Zurich University of Applied Sciences

School of Engineering

Bachelor of Science (BSc) in Informatik

Modul Advanced Software Engineering 1 (ASE1)

Software Architektur

# Grundlagen

Institut für Angewandte Informationstechnologie (InIT)

Walter Eich (eicw) / Matthias Bachmann (bacn)

https://www.zhaw.ch/de/engineering/institute-zentren/init/

# Agenda

## Magisches Viereck

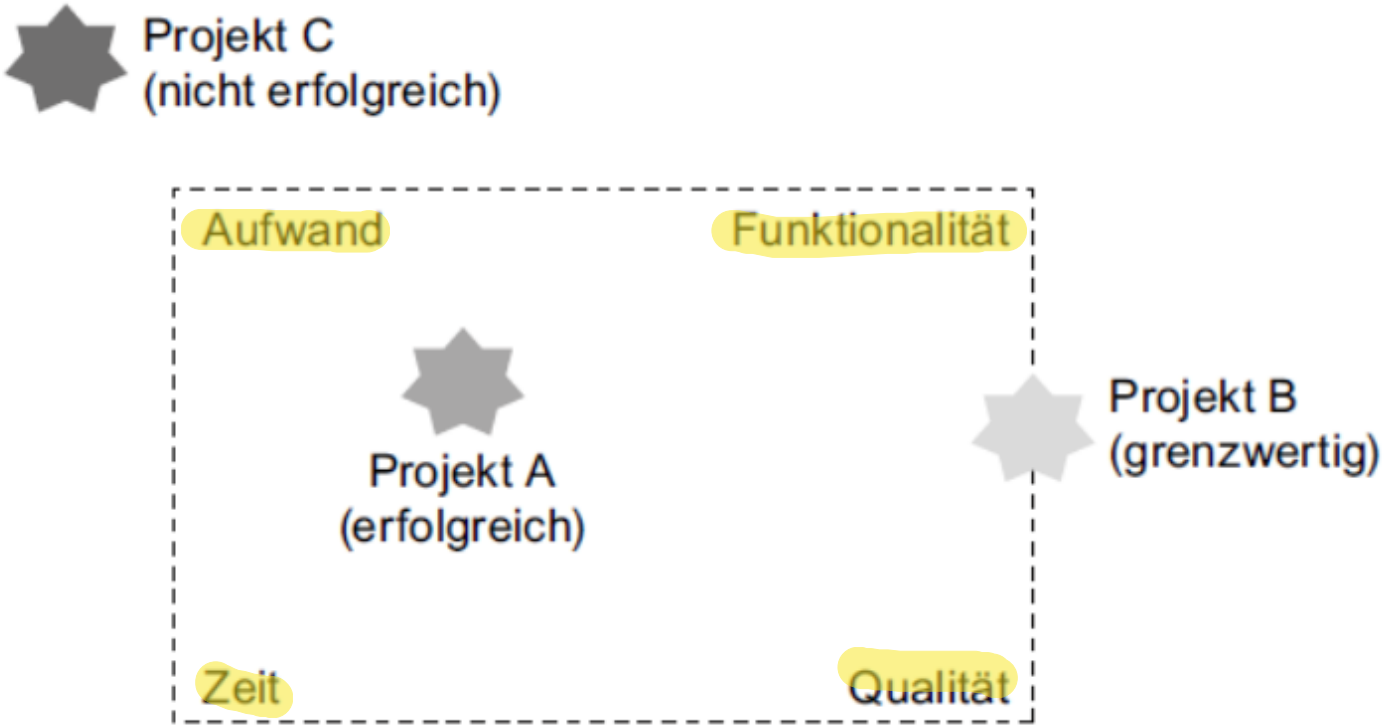


- Softwareintensive Systeme und Softwarearchitekturen
  - Was ist ein softwareintensives System? (LZ 1-10)
  - Ausprägung von softwareintensiven Systemen( (LZ 1-10)
  - Bedeutung der Software Architektur für ein System (LZ 1-7)
- Grundlegende Konzepte von Softwarearchitekturen
  - Was ist eine Software Architektur? (LZ 1.1)
  - Ausprägungen von softwareintensiven Systemen (LZ 1-1)
  - Qualität und Nutzen der Software Architektur (LZ 1-2)
- Der Softwareentwurf aus der Vogelperspektive
  - Ziele und Aufgabe es Software Architekturentwurfs (LZ 1-2)
  - Software Architekturentwurf im Überblick (LZ 1-3. KLZ 1-6)
  - Aufgaben des Software Architekten (LZ 1-4, LZ 1-5)

# Magisches Viereck erfolgreicher Softwareprojekte

Zurich University of Applied Sciences
zh aw School of Engineering

- CHAOS-Report der Standish Group;
  - Wir schaffen es immer noch nicht, wiederholbar qualitativ hochwertige Software zu erschwinglichen Kosten und im vorgegebenen Zeitfenster mit der notwendigen Funktionalität zu erstellen.



# Was ist Softwarearchitektur?



- Will man erfolgreich Software entwickeln, dann sind Requirements Engineering und Architekturentwurf zwei zentrale Schlüsselfaktoren.
  - Bei beiden ist das Risiko von gravierenden Fehlentwicklungen hoch, da früh – d.h. insbesondere bei noch eingeschränktem Wissensstand – Entscheidungen zu treffen sind,
- Aber was ist Softwarearchitektur eigentlich?
  - Was sind die Kernkonzepte?
  - Welche Vorgehensweisen und Ansätze gibt es?
- Inhalt dieses Kapitels:
  - Was ist ein softwareintensives System?
  - Was ist der Zusammenhang mit der Softwarearchitektur?
  - Zentralen Grundbegriffe von Softwarearchitekturen
  - Grundlegendes Vorgehen beim Architekturentwurf

# Was ist ein softwareintensives System? (1)



- Definition System (IEEE 610.12-1990, S. 73):
  - »**system.** A collection of components organized to accomplish a specific function or set of functions.«
- Definition Software (IEEE 610.12-1990, S. 66):
  - »**software.** Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.«

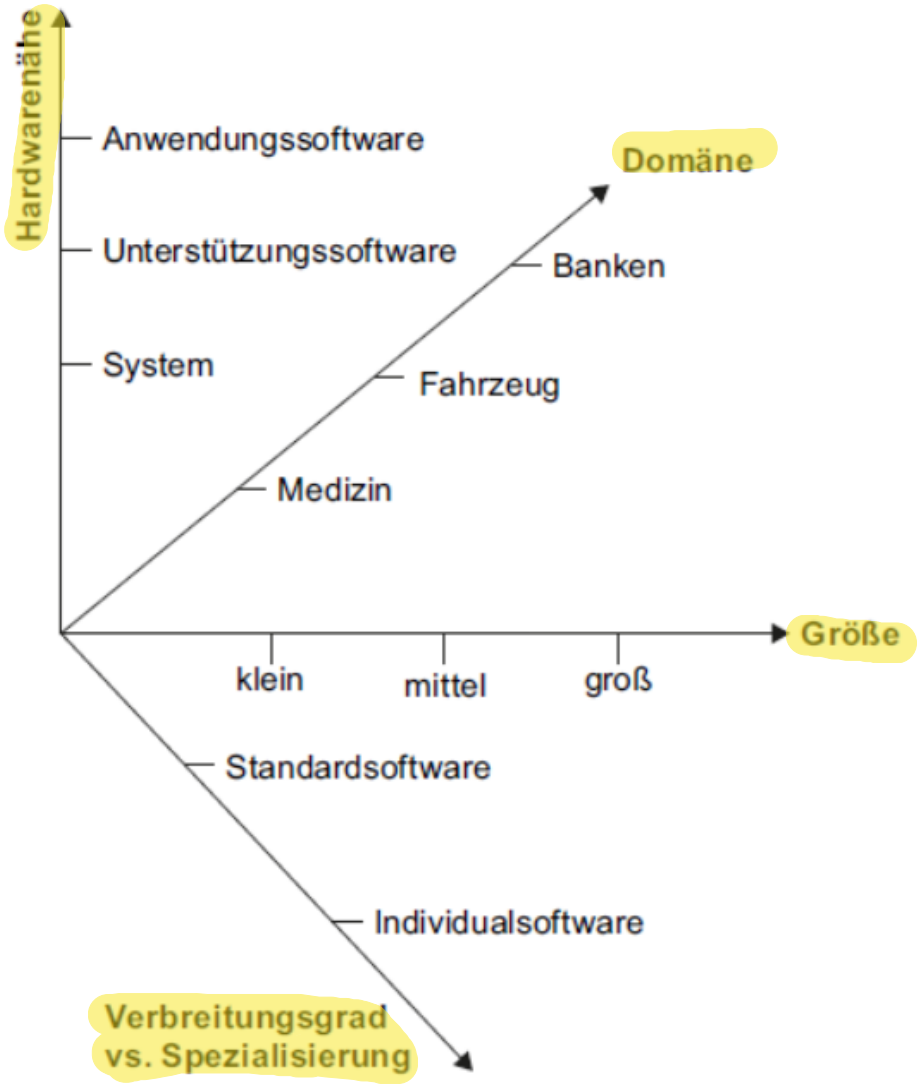
# Was ist ein softwareintensives System? (2)



- Definition softwareintensives System:
  - Ein **softwareintensives System** besteht aus einer Menge von Bausteinen, die so zusammengestellt sind, dass sie gemeinsam den Zweck des Systems erfüllen.
  - Bausteine, die vollständig oder zu wesentlichen Teilen aus Software bestehen, übernehmen dabei essenzielle Aufgaben zur Erfüllung des Systemzwecks.
  - Der Softwareanteil des Systems besteht dabei aus einer Menge von Programmen und weiteren Prozeduren und Daten sowie zugehöriger Dokumentation.

# Kategorisierung von softwareintensiven Systemen

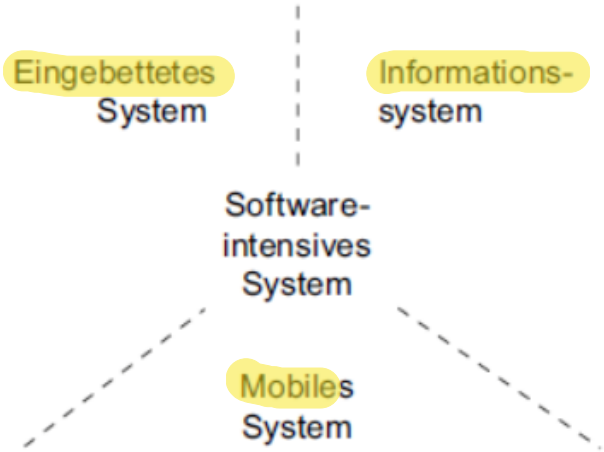




# Kategorisierung von softwareintensiven Systemen

Zurich University of Applied Sciences
zh aw School of Engineering

- Informationssystem
  - Verwaltung und Verarbeitung von Informationen
  - Beispiele: SAP, CAD
- Eingebettete Systeme
  - In physikalische Gegenstände eingebettet
  - Beispiele: Waschmaschinen, Werkzeugmaschinen
- Mobile Systeme
  - Autonome und personenspezifische Systeme
  - Beispiele: SmartPhones, Transportroboter



# Bedeutung der Software Architektur



- Strukturierung von grossen Systemen
- Jedes System hat eine Architektur
  - bewusst oder unbewusst entworfen
- Eine gute Architektur unterstützt das Refactoring aufgrund von geänderten Anforderungen
- Eine gute Architektur ist somit eine Voraussetzung für eine erfolgreiche Software Entwicklung
- Die Architektur definiert die tragenden Elemente der Software

# Grundlegende Konzepte von SW Architekturen



- Hierarchische Zerlegung in eine Menge von Bestandteilen bzw. Bausteinen (Komponenten)
- Bauplan für die Festlegung von Strukturen, technischen Konzepten und Entwurfsentscheidungen für den Bau von komplexen und umfangreichen Softwaresystemen

# Was ist eine Software Architektur?



- »The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.«

- Die **Softwarearchitektur** definiert die grundlegenden Prinzipien und Regeln für die Organisation eines Systems sowie dessen Strukturierung in Bausteinen und Schnittstellen und deren Beziehungen zueinander wie auch zur Umgebung. Dadurch legt sie Richtlinien für den gesamten Systemlebenszyklus, angefangen bei Analyse über Entwurf und Implementierung bis zu Betrieb und Weiterentwicklung, wie auch für die Entwicklungs- und Betriebsorganisation fest.

# Begriffe: Schnittstelle



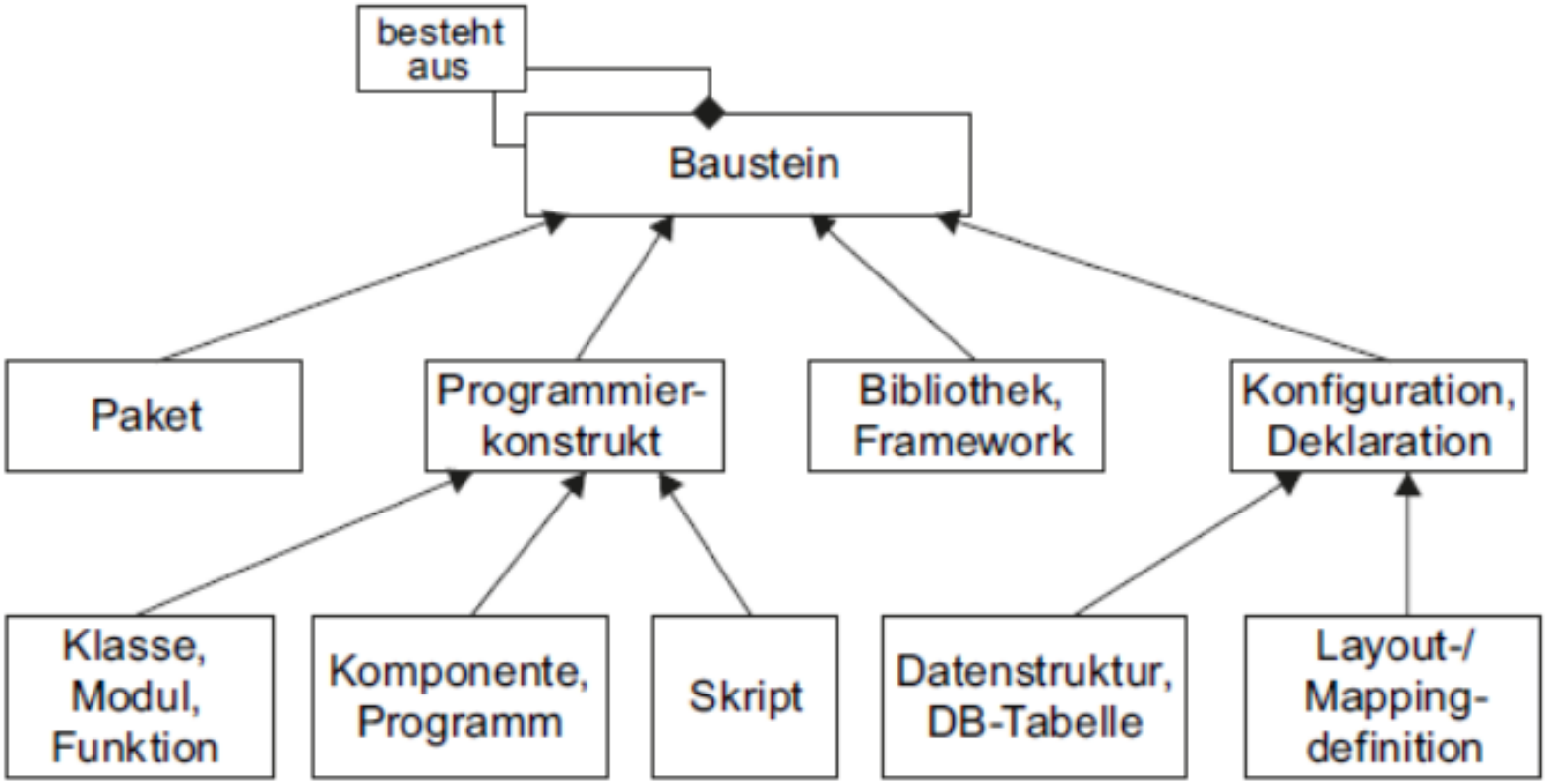
Eine **Schnittstelle** repräsentiert einen wohldefinierten Zugangspunkt zum System oder dessen Bausteinen.

- Dabei beschreibt eine Schnittstelle die Eigenschaften dieses Zugangspunkts, wie z.B. Attribute, Daten und Funktionen.
- Ziel ist es, diese Eigenschaften möglichst präzise mit allen notwendigen Aspekten zu definieren,
  - wie z.B. Syntax,
  - Datenstrukturen,
  - funktionales Verhalten,
  - Fehlerverhalten,
  - nichtfunktionale Eigenschaften,
  - Nutzungsprotokoll der Schnittstelle,
  - Technologien, Randbedingungen und Semantik.

# Begriffe: Bausteine (1)



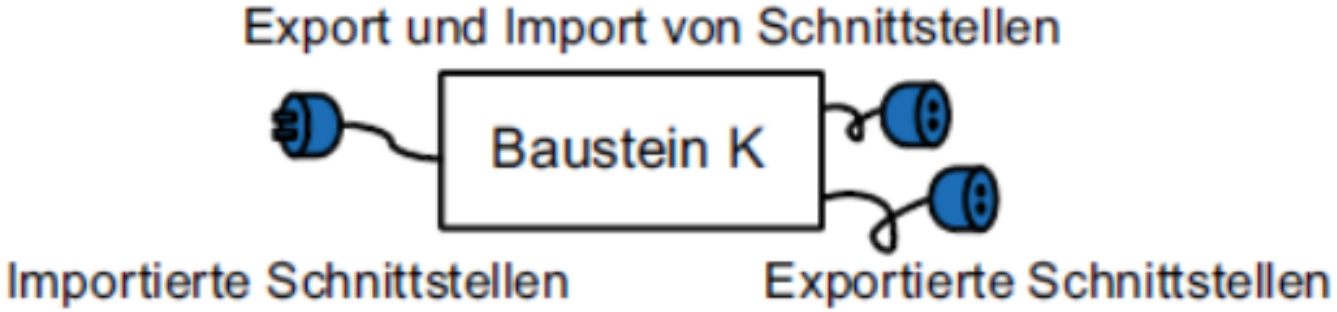
- Beispiele für Bausteine
  - Er beinhaltet sämtliche Software- oder Implementierungsartfakte, die letztendlich Abstraktionen von Quellcode darstellen.



# Begriffe: Bausteine (2)



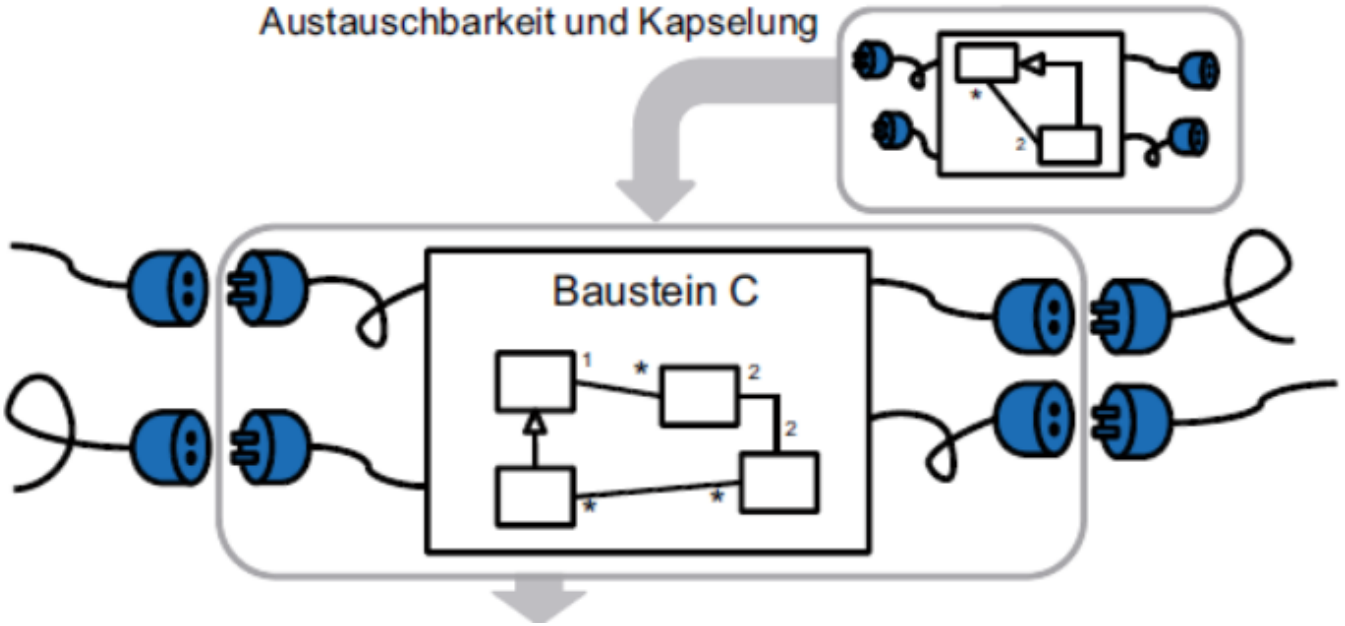
- **Export und Import von Schnittstellen**
  Ein **Baustein** bietet Schnittstellen an, die er im Sinne eines Vertrags garantiert.
  - Diese Garantie gilt aber erst, wenn die von ihm benötigten Schnittstellen im Rahmen einer entsprechenden Konfiguration zur Verfügung gestellt werden



# Begriffe: Bausteine (2)



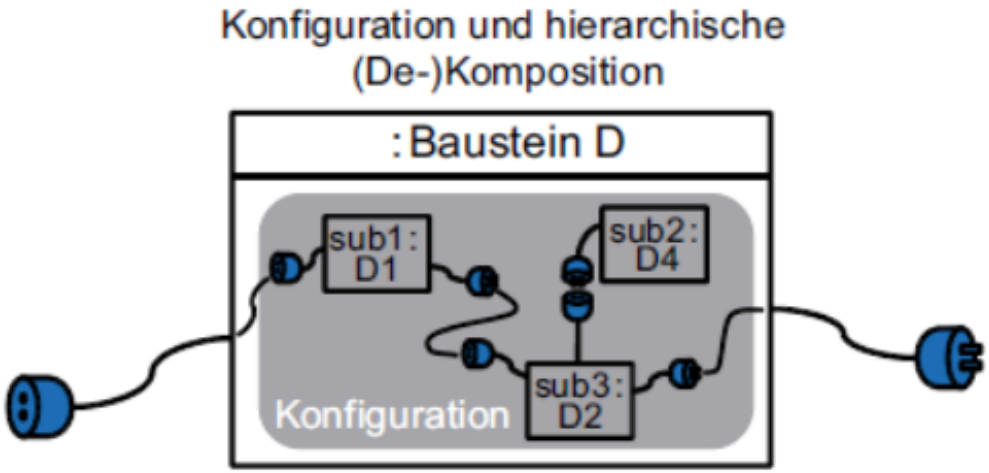
- **Kapselung und Austauschbarkeit**
  Über die angebotenen und benötigten Schnittstellen kapselt der Baustein die Implementierung dieser Schnittstellen.
  - Daher kann er durch andere Bausteine ersetzt werden, die dieselben Schnittstellen exportieren und gegebenenfalls importieren.



# Begriffe: Bausteine (3)



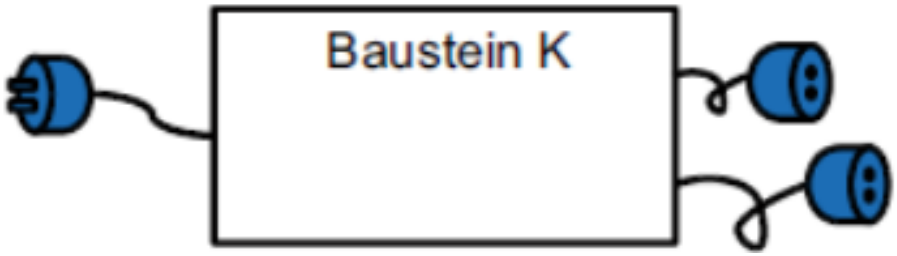
- **Konfiguration und hierarchische (De-) Komposition** Bausteine sind die Einheit der hierarchischen (De-) Komposition eines softwareintensiven Systems.
  - Das heisst, ein (Super-)Baustein kann durch eine entsprechende Konfiguration von anderen (Sub-) Bausteinen und deren Beziehungen implementiert werden.
  - Wir sagen dann auch, dieser (Super-)Baustein kapselt die (Sub-) Bausteine.
  - Dabei kann diese Kapsel auch äussere Schnittstellen auf innere delegieren und umgekehrt.
    So werden die Beziehungen zwischen den Bausteinen definiert.



# Sichten auf einen Baustein (1)



- In der **Blackbox-Sicht** sieht man lediglich die von dem Baustein exportierten und importierten Schnittstellen. Das ist die Sicht des Bausteinnutzers.
  - Diese Sicht respektiert das Geheimnisprinzip, d.h., sie verbirgt das (private) Innenleben des Bausteins.

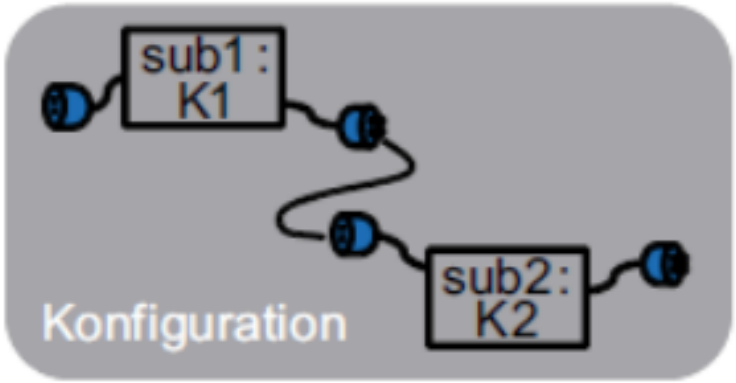



Blackbox-Sicht

# Sichten auf einen Baustein (2)



- Die **Greybox-Sicht** (auch: Konfiguration) zeigt, wie die importierten Schnittstellen von Bausteinen mit den exportierten Schnittstellen (i.d.R.) anderer Bausteine verschaltet werden.
- Das ist die Sicht des Bausteinkonfigurators.

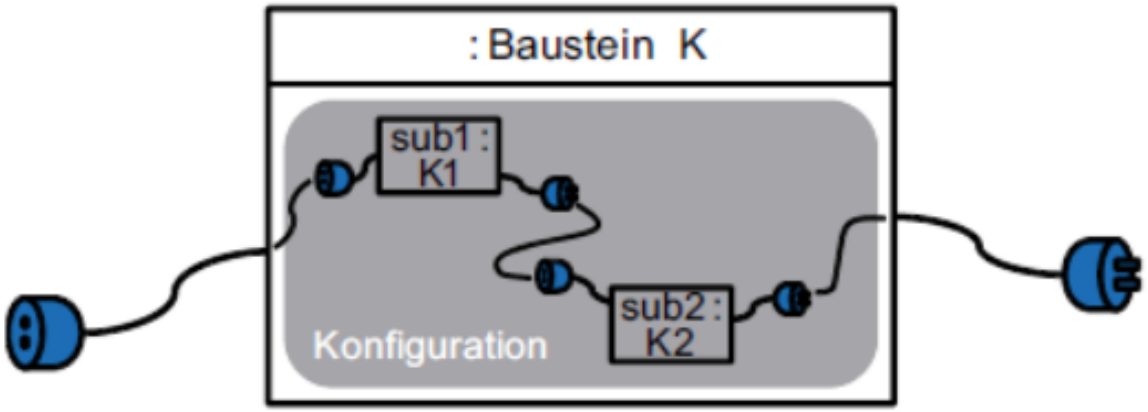



Greybox-Sicht

# Sichten auf einen Baustein (3)



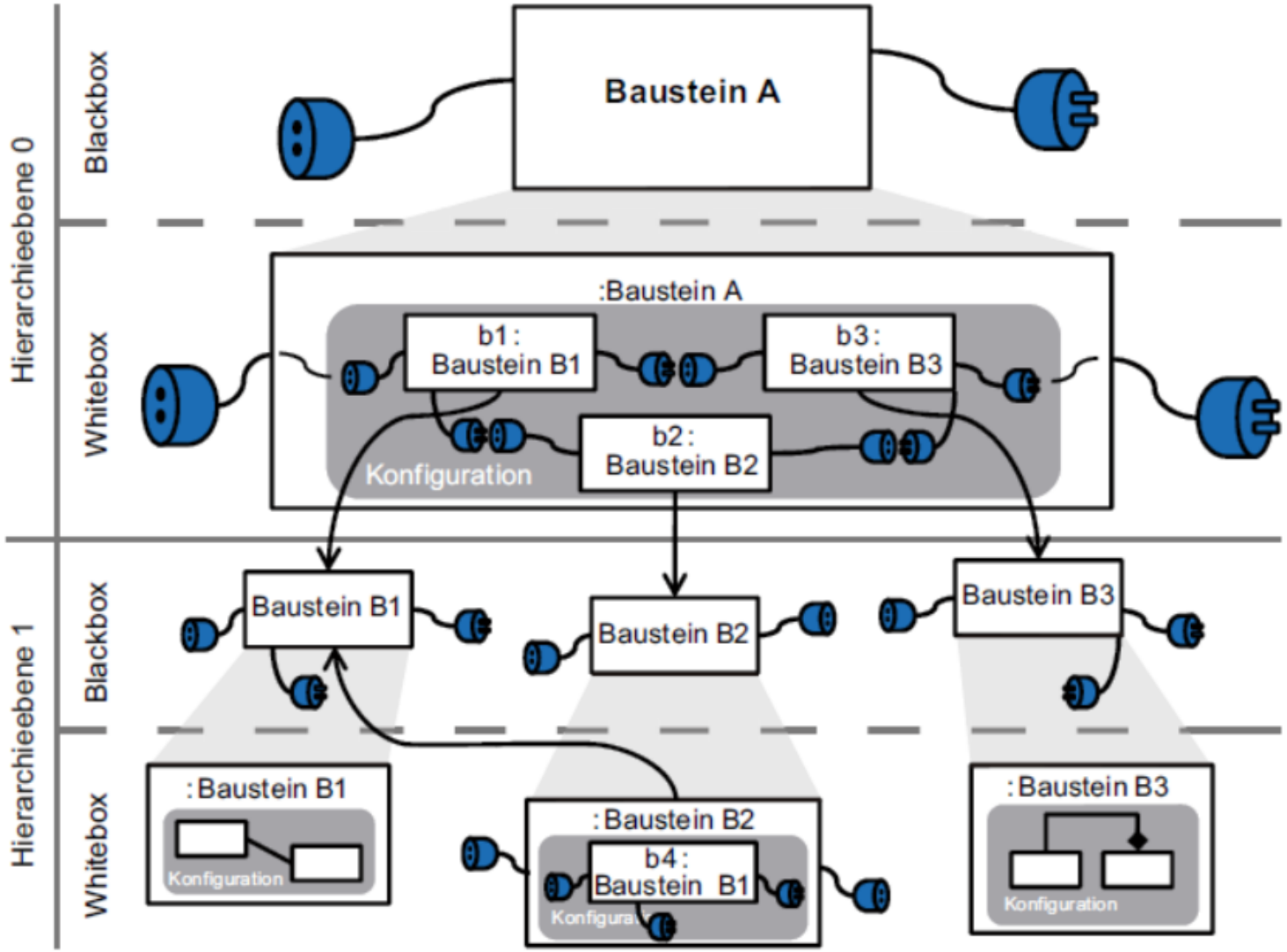
- Die **Whitebox-Sicht** (auch: Glassbox-Sicht) erlaubt einen Blick auf den inneren Aufbau des Bausteins, d.h. seine Zerlegung in die Konfiguration der Subbausteine oder eine andere Art der Implementierung und dabei auch die Delegation seiner exportierten und importierten Schnittstellen auf das Innenleben des Bausteins.
- Das ist die Sicht des Bausteinimplementierers.




Whitebox-Sicht

# Hierarchische (De-)Komposition einer Architektur





# Hierarchische (De-)Komposition

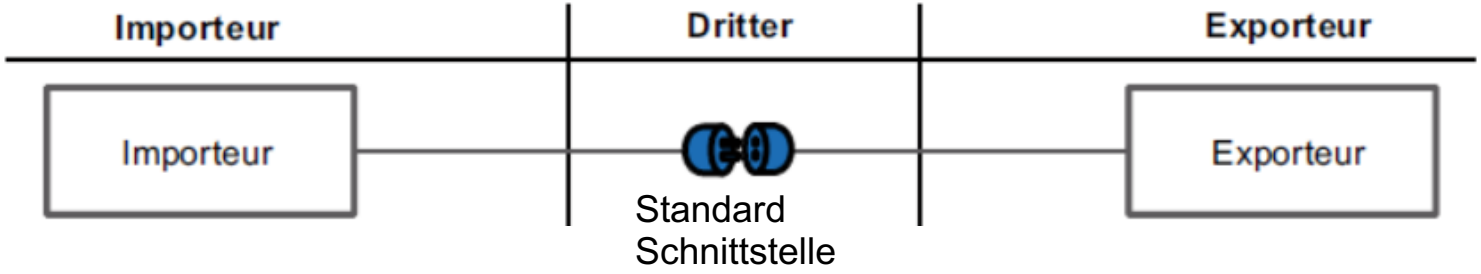


- Wie die vorhergehende Folie zeigt, kann die Blackbox-Sicht eines Bausteins A in einer darunter liegenden Whitebox-Sicht hierarchisch dekomponiert werden.
  - In dieser Whitebox-Sicht zerfällt der Baustein A in seine Subbausteine B1, B2 und B3, die jeweils in einer Greybox-Sicht dargestellt werden.
- Auch für die Subbausteine B1 bis B3 können dann wiederum entsprechende Blackbox- und Whitebox- Sichten auf einer darunter liegenden Ebene vorhanden sein.
- Die Hierarchie wird durch die Hierarchieebenen dargestellt.

# Wer definiert die Schnittstelle? (1)



- **Standardschnittstelle** Vereinbarung
  - Diese Schnittstelle wird von ausserhalb definiert. Sowohl der importierende als auch der exportierende Baustein halten sich daran.




- **Angebotene Schnittstelle** Angebot
  - Hier definiert der Baustein, der die Schnittstelle exportiert, die Schnittstelle. Neben der Standardschnittstelle ist das die meistverwendete Schnittstellenart.

# Wer definiert die Schnittstelle? (2)



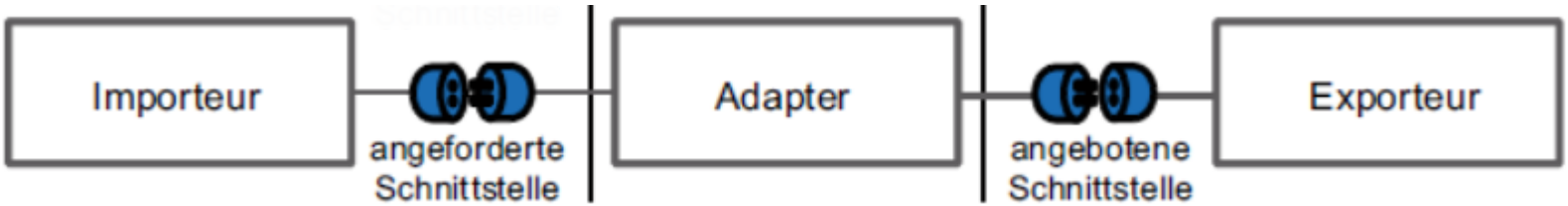
- Angeforderte Schnittstelle
  - Hier definiert der Baustein, der die Schnittstelle importiert, die Schnittstelle.
  - Gerade bei Frameworks ist diese Konstellation häufig anzutreffen.
  - Über derartige Schnittstellen kann man Bausteine mit spezifischer Funktionalität in ein Programmgerüst einbringen.

Beispiel

# Wer definiert die Schnittstelle? (3)



- Unabhängige Schnittstellen
  - In diesem Fall haben exportierender und importierender Baustein jeweils eigene Schnittstellen definiert. Damit erhöht sich die Entkopplung der Bausteine.
  - So können sie unabhängig voneinander entwickelt und getestet werden.
  - Allerdings führt dies dazu, dass die Schnittstellen im Laufe der Zeit nicht mehr identisch sind. Deshalb muss ein Adapter dazwischengesetzt werden.

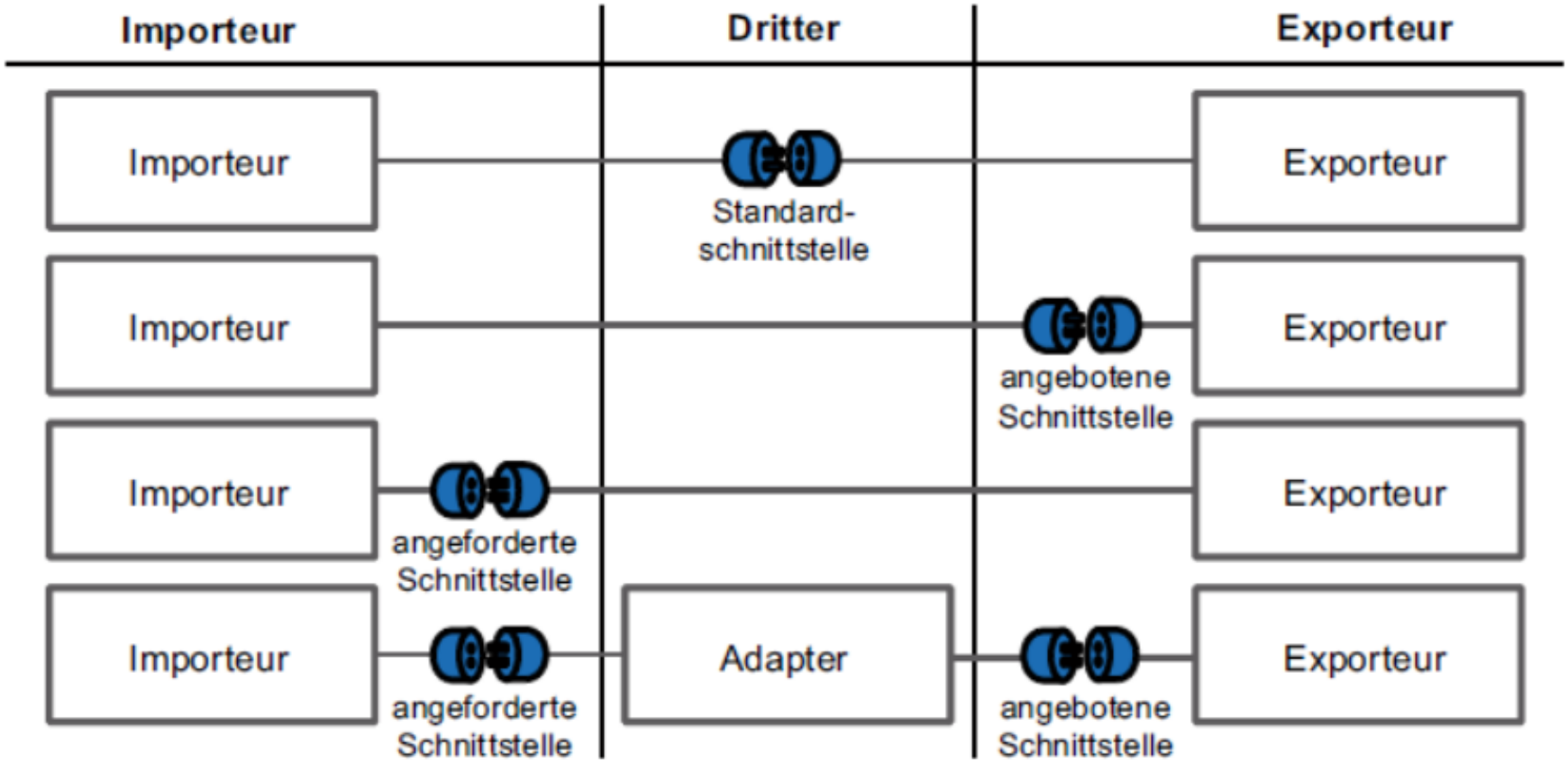


# Wer definiert die Schnittstelle? (4)

Zurich University of Applied Sciences

zh aw School of Engineering

- Zusammenfassung



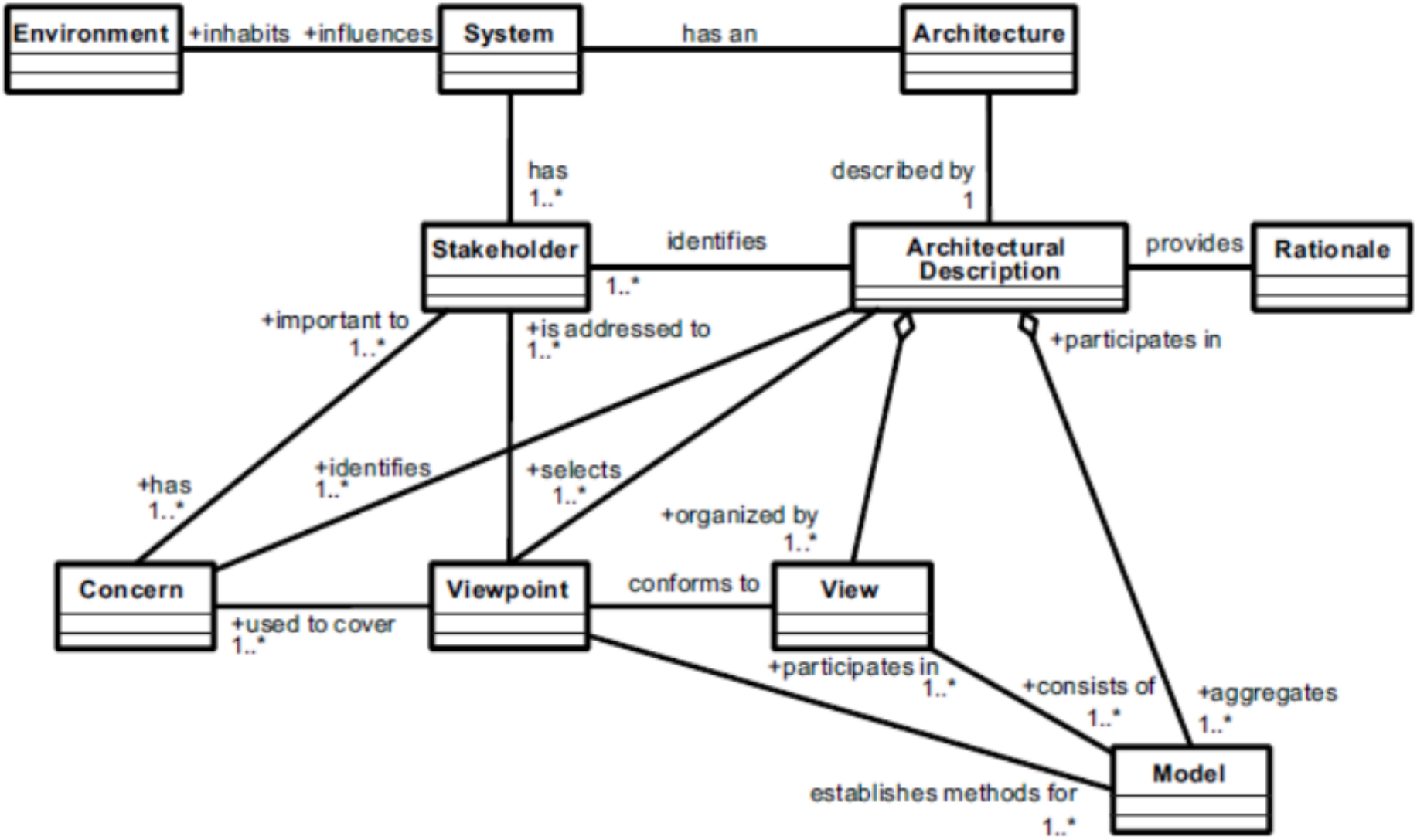
# Konzepte der Beschreibung von Architekturen



- Entsprechend dem IEEE-Standard 1471-2000, Recommended Practice for Architectural Description for Software-Intensive Systems [IEEE 1471-2000], beinhaltet eine Softwarearchitekturbeschreibung eine Menge von Artefakten, um eine Softwarearchitektur darzustellen.
  - Der entsprechende Standard definiert ein konzeptionelles Begriffsmodell für Architekturbeschreibungen.
  - Die nachstehende Abbildung zeigt den für uns relevanten Ausschnitt aus dem Begriffsmodell.

# Kernelemente des konzeptionellen Modells
## gemäss IEEE-Standard 1471-2000 (1)





# Kernelemente des konzeptionellen Modells gemäss IEEE-Standard 1471-2000 (2)

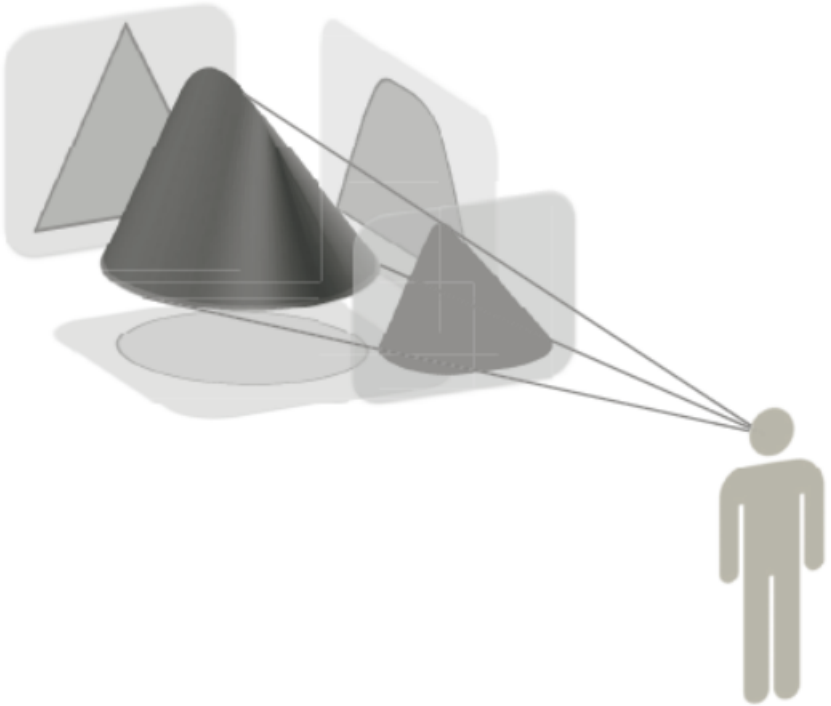


- Dabei wird ein System (System) von seiner Umgebung (Environment) beeinflusst und umgekehrt.
- Jedes System hat eine Architektur (Architecture).
- Die Architektur wird laut Standard durch eine einzige Architekturbeschreibung (Architectural Description) beschrieben.
- Darüber hinaus hat ein System eine Reihe von Interessenvertretern (Stakeholder).
- Die Interessenvertreter haben eine Menge von Anliegen (Concerns).
- Die Architekturbeschreibung nimmt die Anliegen der Interessenvertreter auf und begründet damit die getroffenen Architekturentscheidungen in den Begründungen (Rationales).

# Kernelemente des konzeptionellen Modells gemäss IEEE-Standard 1471-2000 (3)



- Architekturbeschreibungen sind Sichten (View) auf die Architekturbeschreibungen
  - Sichten sind Projektionen der Softwarearchitektur
  - Abhängig vom Standpunkt (Viewpoint) hat man eine unterschiedliche Sicht auf die Architektur. Diese Standpunkte sind dabei durch Interessenvertreter und deren Anliegen motiviert.



# Architektursichten

- IEEE-Standard Sichten:
  - »functional view«, »physical view« oder »technical view«.
- Sichten im Verständnis von Philippe Kruchten (1995): „Architectural Blueprints – „The „4+1“ View Model of Software Architecture“;

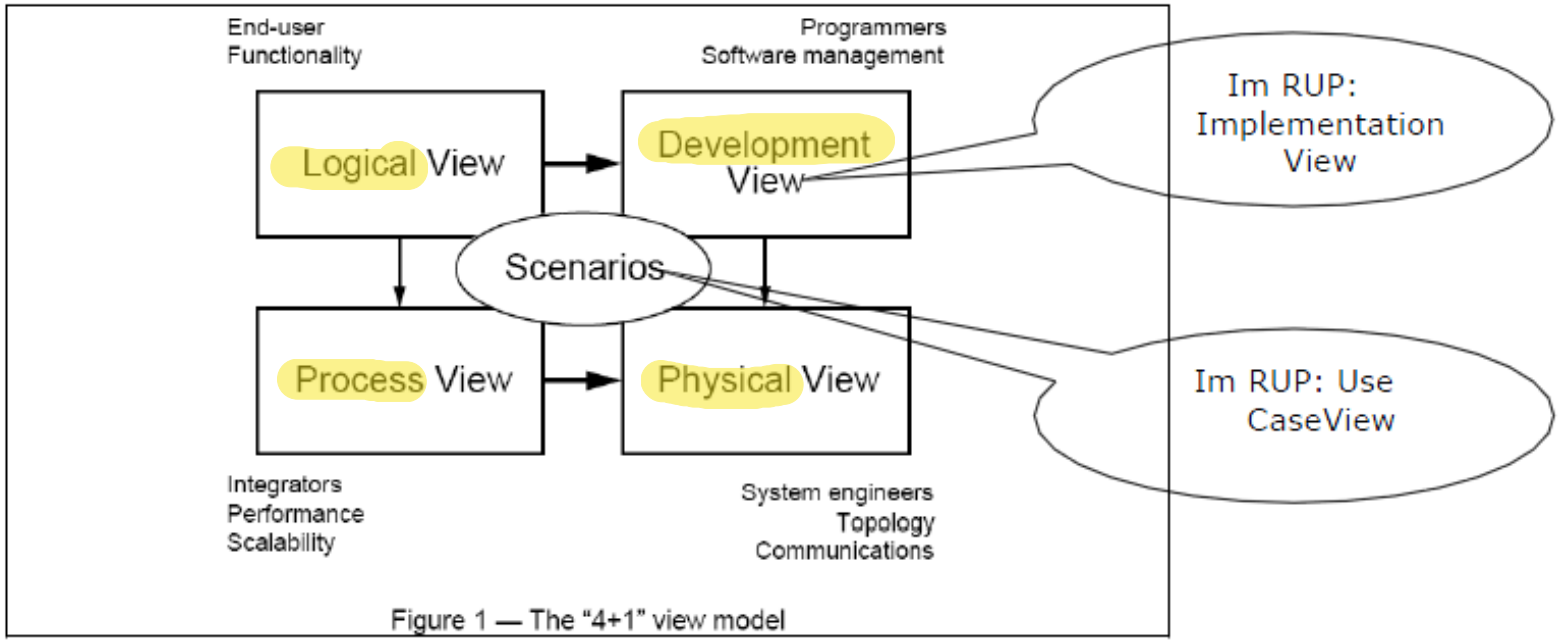
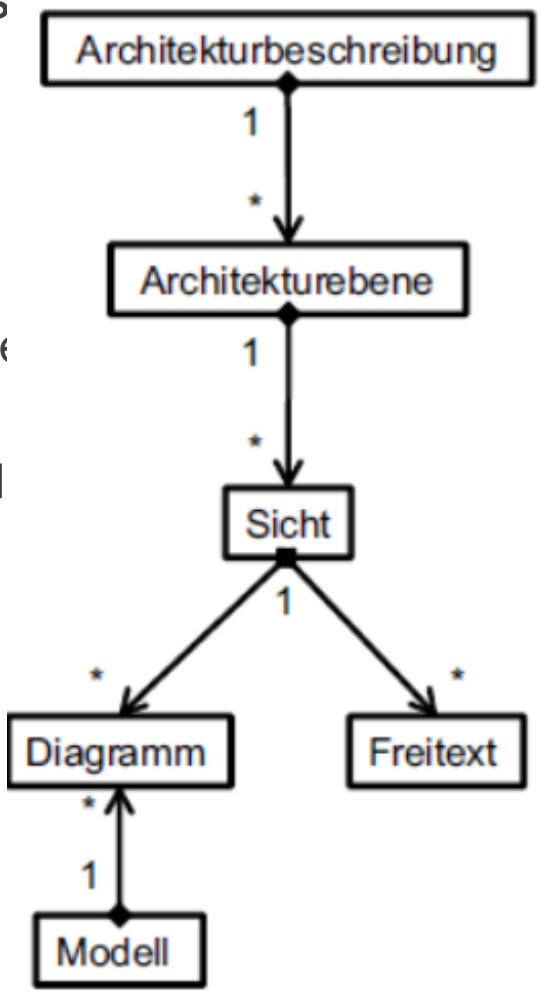


Figure 1 — The “4+1” view model

# Konzeptionelles Modell der Beschreibung von Softwarearchitekturen

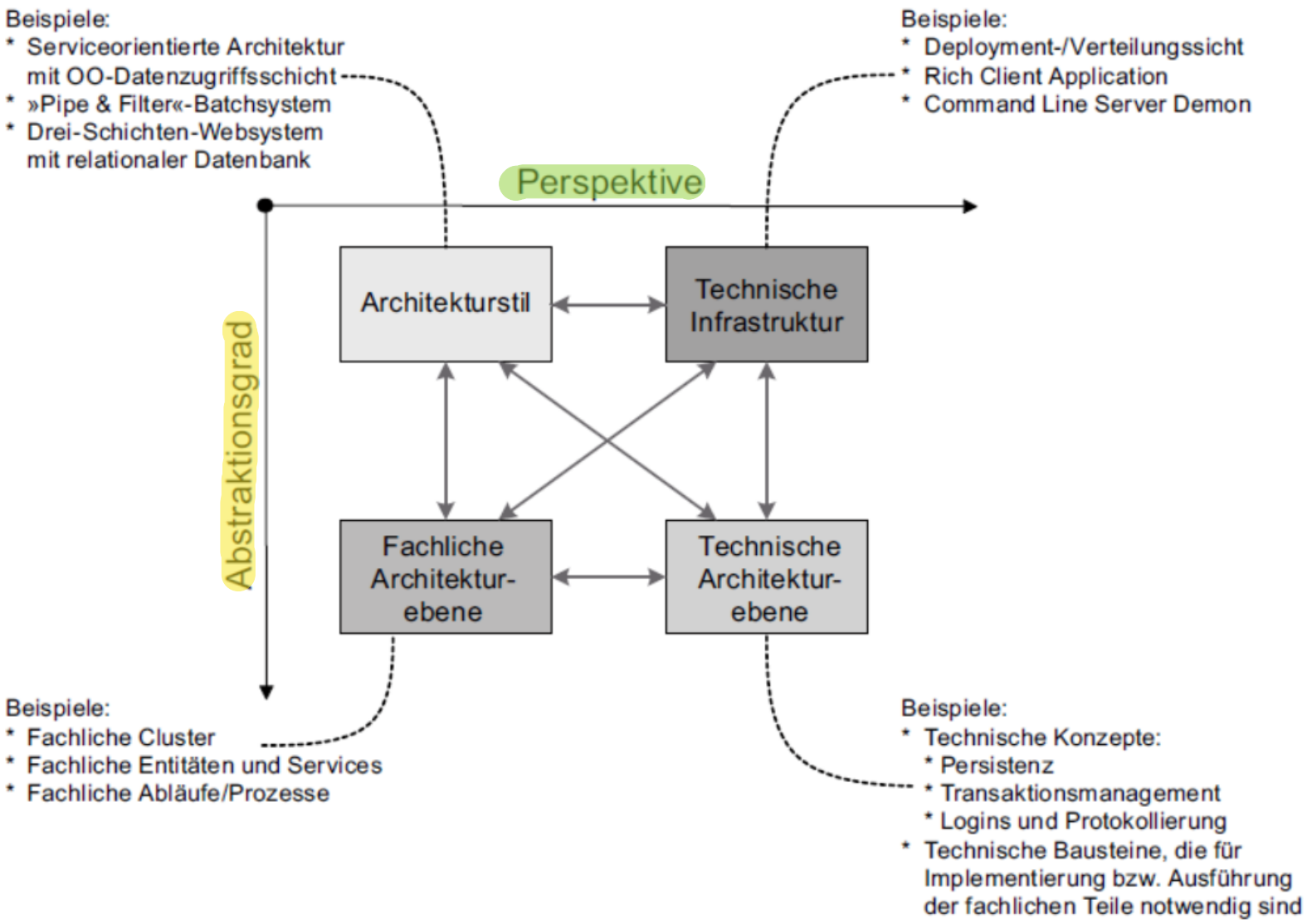


- Eine Architekturbeschreibung besteht dabei aus einer Menge von Architekturebenen.
  - Eine Architekturebene fasst eine Menge von Sichten (z.B. statische Sichten) zu einer sinnvollen Beschreibungseinheit zusammen.
  - Sichten enthalten sowohl Texte als auch Diagramme, die wiederum in Modellen abgelegt sind.
  - **Beispiel**: eine Architekturebene kann eine statische und eine dynamische Sicht beinhalten



# Architekturbeschreibung und Architekturebenen

Zurich University of Applied Sciences
zh aw School of Engineering



# Architekturebenen (1)



- Die vorhergehende Folie zeigt eine beispielhafte Aufteilung der Architekturbeschreibung in die vier Architekturebenen:
  - Architekturstil,
  - technische Infrastruktur,
  - fachliche Anwendungsarchitekturebene und
  - technische Architekturebene.
- Diese vier Ebenen unterscheiden sich wie dargestellt in zwei Dimensionen:
  - sowohl durch die Abstraktionsstufe als auch
  - durch die betrachtete Perspektive (fachlich vs. technisch).

# Architekturebenen (2)



- Architekturebenen können für sich alleine betrachtet werden, allerdings beeinflussen sie sich gegeneinander und sind somit voneinander abhängig.
  - Konkretere Architekturebenen nehmen die fachlichen und technischen Vorgaben des Architekturstils und der technischen Infrastruktur auf.
  - Umgekehrt beeinflussen die konkreten Entwürfe der fachlichen und technischen Architekturebene die Vorgaben in den höheren Architekturebenen.
  - Die technische Architektur kann konkrete Konzepte für die Persistenz oder das Transaktionsmanagement in der fachlichen Architekturebene vorgeben.
  - Umgekehrt liefert die fachliche Architekturebene die Vorgaben dafür, welche Persistenzkonzepte überhaupt benötigt werden.

# Architekturebenen (3)



- Der Architekturstil ist dabei die zentrale Architekturmetapher des Systems, wie z.B.:
  - »Unser Softwaresystem ist nach einer Drei-Schichten-Architektur unter Verwendung eines Model-View-Controllers in der Präsentationsschicht und einem objektrelationalen Mapping in der Datenhaltungsschicht strukturiert.«
- Die technische Infrastruktur hingegen fixiert die Netzwerkschnitte der Architektur, wie z.B.:
  - »Wir haben einen Thin Client mit einem Web- und Application-Container und einer relationalen Datenbank.«

# Wechselwirkung zwischen Softwarearchitektur und Umgebung (1)



- Die Architektur eines Systems steht nicht im luftleeren Raum entsteht.
  - Sie wird von der Umgebung beeinflusst und umgekehrt.
  - Für die eigentliche Softwarearchitektur umgebenden Elemente ebenfalls häufig der Begriff Architektur verwendet, wie z.B. eine Geschäftsprozessarchitektur.

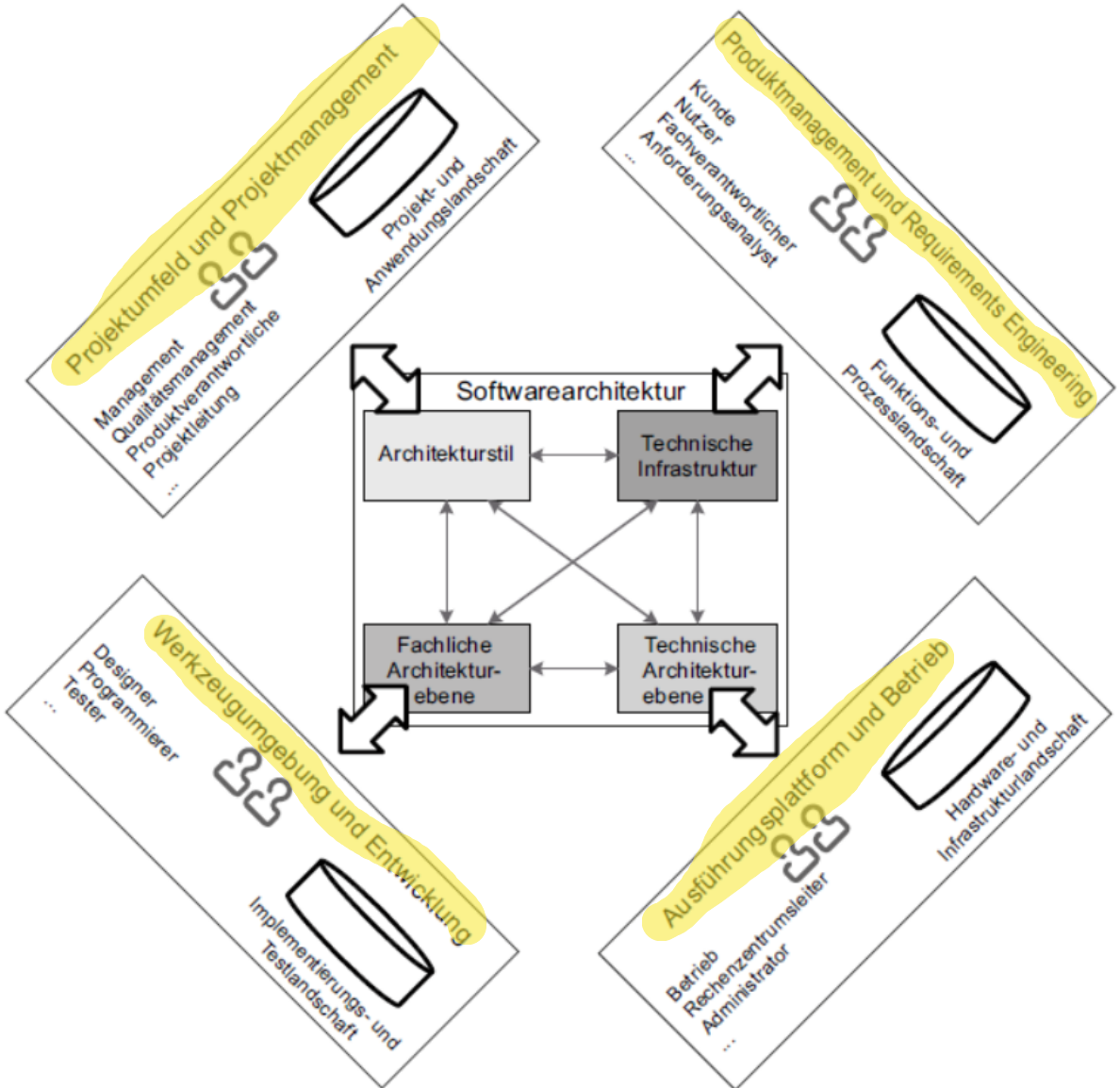
# Wechselwirkung zwischen Softwarearchitektur und Umgebung (2)

Zurich University of Applied Sciences
zh aw School of Engineering

- Umgebungsbereiche im Kontext der Softwarearchitektur:
  - **Projektumfeld und Projektmanagement:** Im Umfeld eines Systems existiert in der Regel eine Vielzahl von Anwendungen und Projekten.
  - **Produktmanagement und Requirements Engineering:** Produktmanagement und Requirements Engineering liefern funktionale und insbesondere nichtfunktionale Anforderungen an das System. Diese Anforderungen können sich im Laufe des Projekts verändern und somit auch die Architektur beeinflussen.
  - **Ausführungsplattform und Betrieb:** Ausführungsplattform und Betriebsorganisation sind meist in den Organisationen schon vorhanden. Neue Softwaresysteme sollten diese, wenn möglich, nutzen. Dementsprechend müssen im Rahmen des  rchitekturentwurfs diese berücksichtigt werden.
  - **Werkzeugumgebung und Entwicklung:** Letztlich muss die Architektur umgesetzt werden. Hierfür sind entsprechende Werkzeuge und eine Entwicklungsorganisation notwendig.

# Wechselwirkung zwischen Softwarearchitektur und Umgebung (3)





# Qualität und Nutzen der Softwarearchitektur



- Die Qualität der Softwarearchitektur kann nur im Kontext konkreter Qualitätsziele bewertet werden
- Diese Qualitätsziele werden von den Anforderungen, insbesondere von den nichtfunktionalen Anforderungen und geforderten Qualitätseigenschaften abgeleitet. Sie sind somit spezifisch für das konkrete Softwaresystem.
- Übergeordnete Qualitätsmerkmale, die sogenannten FURPS-Merkmale aus ISO-9126 bzw. ISO 25010:
  - Functionality (*Funktionalität*)
  - Usability (*Benutzbarkeit*)
  - Reliability (*Zuverlässigkeit*)
  - Performance (*Effizienz*)
  - Supportability (*Änderbarkeit*)

# Qualitätsmerkmale nach ISO 9126

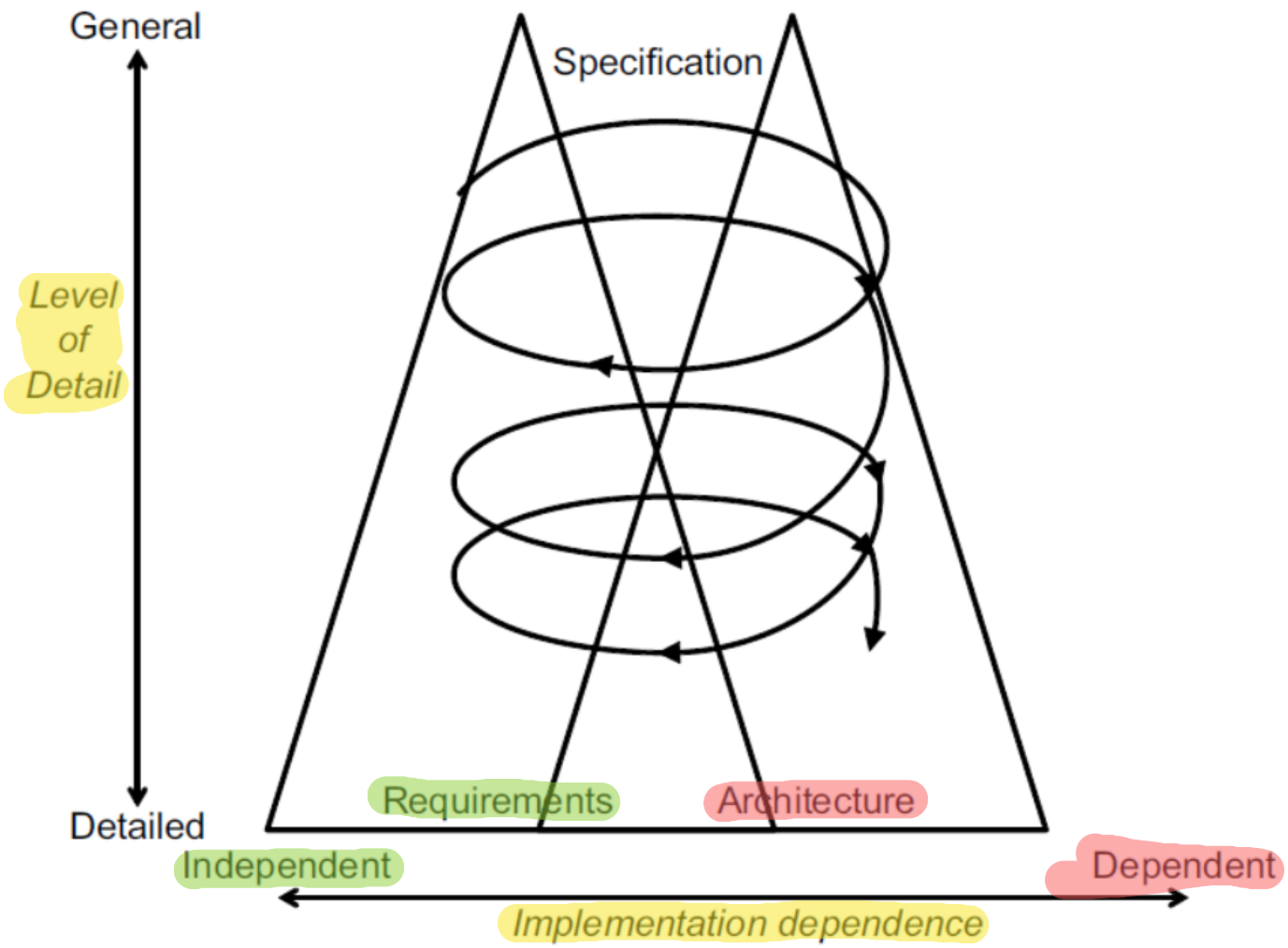


| | |
|---|---|
| **Funktionalität** | ■ Angemessenheit<br>■ Richtigkeit<br>■ Interoperabilität<br>■ Sicherheit<br>■ Ordnungsmäßigkeit |
| **Zuverlässigkeit** | ■ Reife<br>■ Fehlertoleranz<br>■ Wiederherstellbarkeit |
| **Benutzbarkeit** | ■ Verständlichkeit<br>■ Erlernbarkeit<br>■ Bedienbarkeit |
| **Effizienz** | ■ Zeitverhalten<br>■ Verbrauchsverhalten |
| **Änderbarkeit** | ■ Analysierbarkeit<br>■ Modifizierbarkeit<br>■ Stabilität<br>■ Prüfbarkeit |
| **Übertragbarkeit** | ■ Anpassbarkeit<br>■ Installierbarkeit<br>■ Austauschbarkeit<br>■ Konformität |

# Der Softwarearchitekturentwurf aus der Vogelperspektive

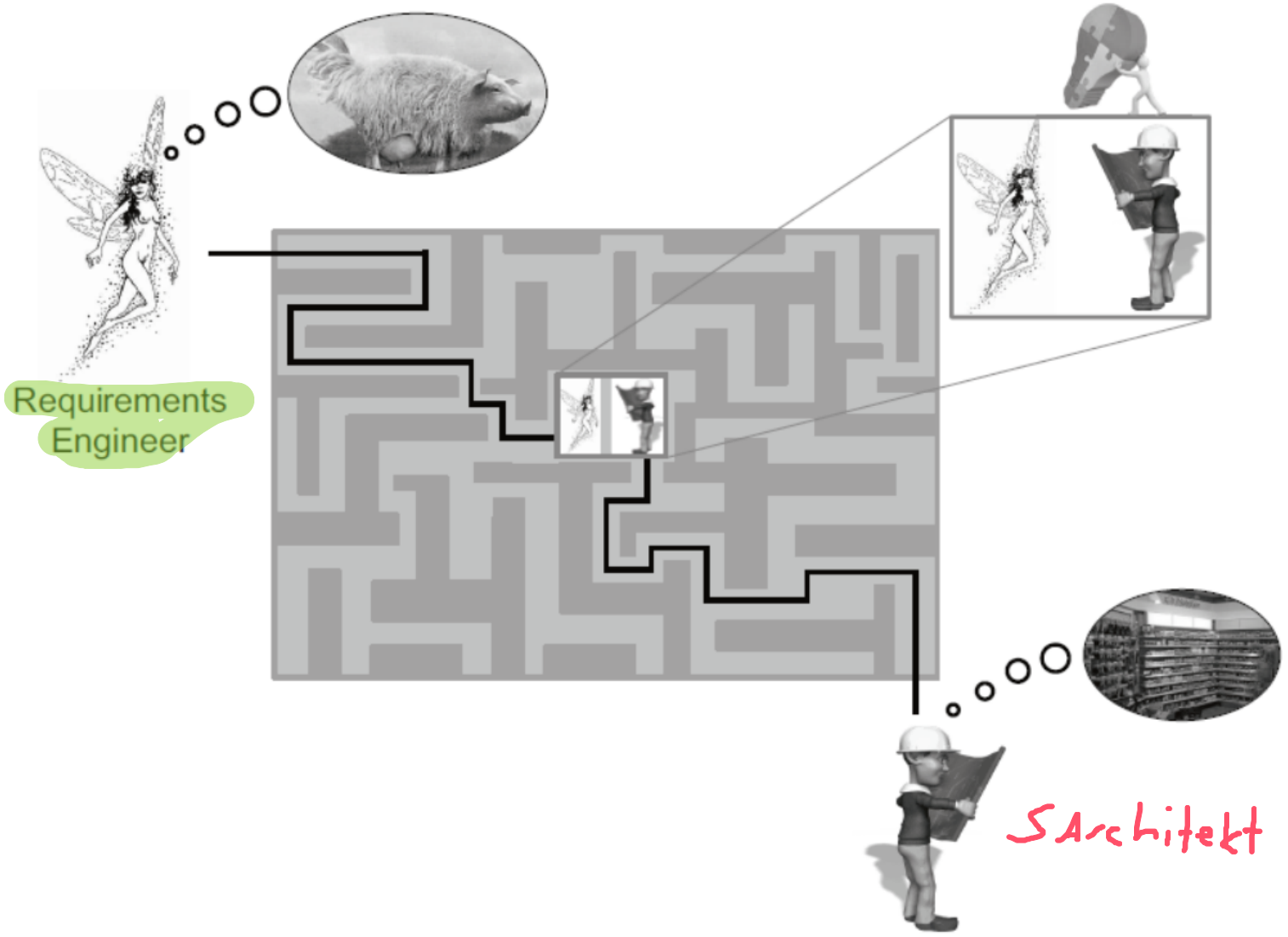


- Twin Peak Model:
  - it



# Requirements <-> Architektur

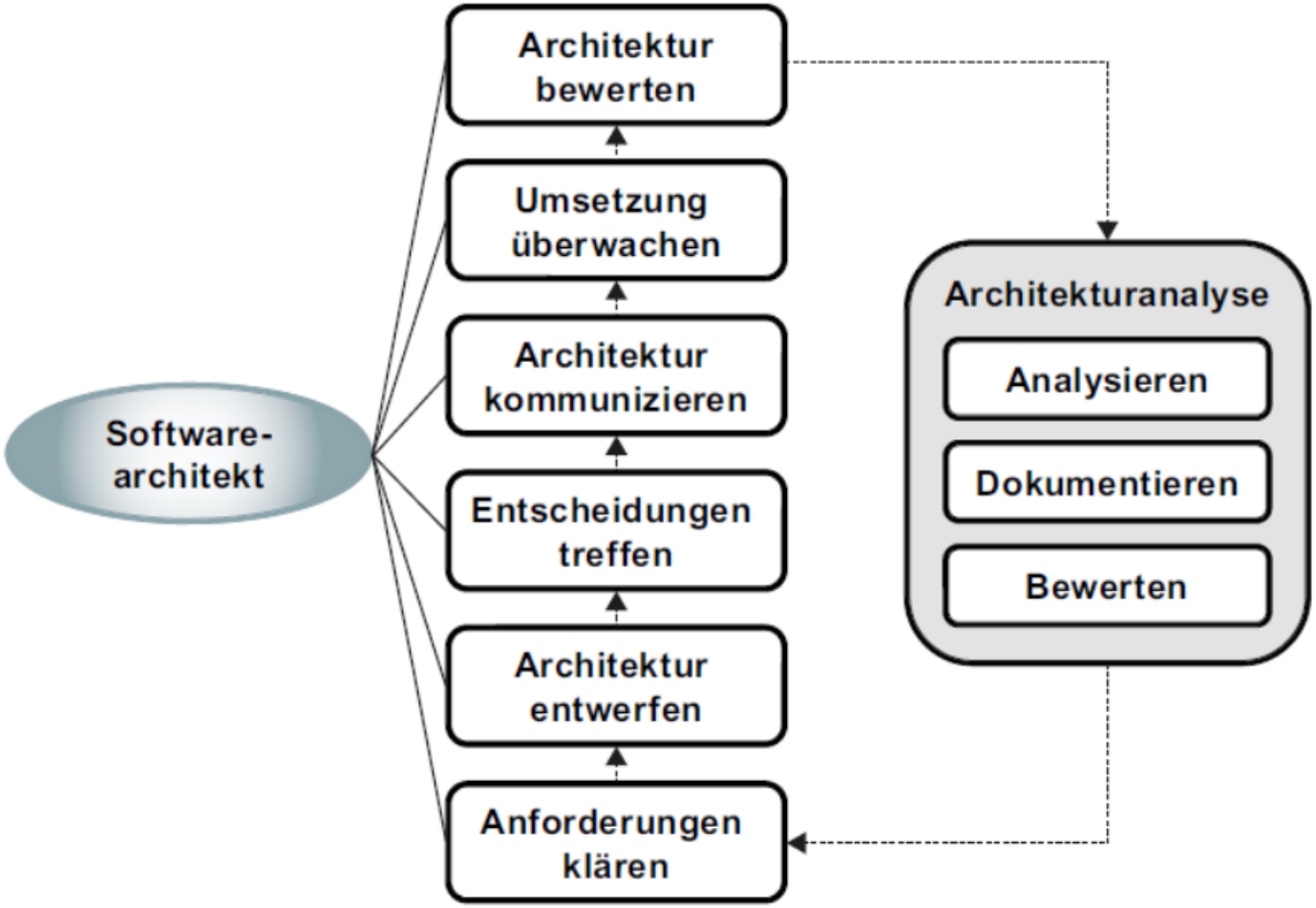




# Der Softwarearchitekturentwurf im Überblick



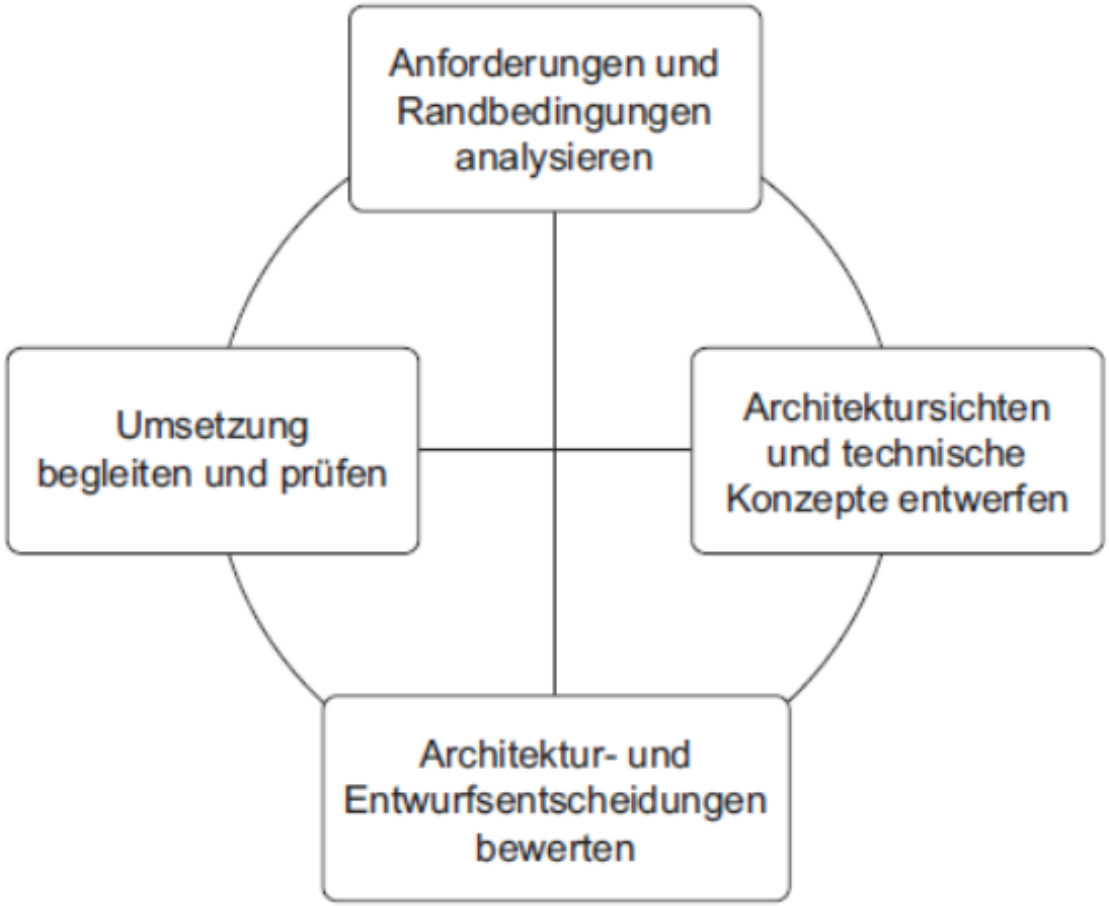
- Architecture Business Cycle



# Der Softwarearchitekturentwurf im Überblick (2)



- Alternatives Modell



# Der Softwarearchitekturentwurf im Überblick (3)

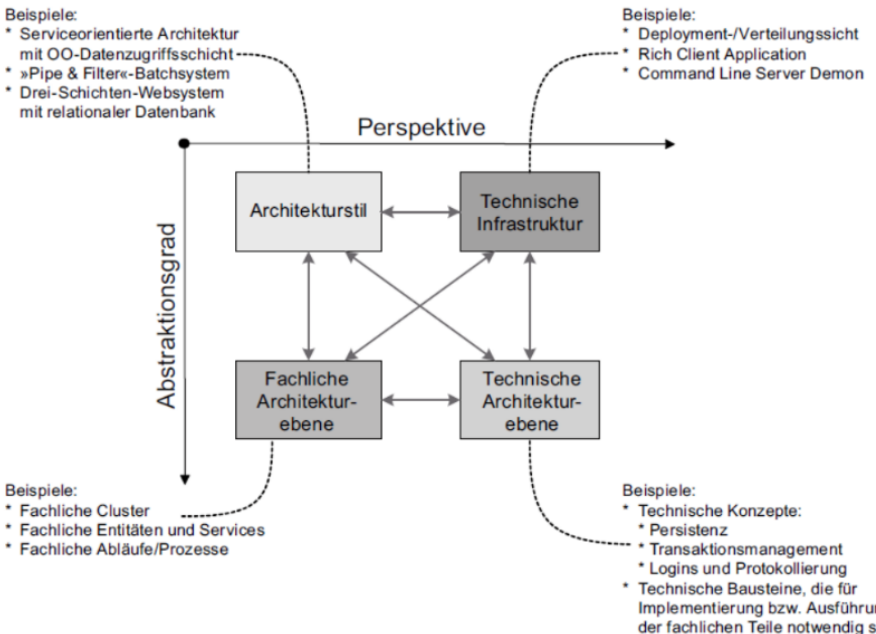


- Anforderungen und Randbedingungen analysieren:
  - Zentrale Aufgabe der Architekturanalyse ist es, die funktionalen und insbesondere nichtfunktionalen Anforderungen aus dem Requirements Engineering im Kontext der anderen umgebenden Bereiche zu untersuchen.
  - Dabei müssen Qualität und Stabilität der Anforderungen überprüft werden.
  - Lücken in den Anforderungen müssen aufgedeckt werden.
  - Gerade bei den nichtfunktionalen Anforderungen muss hier noch meist nachgebessert werden, da die Anforderungsträger diese häufig als selbstverständlich verstehen.
  - In Zusammenarbeit mit allen Projektbeteiligten – insbesondere den Designern und Entwicklern – ist ein erstes Verständnis für Architekturstil und technische Infrastruktur zu entwickeln.

# Der Softwarearchitekturentwurf im Überblick (4)

Zurich University of Applied Sciences
zh aw School of Engineering

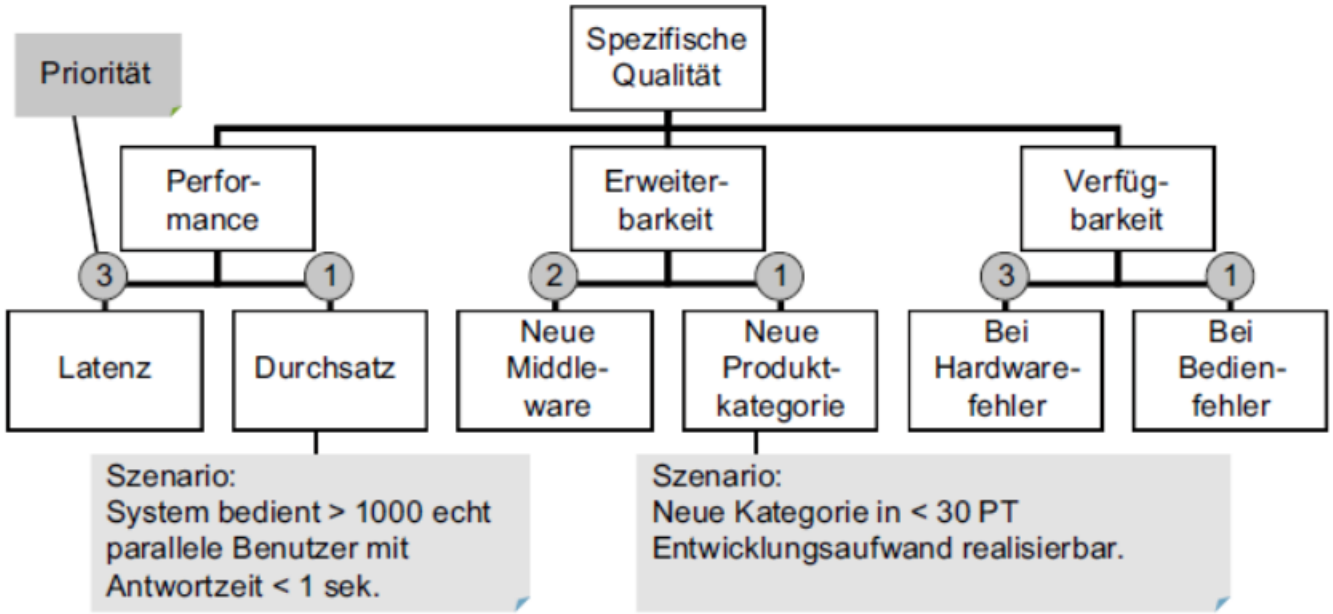
- **Architektursichten und technische Konzepte entwerfen:**
  - Hier wird die Architektur detaillierter ausgearbeitet. Es erfolgt insbesondere auch die sichtenbasierte Beschreibung der unterschiedlichen Architekturebenen wie z.B. der fachlichen und der technischen.
  - Dabei gilt es, die funktionalen Anforderungen auf die entsprechende fachliche Architekturebene herunterzubrechen, sowie für die relevanten Aspekte aus den nichtfunktionalen Anforderungen entsprechende querschnittliche Lösungsbausteine in der technischen Architekturebene zu konzipieren und zu dokumentieren



# Der Softwarearchitekturentwurf im Überblick (4)



- **Architektur und Entwurfsentscheidungen bewerten**
  - Die erarbeitete Architektur muss qualitätsgesichert werden. Hier können die unterschiedlichen Techniken angewendet werden, angefangen von den diversen Reviewtechniken über technische Prototypen und Tests bis hin zu analysierenden und bewertenden Verfahren.
  - Entscheidend hierbei ist, dass man aus den Anforderungen konkrete Szenarien abgeleitet hat, um die Qualität der Architektur sicherzustellen



# Der Softwarearchitekturentwurf im Überblick (4)



- **Umsetzung begleiten und überprüfen**: Von ihrer Bedeutung her häufig unterschätzt ist die vierte Aufgabe des Architekten:
  - das Kommunizieren der Softwarearchitektur an alle Projektbeteiligten.
  - Nur wenn alle – vom Entwickler bis zum Kunden – die Softwarearchitektur verstanden und akzeptiert haben, wird sie erfolgreich umgesetzt werden und so ihre erhoffte Wirkung entfalten.
  - Dabei muss die Softwarearchitektur natürlich so kommuniziert werden, dass der Kommunikationspartner diese auch verstehen kann. Das heisst, dem Kunden wird man die Architektur auf einem anderen Detaillierungsgrad erklären als dem Entwickler.

# Wechselspiel der Tätigkeiten (1)

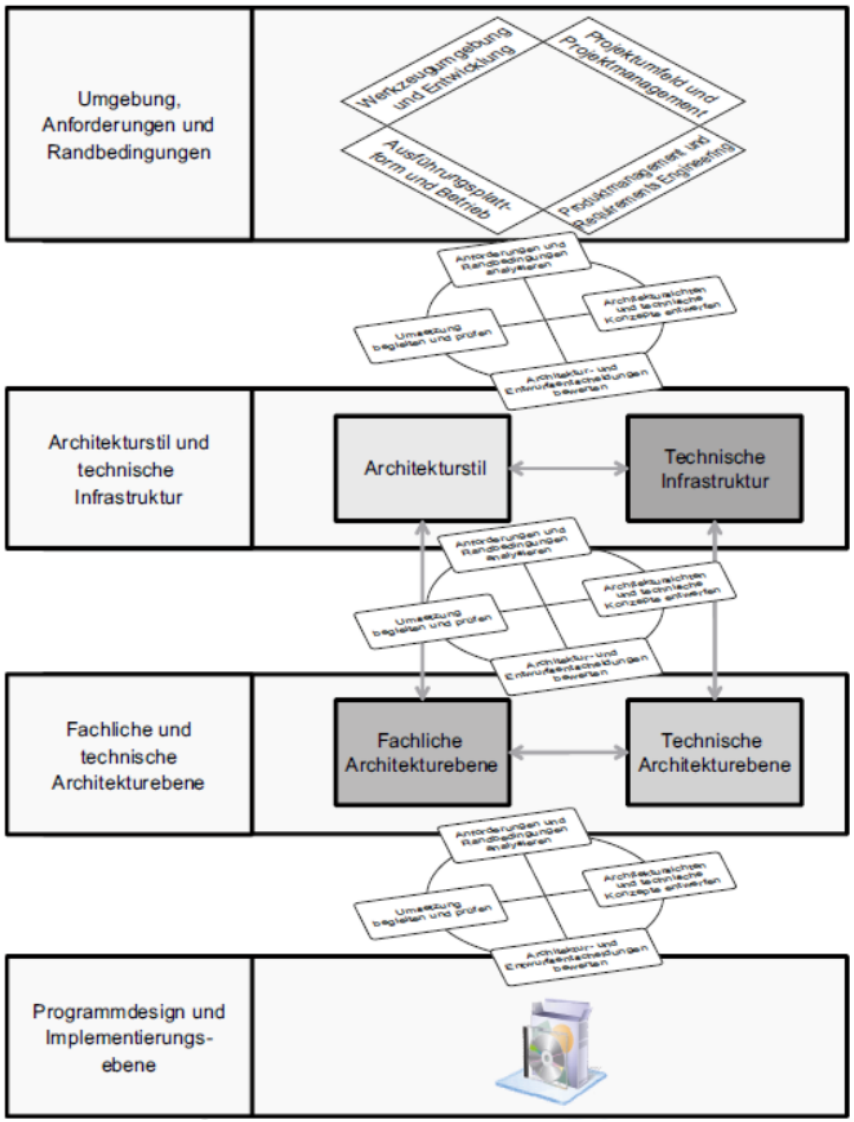


- Tätigkeiten
  - Anforderungen und Randbedingungen analysieren
  - Architektursichten und technische Konzepte entwerfen
  - Architektur und Entwurfsentscheidungen bewerten
  - Umsetzung begleiten und prüfen

- Mit Top-Down oder Bottom-Up Wechsel der Abstraktionsstufen:
  - Vorgaben und Randbedingungen der umgebenden Bereiche
  - Softwarearchitektur mit Abstraktionsstufen, beispielsweise
    - Architekturstil und technische Infrastruktur
    - fachliche und technische Architekturebene
  - Programmdesign und Implementierung

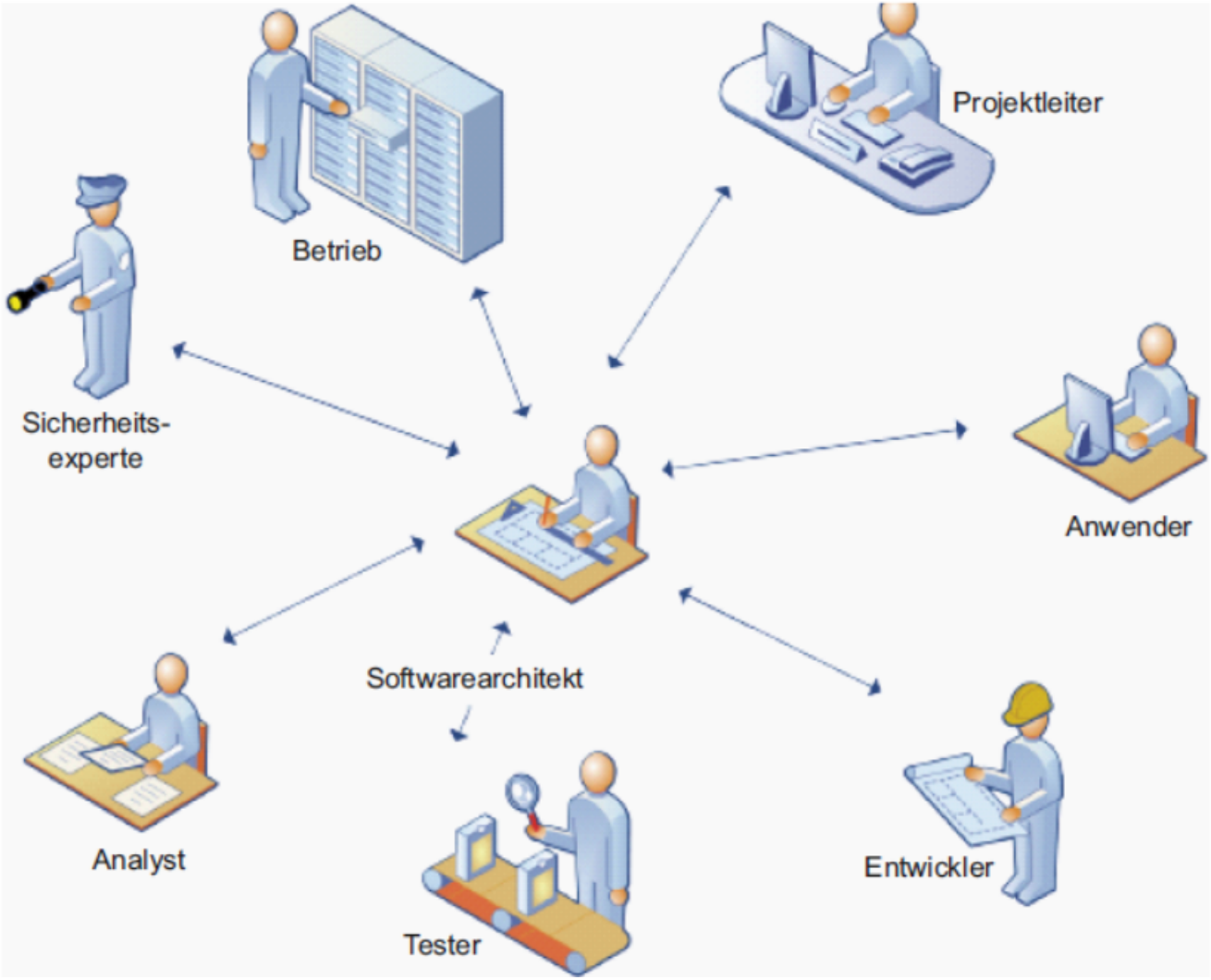
# Wechselspiel der Tätigkeiten (2)





# Aufgaben und Bezug zu den anderen Rollen (1)





# Aufgaben und Bezug zu den anderen Rollen (2)



- **Kommunikations- und Diskussionsplattform:**
  - Anhand der Architektur stellt der Architekt gegenüber dem Requirements Engineer, dem Kunden und ggf. dem Nutzer die Machbarkeit der Anforderungen dar.
  - Dabei unterstützt er bei dem Zuordnen, Priorisieren und Reflektieren von funktionalen und nichtfunktionalen Anforderungen.
  - Er zeigt Integrationsmöglichkeiten von existierenden Lösungen und Systemen auf und gleicht die Anforderungen mit der existierenden Systemarchitektur und Hardware ab.
  - Er erarbeitet, evaluiert und bewertet alternative Lösungsansätze.
  - Berät den Projektleiter bei der Projekt und Iterationsplanung, unterstützt die Risikoanalyse und -vermeidung und unterstützt so bei der Definition der Arbeitsstrukturierung und -verteilung.

# Aufgaben und Bezug zu den anderen Rollen (3)



- **Design- und Implementierungsplan:**
  - Für die Entwickler ist der Architekt ein zentraler Ansprechpartner.
  - Er legt die Bausteine des Systems sowie deren Schnittstellen und Interaktionsmuster fest.
  - Die Integration von neuen Technologien und innovativen Lösungsansätzen muss er vorantreiben und mit den Entwicklern diskutieren.
  - Er leitet die Erarbeitung, Einführung, Schulung und Überprüfung von Programmierrichtlinien.
  - Er hilft den Entwicklern bei der Erstellung von Prototypen und exemplarischen Teillösungen und forciert die Wiederverwendung von existierenden Implementierungen und Teilimplementierungen.
  - Er erklärt die Architektur, macht Entwicklungsvorgaben, gibt seine Erfahrung weiter und führt Codereviews durch.
  - Ebenso unterstützt er die Tester.

# Zusammenfassung - Fragen



- Softwareintensive Systeme und Softwarearchitekturen
  - Was ist ein softwareintensives System? (LZ 1-10)
  - Ausprägung von softwareintensiven Systemen( (LZ 1-10)
  - Bedeutung der Software Architektur für ein System (LZ 1-7)
- Grundlegende Konzepte von Softwarearchitekturen
  - Was ist eine Software Architektur? (LZ 1.1)
  - Ausprägungen von softwareintensiven Systemen (LZ 1-1)
  - Qualität und Nutzen der Software Architektur (LZ 1-2)
- Der Softwareentwurf aus der Vogelperspektive
  - Ziele und Aufgabe es Software Architekturentwurfs (LZ 1-2)
  - Software Architekturentwurf im Überblick (LZ 1-3. KLZ 1-6)
  - Aufgaben des Software Architekten (LZ 1-4, LZ 1-5)