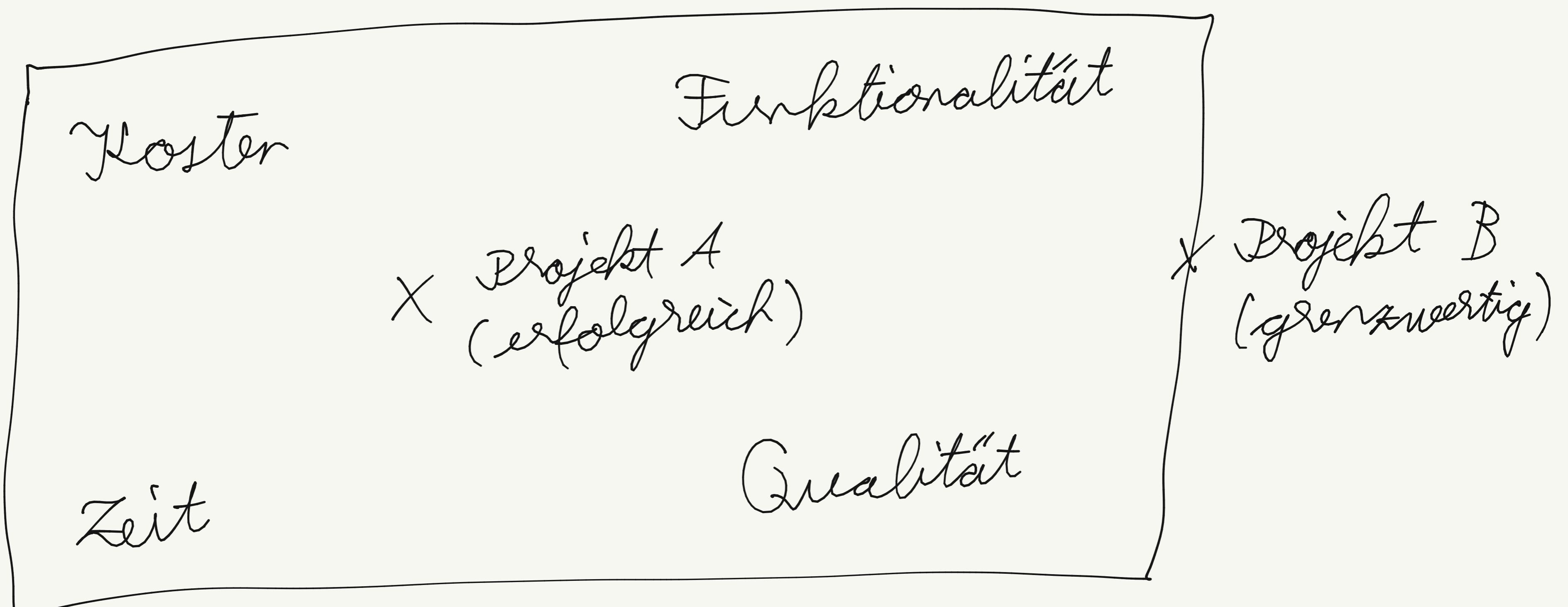


Vorlesung 1

Grundlegende Konzepte

Das magische Viereck erfolgreich
Softwareprojekte:



Projekt C
(unerfolgreich)

- Softwaresysteme sind ein zentrales Rückgrat unserer Gesellschaft
- Die Fähigkeiten IT-Projekte durchzuführen muss verbessert werden
=> Software Engineering / Softwarearchitektur

Warum sind Kenntnisse über Softwarearchitektur wichtig?

- Komplexität von Softwareprojekten
- Skalierbarkeit und Leistung
- Sicherheit und Datenschutz
- Technologische Vielfalt
- Agilität und kontinuierliche Bereitstellung

Big Ball of Mud

- Begriff, um eine unstrukturierte und chaotische Softwarearchitektur zu beschreiben
- Typische Merkmale:
 - Mangel an Struktur
 - Schlechte Modularisierung
 - Spaghetti - Code
 - Technologische Vielfalt
 - Wachsende Komplexität
- Die Entstehung eines BBOM muss unter allen Umständen vermieden werden
- ⇒ Software Architektur - Arbeit ist hier von entscheidender Bedeutung!

Technische Schulden

- Entstehen, wenn Teams bewusst Entscheidungen treffen, die kurzfristig Vorteile bringen, aber langfristig Auswirkungen auf die Qualität haben
- Ähnlich wie finanzielle Schulden, müssen technische Schulden irgendwann „zurückgezahlt“ werden durch Verbesserung des Source Codes

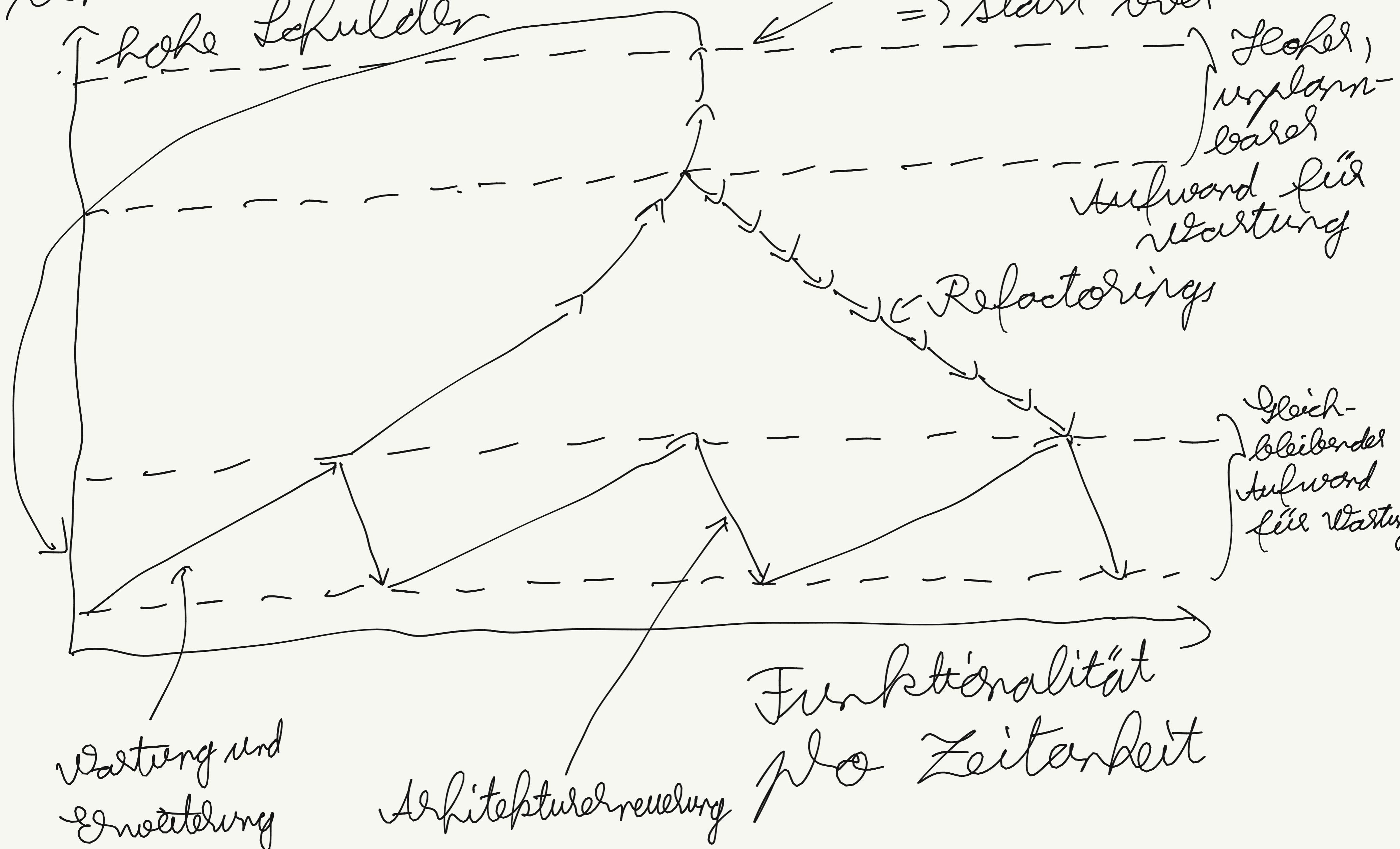
Ursachen und Art der technischen Schulden

- Schnelle Lösungen
 - Veraltete Technologien
 - Fehlende Tests
 - Schlechte Codequalität
 - Nicht behandelte Probleme
- Rückzahlung technischer Schulden erfordert Zeit, Ressourcen und Engagement des Entwicklungsteams

Entwicklung und Effekt vor technischer Schuldner

Entwickeln an technischer Schuldner

hohe Schuldner



Arten von IT-Architekten und Aufgaben

• Enterprise Architekt:

- Starke strategische Ausrichtung
- IT-Business-Alignment
- Technologie Roadmaps

• Solution / System Architekt

- Leadership
- Eingebunden in Anforderungsspezifikation
- Systemintegration

• Software Architekt

- Erstellen und Manager von Softwarearchitekturen
- Enge Zusammenarbeit mit Entwicklungsteam
- Kooperation, Teamwork, Netzwerken

Typische Aufgaben von Softwarearchitekten:

- Architektonische Planung
- Technologieauswahl
- Code - Reviews und Best Practices
- Skalierung und Performance
- Risikomanagement
- Kommunikation und Zusammenarbeit
- Wartbarkeit und Erweiterbarkeit
- Qualitätssicherung

Wann ist eine Softwarearchitektur erfolgreich?

- Der Erfolg einer Softwarearchitektur bemisst sich am Erfolg der Software!
 - Beitrag zur Weiterleitung leisten!
- Der Erfolg bemisst sich nicht an der:
 - Qualität der Diagramme
 - Anzahl der Diagramme
 - Komplexität der Diagramme
 - Anzahl der verwendeten Architekturnmustern
 - ...

Notwendige Fähigkeiten von IT-Architekten

- Soziale
- Technische
- Methoden
- Frameworks
- Business
- Zertifikate
- Rechtlich

Definition Softwarearchitektur und Schnittstelle

Softwarearchitektur = Fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes "

→ "Grundlegende Konzepte oder Eigenschaften einer Entität in ihrer Umgebung und leitende Prinzipien für die Realisierung und Entwicklung dieser Entität und der damit verbundener Lebenszyklusprozesse"

"Fundamental concepts or properties"

- Are usually intended to be embodied in the entity's components, the relationships between components, and the relationships between the entity and its environment
- Sollte in der Regel durch die Komponenten der Entität, die Beziehungen zwischen den Komponenten und durch die Beziehungen zwischen der Entität und ihrer Umgebung verkörpert werden

Architecture Entity (Architektureinheit)

= thing being considered, described, discussed, studied or otherwise addressed during the architecting effort

Beispiele:

- Unternehmen
- Organisationen
- System (insbesondere Softwaresysteme)
- Subsysteme

Zentrale Aspekte der Definition von Softwarearchitektur

- Fundamentale Konzepte und Eigenschaften eines Systems:
 - Die Komponenten des Systems
 - Die Beziehungen / Schnittstellen zwischen den Komponenten
 - Die Beziehungen zwischen dem System und seiner Umgebung

Schnittstelle

Definition: Eine Schnittstelle repräsentiert einen wohldefinierten Zugangspunkt zum System oder zu dessen Komponenten.
Dabei beschreibt eine Schnittstelle die Eigenschaften dieses Zugangspunkts, wie z.B: Attribute, Daten und Funktionen

Arter vor Schnittstellen

- Angebotene Schnittstelle
 - wird von einer Komponente bereitgestellt
 - definiert welche Dienste oder Funktionen verfügbar sind
 - andere Komponente kann dieses Schnittstelle nutzen, um Dienste in Anspruch zu nehmen
- Angeforderte Schnittstelle
 - wird von einer Komponente benötigt
 - definiert Abhängigkeiten und Anforderungen an andere Komponenten
 - Komponente kann nur dann funktionieren, wenn alle angeforderten Schnittsteller von anderen Komponenten bereitgestellt werden

Verantwortung für Schnittstellen:

- Normungsorganisationen
 - für allgemein akzeptierte Schnittstellen
 - meist standardisiert und definiert durch Organisationen wie zum Beispiel ISO, IEEE etc.

- Beispiele: USB, HDMI, TCP/IP
- Normungsorganisationen definieren Schnittsteller; Implementierung und Wartung werden meist durch Drittanbieter vorgenommen

• Schnittstellenanbieter

- Schnittstellenanbieter können verantwortlich für die Definition, Entwicklung, Wartung und Dokumentation der Schnittstelle sein (ist meistens der Fall)

- Beispiel: Google Maps API

(Google ist Anbieter und übernimmt auch die Verantwortung für die Definition, Entwicklung, Wartung und Dokumentation)

Schnittstellenverwendel

- Es kann vorkommen, dass ein Verwender der Schnittstelle gleichzeitig die Gesamtverantwortung oder Teile der Verantwortlichkeit für angebotene Schnittstellen übernimmt
- Beispiel: Entwicklung einer maßgeschneiderten API durch ein Softwareentwicklungsunternehmen für ein E-commerce - Unternehmen
 - E-commerce - Unternehmen:
 - Ist durch ein E-commerce - Unternehmen beauftragtes Unternehmen
 - Trägt inhaltliche Verantwortung für die Schnittsteller
 - Definiert Anforderungen an die Schnittsteller
 - Softwareentwicklungsunternehmen:
 - Implementiert die Schnittsteller, ist ggf. zusätzlich verantwortlich für Dokumentation, Wartung und weitere Aspekte

Komponenten

Modul

→ kann aus einem oder mehreren Modulen bestehen

· ist ein:

→ Konfiguration / Deklaration:

- Datenstruktur
- DB - Tabelle
- O/R - Mapping

→ Programmierstruktur:

- Klasse
- Funktion
- Skript

→ Bibliothek, Framework:

- React.js
- Flutter
- JUnit
- Bootstrap
- Unity

→ Komponente:

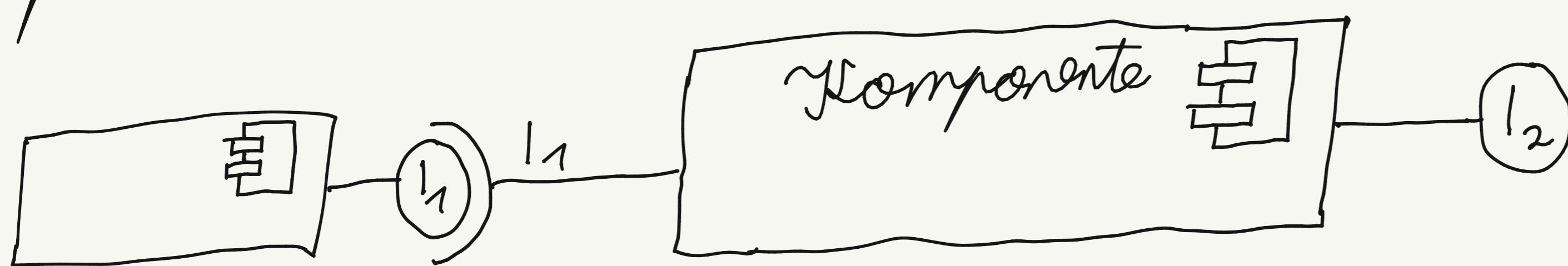
- Ließe Kernmerkmale einer Komponente

In der Praxis werden diese Begriffe je nach Kontext häufig unterschiedlich interpretiert

Fern Eigenschaften einer Komponente:

- Export und Import von Schnittstellen
- Kapselung und Austauschbarkeit
- Konfigurierbarkeit

Export und Import von Schnittstellen

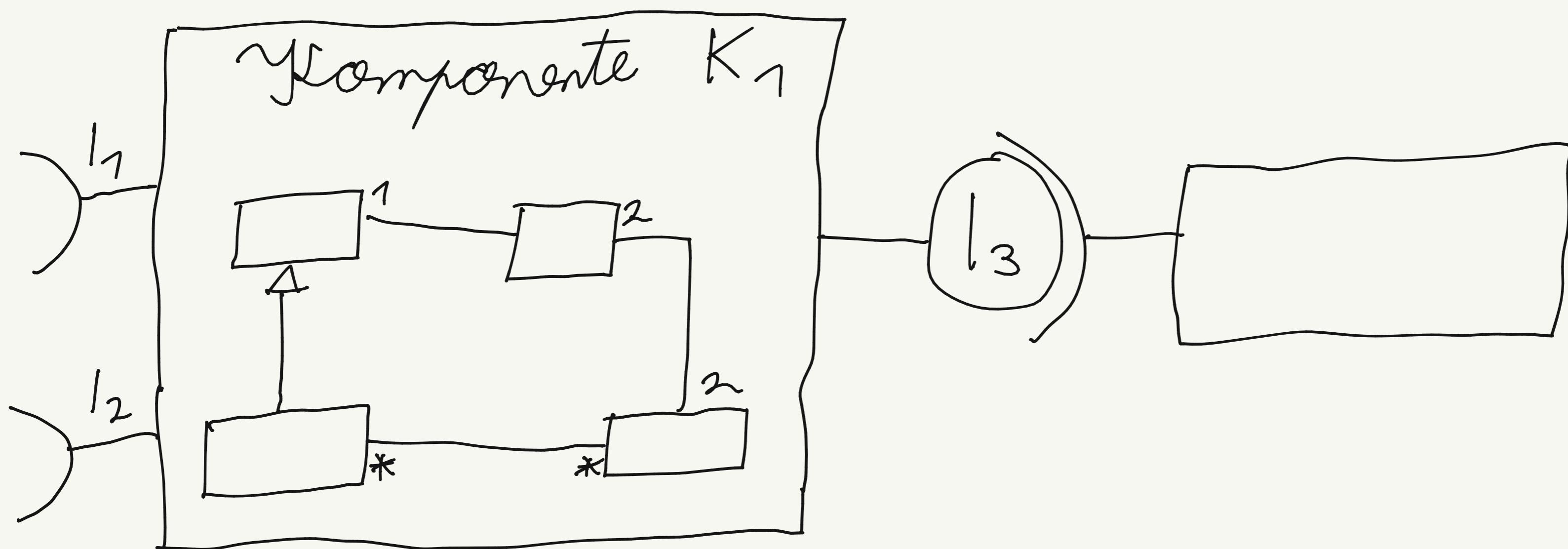
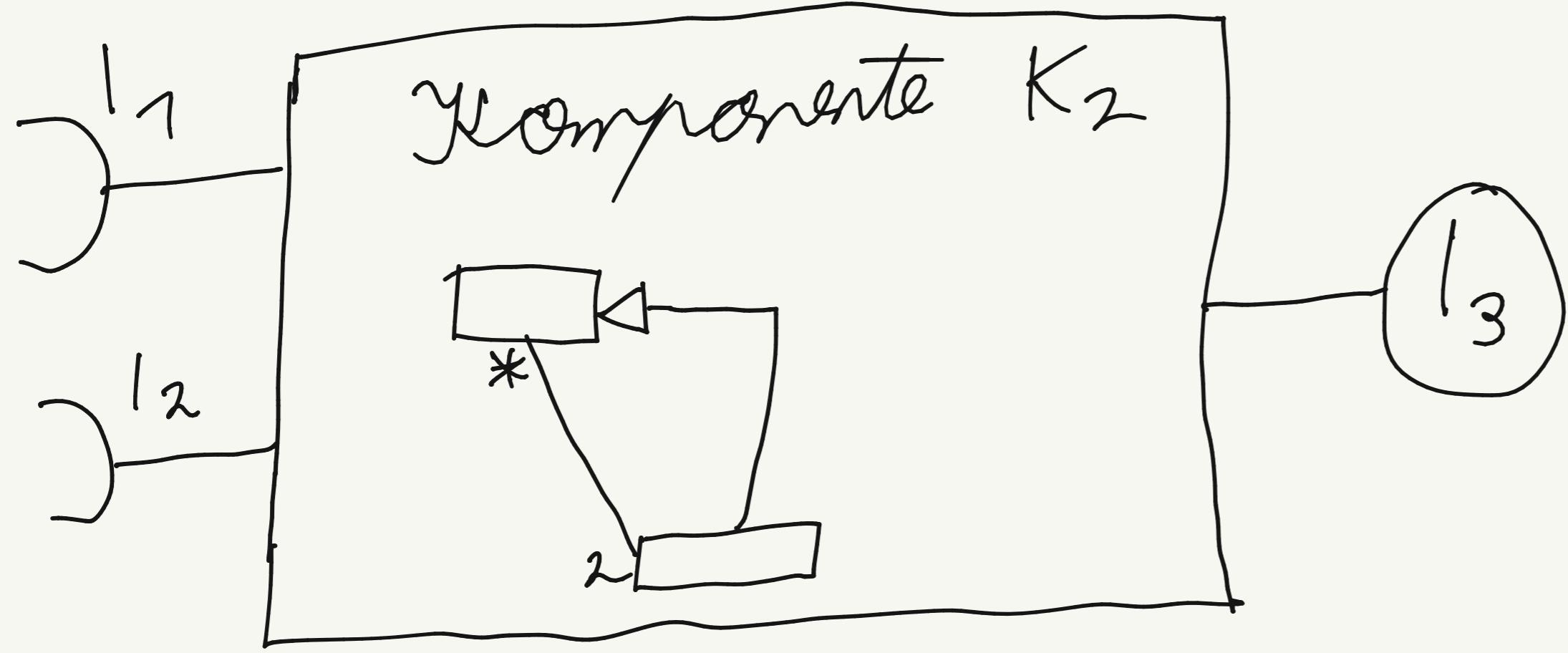


Importiert
Schnittstelle

Exportiert
Schnittstelle

- eine Komponente bietet Schnittstellen an, die sie im Zuge eines Vertrages garantiert
- Diese Garantie gilt unter der Bedingung, dass die von ihr benötigter Schnittstellen im Rahmen einer entsprechender Konfiguration bereitgestellt werden

Austauschbarkeit und Kapselung



- Über die Schnittstelle kapselt eine Komponente die Implementierung
- Solange alle Schnittstellenverträge erfüllt werden, können Komponenten ersetzt werden

Konfigurierbarkeit

Arten von Konfigurierbarkeit

- Konfiguration von Eigenschaften und Verhalten:
 - Allgemeine Einstellungen, Aussehen, Verhalten
- Komposition von Komponenten
 - Verbindungen zwischen Komponenten
 - zur Erstellung eines vollständigen Anwendungssystems
- Dynamische vs. statische Konfiguration

Voraussetzung von Konfigurierbarkeit: Effektives Konfigurationsmanagement

- Konfigurationsdateien
- Versionierung
- Fehlermanagement