Hochschule Konstanz
Fakultät Informatik

Grundlagen der Software-Architektur

Fraunhofer ACES

In dieser Vorlesung werden unter anderem Methoden und Materialien aus folgenden Quellen verwendet

Fraunhofer ACES - Architecture-Centric Engineering Solutions:

- Fraunhofer-Ansatz zur Modellierung von Software- und Systemarchitekturen
- Gesammelte Best Practices aus der Literatur,
  skaliert und zugeschnitten für effektive Architektur in der Praxis
- Basierend auf mehr als 20 Jahren Erfahrung in der Architektur in verschiedenen Bereichen:
  Eingebettete Systeme, Informationssysteme, intelligente Ökosysteme
- Die Methode kann von allen frei verwendet werden (das Material der Vorlesung hier allerdings nicht ohne Rücksprache mit mir)

Die Methode wird am **Fraunhofer IESE** Institut in Kaiserslautern entwickelt. Dort gibt es eine Arbeitsgruppe, die sich der Softwarearchitektur widmet

- siehe https://www.iese.fraunhofer.de/de/leistungen/software-architektur.html für weitere Einzelheiten

Software-Architektur: Was und Wozu?

- pingo.coactum.de, Code 871775 eingeben
  - Kann auch von einem Mobiltelefon aus genutzt werden

- https://pingo.coactum.de/871775

Haftungsausschluss: Lesen Sie die Datenschutzbestimmungen und entscheiden Sie selbst, ob Sie an den Umfragen teilnehmen möchten.

Definitionen

- Software architecture is the **structure or structures** of the **system**, which comprise software **elements**, the **externally visible properties** of those elements, and the **relationships** among them.

[Software Architecture in Practice, L.Bass, P.Clements, R.Kazman]

- Software architecture is the fundamental **concepts or properties of a system** in its **environment** embodied in its **elements**, **relationship**s, and in the **principles** of its **design and evolution**.

[Systems and software engineering — Architecture description, ISO Standard 42010]

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

[E. Woods]

- Software architecture is the set of **principal design decisions** made about the **system**.

[Software Architecture: Foundations, Theory, and Practice , E.Dashofy, N.Medvidovic, R. Taylor.]

Deutsche Übersetzung

- Softwarearchitektur ist die **Struktur oder die Strukturen** des **Systems**, die aus **Softwareelementen**, den **äußerlich sichtbaren Eigenschaften** dieser Elemente und den **Beziehungen zwischen ihnen** besteht.

[Software Architecture in Practice, L.Bass, P.Clements, R.Kazman]

- Die Software-Architektur umfasst die grundlegenden **Konzepte oder Eigenschaften eines Systems** in seiner **Umgebung**, die in seinen **Elementen** und Beziehungen sowie in den **Grundsätzen** seines **Designs und seiner Entwicklung zum** Ausdruck kommen.

[Systems and software engineering — Architecture description, ISO Standard 42010]

- Die Softwarearchitektur ist die **Gesamtheit der Entwurfsentscheidungen**, die, wenn sie falsch getroffen werden, zum Abbruch Ihres Projekts führen können.

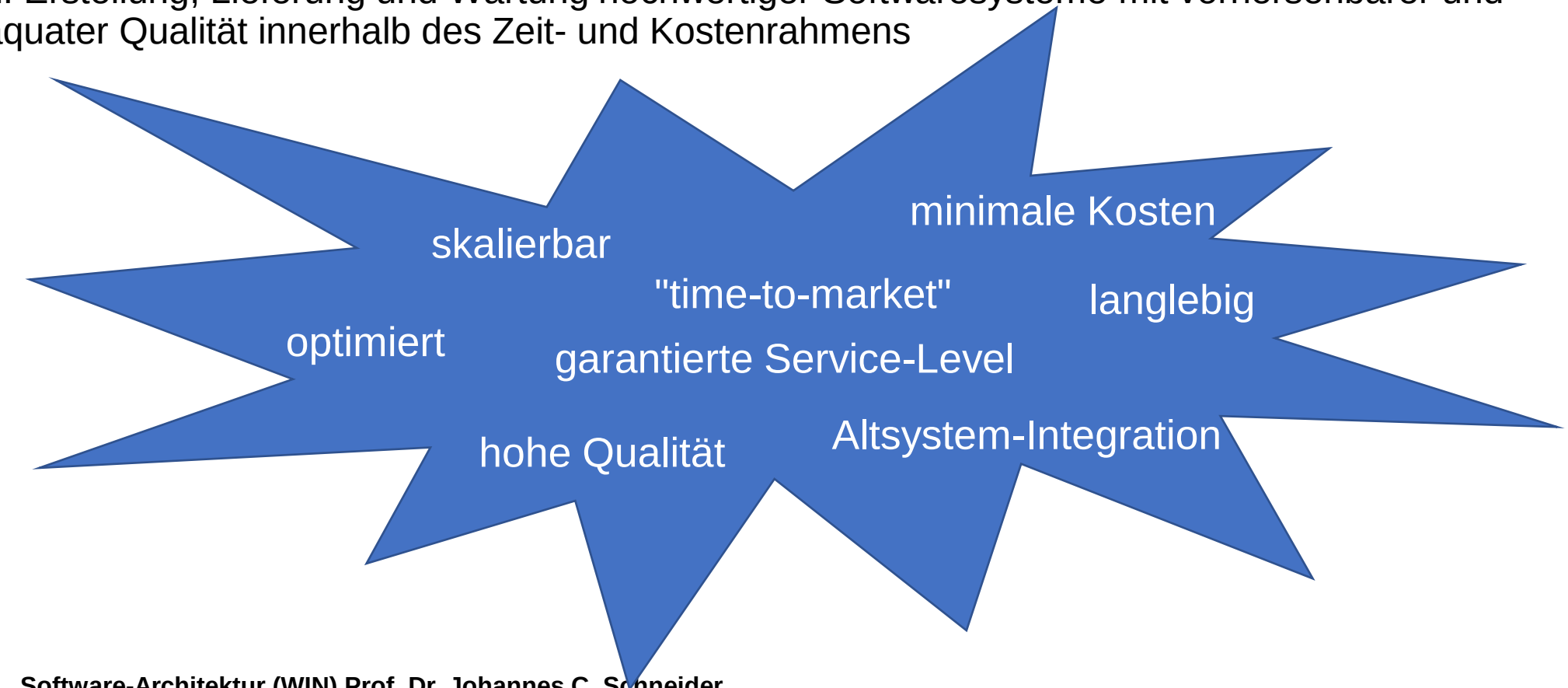
(E. Woods)

- Die Software-Architektur ist die Gesamtheit der **wichtigsten Entwurfsentscheidungen**, die für das **System** getroffen werden.

[Software-Architecture: Foundations, Theory, and Practice , E. Dashofy, N. Medvidovic, R. Taylor].

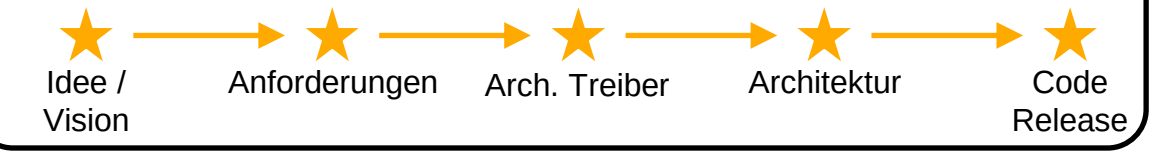
Warum Architektur?

- Architekturarbeit: Auf die **Management-Ziele und Zielvorgaben** kommt es an!
- d. h. Erstellung, Lieferung und Wartung hochwertiger Softwaresysteme mit vorhersehbarer und adäquater Qualität innerhalb des Zeit- und Kostenrahmens



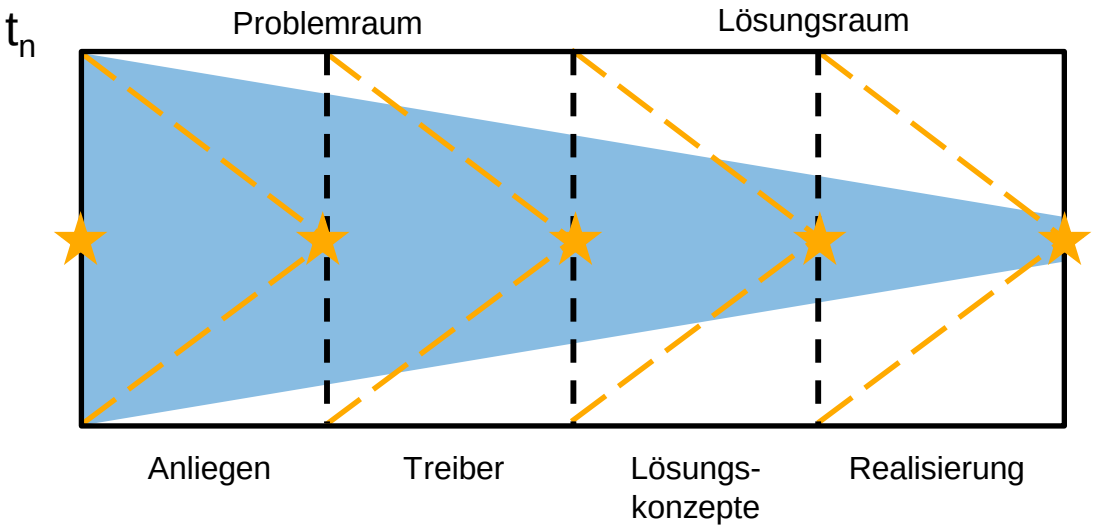
Warum Architektur? Es geht darum, die Lücken zu überbrücken!

Informationsfluss im Software-Engineering



Anliegen (*Concerns)*
versus
Ergebnisse

Architekturarbeit

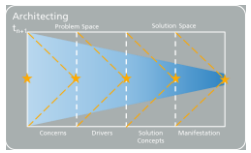


Problemraum
vs.
Lösungsraum

Evolution

t_{n-m} → t_n → t_{n+p}

Änderungen geschehen, Systeme folgen!



- Die technischen Schulden der *Vergangenheit* beseitigen
- Vorbereiten/Lösen von *aktuellen* Herausforderungen
- *Künftige* Änderungen/Bedürfnisse antizipieren

Gegenwart
vs.
Vergangenheit und Zukunft

Eine gut dokumentierte Software-Architektur...

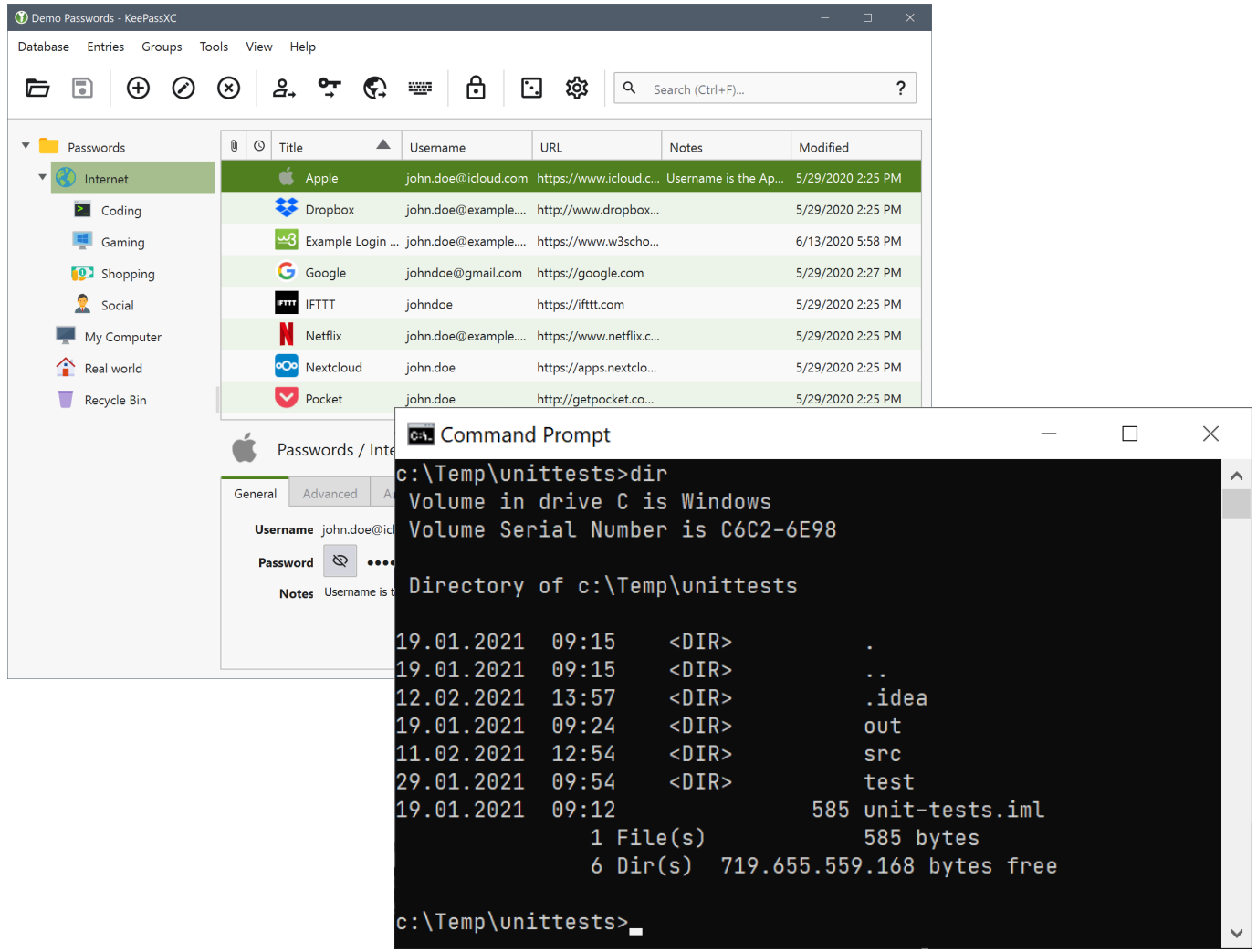
- ... bietet **Anleitung**
  - Plan für den Aufbau eines Systems
  - Technische Leitung und Koordination
  - Standards und Konsistenz
- ... gleicht **technische Risiken** aus
  - Identifizierung und Risikominimierung
  - Definition von Lösungskonzepten
  - Antizipation (Vorbereitung) von Veränderungen
- ... ermöglicht **Kommunikation**
  - Klare technische Vision und Roadmap
  - Explizite Dokumentation der Kommunikation
- ... verwaltet die **Komplexität von Software**
  - Zu erstellende Produkte
  - Verknüpfung der Systeme
  - Integration mit Altsystemen
  - Zusammenarbeit von Organisationseinheiten

Beobachtung: Jedes Haus hat eine Architektur

– Struktur, Materialien, Raumaufteilung, Strom-/Gas-/Wasserversorgung

Jede Software hat auch eine Architektur!

- Komponenten, Schichten
- Datenmodelle
- Art der Benutzeroberfläche
- Lokal oder verteilt
- Technologien und Bibliotheken
- ...



von https://keepassxc.org/screenshots/

Wiederholung: Definitionen

- Die Softwarearchitektur ist die **Gesamtheit der Entwurfsentscheidungen**, die, wenn sie falsch getroffen werden, zum Abbruch Ihres Projekts führen können.

(E. Woods)

- Die Software-Architektur ist die Gesamtheit der **wichtigsten Entwurfsentscheidungen**, die für das System getroffen werden.

[Software-Architektur: Foundations, Theory, and Practice , E. Dashofy, N. Medvidovic, R. Taylor].

Implizite versus explizite Entscheidungen

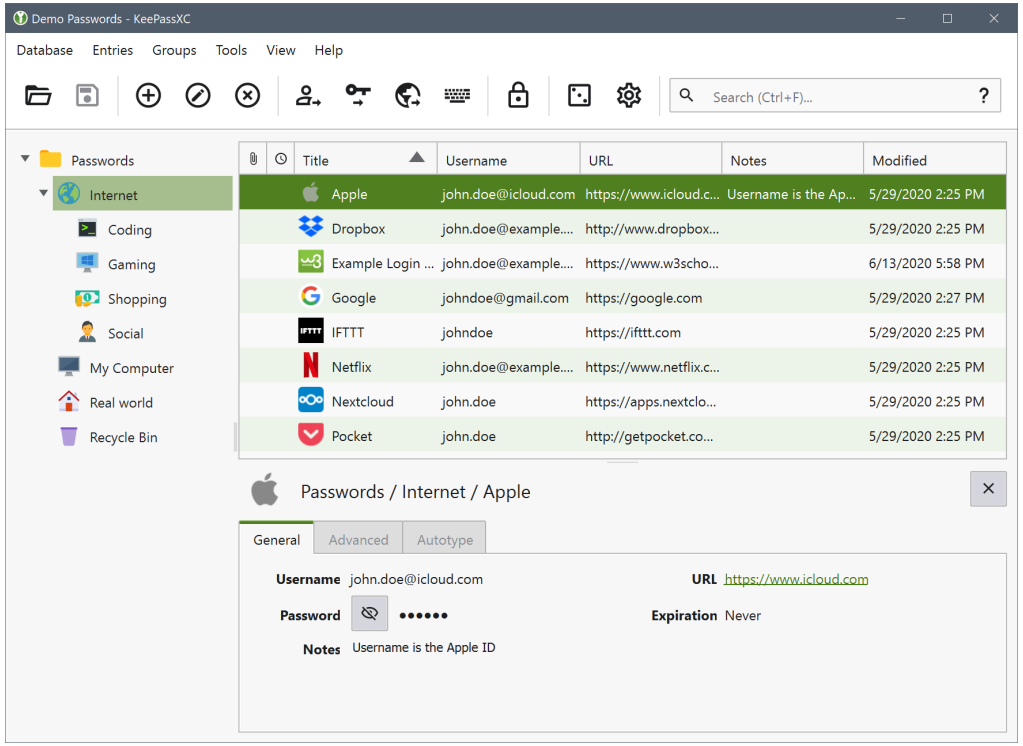
- Entscheidung für die Architektur: Diese Holzleiter verwenden

- Wurde diese Entscheidung im Vorfeld **explizit** getroffen?
  - Wir verwenden diese Leiter, um einen bequemen und stabilen Zugang zum "Gebäude" zu schaffen.

- Oder eher **implizit** während der Erstellung?
  - Wir benutzen diese Leiter, weil wir eine herumliegen hatten.

Implizite versus explizite Entscheidungen

- Architektur-Entscheidung: Qt als UI-Framework verwenden
- Wurde diese Entscheidung im Vorfeld **explizit** getroffen?
  - Wir haben uns für Qt entschieden, da es für Linux, MacOS und Windows verfügbar ist und eine aktive Community hat.
- Oder eher **implizit** bei der Umsetzung?
  - Der Entwickler Alex stellt Qt vor, weil es das einzige UI-Framework ist, das er kennt.



Man kommt nicht umhin, eine Architektur zu haben…

… also sollten Sie* sich besser auf eindeutige, wohlüberlegte Entscheidungen stützen!

* als Architekt:innen!

Sind **alle Entscheidungen** im Hinblick auf die Softwarearchitektur **relevant**?

– Nein! (z.B. Farbe des Abbrechen-Icons eher nicht relevant für die Architektur)

Welche Entscheidungen sind relevant?

– Entscheidungen, die das System als Ganzes betreffen

– Entscheidungen, die schwer und daher kostspielig zu ändern sind

– Entscheidungen, bei denen ein hohes Risiko besteht, dass sie falsch sind (z. B. bei neuen oder unbekannten Technologien)

Beispielhafte Designentscheidungen

Beispiel

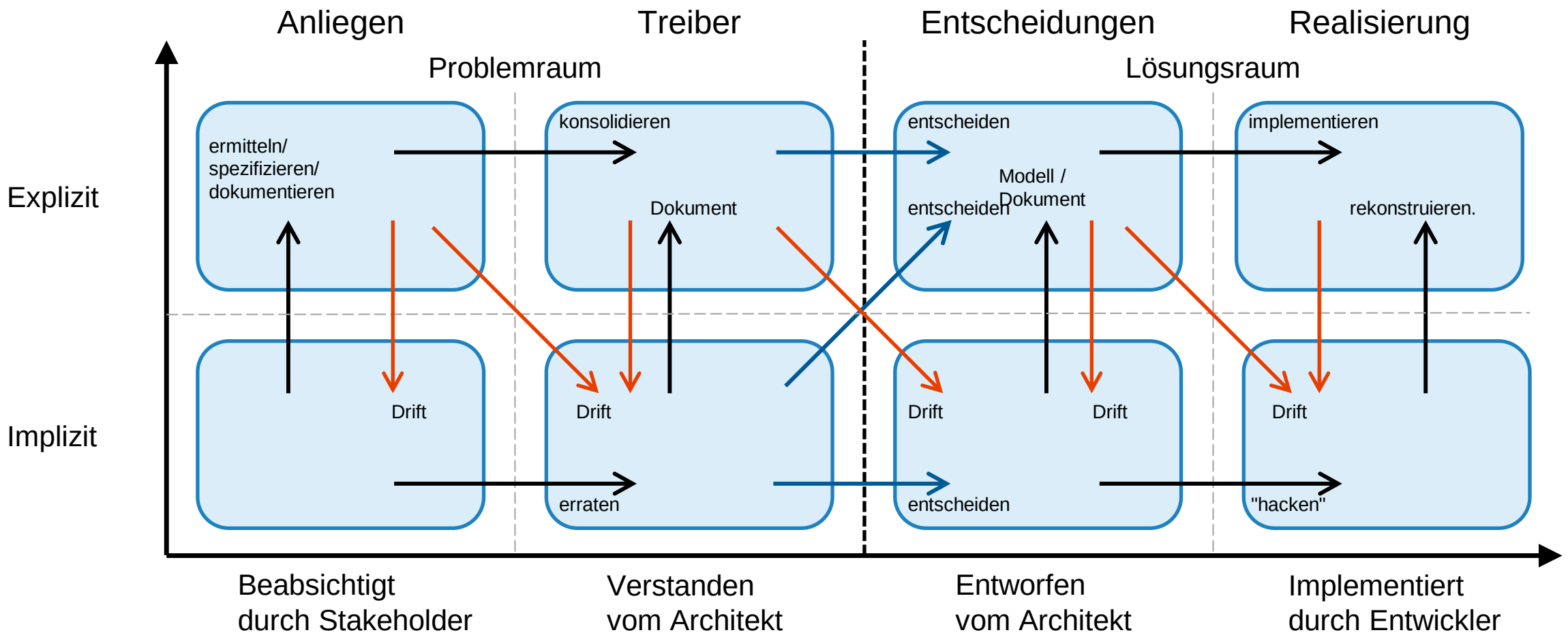
- Programmiersprache Java
- Eine zentrale Datenbank
- Microservice-Architektur
- Nutzung einer Platform-as-a-Service (PaaS) eines Cloud-Anbieters
- Verwendung eines Frameworks zur App-Erstellung für iOS- und Android-Geräte
- JSON als Datenformat
- Komprimierung der Daten zwischen Client und Server aufgrund geringer Netzwerkbandbreite
- Outsourcing der Implementierung einer Komponente
- ...

Ein Wort zur "guten" Architektur

- Es gibt **keine "gute" Architektur**!
  → absolute Messung nicht möglich
  → ein Vergleich zwischen der Architektur verschiedener Systeme ist wenig sinnvoll

- **Die Architektur** muss ihrem **Zweck angemessen (adäquat)** sein
  → immer anhand des Systems und seiner Anforderungen messen/beurteilen

Arbeiten Sie kontinuierlich gegen das Abdriften in den "impliziten Bereich"!



Architektur versus Architekturarbeit

- "Architektur" kann beides bedeuten:
  - Die **Aktivitäten** im Kontext der Softwarearchitektur:
    - Gestaltung der Architektur
    - Dokumentation der Architektur
    - Kommunikation und Verhandlung mit Stakeholdern
  - und die **Artefakte**, die während der Aktivitäten produziert werden:
    - die Systemstruktur, wie sie sich im Code manifestiert
    - Modelle und Sichten
    - Leitlinien (Guidelines)
    - dokumentierte Entscheidungen

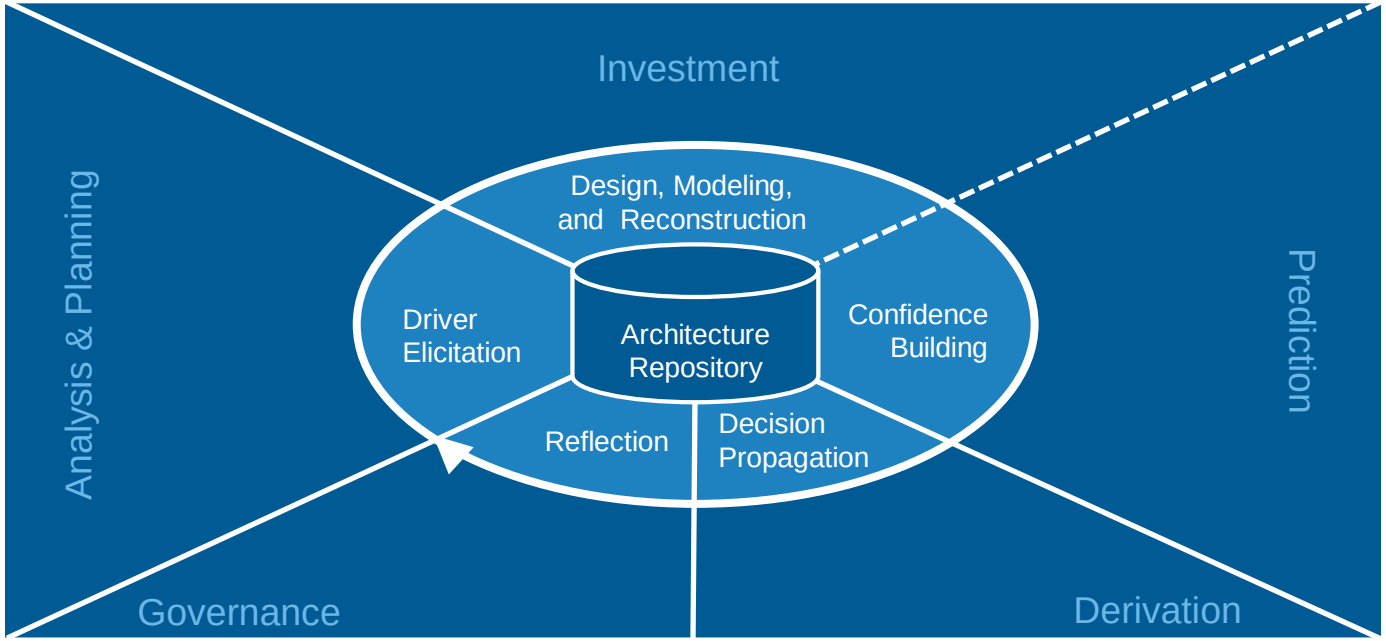
Zur Unterscheidung verwenden wir die Begriffe

- **Architekturarbeit**
  engl. *Architecting*

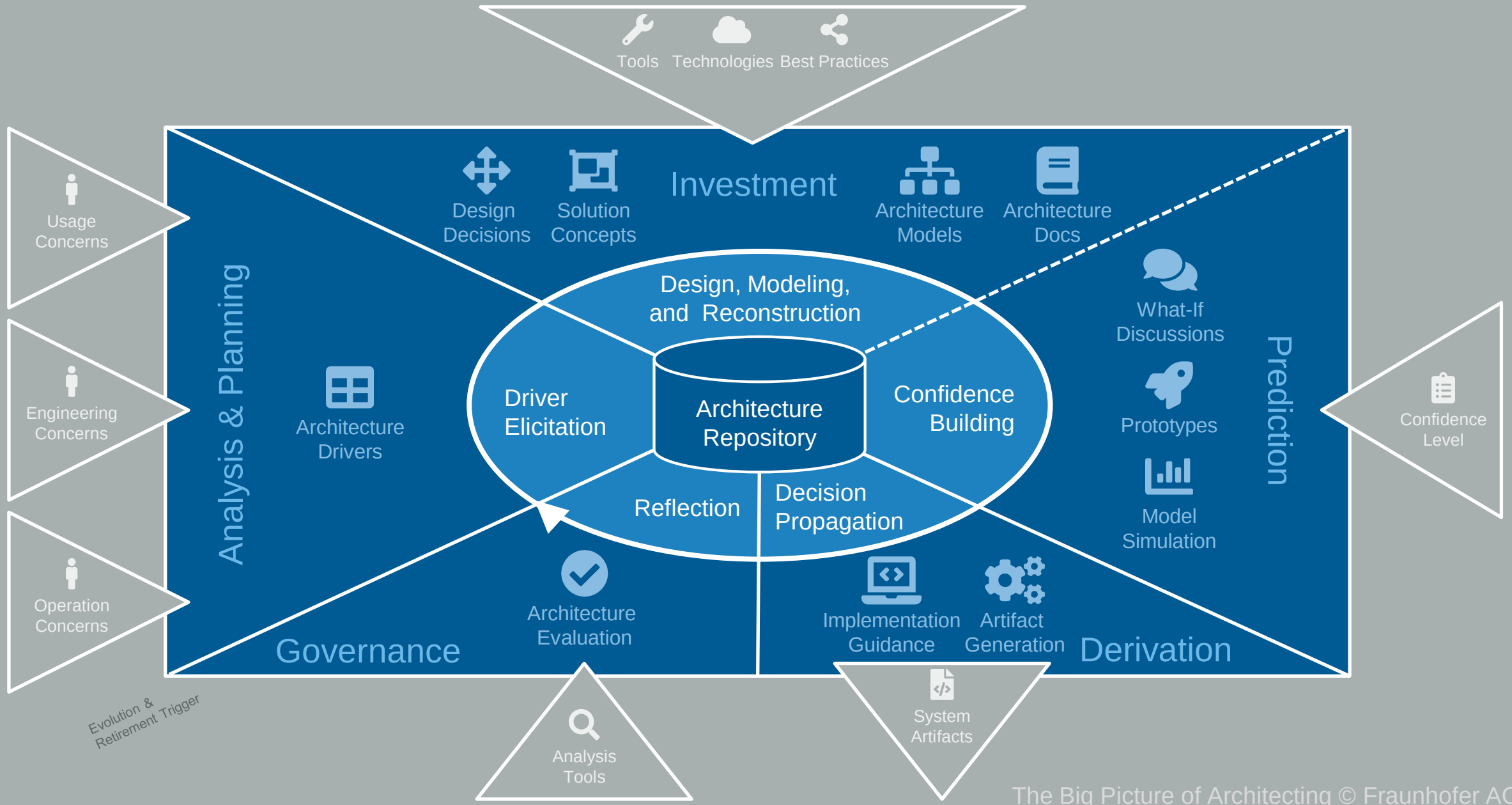
- **Architektur**, und
  engl. *Architecture*

Architektur

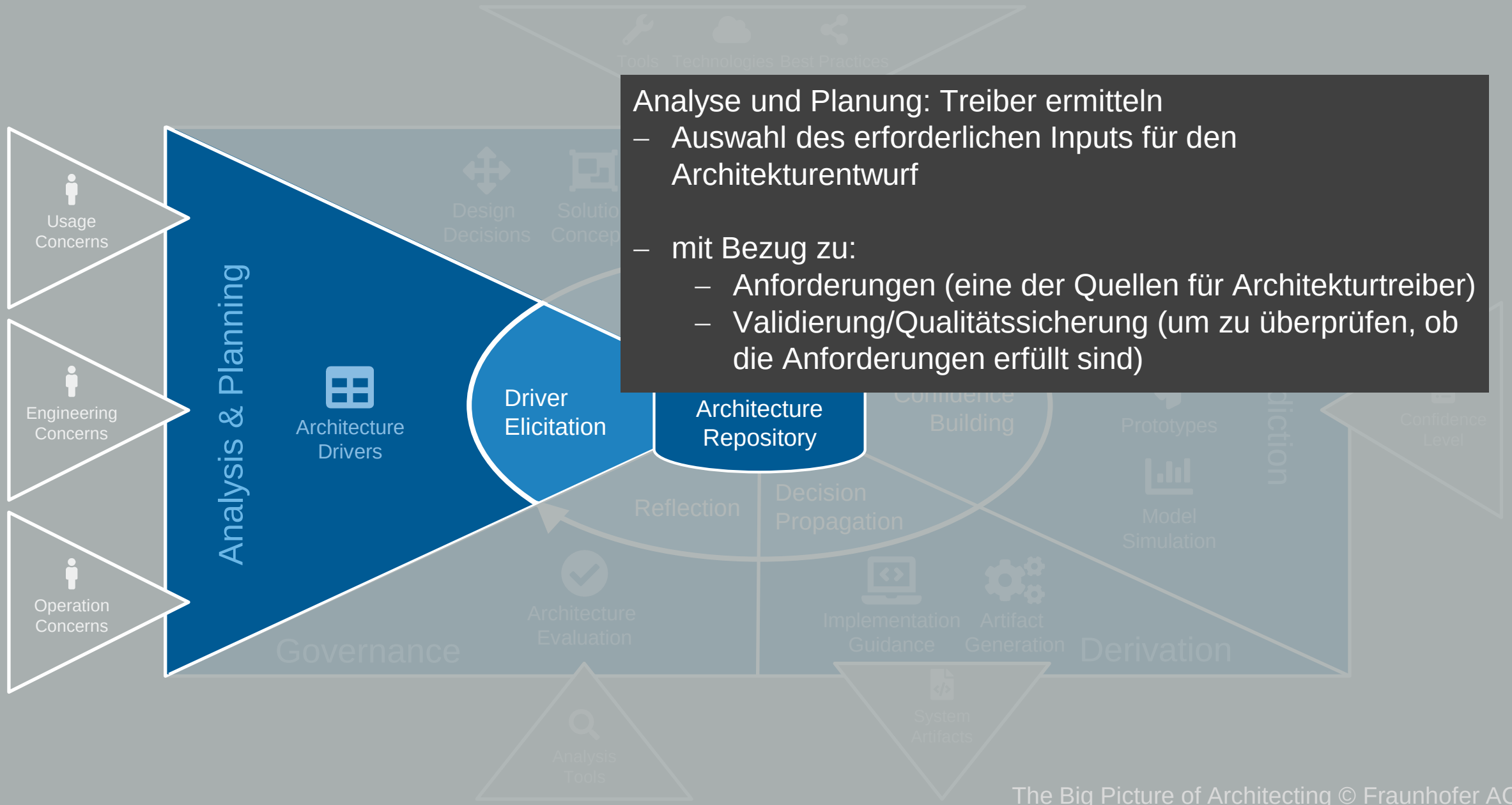
- Architektur ist keine Phase mit klar festgelegtem, zeitlichen Rahmen
  - ~~"Man beginnt mit Archtitekturarbeit, wenn die Anforderungen klar sind, und endet, bevor die Umsetzung beginnt"~~ - falsch!
- Architekturarbeit bezieht sich auf den **gesamten Software-Lebenszyklus** von der Spezifikation bis zum Betrieb und unterstützt diesen.



The Big Picture of Architecting



The Big Picture of Architecting

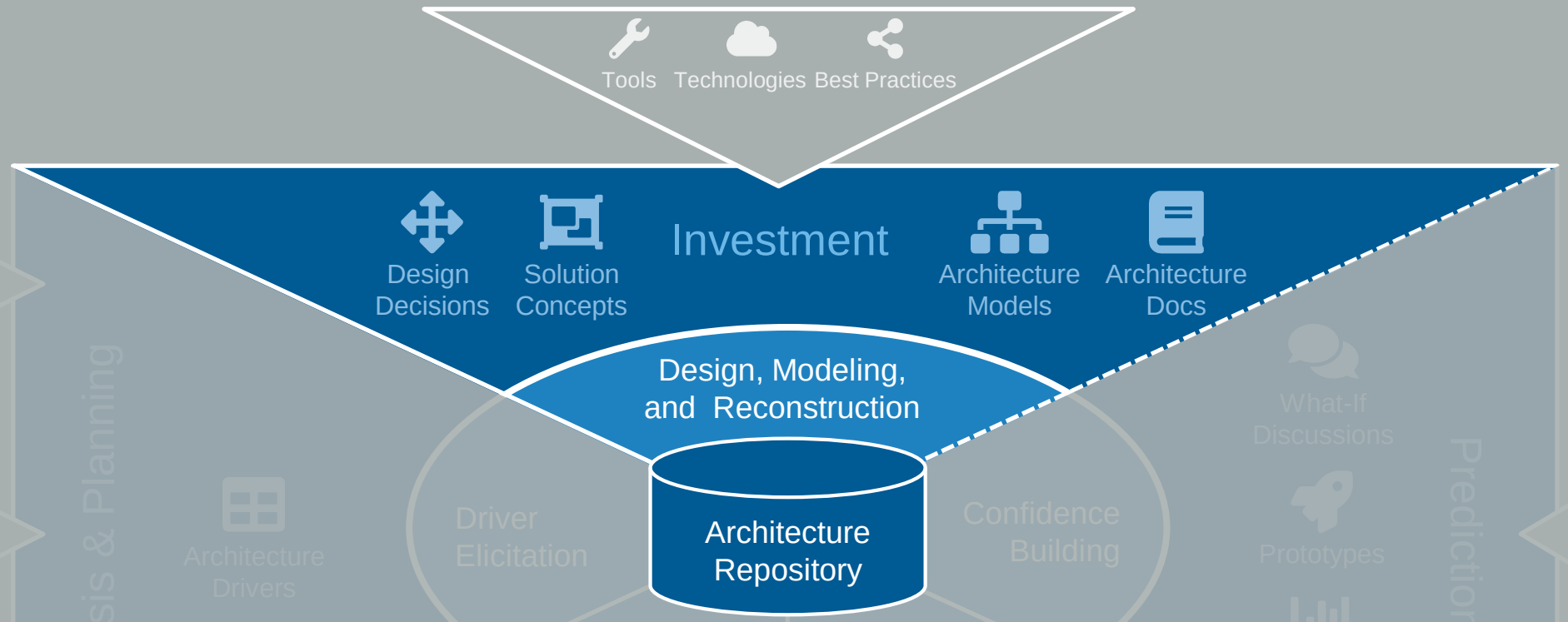


Analyse und Planung: Treiber ermitteln

- Auswahl des erforderlichen Inputs für den Architekturentwurf

- mit Bezug zu:
  - Anforderungen (eine der Quellen für Architekturtreiber)
  - Validierung/Qualitätssicherung (um zu überprüfen, ob die Anforderungen erfüllt sind)

Investment



Investition: Entwurf, Modellierung und Rekonstruktion

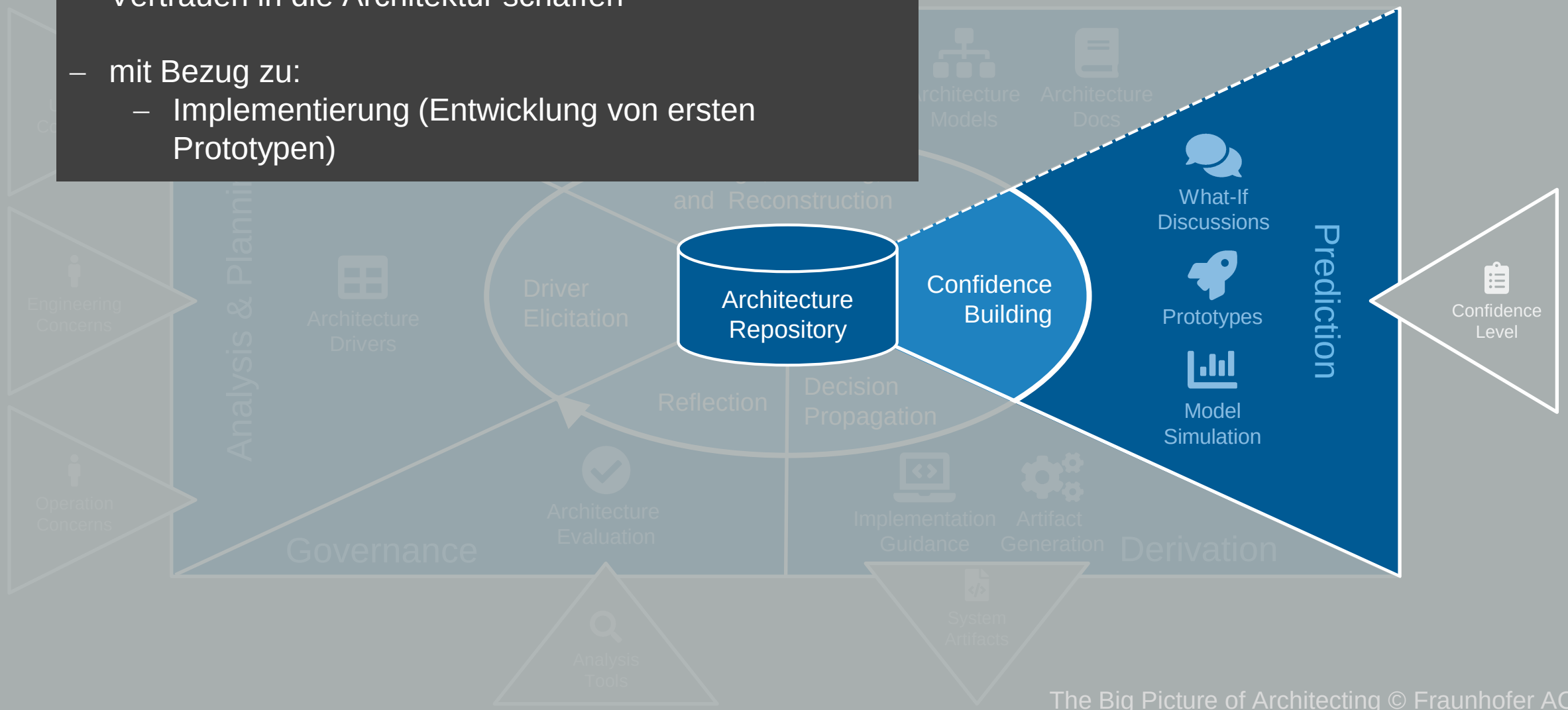
- In die Architektur investieren

- mit Bezug zu:
  - Architekturentwurf
  - (Quellcode des bestehenden Systems)

Prediction

Vorhersage: Vertrauen aufbauen
- Vertrauen in die Architektur schaffen

- mit Bezug zu:
  - Implementierung (Entwicklung von ersten Prototypen)



Derivation

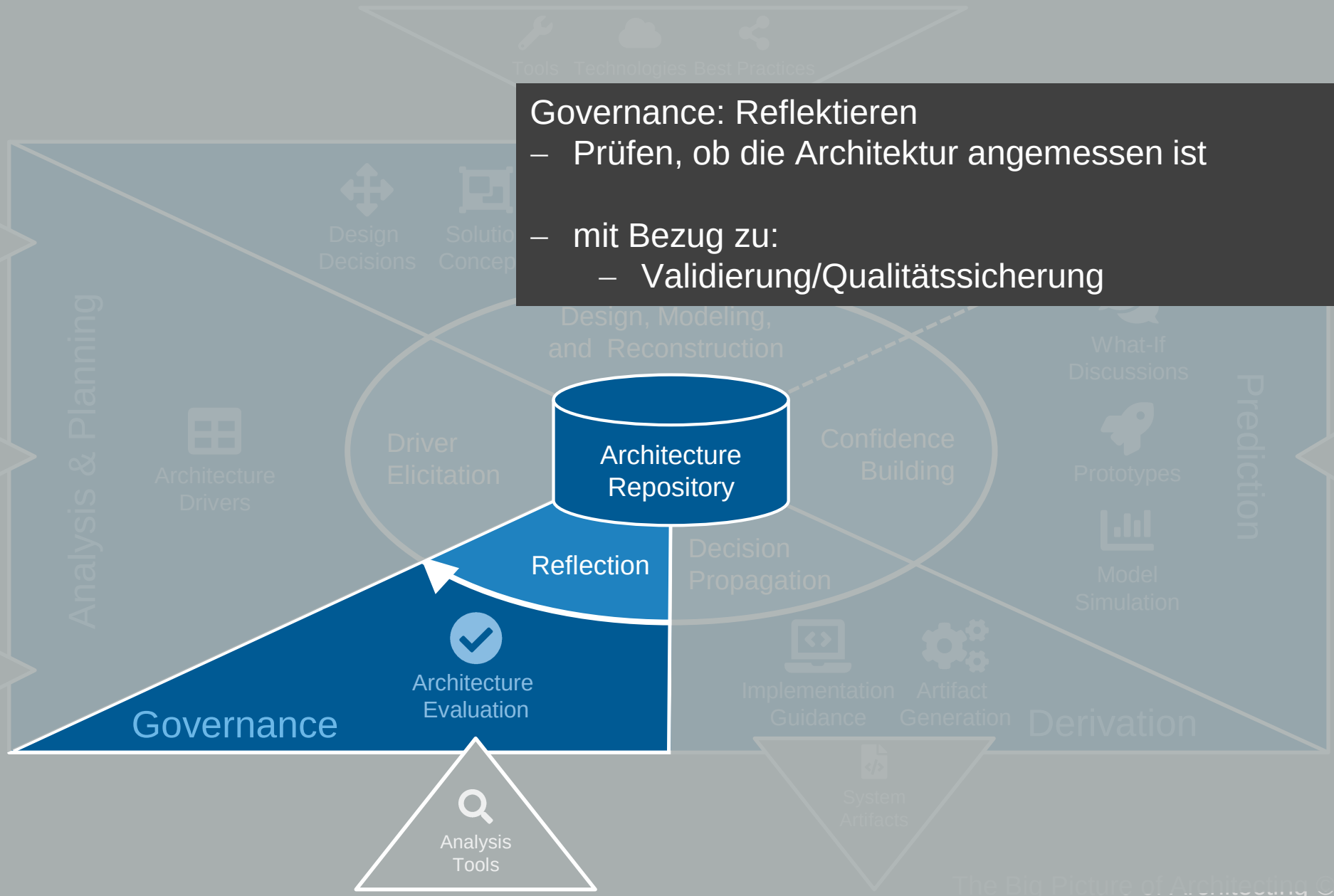
Ableitung: Entscheidungen verbreiten
- das System realisieren

- mit Bezug zu:
  - Implementierung

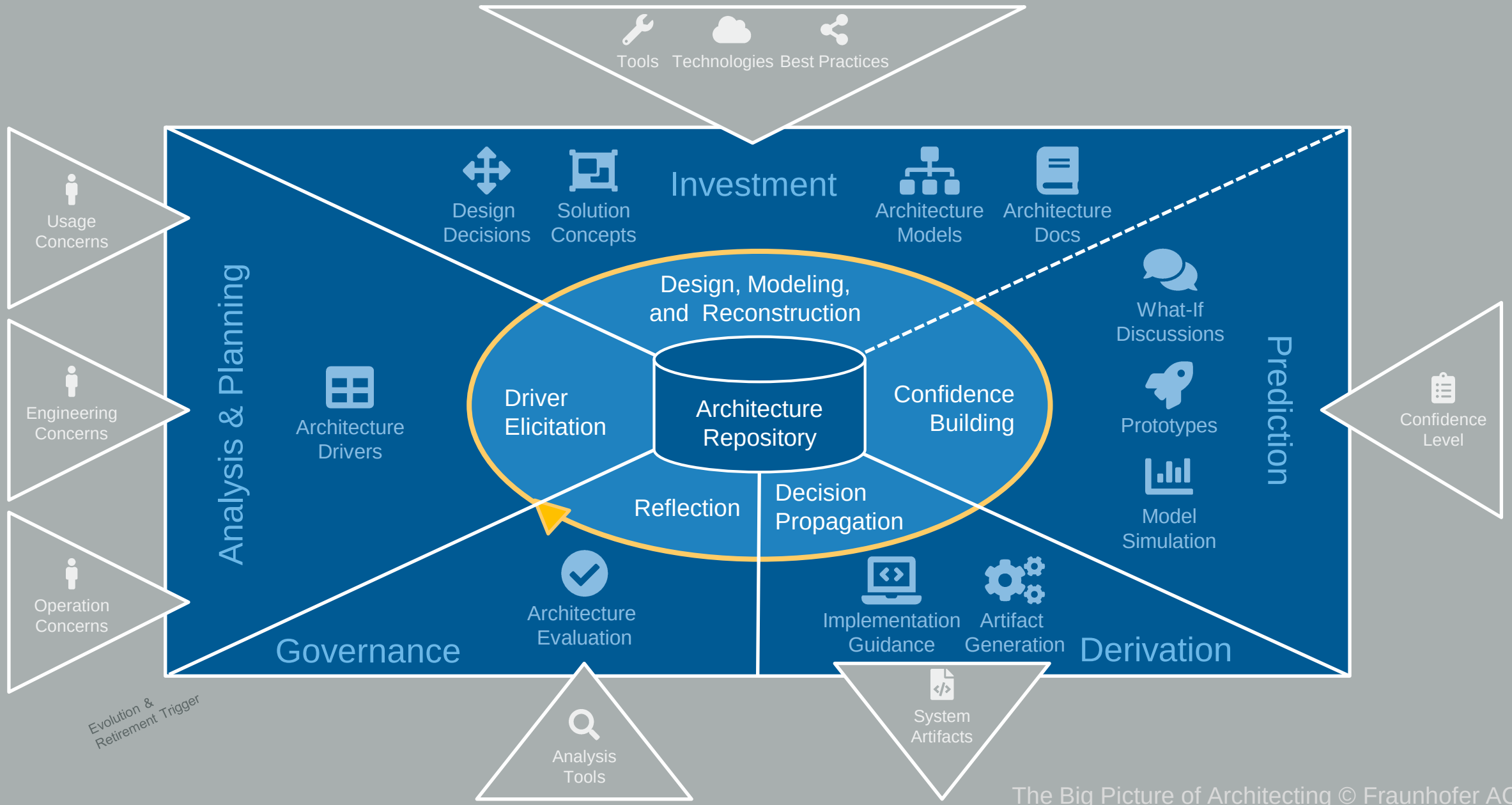
Governance

Governance: Reflektieren

- Prüfen, ob die Architektur angemessen ist
- mit Bezug zu:
  - Validierung/Qualitätssicherung

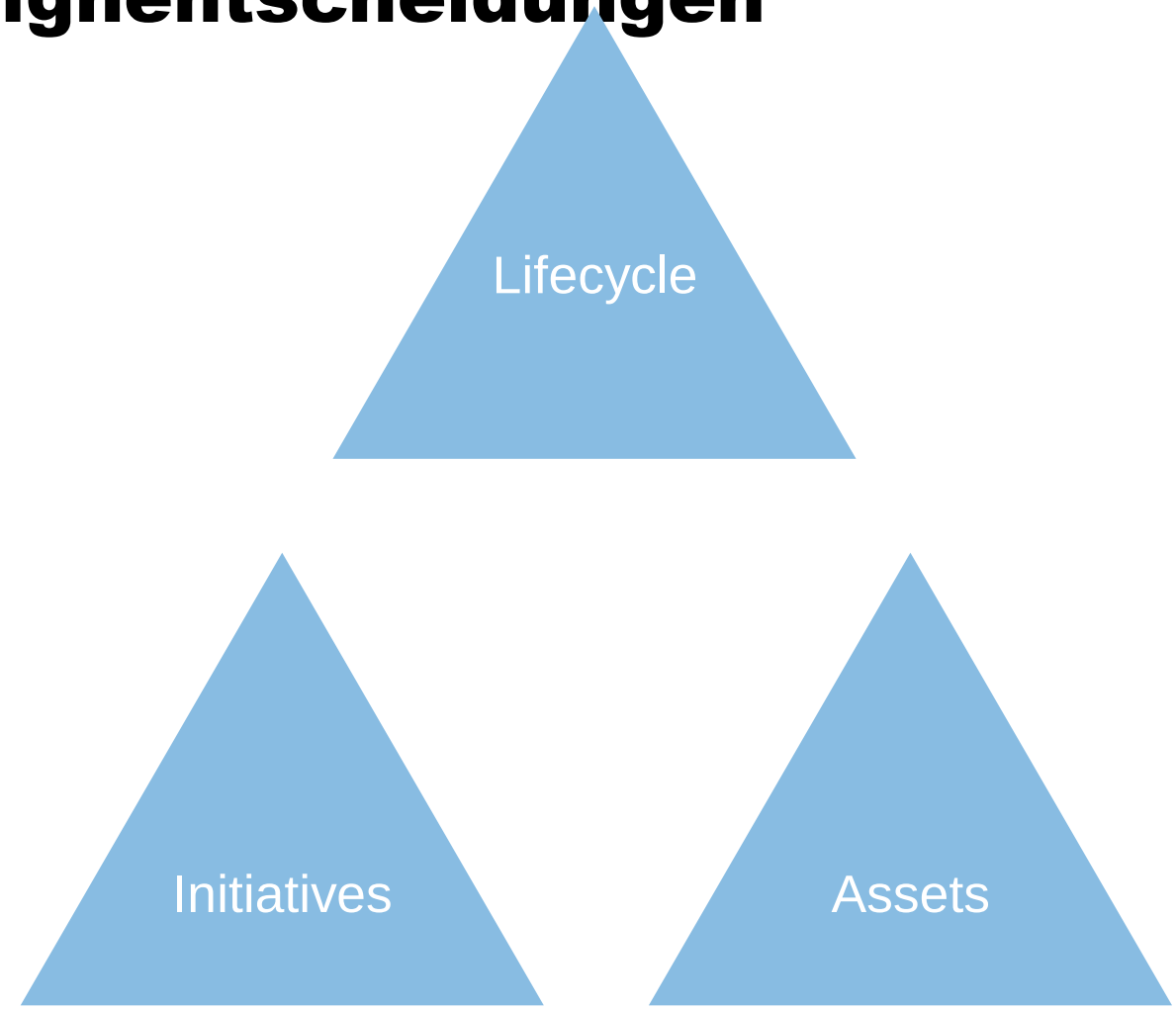


Es ist ein Kreislauf!



Auswirkungen von Designentscheidungen

In der Architektur können sich **Designentscheidungen** auf viele verschiedene Aspekte auswirken



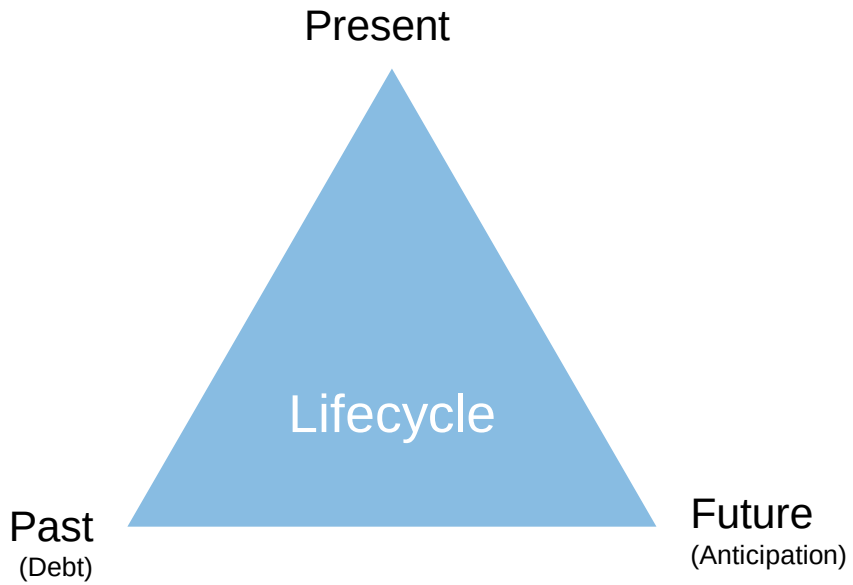
Lifecycle

Lebenszyklus der Software, d.h. Erstellung, Entwicklung und Stilllegung.
Das sollte jeder Architekt immer auf dem Radar haben.

Vergangenheit: Bereits bestehendes System und technische Schulden.

Gegenwart: Die aktuelle Weiterentwicklung der Software (neue/geänderte Funktionen), um das nächste Inkrement zu erzeugen

Zukunft: mittel- bis langfristige Planung; auch geplante Lebensdauer der Software.



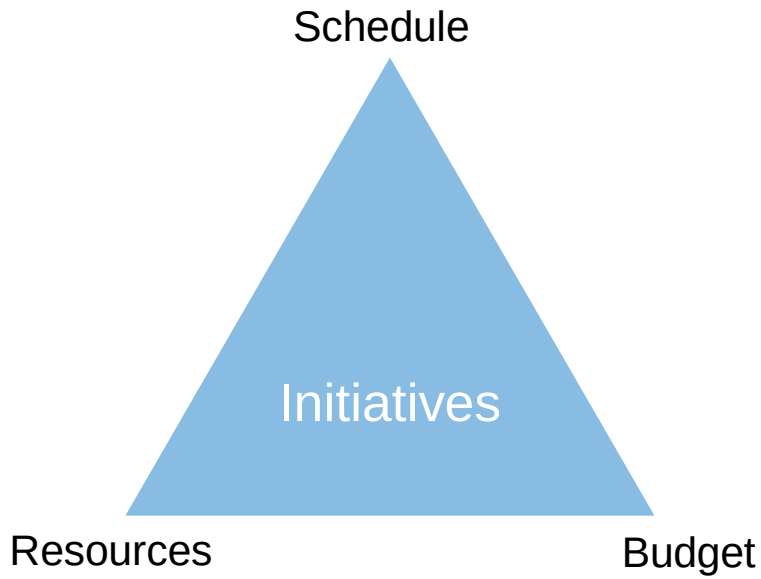
Initiatives

Initiativen beschreibt die Planung der zu erledigenden Aufgaben. Dies sind typische Projektmanagement-Themen, aber auch Softwarearchitekt:innen müssen diese Rahmenbedingungen beachten (z.B. bei Make-or-Buy-Entscheidungen)

Ressourcen: die verfügbaren Personen.

Zeitplan: Der Zeitplan, Beginn, Ende des Projekts, Zwischenergebnisse (Meilensteine).

Das Budget: Das verfügbare Budget zur Finanzierung des Projekts.



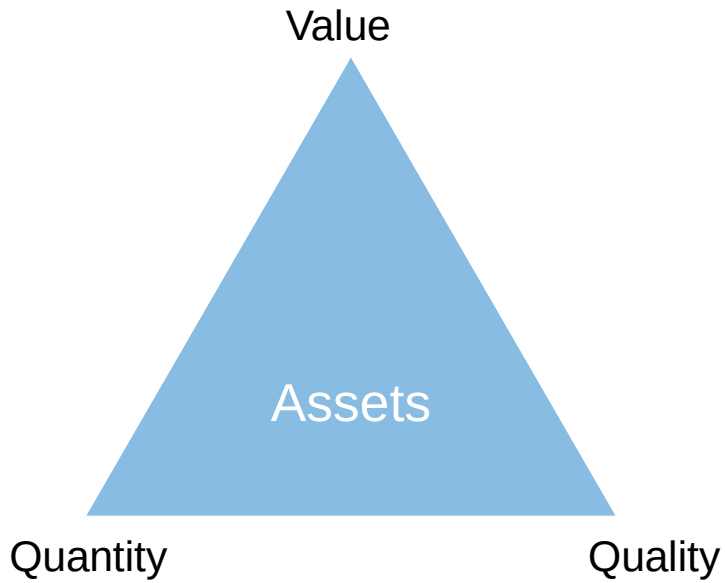
Assets

Assets (*Wirtschaftsgüter*) sind das Ergebnis des Projekts

Wert: Wert für Kunden und für das Entwicklerunternehmen

Menge: Umfang der Lösung (z. B. Anzahl der Funktionen)

Qualität: Qualität der Lösung. Wie gut/schnell/zuverlässig funktioniert das System?



Das Bermuda-Dreieck der Architektur

… hier alles zusammen betrachtet.

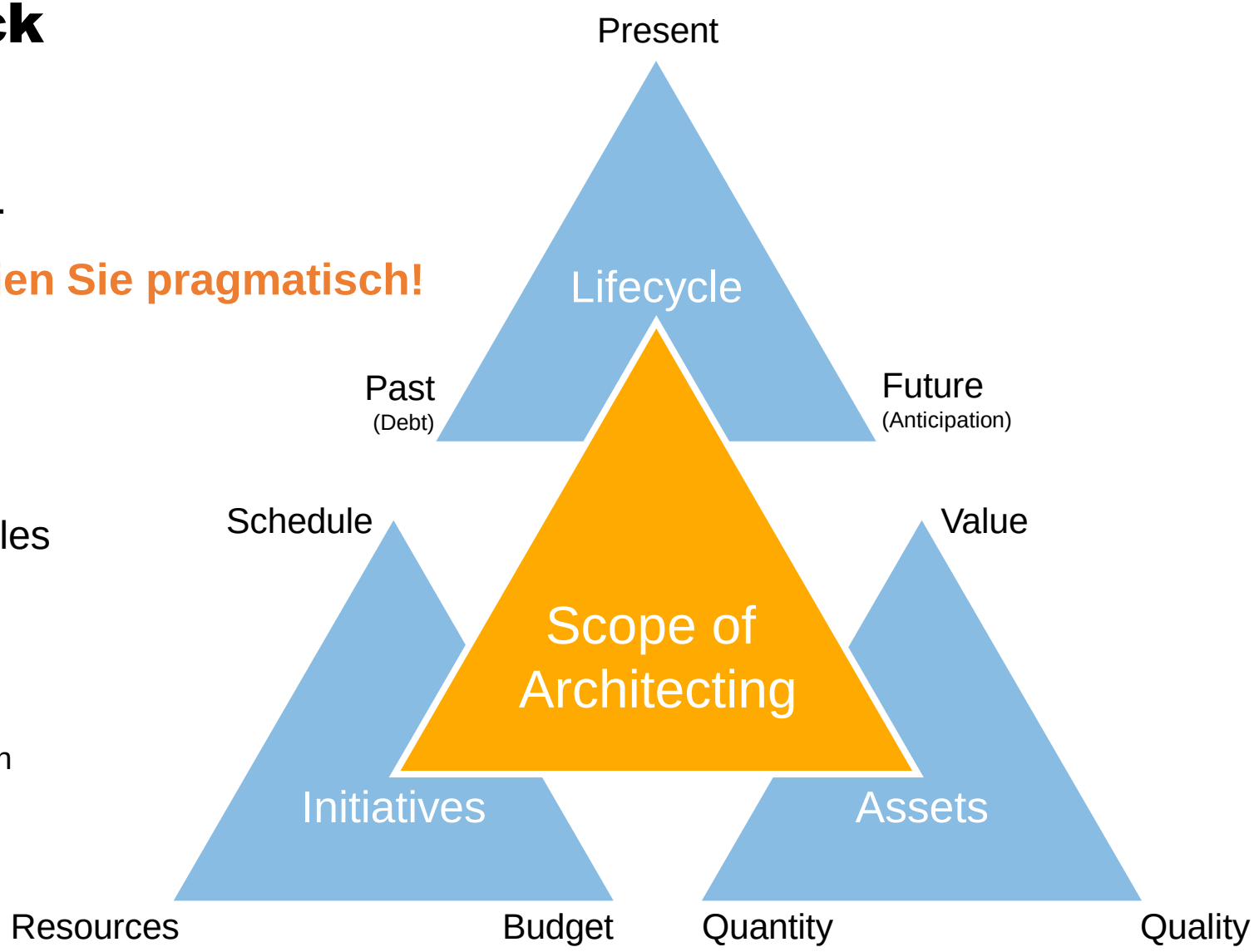
Denken Sie daran:

Seien Sie pragmatisch!

- Verlieren Sie sich nicht & Verlieren Sie Ihre Investitionen nicht!

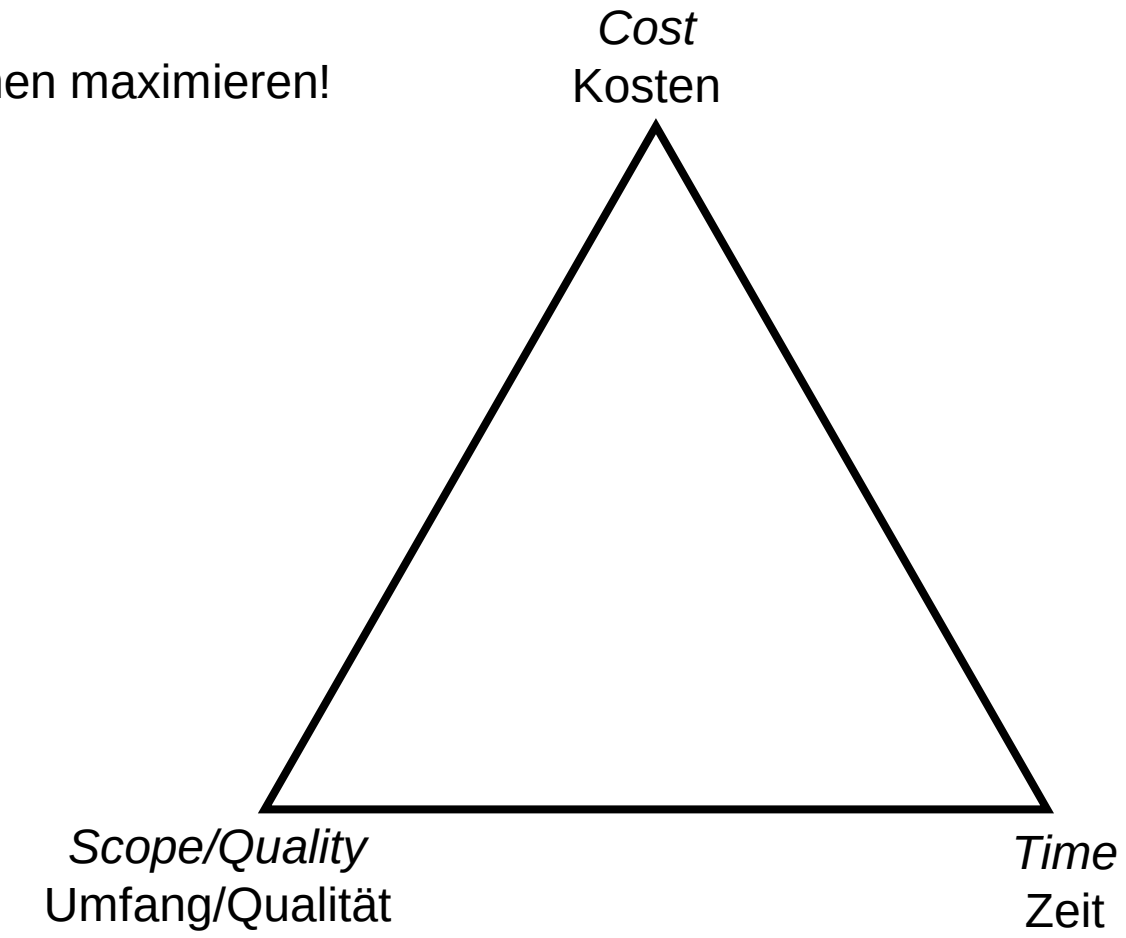
- Halten Sie mit der Architektur alles im Gleichgewicht!

Hinweis: Eine Änderung in einer Dreiecksdimension wirkt sich auf die anderen aus!



Hintergrund: Dreieck des Projektmanagements

- Teil eines jeden Projektmanagementkurses
- Unterm Strich: Man kann nicht alle Dimensionen maximieren!



Geschichte - Entstehung des Bereichs Softwarearchitektur

- 1992 Dwayne Perry and Alexander Wolf
  “Foundations for the Study of Software Architectures”
  in Software Engineering Notes

- 1993 David Garlan and Mary Shaw
  “An Introduction to Software Architecture”
  in Advances in Software Engineering and Knowledge Engineering

- 1994 Special Issue on Software Architecture
  in IEEE Transactions on Software Engineering

- 1995 Special Issue on Software Architecture
  in IEEE Software Magazine

Heute - sehr aktiver und etablierter Bereich

- Verfügbare Ressourcen
  - Dutzende von Büchern über Software-Architektur
  - Viele Informationen im Internet verfügbar
- Architekt:innen und Architekturgruppen in vielen Unternehmen
- Praktikerkonferenzen
  - OOP, SEACON, SATURN, O'Reilly Architektur Konferenz, The Architecture Gathering
- Akademische Community
  - Dedizierte Konferenzen
    - ICSA (IEEE International Conference on Software Architecture)
    - ECSA (European Conference on Software Architecture)
    - Spezielle Workshops, die mit den meisten Software-Engineering-Konferenzen verbunden sind
  - Fester Bereich für Architektur im Journal of Software and System Engineering (JSS)
- Wichtige Schlagworte und Trends im Bereich der Architektur:
  - MDA, SOA, MicroServices, SCS, CQRS, Serverless ...

Diskussion: Dokumentation der Architektur

- In den nächsten Abschnitten werden wir systematisch lernen, wie man eine Softwarearchitektur modelliert und dokumentiert.

- Allerdings sagt Chris:

"
  - **Wir brauchen die Software-Architektur nicht zu dokumentieren**.
  - Sie ist im Code manifestiert.
  - Die Dokumentation ist sehr schnell veraltet.
  - **Der Code ist die einzige Wahrheit.**
  - Außerdem bin ich der Experte. Ich kann jeden Teil des Systems mit Hilfe des Codes erklären. Immer!

"

- Was meinen Sie dazu? Hat Chris recht?

Diskussion: Dokumentation der Architektur

- Chris irrt sich:
  - Der Code ist zu feinkörnig, um sich einen **Überblick über das System als Ganzes zu verschaffen**
  - Implementierung braucht Anleitung (**explizite** Architekturdokumentation)
  - Der Code ist das Ergebnis von **Designentscheidungen**
    - Getroffene Entscheidungen können (zumindest teilweise) rekonstruiert werden
    - Aber **warum wurden** diese Entscheidungen getroffen?
    - Welche **Alternativen** wurden verworfen?
  - Architekturdokumentation ist nicht nur für die **Implementierung** hilfreich, sondern auch für **Projektmanagemententscheidungen** (z.B. Make-or-Buy)
  - Nicht skalierbar für große Projekte - Chris ist eine begrenzte Ressource
  - Und nicht zuletzt: Chris kann krank werden oder das Team verlassen. Wer ist dann der Experte?

Grundlagen der Architektur

"Jedes System hat eine Architektur!"

"Man kann nicht verhindern, dass man eine Architektur bekommt ...!"

"Man kann nur vermeiden, eine falsche zu bekommen ...!"

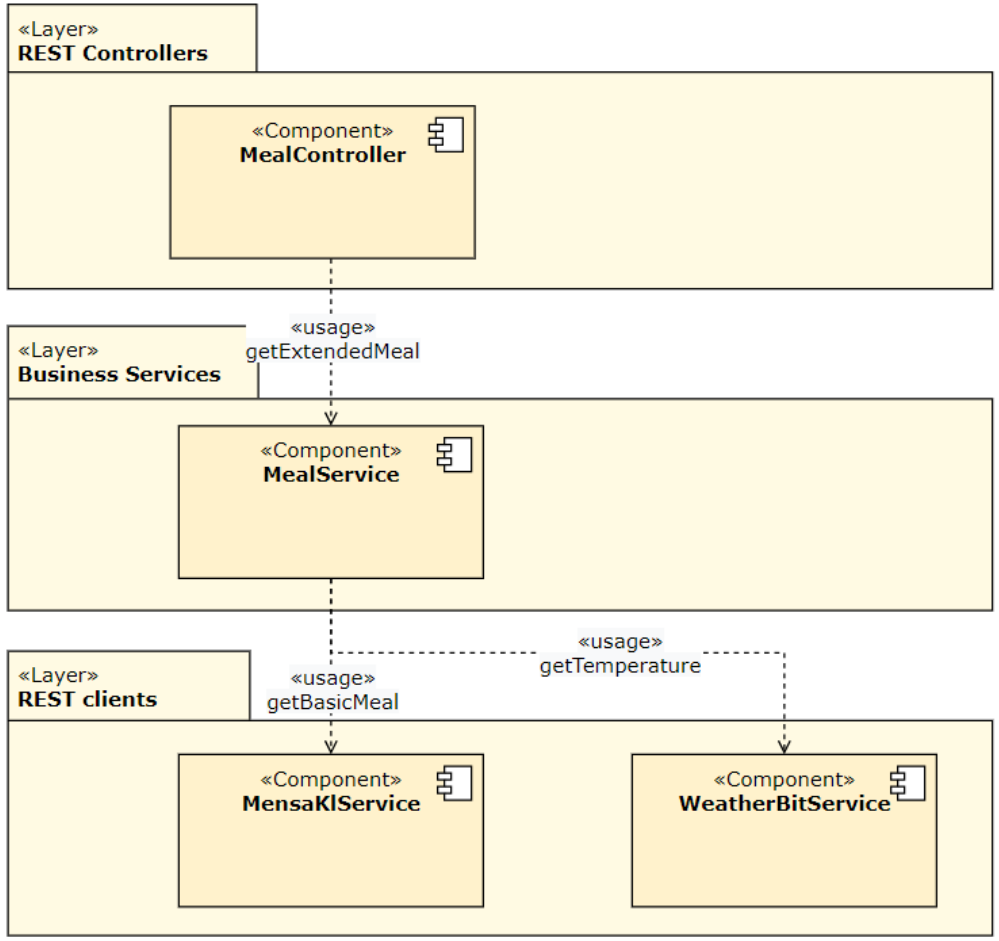
"Obwohl das System eine Architektur hat, ist sie vielleicht nicht bekannt!"

"Die tatsächlich implementierte Architektur kann sich stark von der beabsichtigten Architektur unterscheiden!"

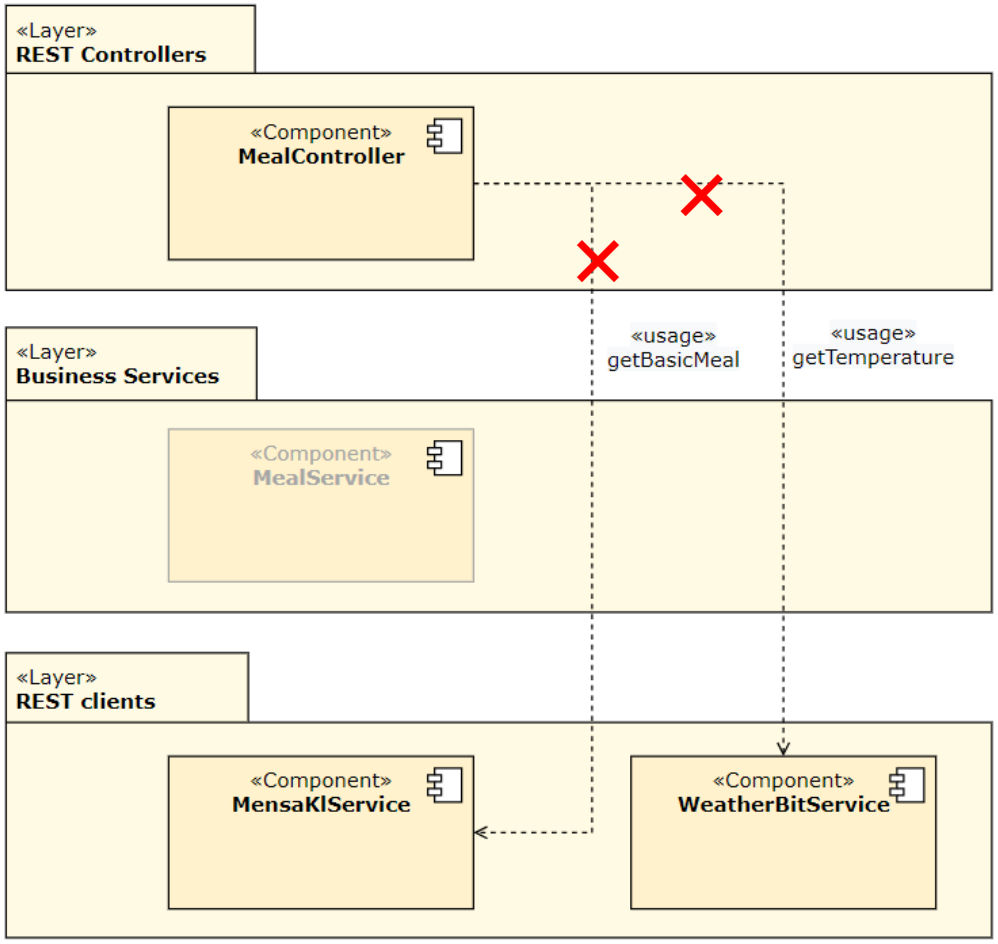
Diese Abweichung nennt man **Erosion**.

Beispiel Beabsichtigte vs. implementierte Architektur

Beabsichtigt

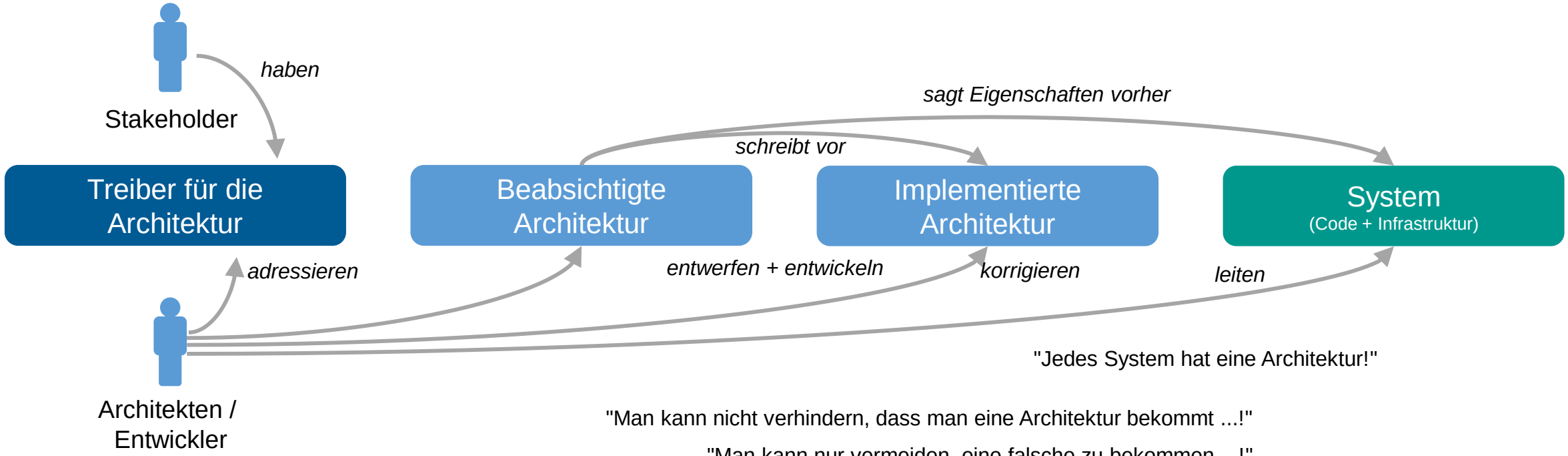


Implementiert



Ebenenprinzip "nur Zugang zur nächsten Ebene" verletzt

Grundlagen der Architektur



"Jedes System hat eine Architektur!"

"Man kann nicht verhindern, dass man eine Architektur bekommt ...!"

"Man kann nur vermeiden, eine falsche zu bekommen ...!"

"Obwohl das System eine Architektur hat, ist sie vielleicht nicht bekannt!"

"Die tatsächlich implementierte Architektur kann sich stark von der beabsichtigten Architektur unterscheiden!"

"Wenn nicht klar ist, wer die Entscheidungen trifft, wird irgendjemand sie treffen!"

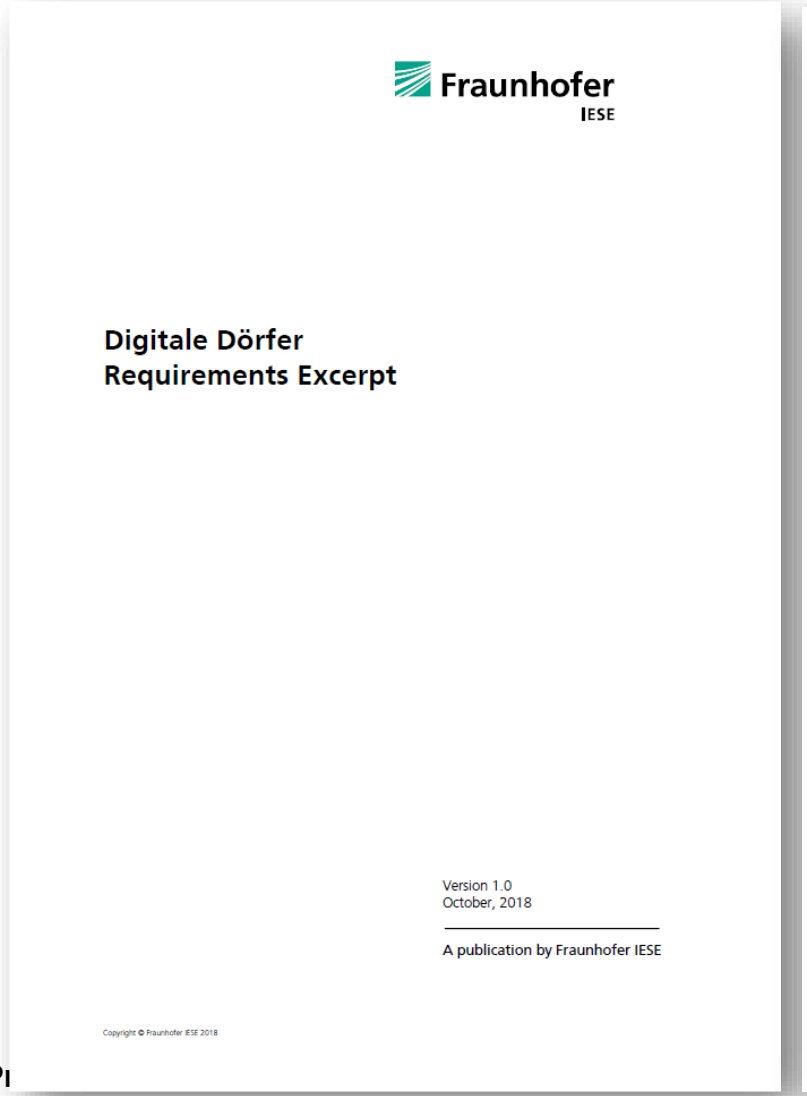
"Eine Architektur ist nicht gut oder schlecht. Sie ist angemessen oder nicht!"

"Architektur ist keine Phase!"

TODO für die nächste Vorlesung: Dokument lesen

Beispiel

- Verfügbar in Moodle
- Dies ist ein reduzierte Menge von Anforderungen zum Zweck eines lesbaren Beispiels. Es handelt sich um einen Auszug aus dem eigentlichen Anforderungsdokument von DD



Fraunhofer IESE

Digitale Dörfer
Requirements Excerpt

Version 1.0
October, 2018

A publication by Fraunhofer IESE

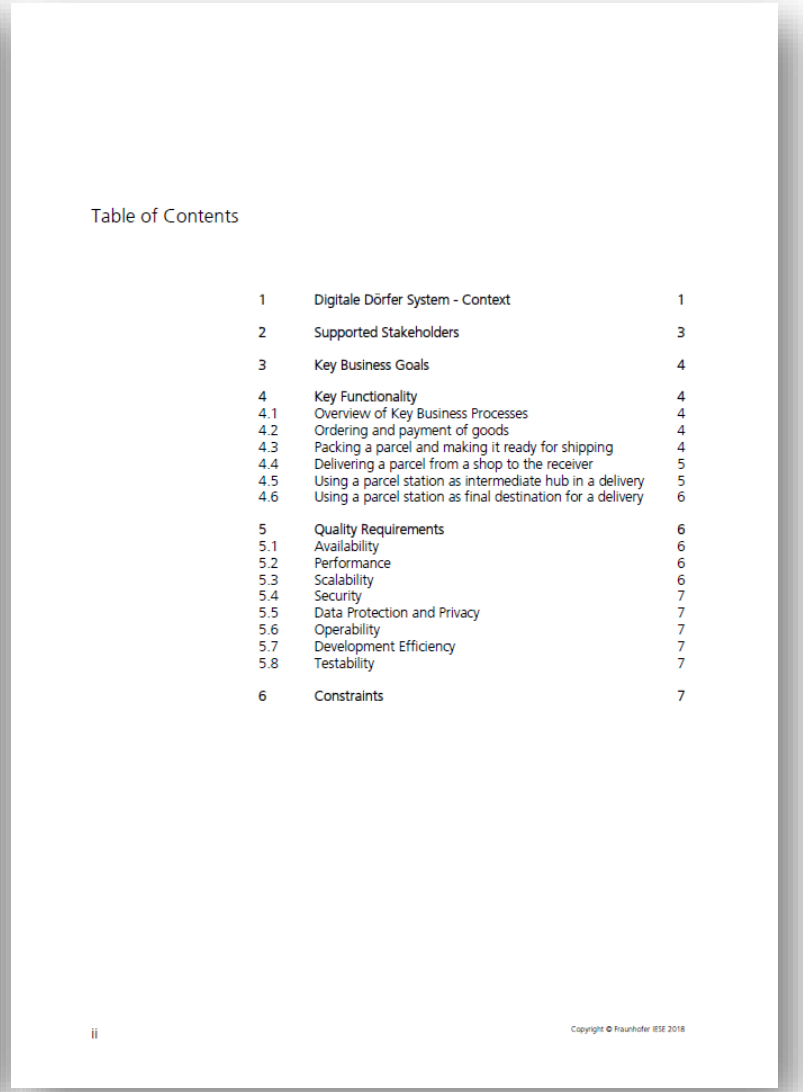


Table of Contents