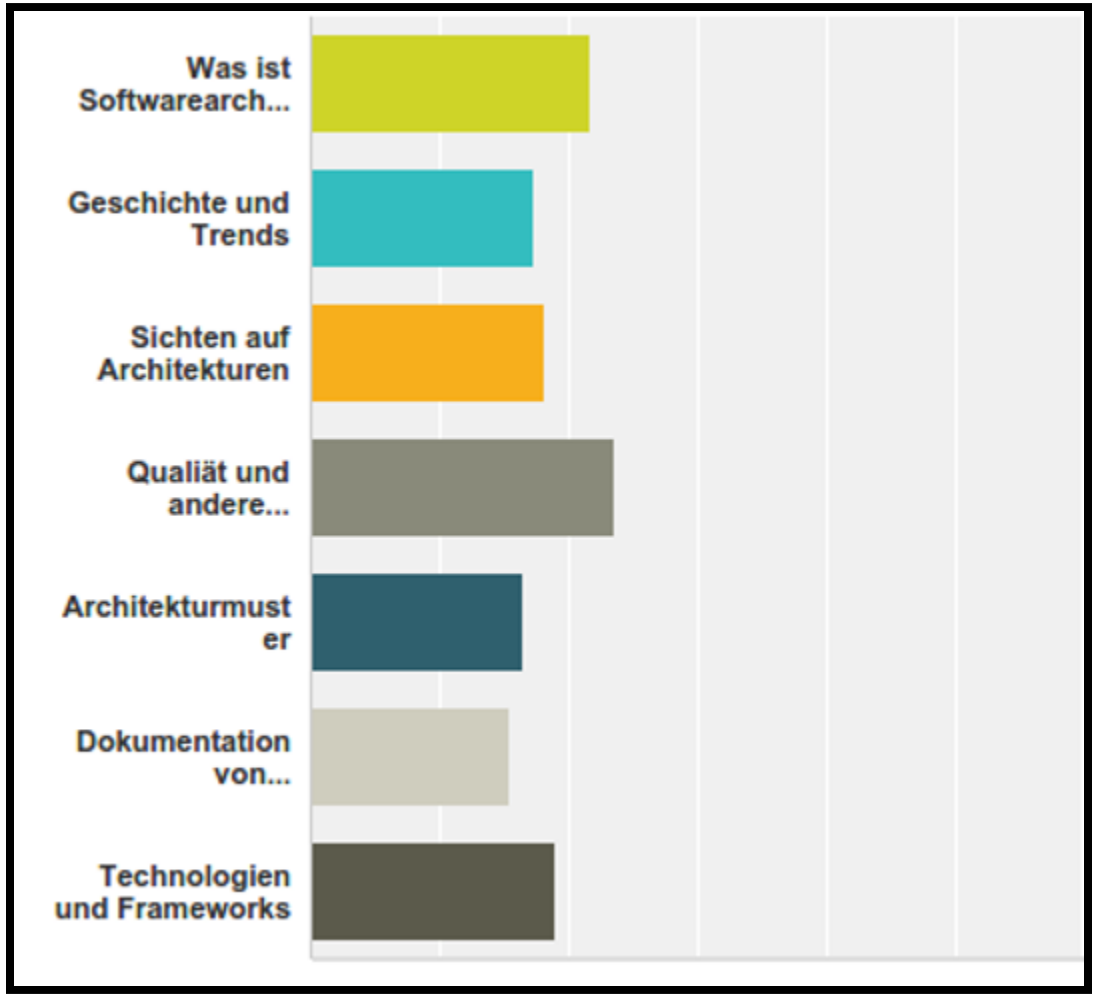
Bildquelle: Starke / "Effektive Softwarearchitekturen" (5. Auflage)

Wie Architekturen nicht entstehen sollten

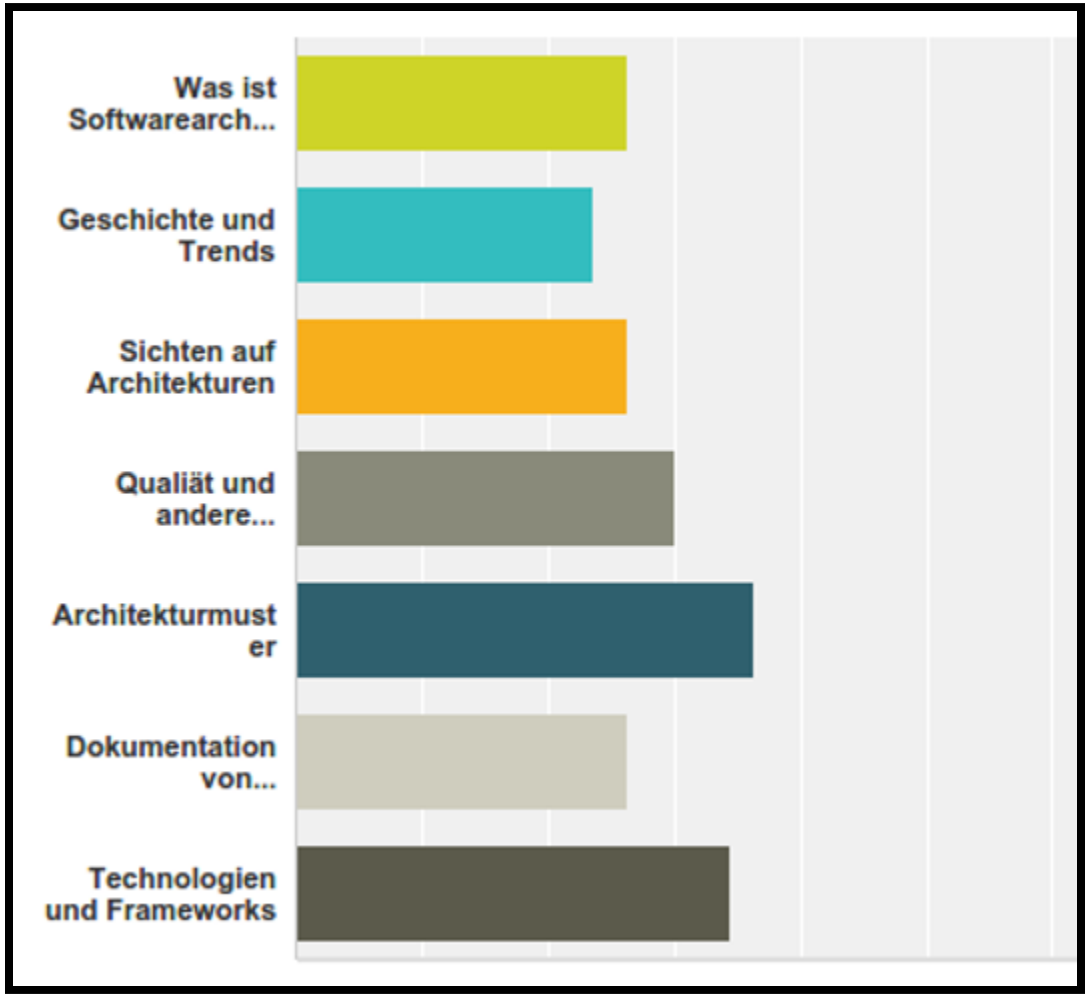
- im Architekturkomitee, das jeden Donnerstag um 15 Uhr im großen Besprechungsraum tagt
- im Elfenbeinturm ohne Kunden, Auftraggeber, Projektleitung und Realisierungsteam
- ausschließlich auf bunten Marketing-Folien
- "Wir machen jetzt {Name-der-Technologie}!"

Ergebnisse der Umfrage

Wieviel Vorwissen haben Sie zu den einzelnen Bereichen?



Welche Schwerpunkt wünschen Sie sich?



Was ist Softwarearchitektur?

Geschichte und Trends

Sichten auf Architekturen

Qualiät und andere nichtfunktionale Anforderungen

Architekturmuster

Dokumentation von Architekturen

Technologien und Frameworks

Seit wann gibt es den Begriff der Softwarearchitektur?

Konferenz über Softwaretechnik in Rom

Software Engineering Techniques. Report of a Conference Sponsored by the NATO Science Committee. Scientific Affairs Division, NATO, 1970, S. 12.

Warum?

Die Systeme wurden in den 1960ern so komplex, dass sie von mehreren Teams entwickelt werden mussten.

Beispiel: IBM OS/360

Planung

- Entwicklungskosten: 40 Mio. USD
- Lines of Code: 1 Mio.
- Fertigstellung: 1965

Beispiel: IBM OS/360

Realität

- Entwicklungskosten: 500 Mio. USD (Faktor 12,5)
- Lines of Code: 10 Mio. (Faktor 10)
- Fertigstellung: 1967 (2 Jahre zu spät)

Beispiel: IBM OS/360

Fun Facts

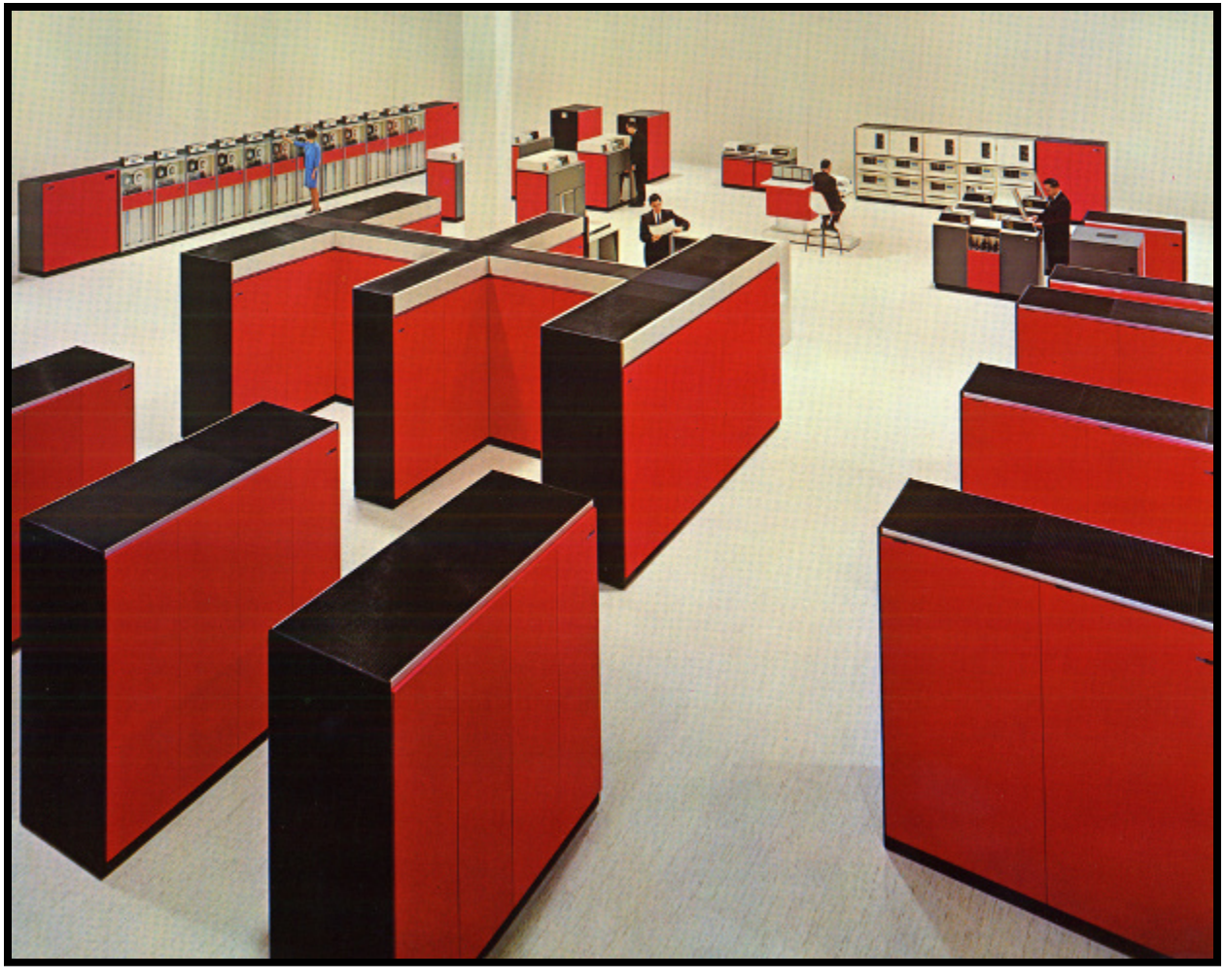
- PCP: Primary Control Program (genau ein Task ausführbar)
- MFT: Multiprogramming with a Fixed number of Tasks (Anzahl Tasks fest vorgegeben)
- MVT: Multiprogramming with a Variable number of Tasks (Echte dynamische Speicherverwaltung)

Beispiel: IBM SYSTEM/360

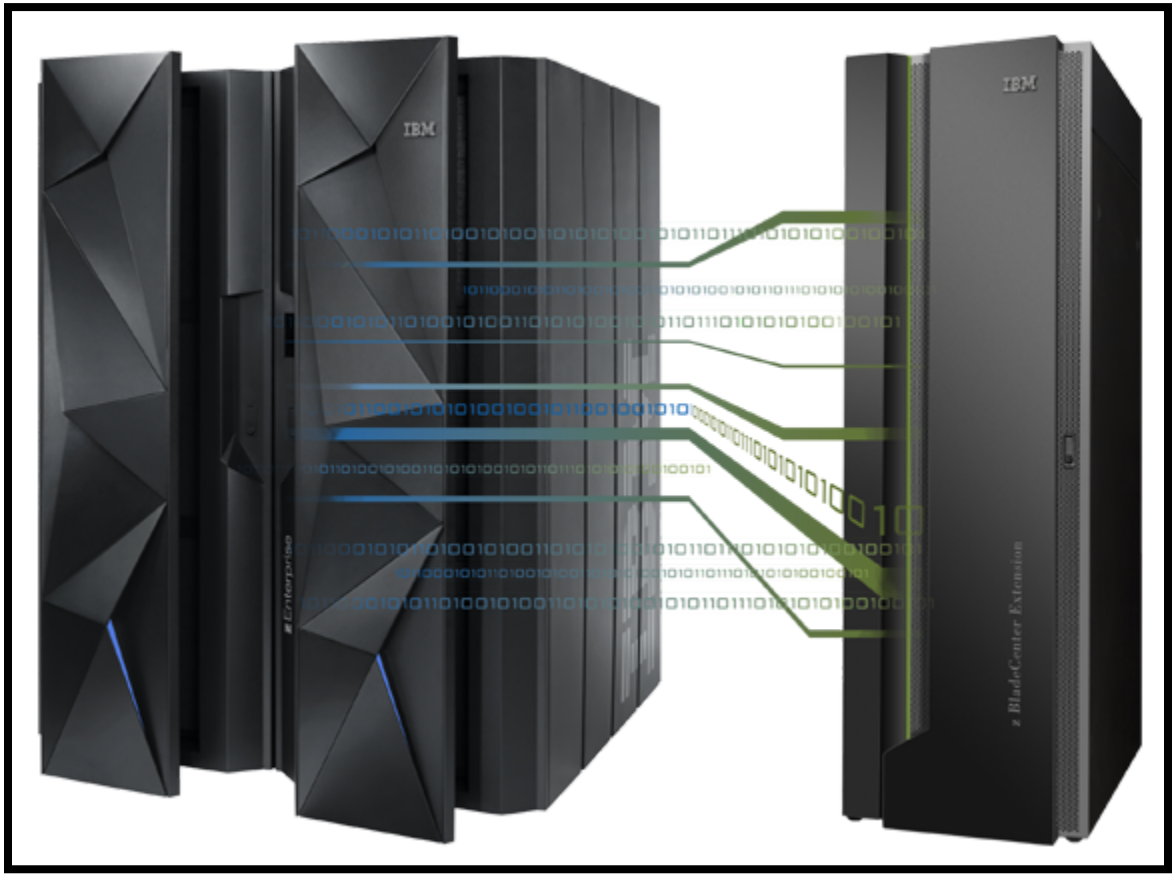
Mainframes damals und heute

- Mainframes verwalten heute 80 % aller Unternehmensdaten
- Mainframes verarbeiten heute täglich 30 Mrd. Unternehmenstransaktionen (z.B. Banking, Flugbuchungen, ...)
- Modell EC12 (2012): 5,5GHz CMOS Prozessor, 3 TB Ram
- Erstes Modell damals: 0,0018 MIPS, 8 KByte Ram
- Vergleich: Ein iPhone 5S schafft 18200 MIPS

1968: IBM System/360 m85



2012: IBM EC12



Märchenstunde

Softwarearchitektur im Lauf der Zeit

Softwarearchitektur im Lauf der Zeit

Erste Beschreibung von ”Dekomposition, Zerlegung, Entwurf”

- 1970er: Eher im Kontext von Hardware genutzt
- 1972: *"On the criteria to be used in decomposing systems into modules"* von D. L. Parnas
- 1975: *"The Mythical Man Month"* von Frederick Brooks

Softwarearchitektur im Lauf der Zeit

Unabhängiges Teilgebiet der Softwaretechnik

Konzept der Schnittstellen und Konnektoren

- 1992: *"Foundations for the Study of Software Architecture"* von Dewayne Perry und Alexander Wolf
- 1995: *"Software Architecture Analysis Method"* des Software Engineering Institute

Softwarearchitektur im Lauf der Zeit

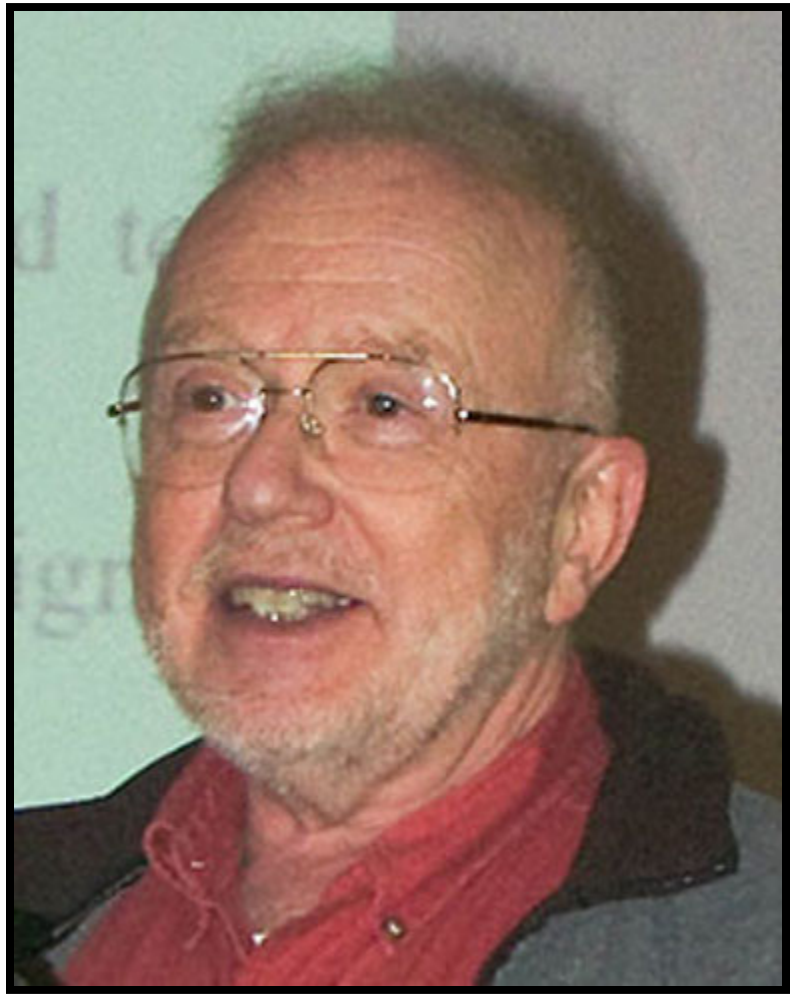
Allgemeine Verbreitung und ”Stand der Technik”

- 2000: *"IEEE 1471:2000 Norm Recommended Practice for Architectural Description of Software-Intensive Systems"*
- 2003: Zertifizierung als Softwarearchitekt durch die iSAQB (International Software Architect Qualification Board)
- 2003: UML 2.0 ist geeignet um Softwarearchitekturen zu beschreiben

Pioniere der Softwarearchitektur

Pioniere der Softwarearchitektur

David Parnas



* 10. Februar 1941, New York

Pioniere der Softwarearchitektur

David Parnas

Erfinder des *Modulkonzepts* und des *Geheimnisprinzips*

Schaffung der Grundlage der *objektorientierten Programmierung*

Pioniere der Softwarearchitektur

Frederick Brooks

* 19. April 1931, North Carolina

Pioniere der Softwarearchitektur

Frederick Brooks

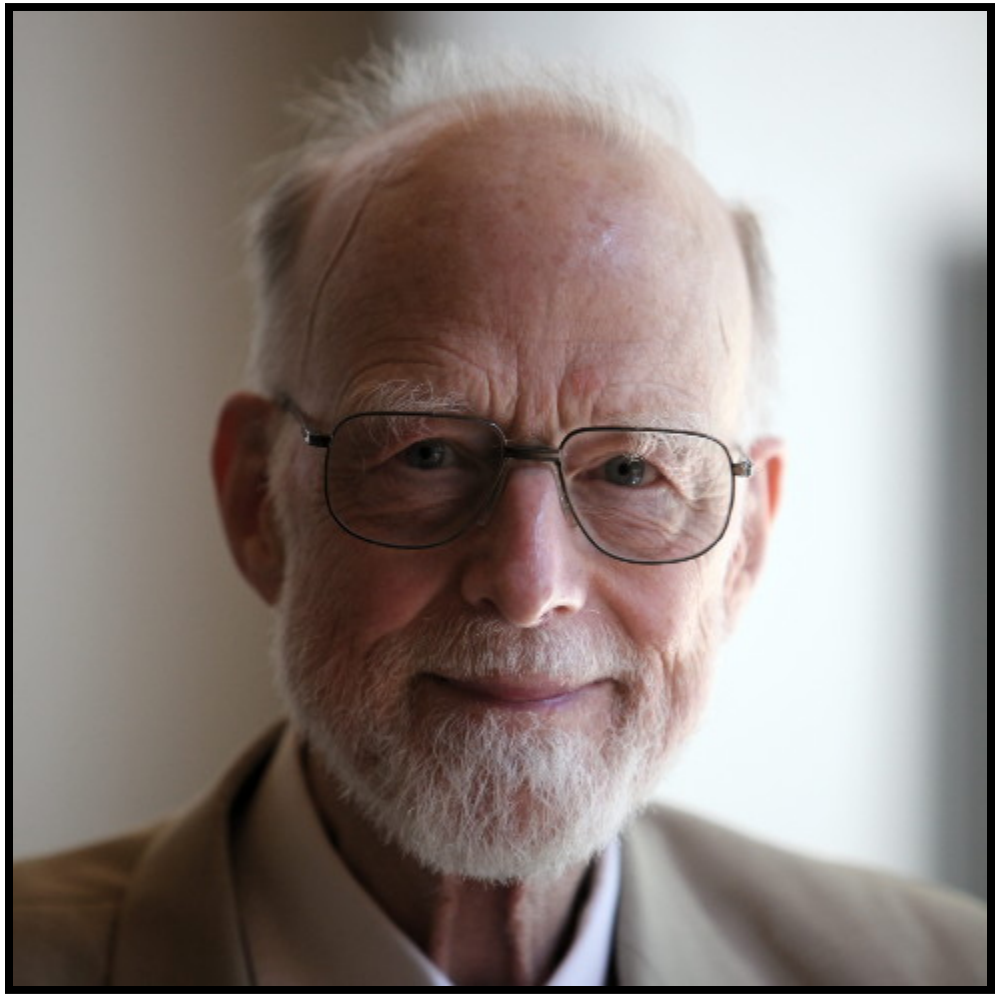
Schrieb das erste 'ehrliche' Buch über Software-Projektmanagement

"Adding manpower to a late software project makes it later."

(The Mythical Man Month: Essays on Software Engineering)

Pioniere der Softwarearchitektur

Tony Hoare



* 11. Januar 1934, Sri Lanka

Pioniere der Softwarearchitektur

Tony Hoare

- Entwickler des Quicksort-Algorithmus
- Erfinder des Hoare-Kalküls zum Beweisen der Korrektheit von Algorithmen
- Entwickler der Prozessalgebra *Communicating Sequential Processes (CSP)*, Grundlage der Programmiersprachen Ada, Occam und Go

Pioniere der Softwarearchitektur

Edsge Dijkstra

* 11. Mai 1930, Rotterdam

† 6. August 2002, Nuenen

Pioniere der Softwarearchitektur

Edsge Dijkstra

- Entwickler des Dijkstra-Algorithmus zur Berechnung eines kürzesten Weges in einem Graphen
- Einführung von Semaphoren zur Synchronisation zwischen Threads
- Entwicklung des Shunting-yard-Algorithmus zur Übertragung der Infixnotation n einen abstrakten Syntaxbaum
- Entwickler des Multitasking-Betriebssystem THE, erste dokumentierte Schichtenstruktur
- Mitentwickler von Algol 60 - Schrieb den den ersten Compiler dafür
  Prägung der Begriffe der strukturierten Programmierung der *Softwarekrise*

Pioniere der Softwarearchitektur

Per Brinch Hansen

* 13. November 1938 in Frederiksberg

† 31. Juli 2007

Pioniere der Softwarearchitektur

Per Brinch Hansen

- Entwickler des RC-4000-Minicomputer und dessen Betriebssystems (1969): Erste Implementierung des Mikrokern-Konzepts
- Erfinder des Monitor-Konzepts für das Concurrent Programming
- Entwickler von *Concurrent Pascal*, der ersten nebenläufigen Programmiersprache
- Entwickler von *SuperPascal* zur Darstellung paralleler Algorithmen
- Von Per Brinch Hansen stammt die dänische Bezeichnung *Datamat* für Computer

Pioniere der Softwarearchitektur

Friedrich Bauer

* 10. Juni 1924 in Regensburg

✝ 26. März 2015

Pioniere der Softwarearchitektur

Friedrich Bauer

- Erfinder des Stack-Konzepts ("Kellerspeichers")
- Hielt 1967 an der Technischen Universität München die erste offizielle Informatikvorlesung in Deutschland
- Ausrichter der ersten Computerausstellung im Deutschen Museum 1988
- Autor mehrerer Standardwerke zur Kryptologie

Pioniere der Softwarearchitektur

Niklaus Wirth

* 15. Februar 1934 in Winterthur

Pioniere der Softwarearchitektur

Niklaus Wirth

- Erfinder des *Wirthschen Gesetzes*, nach dem sich die Software schneller verlangsamt als sich die Hardware beschleunigt.
- Mitentwickler Programmiersprache Euler
- Entwickler derProgrammiersprache PL360, die 1968 auf dem System IBM /360 implementiert wurde
- Mitentwickler der Programmiersprache Algol
- Entwickler der Programmiersprache Pascal
- Erweiterung der formalem Sprache Backus-Naur-Form (BNF), zur Erweiterten Backus-Naur-Form (EBNF)
- Entwickler von Modula, Modula-2 und Oberon (1985–1990)
  ”Importierte” 1980 eine der ersten Computermäuse nach Europa, was zur Gründung von Logitech führte

Tools und Frameworks im Laufe der Zeit

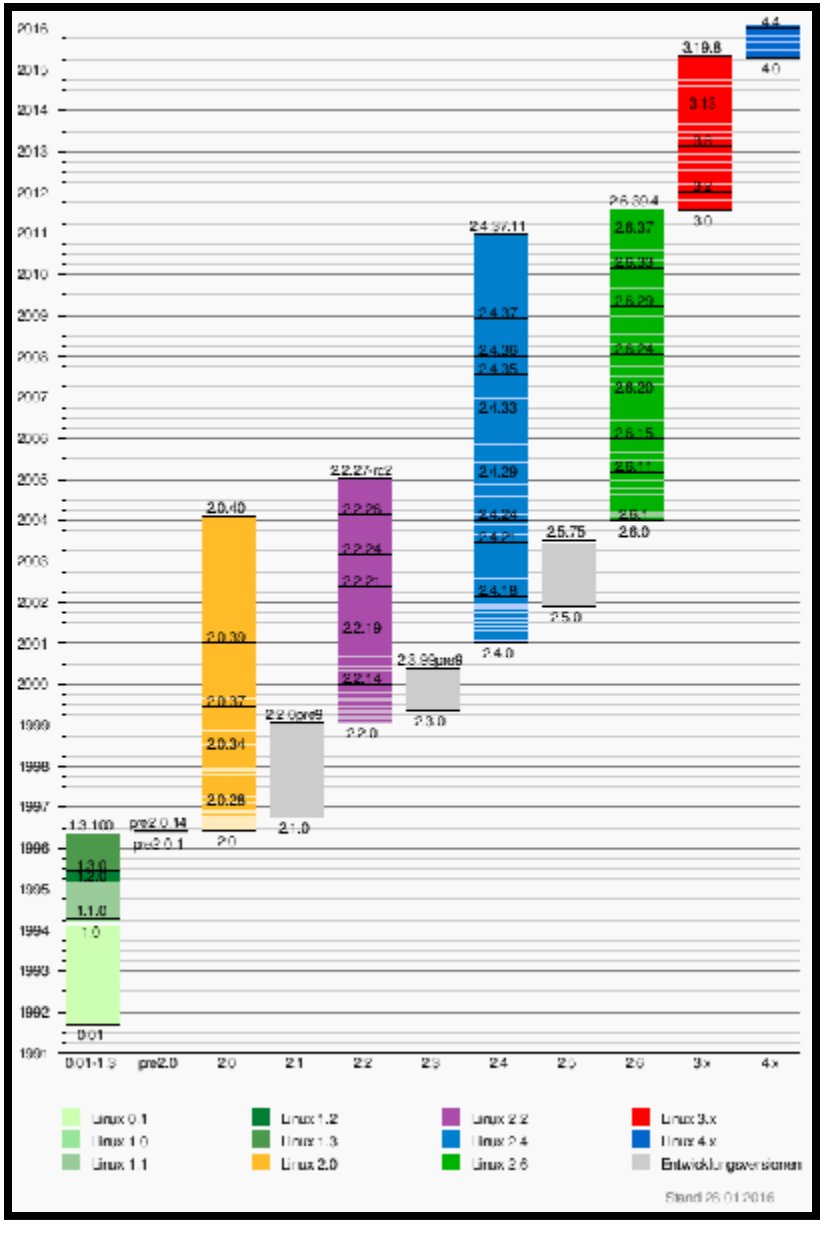
Tools und Frameworks im Laufe der Zeit

Entwicklung des linux-kernels

- 1992: V 0.0.1 / 8k LOC / 230 kB
- 1994: V 1.0.0 / 170k LOC / 1.2 MB
- 1996: V 2.0.0 / 716k LOC / 5.8 MB
- 2011: V 3.0.0 / 14.6 Mio. LOC / 96 MB
- 2015: V 4.0.0 / 19.3 Mio. LOC / 78 MB

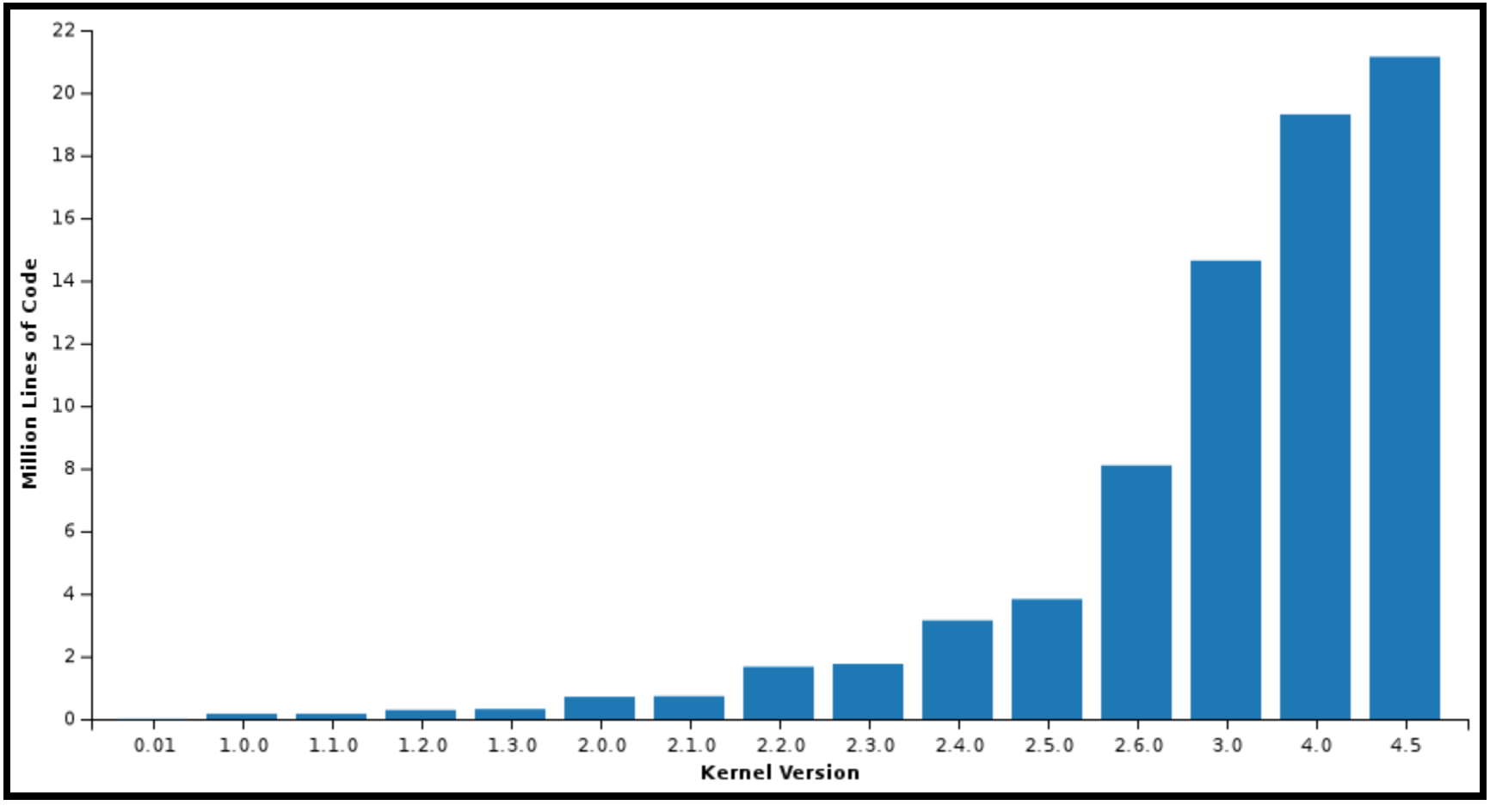
Tools und Frameworks im Laufe der Zeit

Entwicklung des linux-kernels


2016
2015
2014
2013
2012
2011
2010
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000
1999
1998
1997
1996
1995
1994
1993
1992
1991
0.01
1.0
1.1.0
1.2.0
1.3.100
1.3.0
pre2.0.1
pre2.0.14
2.0
2.0.28
2.0.34
2.0.37
2.0.39
2.0.40
2.1.0
2.2.0
2.2.14
2.2.19
2.2.24
2.2.26
2.2.27-rc2
2.3.99pre9
2.4.0
2.4.18
2.4.33
2.4.35
2.4.36
2.4.37
2.4.37.11
2.5.0
2.5.75
2.6.0
2.6.20
2.6.29
2.6.37
2.6.39.4
3.0
3.13
3.19.8
4.0
4.4
0.01-1.3
pre2.0
2.0
2.1
2.2
2.3
2.4
2.5
2.6
3.x
4.x
Linux 0.1
Linux 1.0
Linux 1.1
Linux 1.2
Linux 1.3
Linux 2.0
Linux 2.2
Linux 2.4
Linux 2.6
Linux 3.x
Linux 4.x
Entwicklungsversionen
Stand 26.01.2016

Tools und Frameworks im Laufe der Zeit

Entwicklung des linux-kernels



Tools und Frameworks im Laufe der Zeit

Entwicklung von Ruby on Rails

- 2005: V 1.0.0 / 96k LOC / 3365 Klassen / 8523 Methoden
- 2007: V 2.0.0 / 170k LOC / 5255 Klassen / 13260 Methoden
- 2010: V 3.0.0 / 230k LOC / 8334 Klassen / 19785 Methoden
- 2013: V 4.0.0 / 317k LOC / 9430 Klassen / 24143 Methoden

Was ist Softwarearchitektur?

Geschichte und Trends

Sichten auf Architekturen

Qualiät und andere nichtfunktionale Anforderungen

Architekturmuster

Dokumentation von Architekturen

Technologien und Frameworks

Warum überhaupt Sichten?

”Es ist eine offensichtliche Wahrheit, dass auch eine perfekte Architektur nutzlos bleibt, wenn sie nicht verstanden wird...”

Felix Bachmann und Len Bass in "Software Architecture Documentation in Practice"

1.

Eine einzelne Darstellung kann die Vielschichtigkeit und Komplexität einer Architektur nicht ausdrücken.

- Genauso wenig, wie man nur mit einem Grundriss ein Haus bauen kann.

2.

Sichten ermöglichen die Konzentration auf einzelne Aspekte des Gesamtsystems und reduzieren somit die Komplexität der Darstellung.

3.

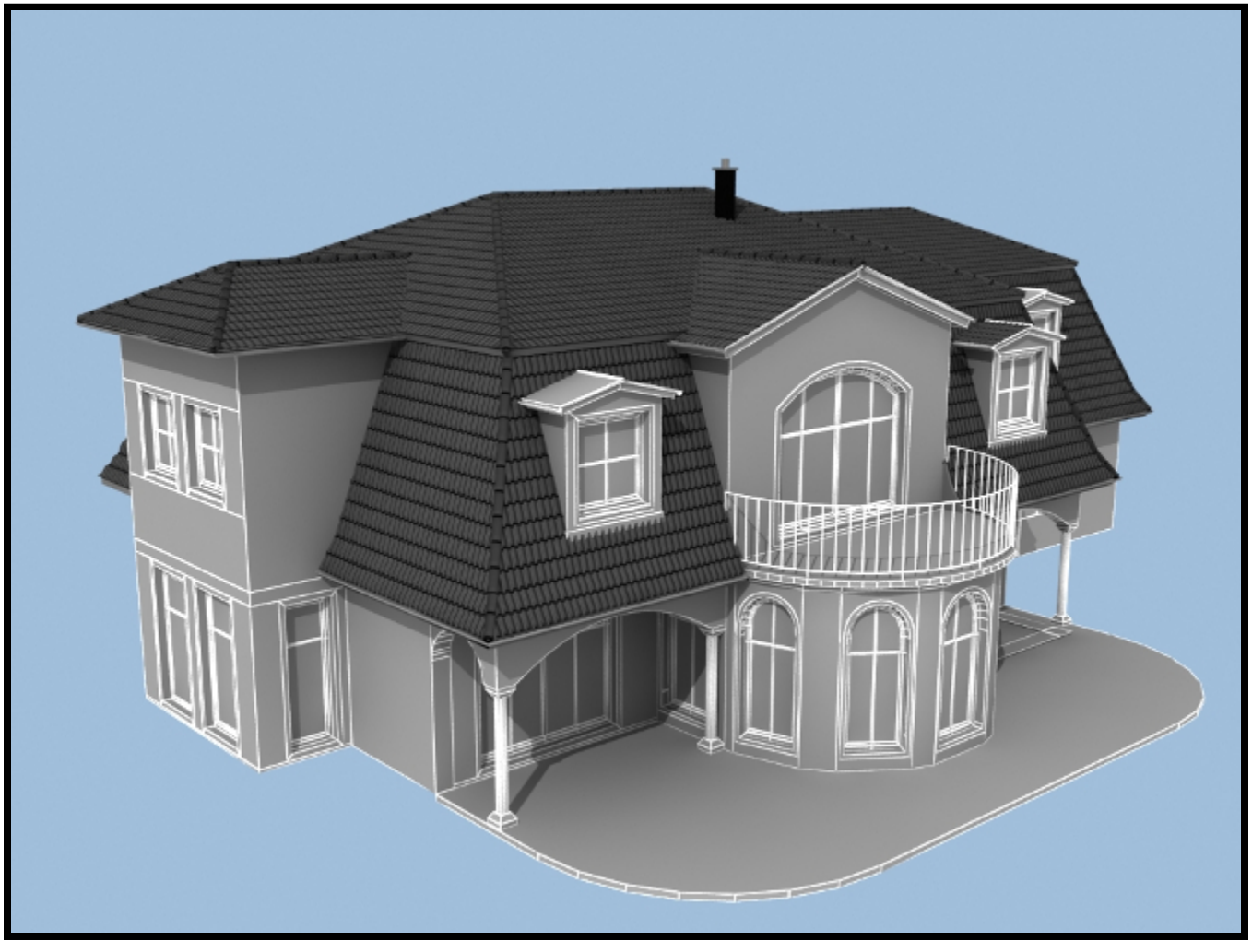
Die Projektbeteiligten haben ganz unterschiedliche Informationsbedürfnisse.

Architekten müssen Projektbeteiligten die Architektur erklären bzw. sie verteidigen/vermarkten

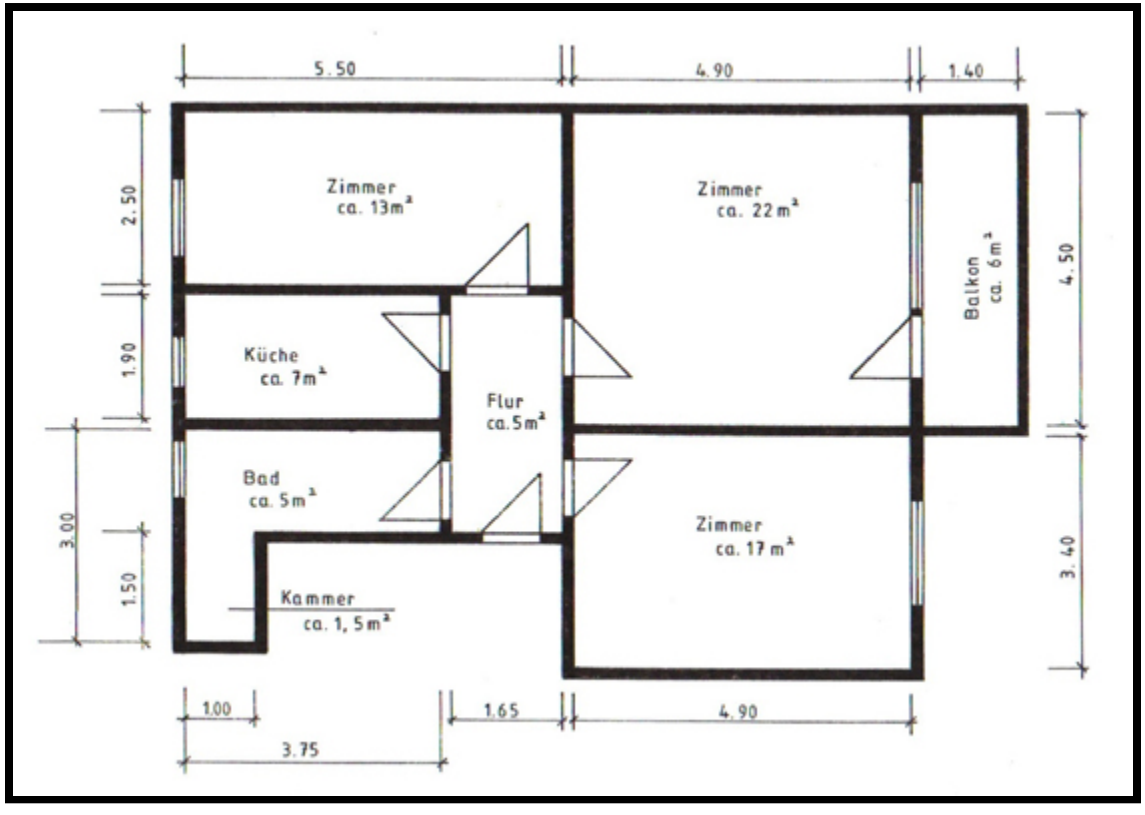
- die entworfenen Strukturen
- die getroffenen Entscheidungen
- ihre Konzepte + Begründungen + Vor- und Nachteile

→ Mit Hilfe von unterschiedlichen Sichten lassen sich viele Aspekte von Architektur verständlich darstellen.

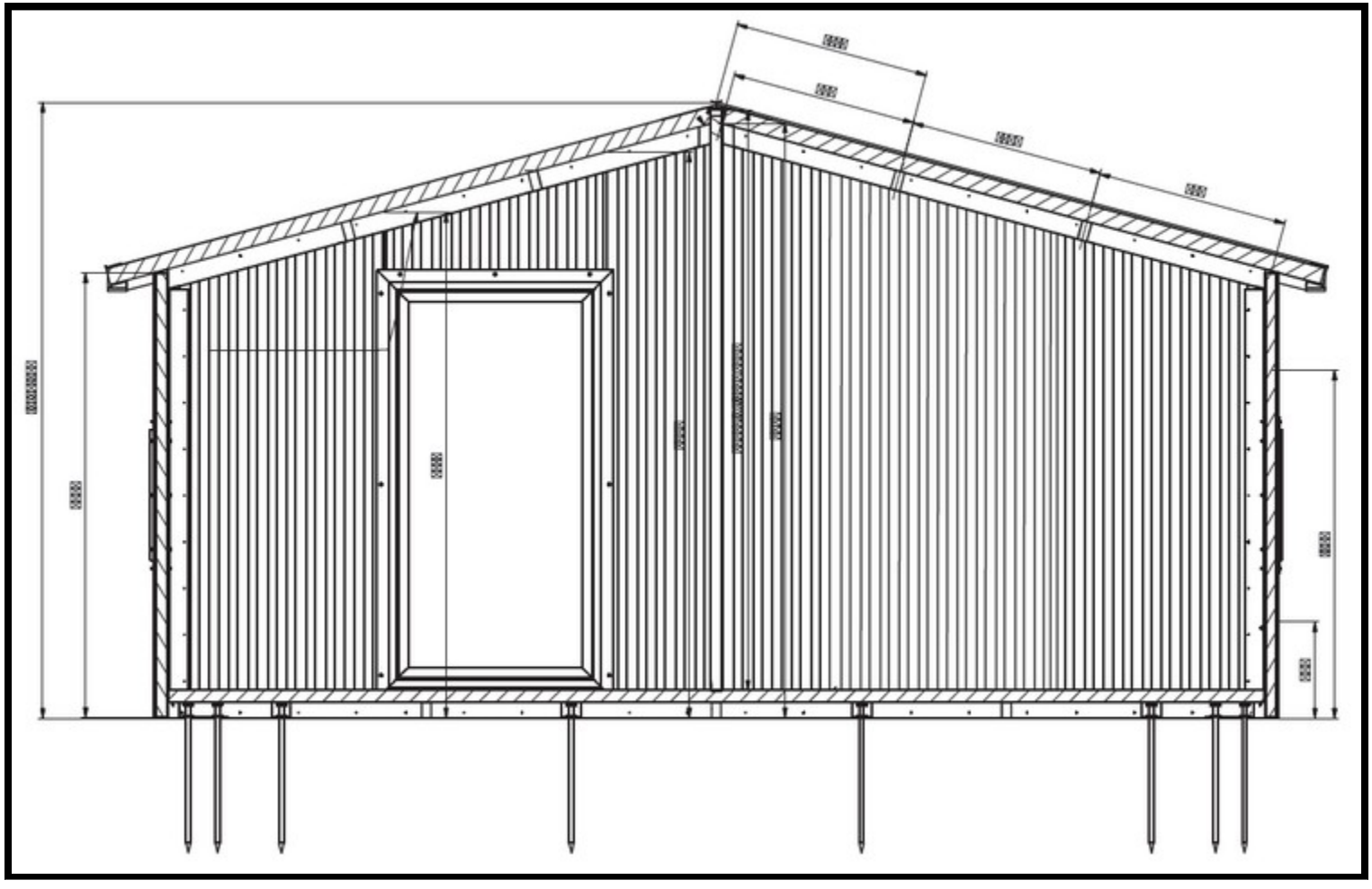
Beispiel: Gebäudearchitektur



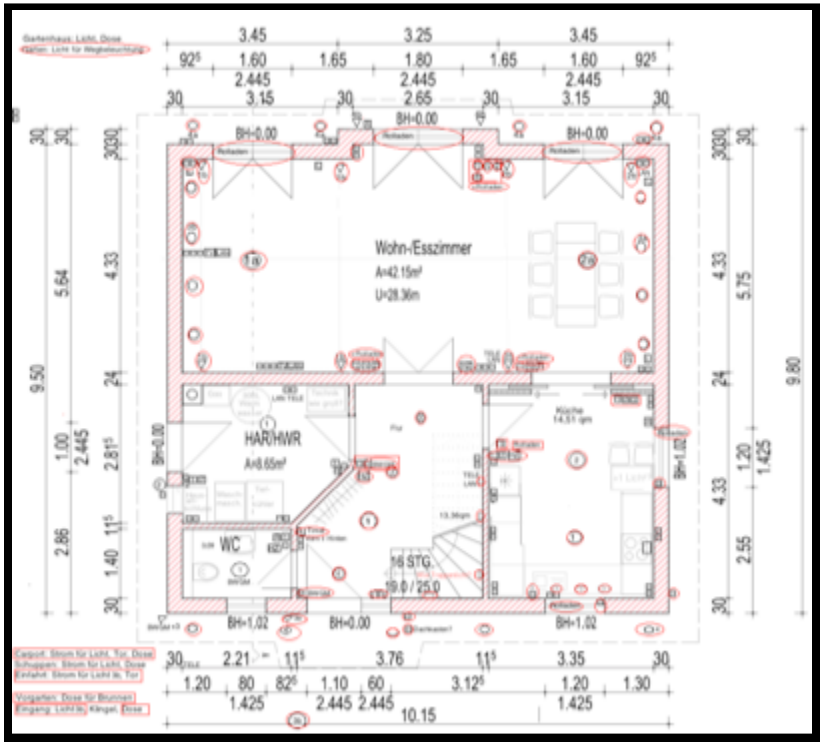
Beispiel: Gebäudearchitektur



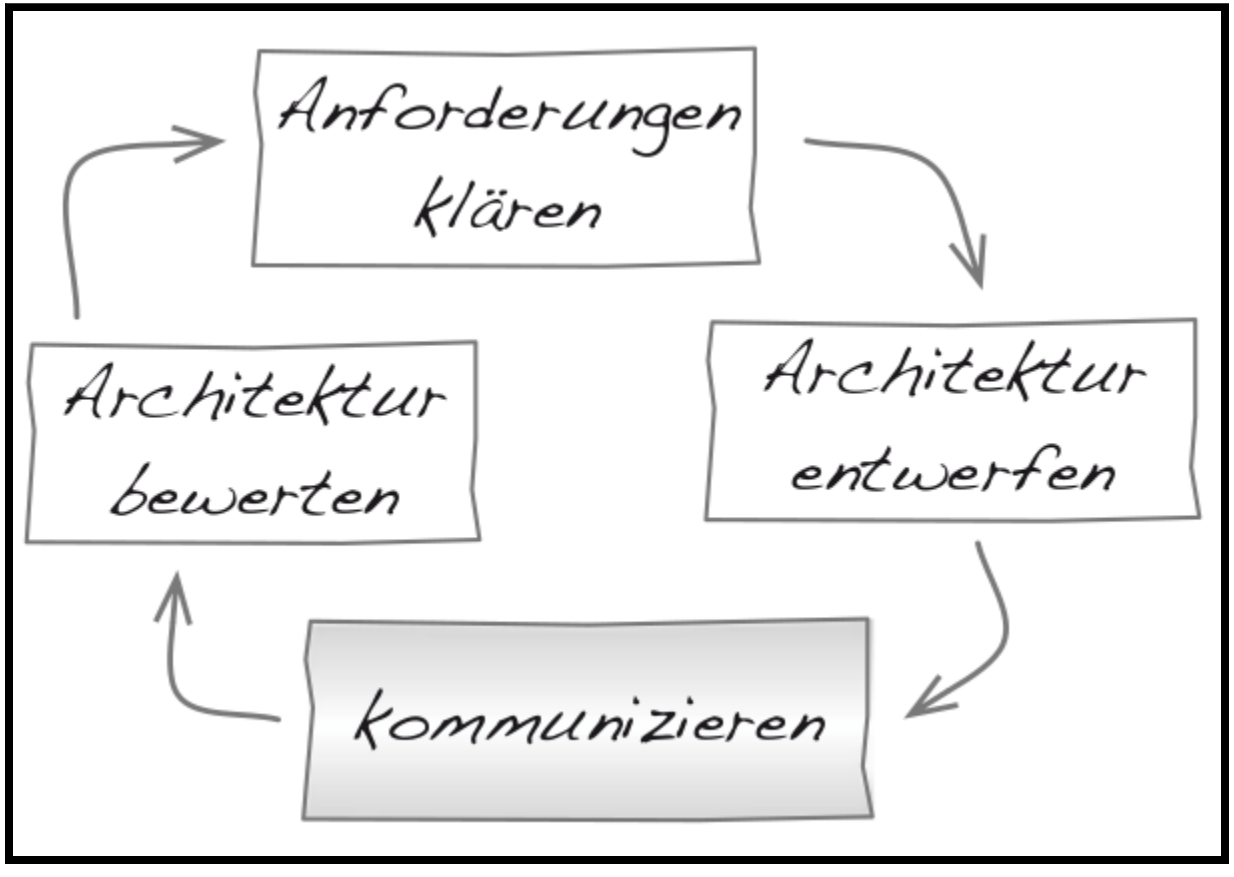
Beispiel: Gebäudearchitektur



Beispiel: Gebäudearchitektur

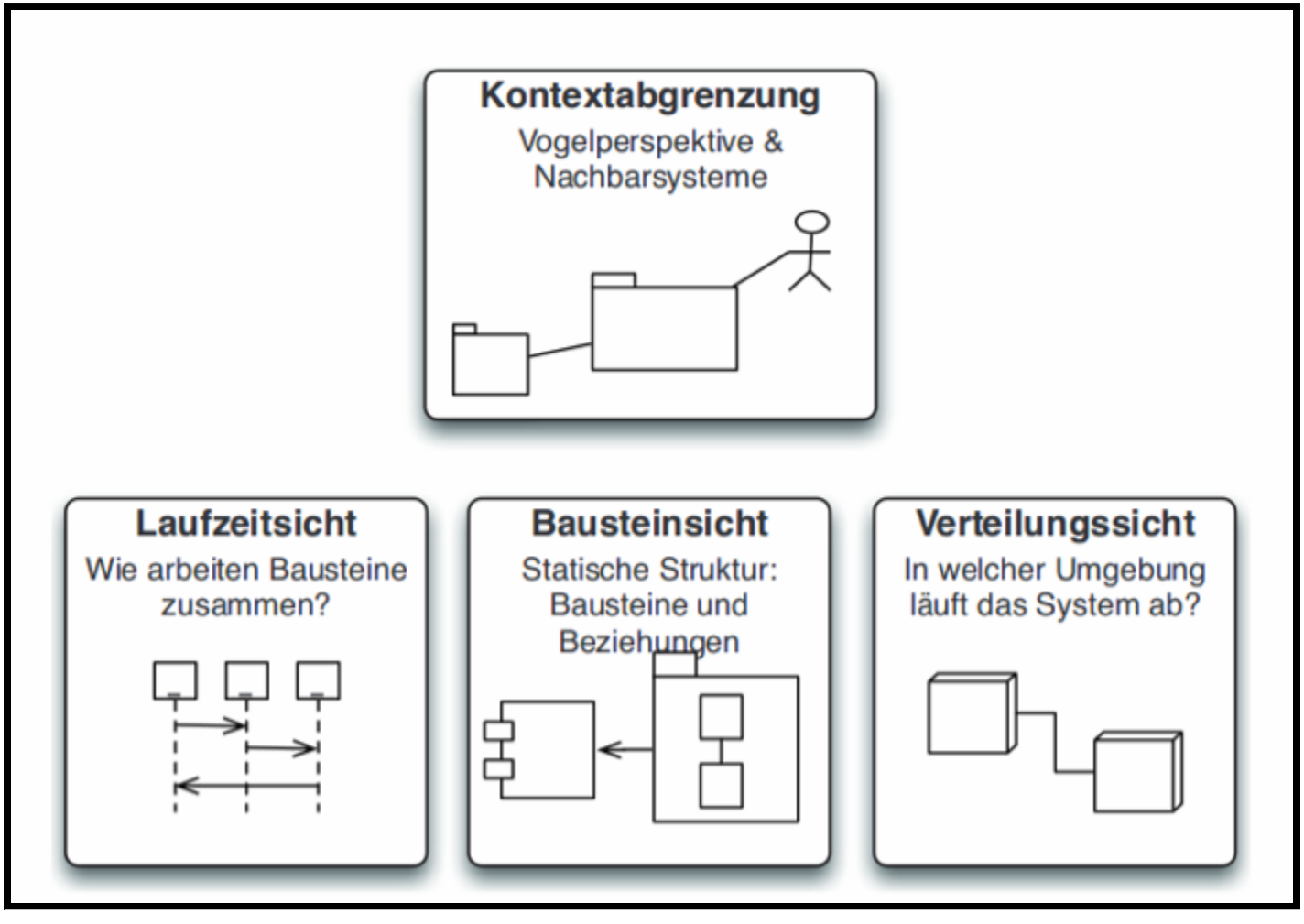


So erreichen Architekten ein gemeinsames Verständnis der Architektur innerhalb des Teams



Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

Überblick über die vier Sichten



Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

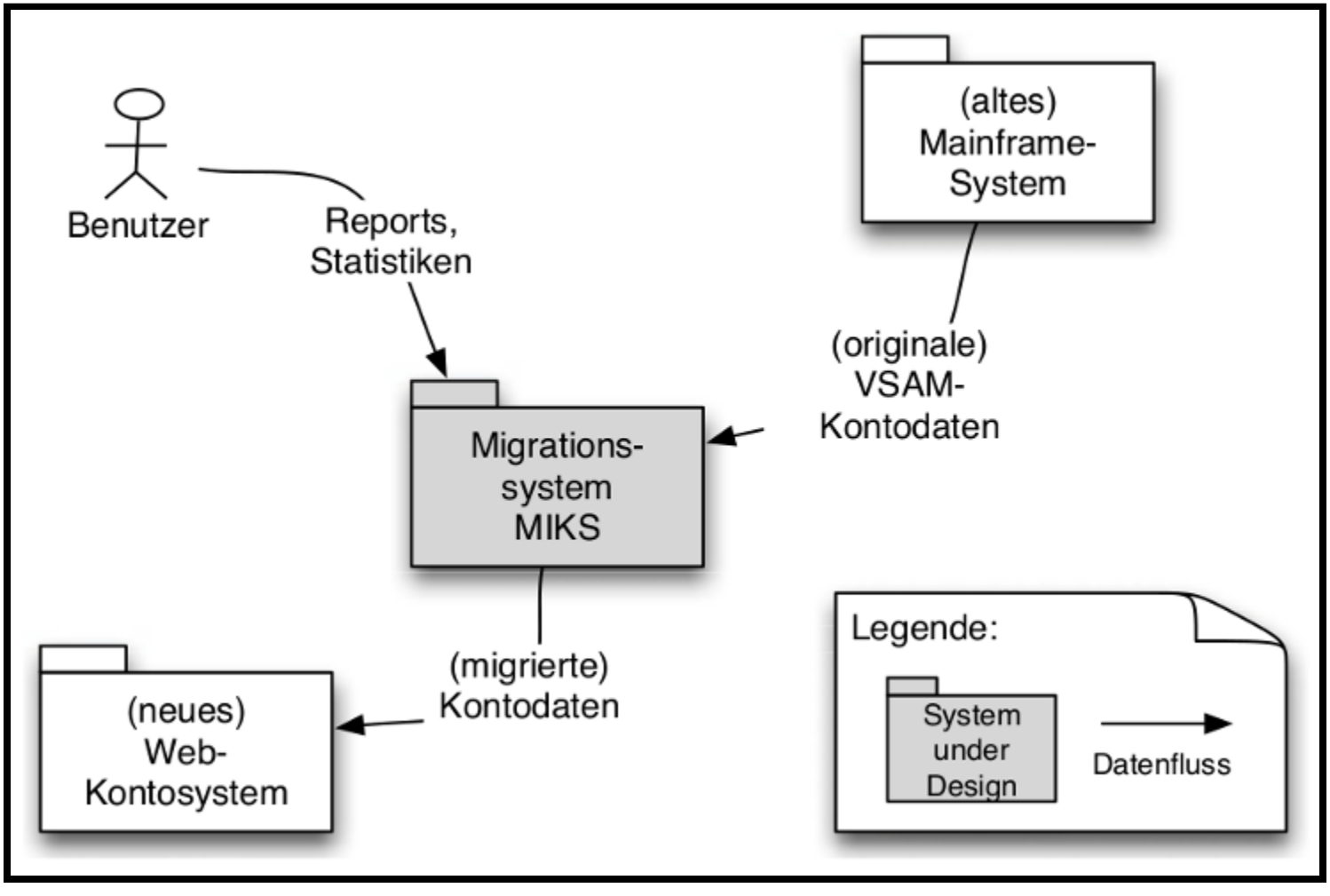
Kontextsicht

- Wie ist das System in seine Umgebung eingebettet?
- zeigt das System als Blackbox in seinem Kontext aus der Vogelperspektive

Kontextsicht - Enthaltene Informationen:

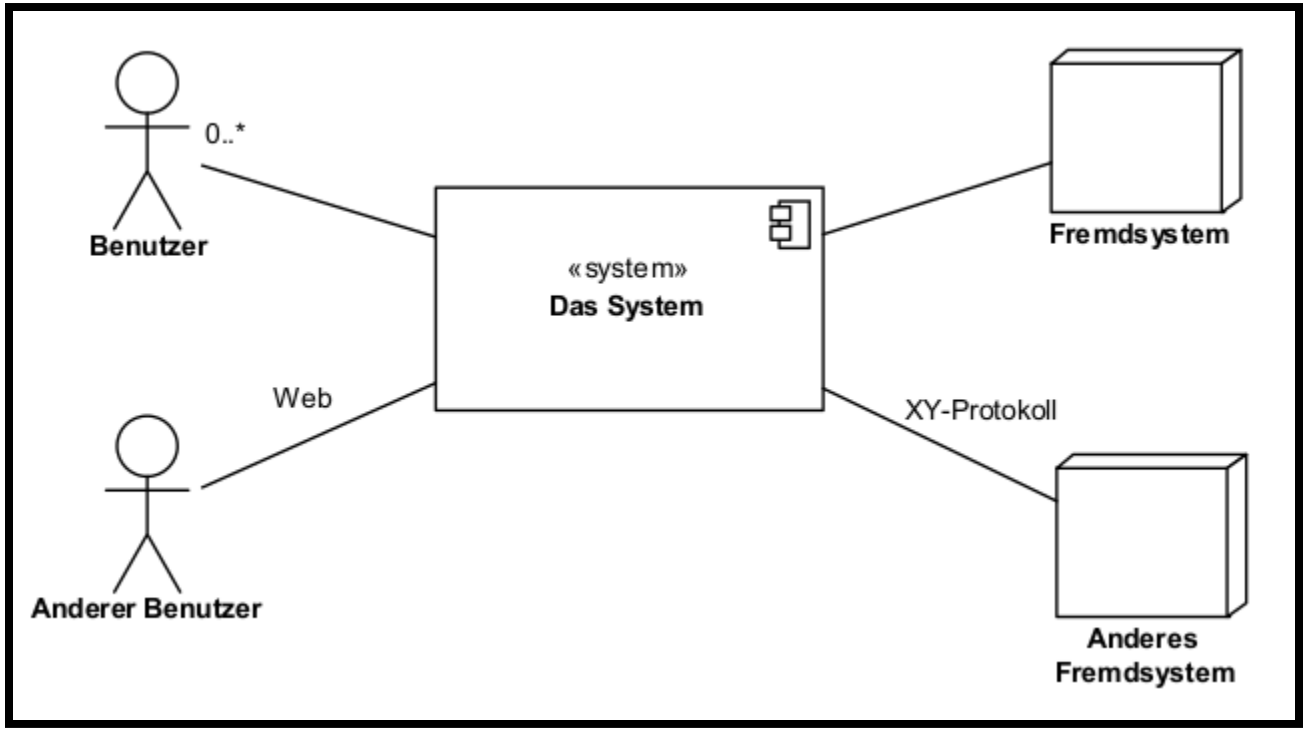
- Schnittstellen zu Nachbarsystemen
- Interaktion mit wichtigen Stakeholdern
- wesentliche Teile der umgebenden Infrastruktur

Kontextsicht - Beispiel



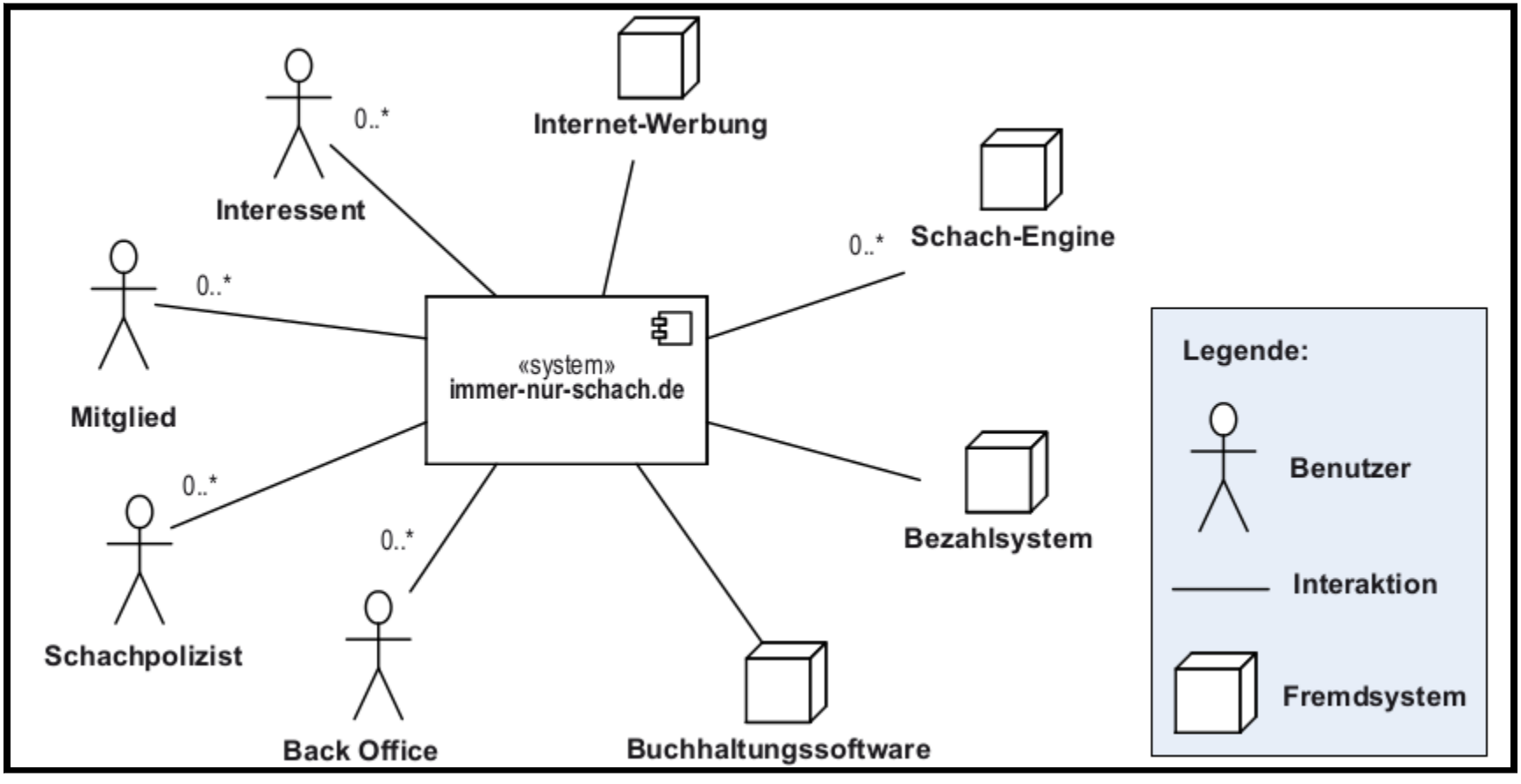
Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

Kontextsicht - Beispiel



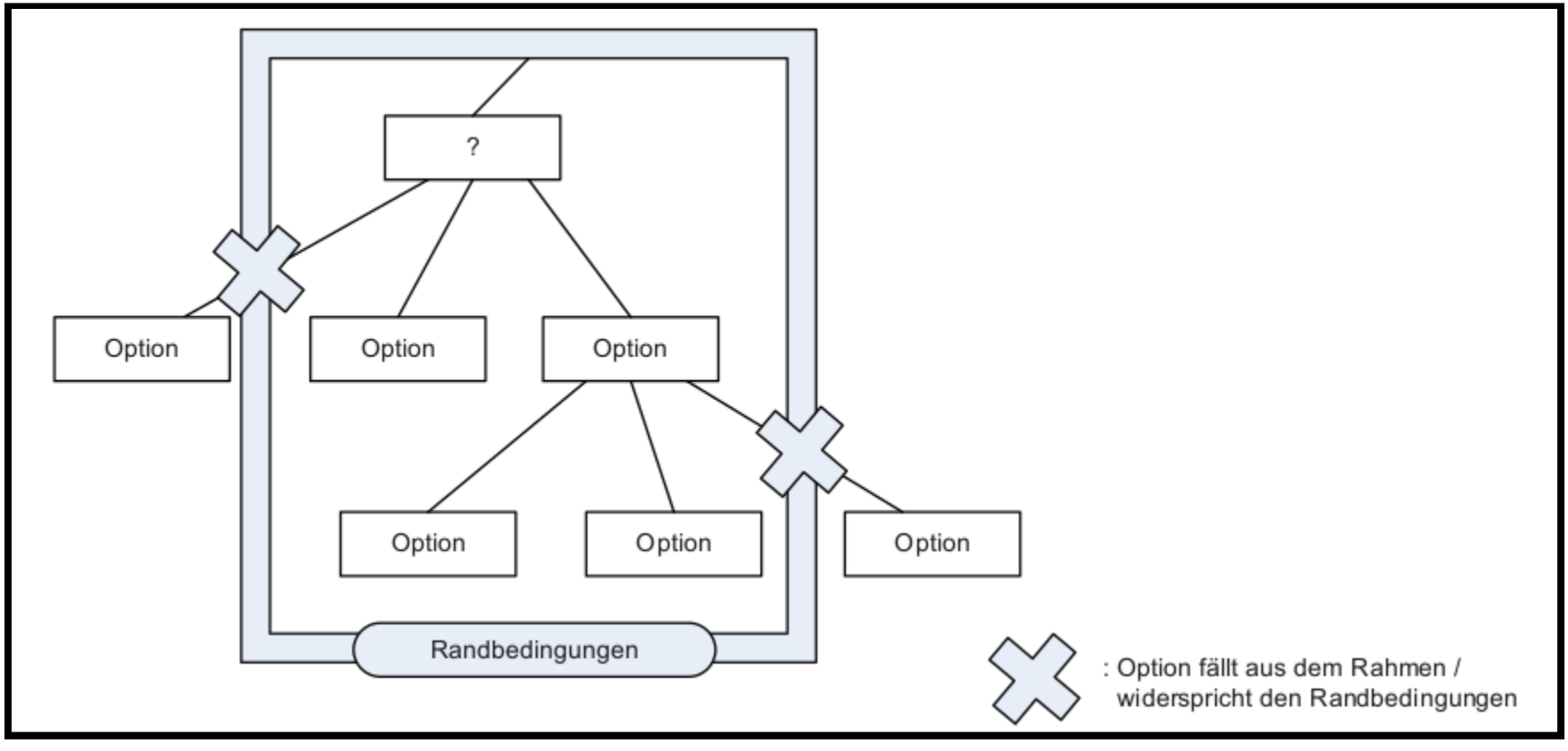
Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Kontextsicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Kontextsicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

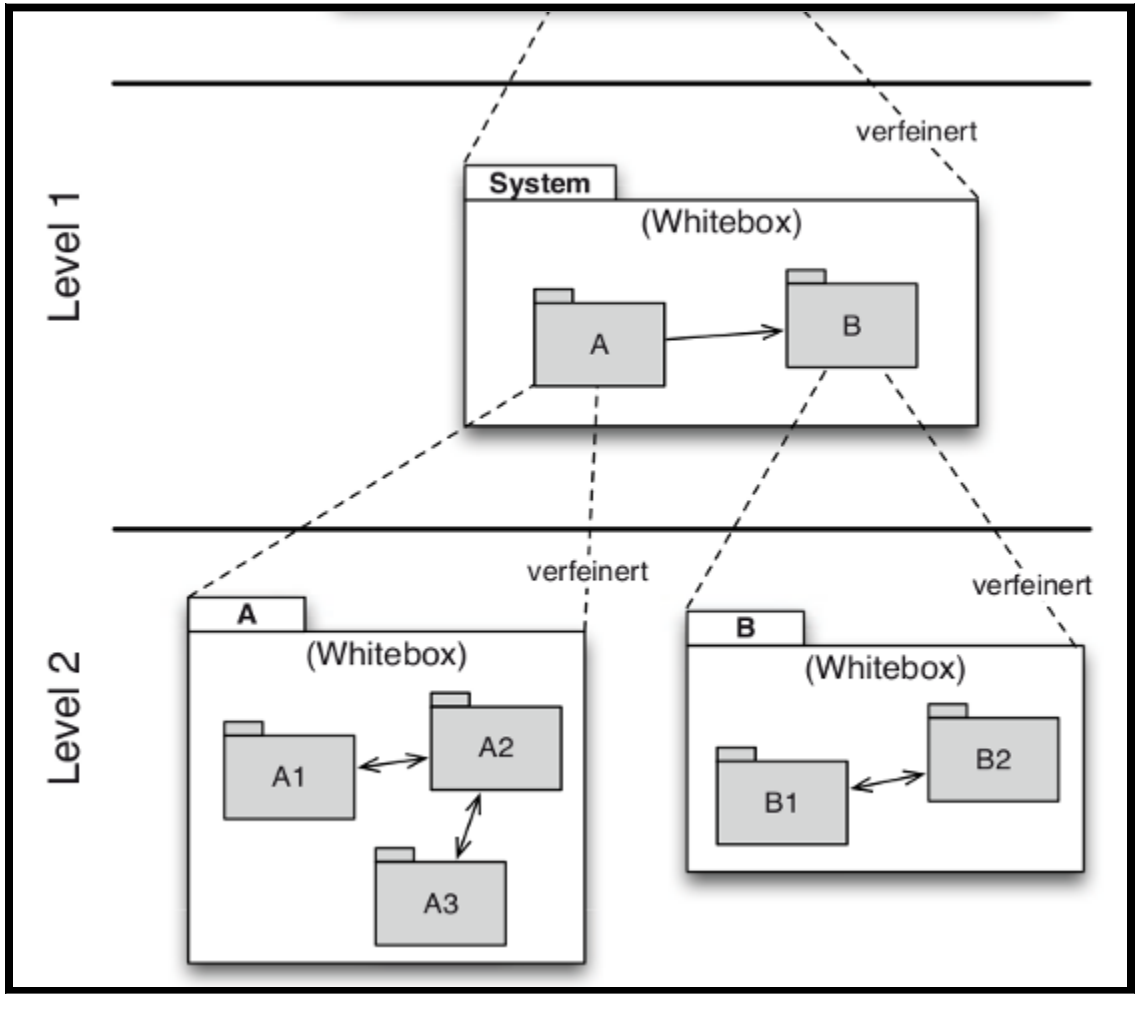
Bausteinsicht

- Wie ist das System intern aufgebaut?
- unterstützt Auftaggeber und Projektleiter bei der Projektüberwachung
- dienent der Zuteilung von Arbeitspaketen
- dient als Referenz für Software-Entwickler

Bausteinsicht - Enthaltene Informationen:

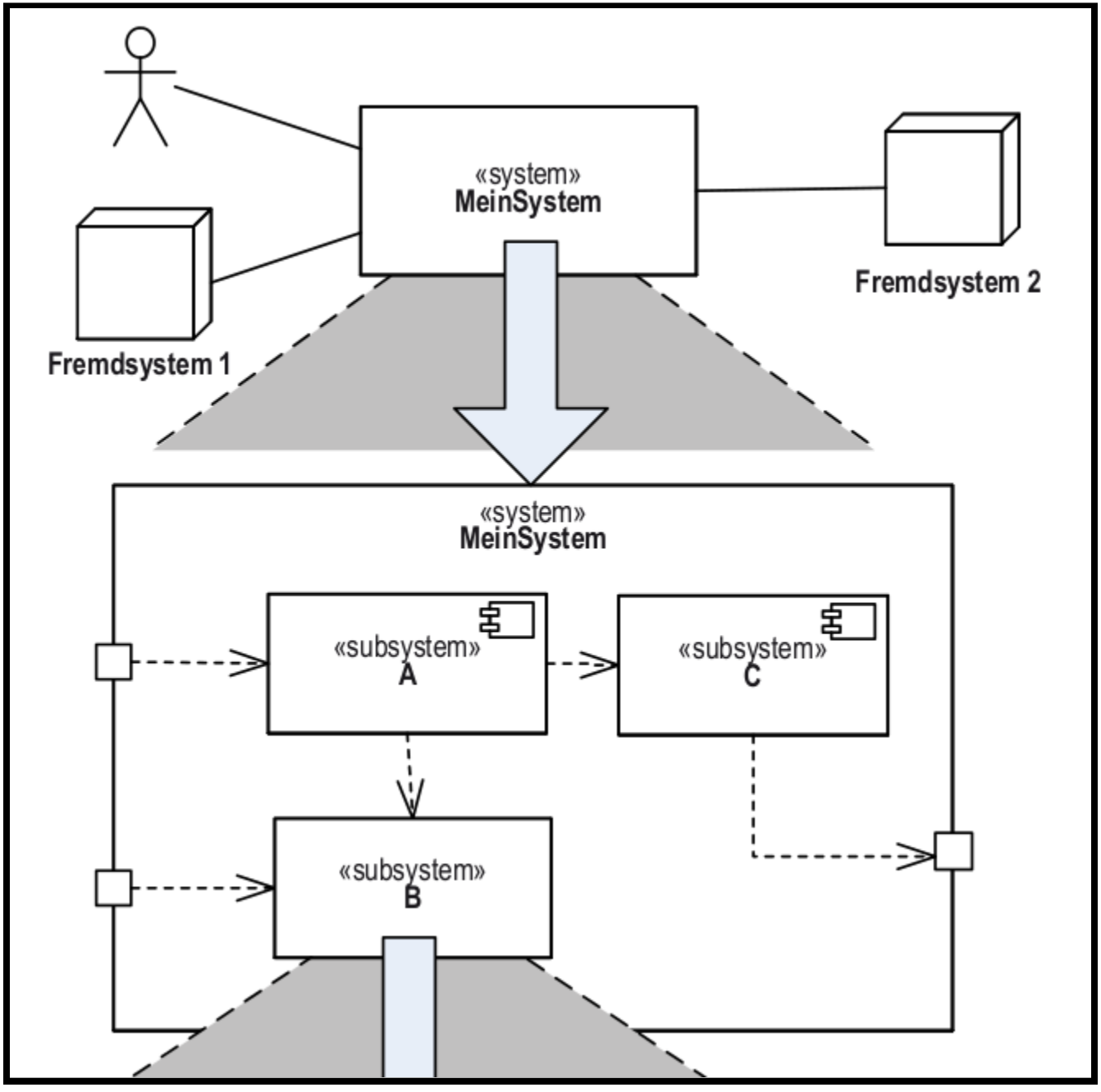
- statische Strukturen der Bausteine des Systems
- Subsysteme
- Komponenten und deren Schnittstellen

Bausteinsicht - Beispiel



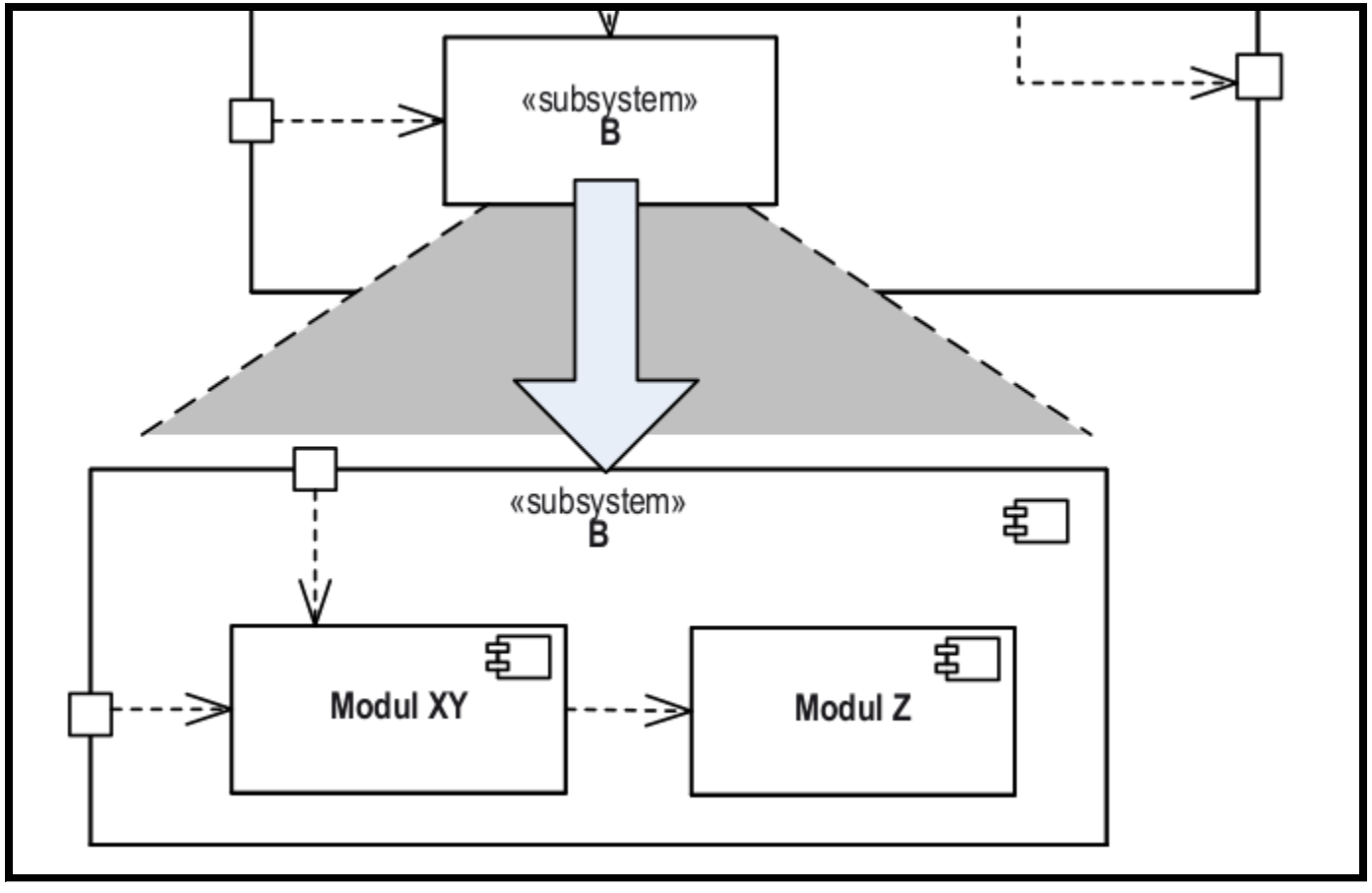
Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

Bausteinsicht - Beispiel



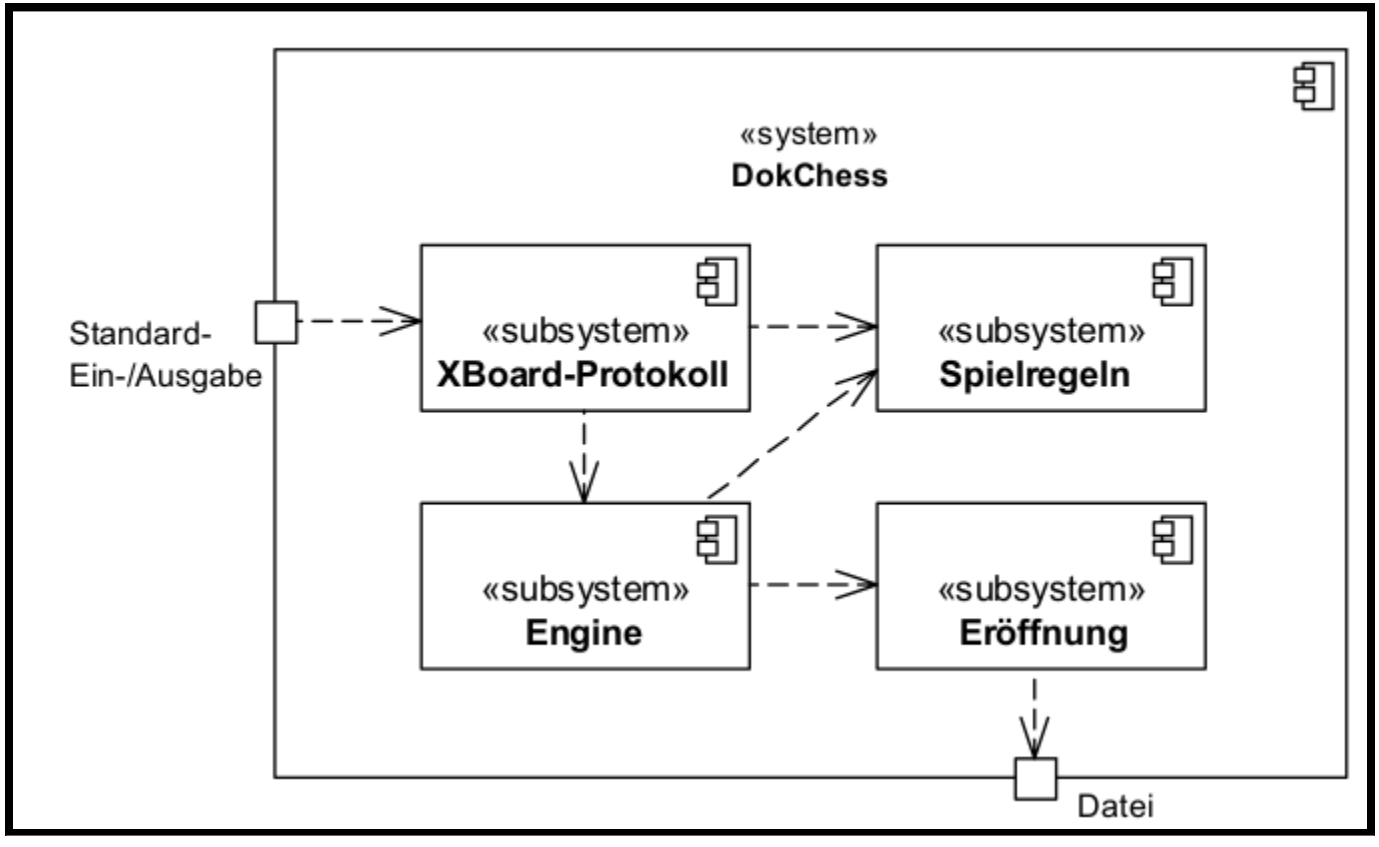
Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Bausteinsicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Bausteinsicht - Beispiel

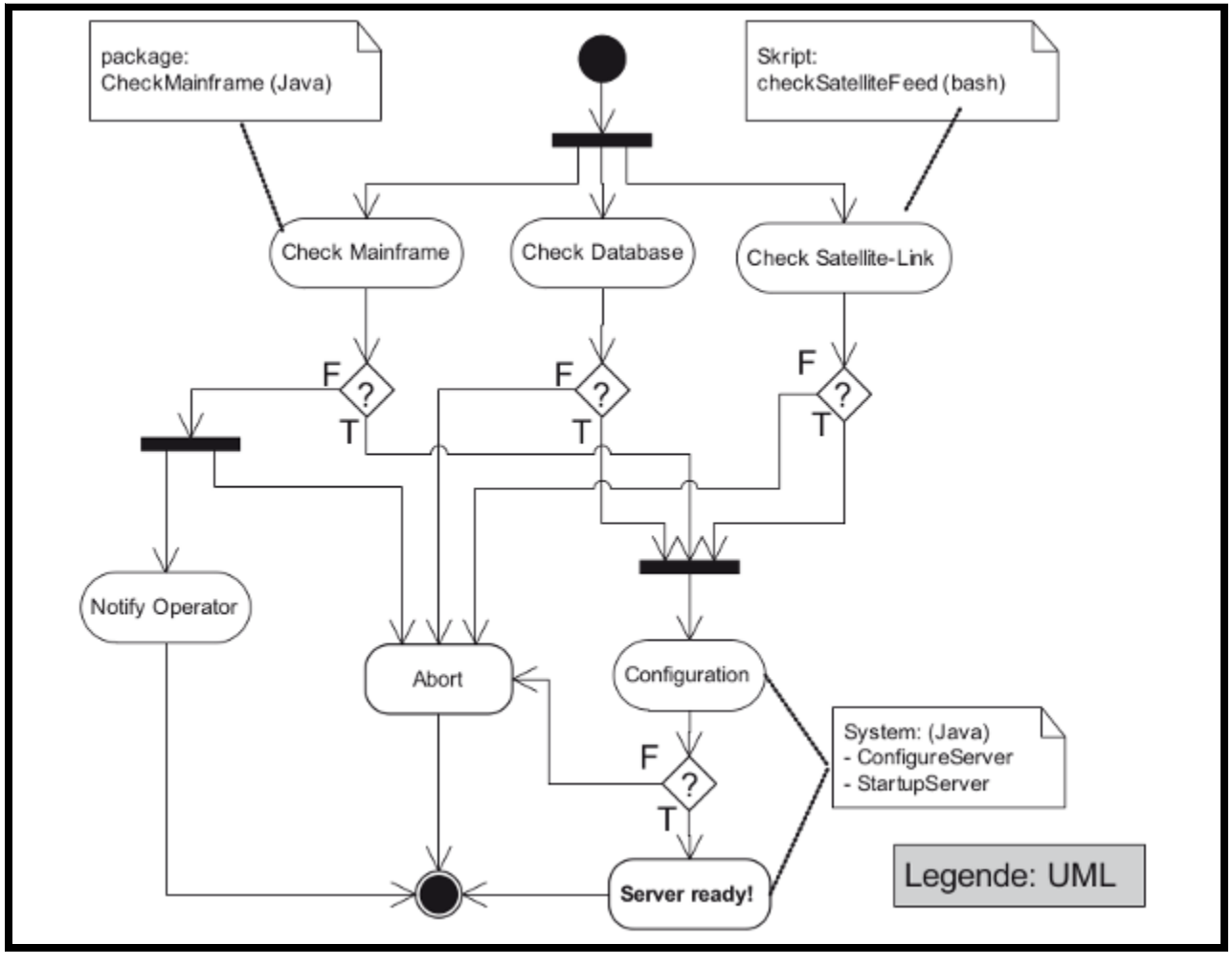


Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Laufzeitsicht

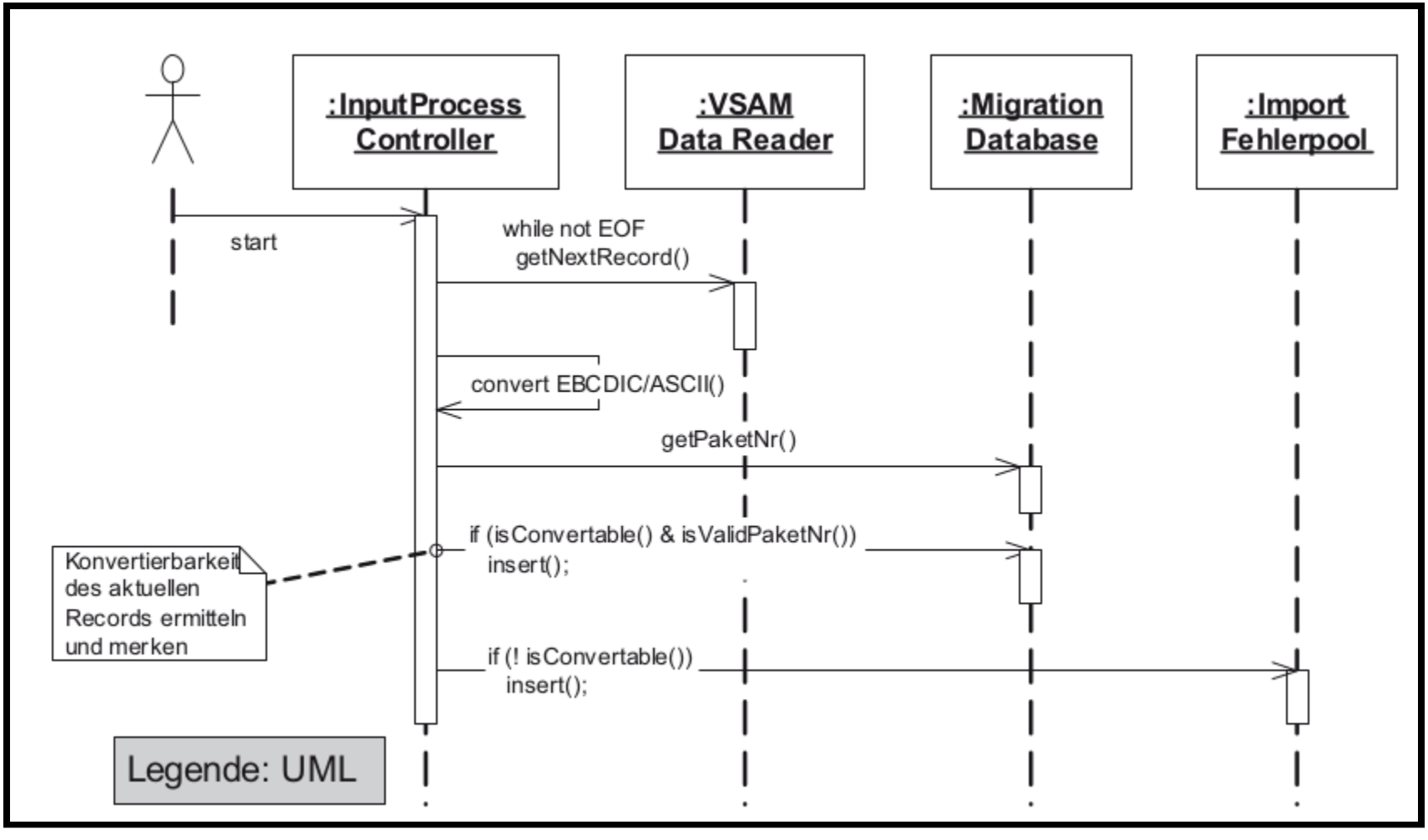
- Wie läuft das System ab?
- Welche Bausteine des Systems existieren zur Laufzeit?
- Wie wirken die Bausteine zusammen?

Laufzeitsicht - Beispiel



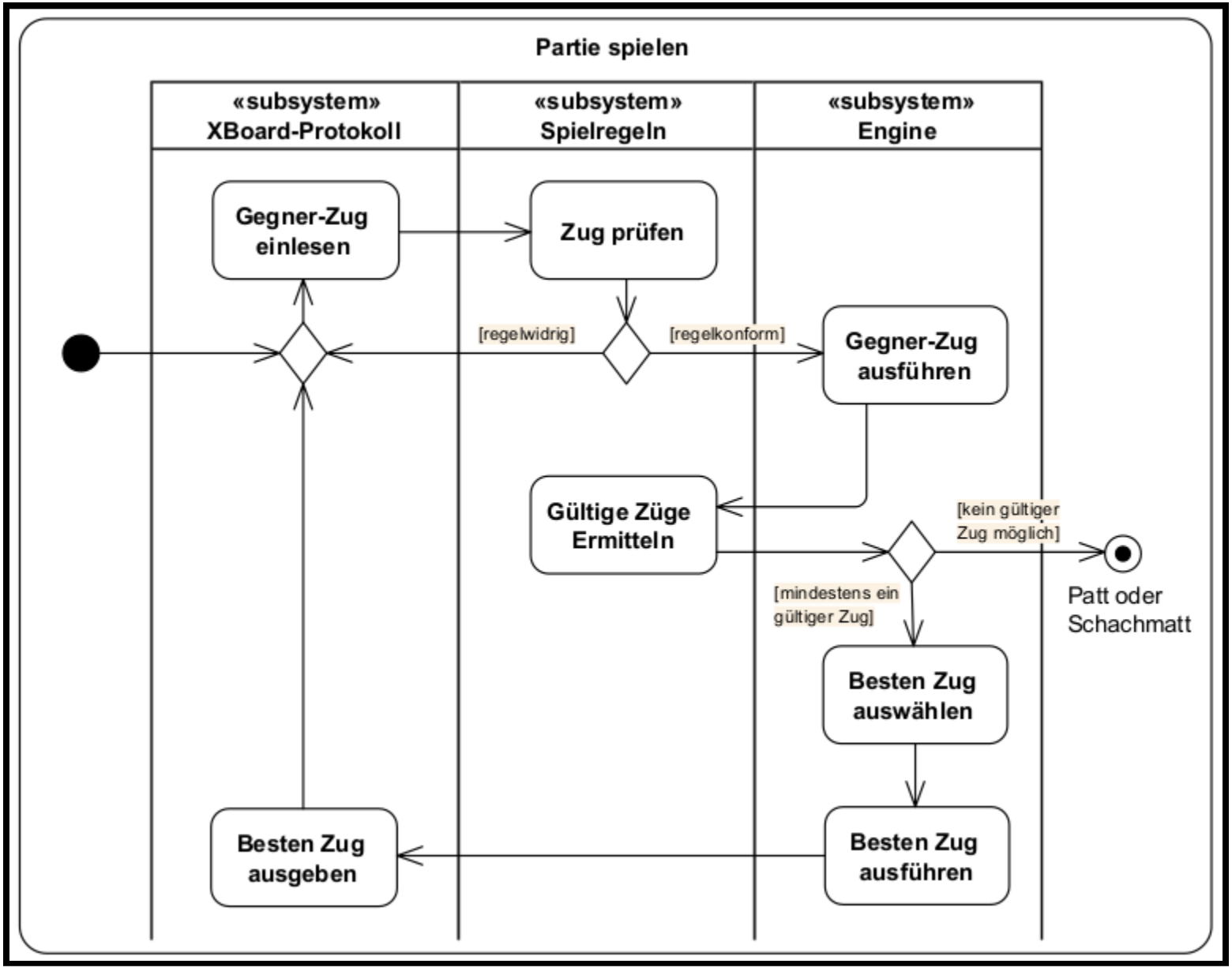
Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

Laufzeitsicht - Beispiel



Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

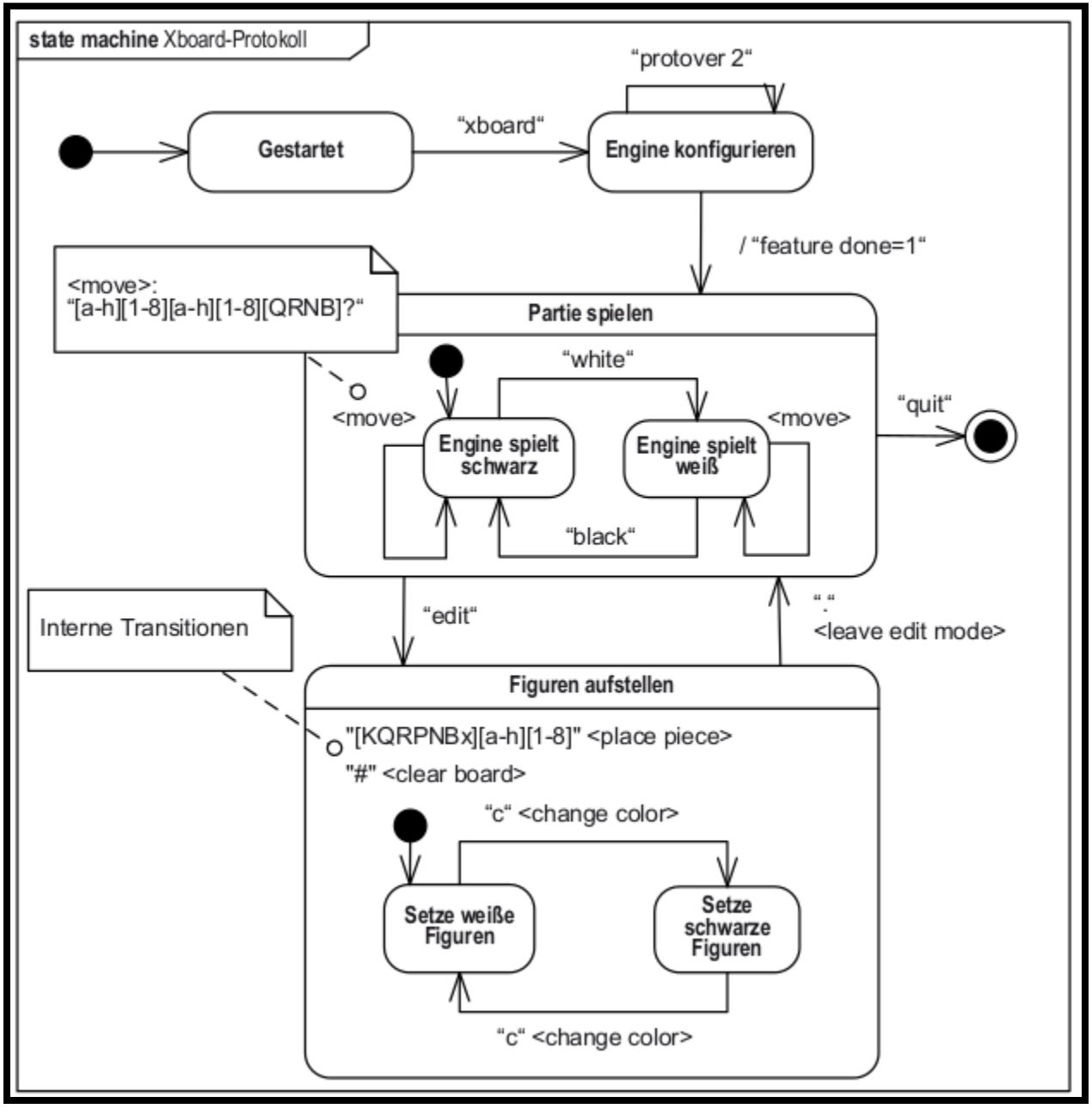
Laufzeitsicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan

Zörner

Laufzeitsicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

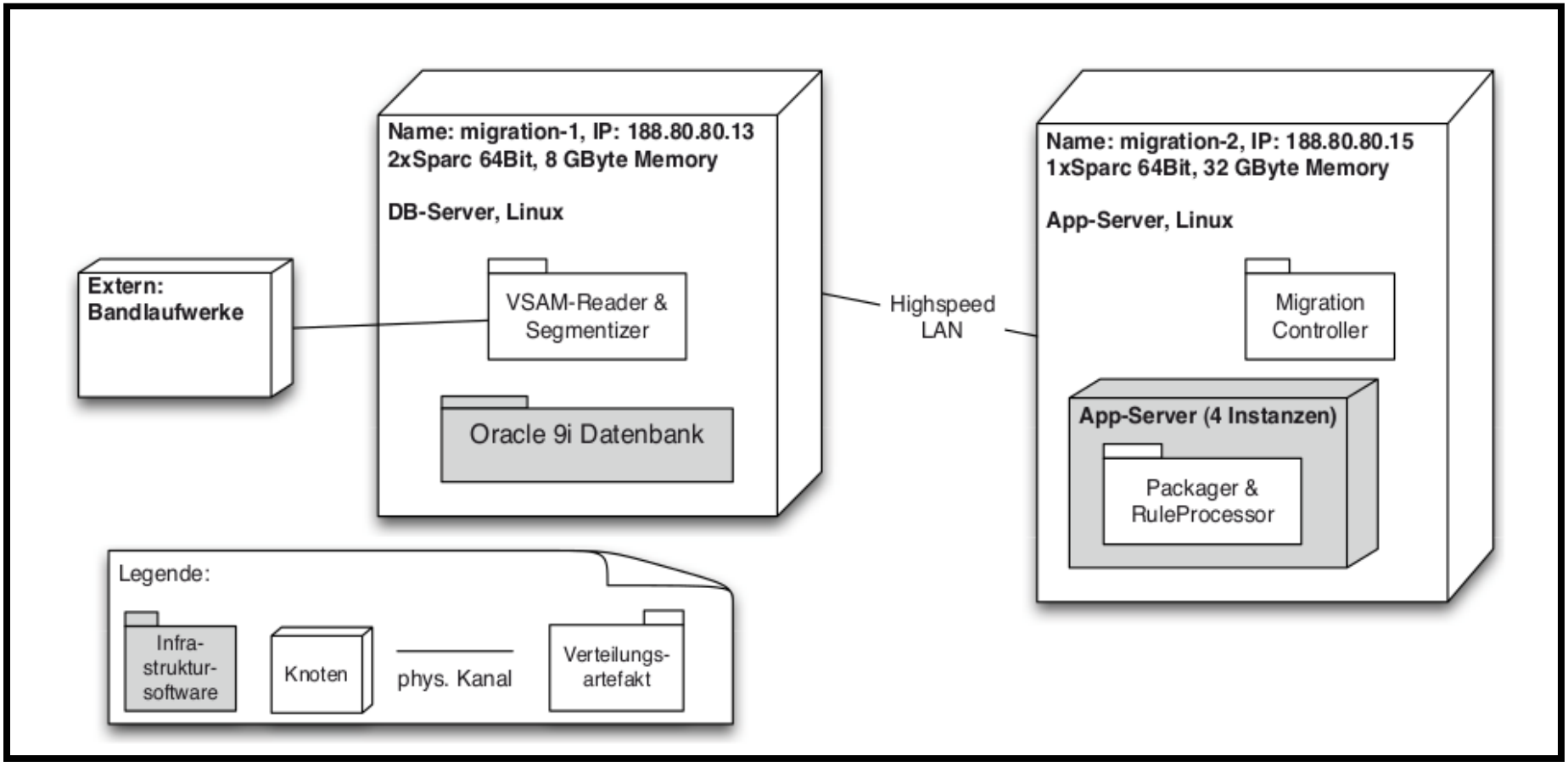
Verteilungssicht / Infrastruktursicht

- In welcher Umgebung läuft das System ab?
- zeigt das System aus Betreibersicht

Verteilungssicht - Enthaltene Informationen:

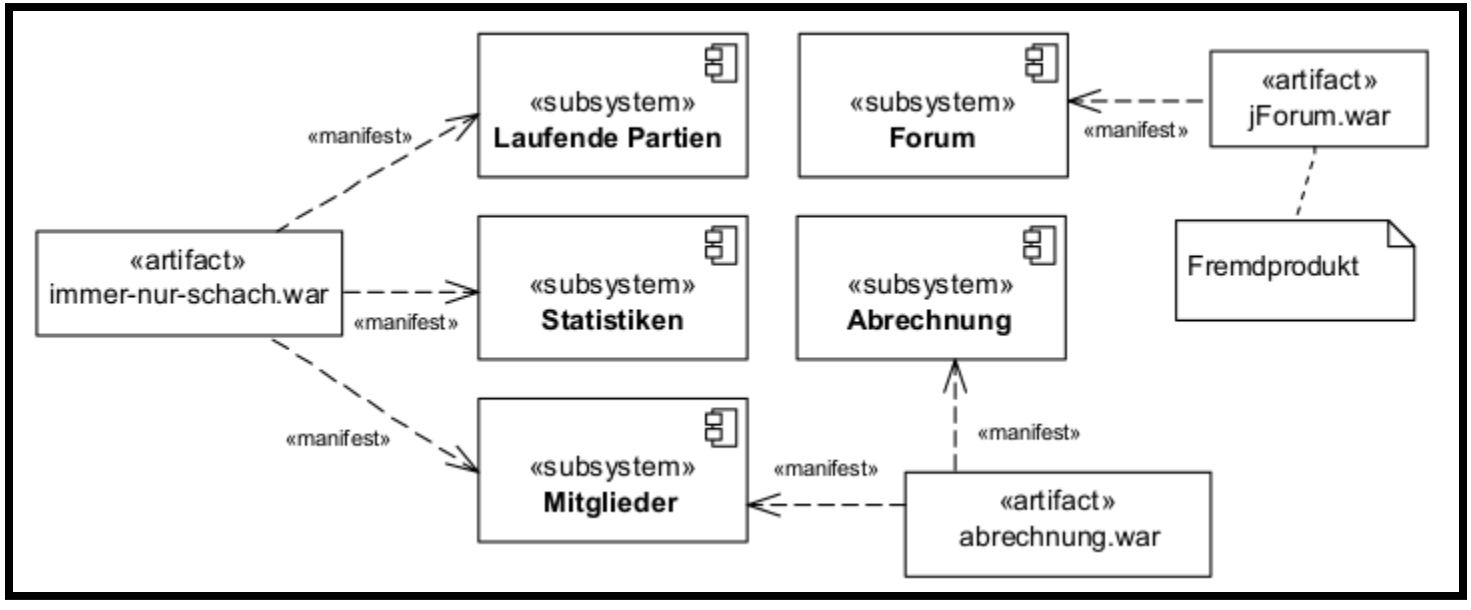
- Hardwarekomponenten: Rechner, Prozessoren
- Netztopologien
- Netzprotokolle
- sonstige Bestandteile der physischen Systemumgebung

Verteilungssicht - Beispiel



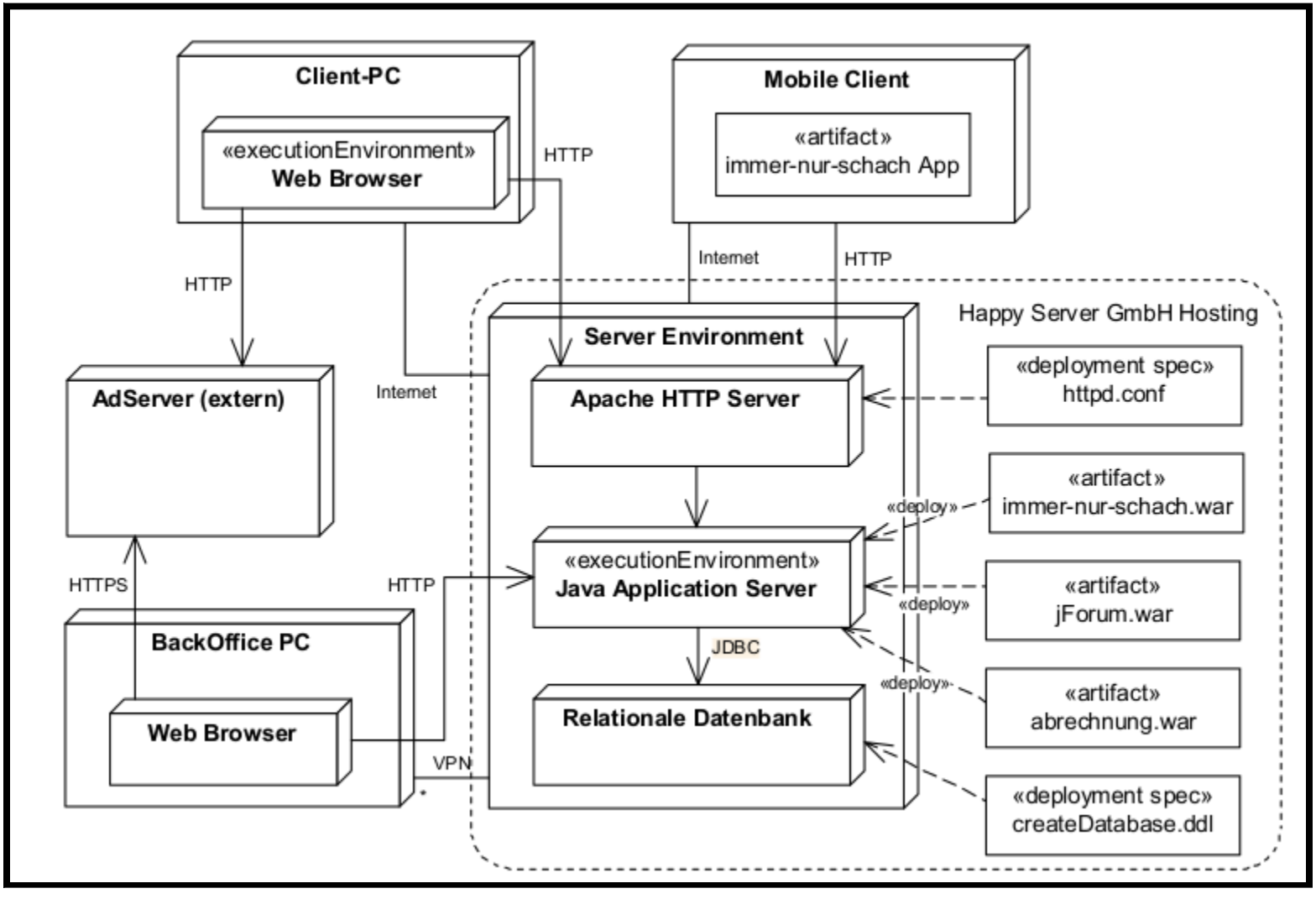
Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke

Verteilungssicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Verteilungssicht - Beispiel



Bildquelle: "Softwarearchitekturen dokumentieren" von Stefan Zörner

Was ist Softwarearchitektur?

Geschichte und Trends

Sichten auf Architekturen

Qualiät und andere nichtfunktionale Anforderungen

Architekturmuster

Dokumentation von Architekturen

Technologien und Frameworks

Fragen?

Unterlagen: ai2016.nils-loewe.de