

## Vorlesung 2

### Dokumentation und Kommunikation

Kommunikation von Softwarearchitektur – Warum so wichtig?

→ Verständnis und Zusammenarbeit

→ Einheitliche Vision

→ Risikominimierung

→ Bessere Entscheidungsfindung

→ Kunde- und Stakeholder-Engagement

→ Dokumentation und Nachvollziehbarkeit

→ Dokumentation und Nachvollziehbarkeit

Arten der Kommunikation von Softwarearchitekturen:

→ Dokumentation

→ Architektaudiogramme

→ Präsentationen

→ Prototypen und Proof-of-Concepts

→ Code und Kommentare

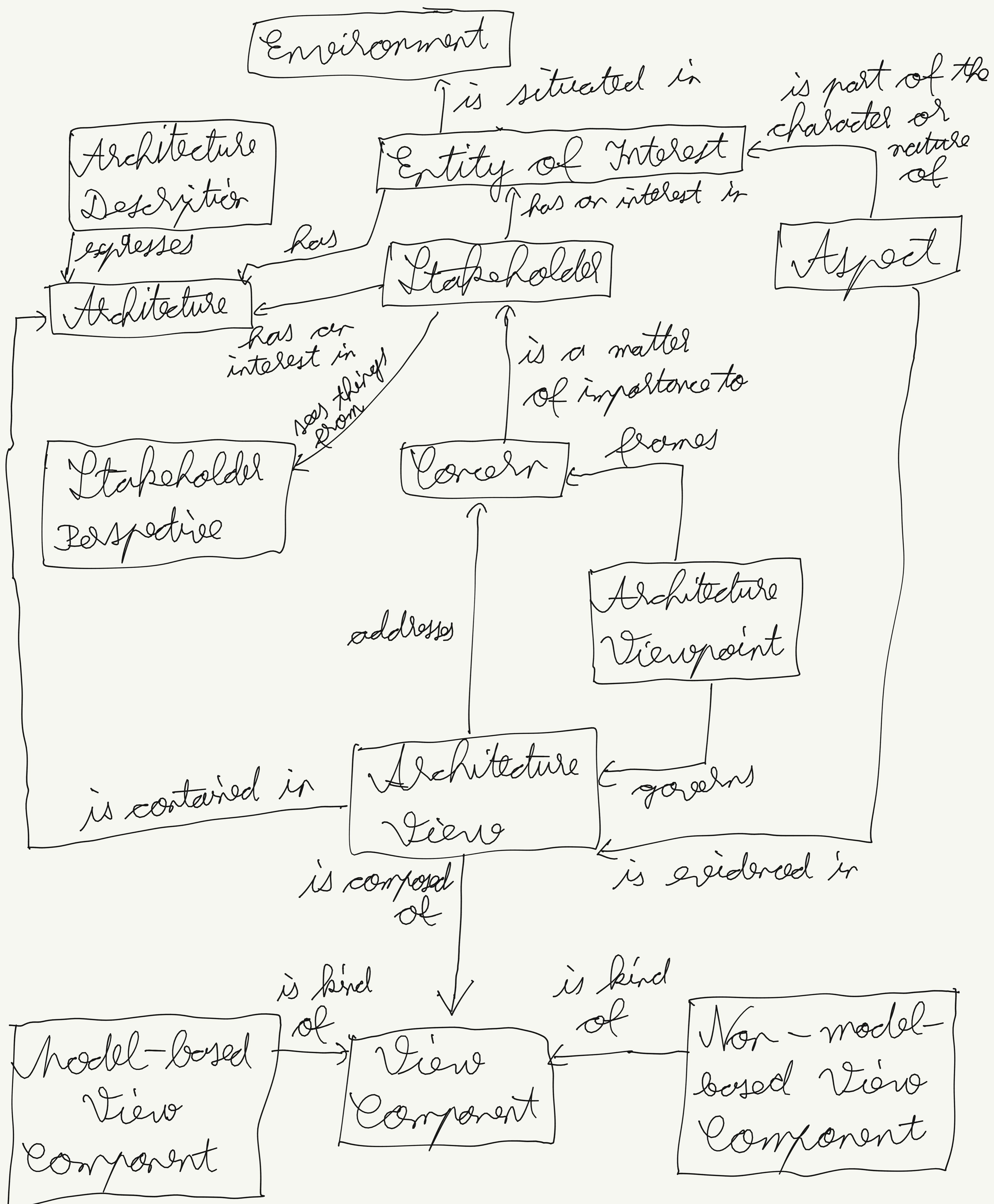
→ Workshops und Brainstorming-Sitzungen

# „Systems and Software Engineering – Architecture Description“

- Norm, die sich mit der Architekturbeschreibung von Systemen befasst
- Legt Grundsätze und Best Practices fest, um effektive Architekturbeschreibungen zu erstellen, zu kommunizieren und zu verwalten
- Definiert Schlüsselkonzepte wie Architektursichter, Architekturrahmenwerke, Architekturbeschreibungssprachen und Architektursbeschreibungsmethoden
- Bietet Richtlinien für die Auswahl und Anpassung von Architektursätzen und -methoden, um den spezifischer Anforderungen und Kontexten von Projekten gerecht zu werden

# Konzeptionelles Modell zur Beschreibung von Softwarearchitekturen nach ISO/IEC/IEEE

42010 - 2022



# Stakeholder

- role, position, individual or organization having a right, a share, a claim, or other interests in an architecture entity or its architecture that reflects their needs and expectations
- Kunde → geringe Kosten
    - hohe Qualität
    - Termintreue
  - Marketing → mehr Funktionalität als konkurrenz
    - kurze Entwicklungszeit
  - Administrator → einfacher Betrieb
    - einfache Wartung
  - Endbenutzer → Funktionalität
    - Performance
    - Stabilität
    - Sicherheit
  - Management → hohe Effektivität, Auslastung und Wachstum
  - Organisation → Einhaltung von Betriebsvereinbarungen, Richtlinien
  - Methodiker → Einhaltung von Standards für Dokumentation, Vorgehen
  - Entwickler → Implementierungsrichtlinien, Design
- ⇒ Softwarearchitekt = ???

## Views und Viewpoints

Concern nach ISO/IEC/IEEE 42010-2022

- Ein concern ist ein Interesse, das sich auf das System, seine Entwicklung oder seinen Betrieb bezieht:
  - Funktionale Anforderungen
  - Nicht-funktionale-Anforderungen
- Rollen in der Architekturbeschreibung:
  - Dienen als Grundlage für Erstellung und Strukturierung der Architektureschichten
  - Lenken der Fokus der Architektuarbeit auf relevante Aspekte
  - Unterstützen Kommunikation mit Stakeholdern
  - Entscheidend für die Validierung und Bewertung der Architektur

Aspects nach ISO/IEC/IEEE 42010:2022

- Aspekte sind bestimmte Perspektiven oder Blickwinkel, unter denen das System analysiert und beschrieben wird
- Helfen dabei, verschiedene concerns der Stakeholder systematisch zu adressieren
- Beispiele: Sicherheits-Aspekt, Leistungs-Aspekt, Wartbarkeitsaspekt, Benutzerfreundlichkeits-Aspekt

## Viewpoint nach ISO/IEC/IEEE 42010-2022

- Ein Viewpoint ist ein definiertes Satz von Konventionen und Regeln zur Darstellung und Beschreibung bestimmter Concerns eines Systems. Verschiedene Viewpoints sind dafür konzipiert, die Interessen und Bedenken unterschiedlicher Stakeholder zu adressieren
- Ein Viewpoint definiert unter anderem:
  - Name des Viewpoints
  - Addressierte Stakeholder
  - Addressierte Concerns (Anliegen der Stakeholder)
  - Notationen / Diagramme
  - Methoden und Techniken
  - Quellangabe
  - Konsistenz- und Vollständigkeits-Checks
- Beispiele:
  - Logische Perspektive
  - Prozessperspektive
  - Physische Perspektive

## Architectural View nach ISO/IEC/IEEE 42010-2022

- Eine Architecture View ist eine Darstellung eines Systems aus der Perspektive eines bestimmten Architecture Viewpoints
- Konkrete Repräsentation eines Aspekts, die dazu dient bestimmte Concerns der Stakeholder zu adressieren
- UML-Diagramme können zum Beispiel Architectural Views sein z.B.: Klassendiagramme, Komponentendiagramme, Sequenzdiagramme

## Views und Viewpoints

• Viewpoint = Standpunkt, Blickwinkel

• View = Sicht

→ Eine „Entity of Interest“ kann über verschiedene Views reduziert werden, da nicht jeder Stakeholder eine Interesse daran hat.

→ Annahme: Es ist nicht möglich, die Merkmale und Qualitätsmerkmale eines komplexen Systems in einem einzigen verständlichen Modell zu erfassen, das von allen Beteiligten verstanden wird und für sie von Nutzen ist.

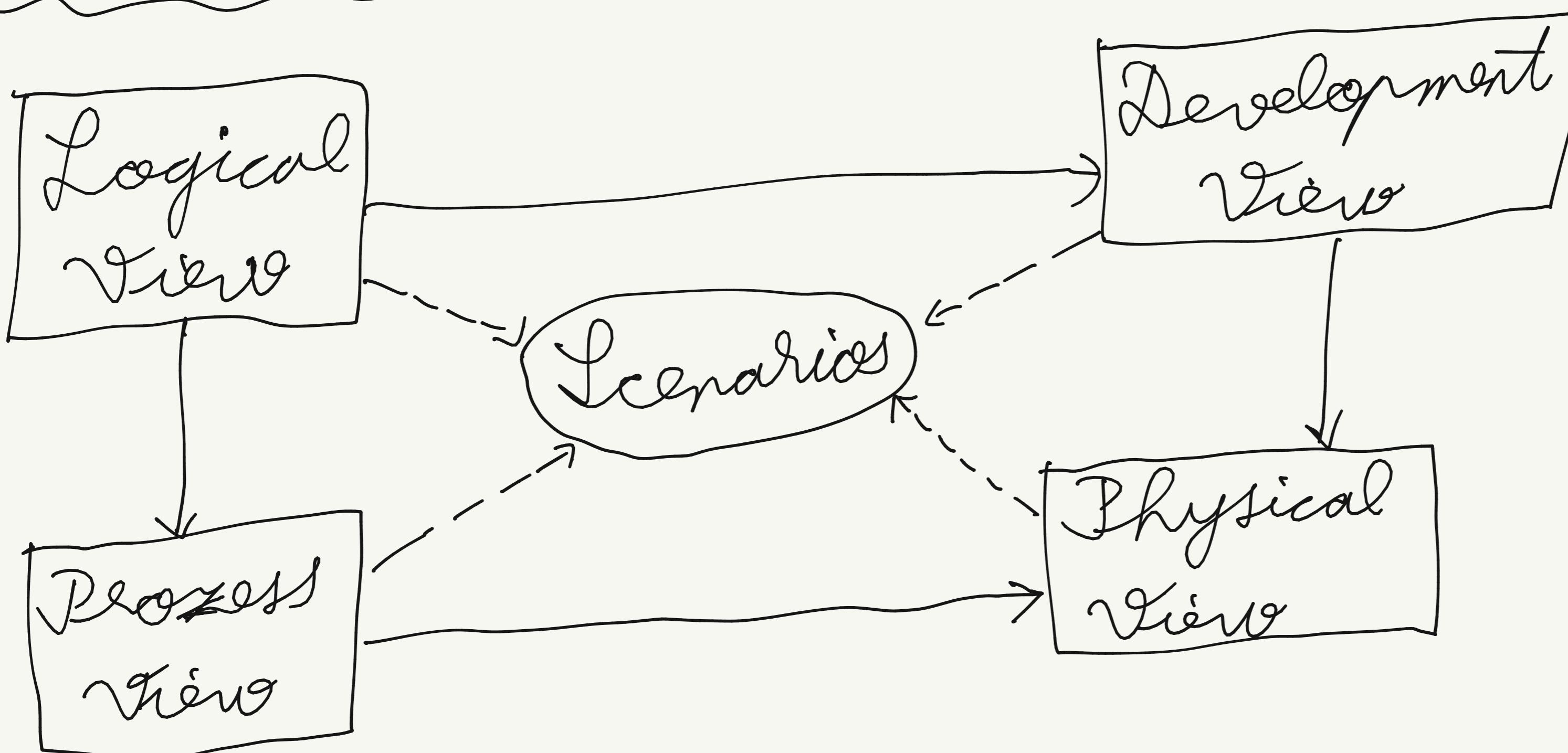
Beispiel:

Stakeholder	Viewpoint	Views	View Components
Administrator, Developer, IT Infrastructure	Deployment	UML Deployment Diagram	Personen, Netzwerknoeuer, Datenbanken
Developer, Tester	Statische Struktur	UML Kompo- nenter Diagramm	Komponenten, Schnittsteller
Developer	Laufzeitsicht	UML Zeitachsen-Diagramm UML Aktivitäts- diagramm	Prozesse, Aktivitäten

## Lichtermodelle

- Strukturierte Darstellungen eines Systems, die verschiedene Aspekte der Architektur aus unterschiedlicher Perspektive oder Lichter zeigen
- Umfassen eine Sammlung von verschiedenen Architektur-Lichtern, die zusammen eine umfassende und kohärente Beschreibung eines Systems liefern
- Die Norm ISO/IEC/IEEE 42010 bietet eine Grundlage zur Erstellung solcher Lichtermodelle

## 4.1 Lichtermodell nach Philippe Kruchter



Logical View – Fokus auf die Strukturierung des Systems in Module und deren Interaktion

Development View – Struktur des Quellcodes sowie Details zum Build-Prozess und anderen Aspekten des Entwicklungsprozesses

Scenarios - Zeigt wie die verschiedenen Lichten zusammenarbeiten anhand verschiedener Szenarien

Physical View - Physische Verteilung des Systems auf Hardware sowie Aspekte wie Netzwerktopologie

Process View - Focus darauf wie Module zur Laufzeit interagieren und wie Daten durch das System fließen

### Siemens' Four View Model

- Conceptual View / Konzeptionelle Schicht
  - Hauptkomponenten und deren Interaktionen
  - Wichtige Architekturmuster, die angewendet werden
  - Views: Zum Beispiel einfache Komponentendiagramme
- Module View / Modul- oder logische Schicht
  - Detaillierte Beschreibung der Module und ihrer Schnittstellen
  - Views: Zum Beispiel Klassen-, Paket- oder Komponentendiagramme
- Execution View / Laufzeitsicht
  - Kommunikationswege, Verhalten in spezifischer Szenarien
  - Views: Zum Beispiel Sequenzdiagramme, Aktivitätsdiagramme oder Zustandsdiagramme
- Physical View / Physische Schicht
  - Hardwarekomponenten und deren Konfiguration
  - Zuordnung der Softwarekomponenten zu Hardwarekomponenten
  - Views: Zum Beispiel Deployment-Diagramme

# Lichtermodell nach der 4<sup>2</sup>

- Kontextabgrenzung
  - Einbettung des Systems in seine Umgebung
  - System als Blackbox
- Bausteinsicht
  - Zerlegung in Subsysteme, Komponenten, Frameworks, ...
- Laufzeitsicht
  - Interaktion von Laufzeitinstanzen
  - Generelle Abläufe innerhalb der Software
- Verteilungssicht
  - Technische Umgebung und Infrastruktur
  - Hardwarekomponenten und deren Interaktion / Netzwerktopologie
  - Deployment - Artefakte

# Struktur von arc42:

## 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-Haut
- 8.4 User Experience

...

## 9. Entwurfsentscheidungen

- 9.1 Entwurfsentscheidung 1
- 9.2 Entwurfsentscheidung 2

...

## 10. Qualitätsanforderungen

- 10.1 Qualitätsbaum

- 10.2 Qualitätszonen

## 11. Risiken und technischer Schulden

## 12. Glossar

## Ebenen in arc42

→ Ebenen werden in arc42 verwendet, um Architekturdokumentation in unterschiedliche Granularitätsstufen zu unterteilen

Typische Ebenen sind:

- Makroarchitektur
  - Gesamtstruktur des Systems
  - Hauptkomponenten und ihre Funktionen
  - Große Interaktionen und Schnittstellen
- Mittlere Ebene
  - Aufteilung des Systems in Komponenten
  - Interaktion zwischen Komponenten
  - Schnittstellen und Kommunikationswege
- Microarchitectur
  - Detaillierte Beschreibung der Bausteine oder Komponenten
  - Interne Struktur und Abhängigkeiten
  - Schnittstellen und Kommunikationswege

## UML - Unified Modelling Language

Jede Sicht braucht geeignete Beschreibungstechniken  
→ mit einer Diagrammart lassen sich nicht alle Aspekte aller Sichten geeignet abbilden. Für jeden Viewpoint werden unterschiedliche Views benötigt.

→ mit der UML steht eine grafische Modellierungssprache zur Verfügung, welche zur Dokumentation von Softwaresystemen verwendet werden kann

Diagramme im UML lassen sich in zwei Hauptgruppen unterteilen:

• Strukturdiagramme:

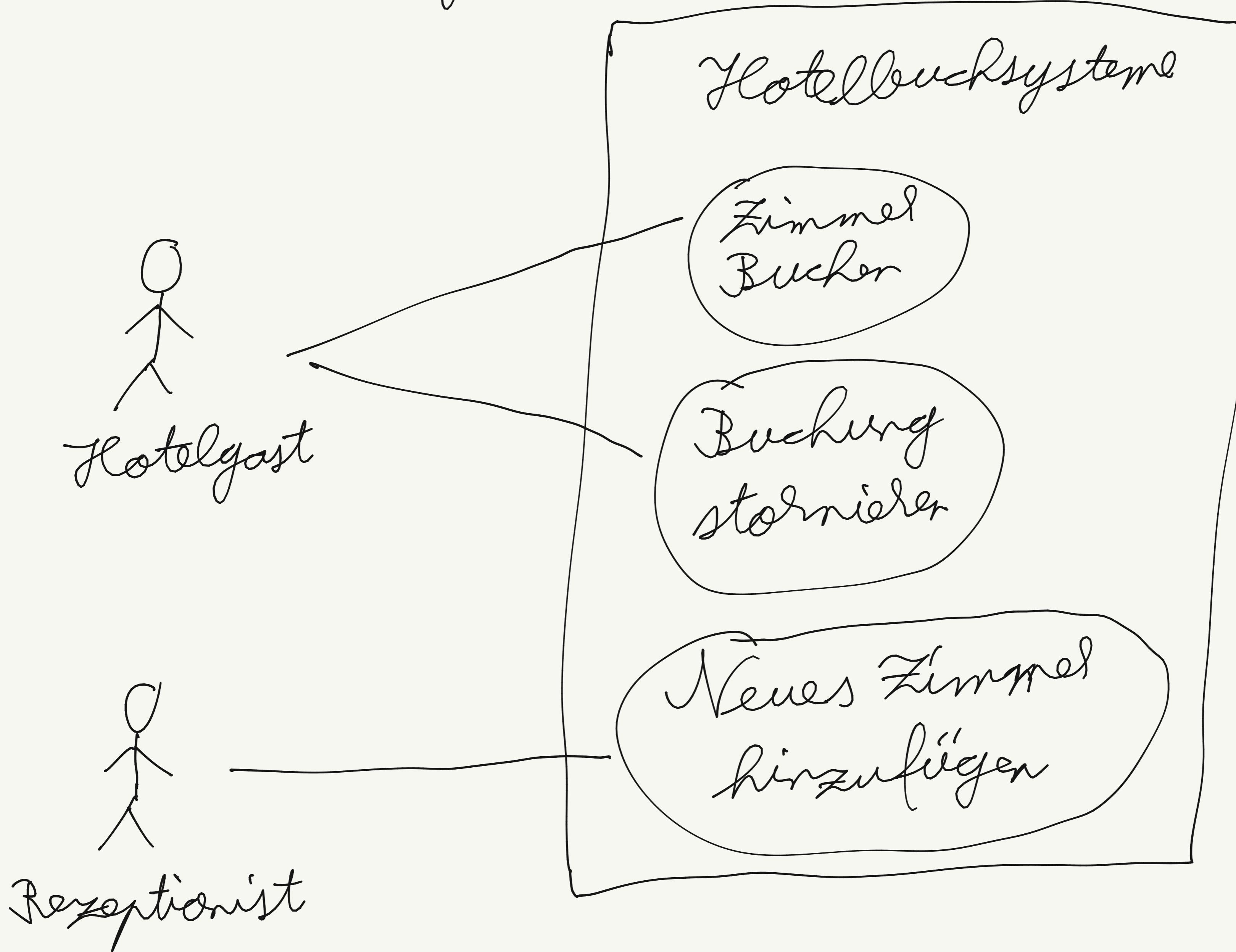
Klassendiagramm, Komponentendiagramm, Verteilungsdiagramm, Kompositionstrukturdiagramm, Paketdiagramm, Objektdiagramm, Profildiagramm

• Verhaltensdiagramme:

Aktivitätsdiagramm, Anwendungsfalldiagramm, Interaktionsübersichtsdiagramm, Kommunikationsdiagramm, Sequenzdiagramm, Zeitverlaufsdiagramm, Zustandsdiagramm

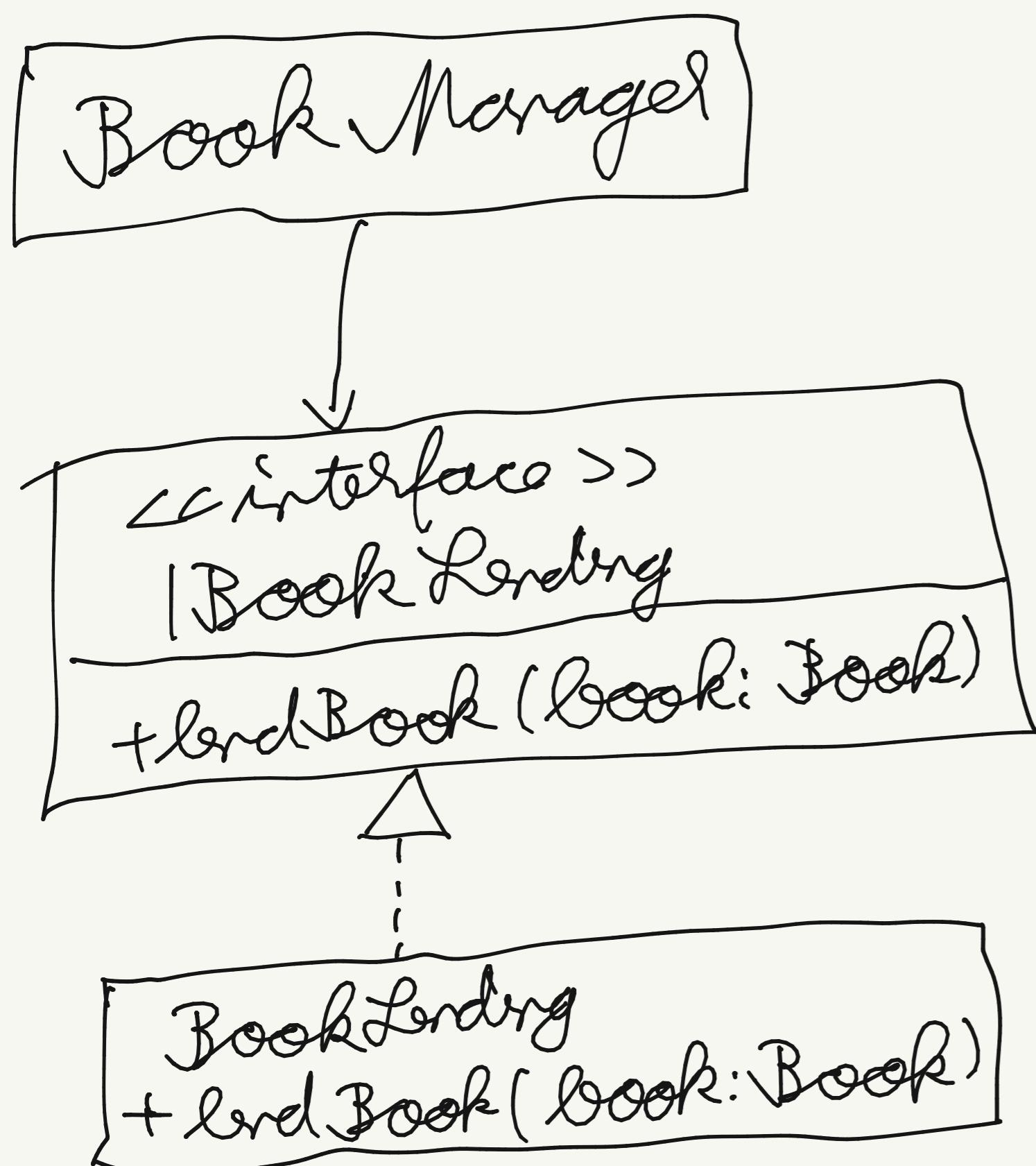
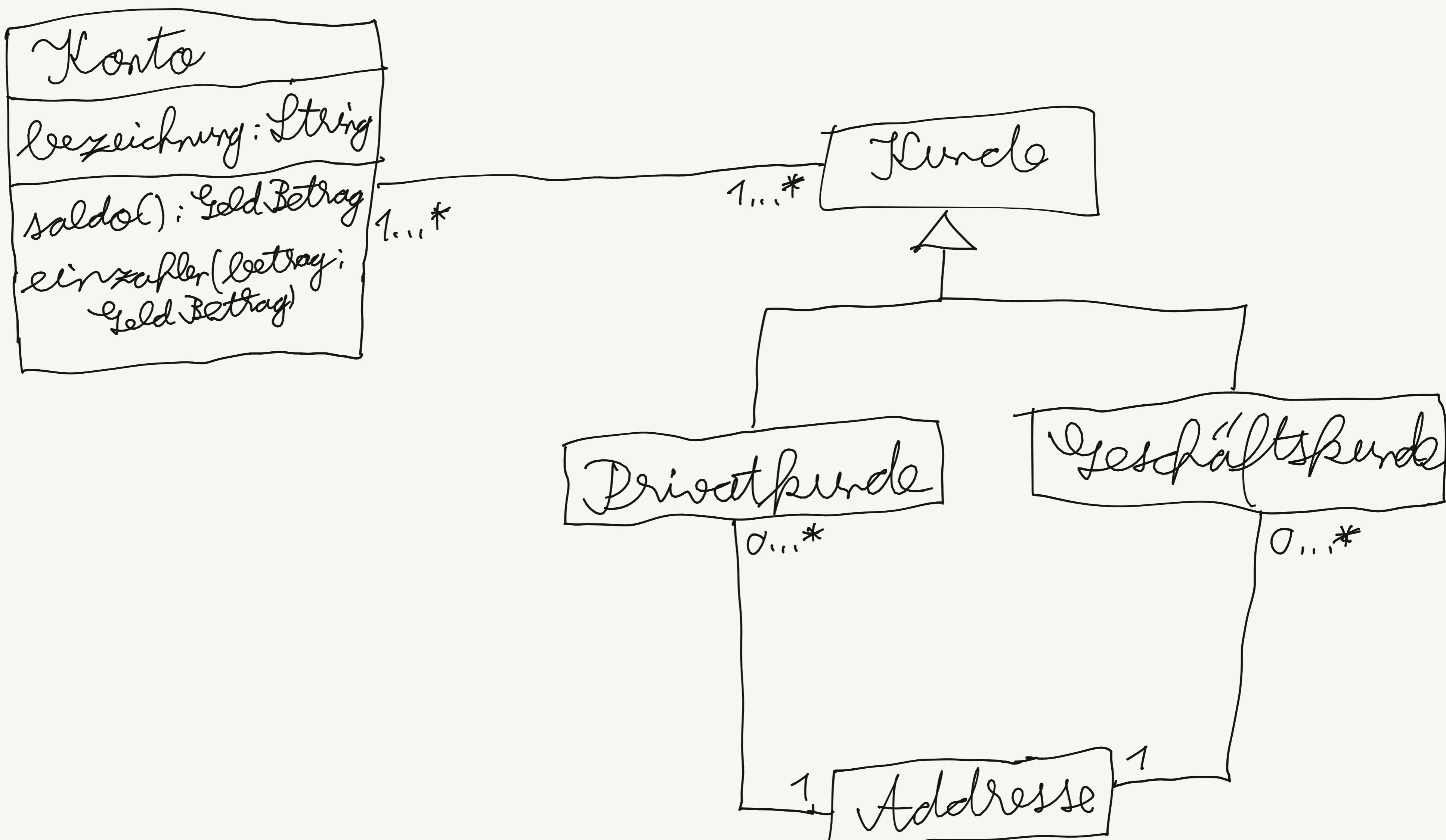
## UML Anwendungsfalldiagramm

- Modellierung von funktionalen Anforderungen eines Systems aus Anwender-Perspektive
- Bestandteile:
  - Systemgrenze, inklusive Name des Systems
  - Akteure (Piktogramme) / externe Systeme
  - Anwendungsfälle (Ellipsen)
  - Beziehungen

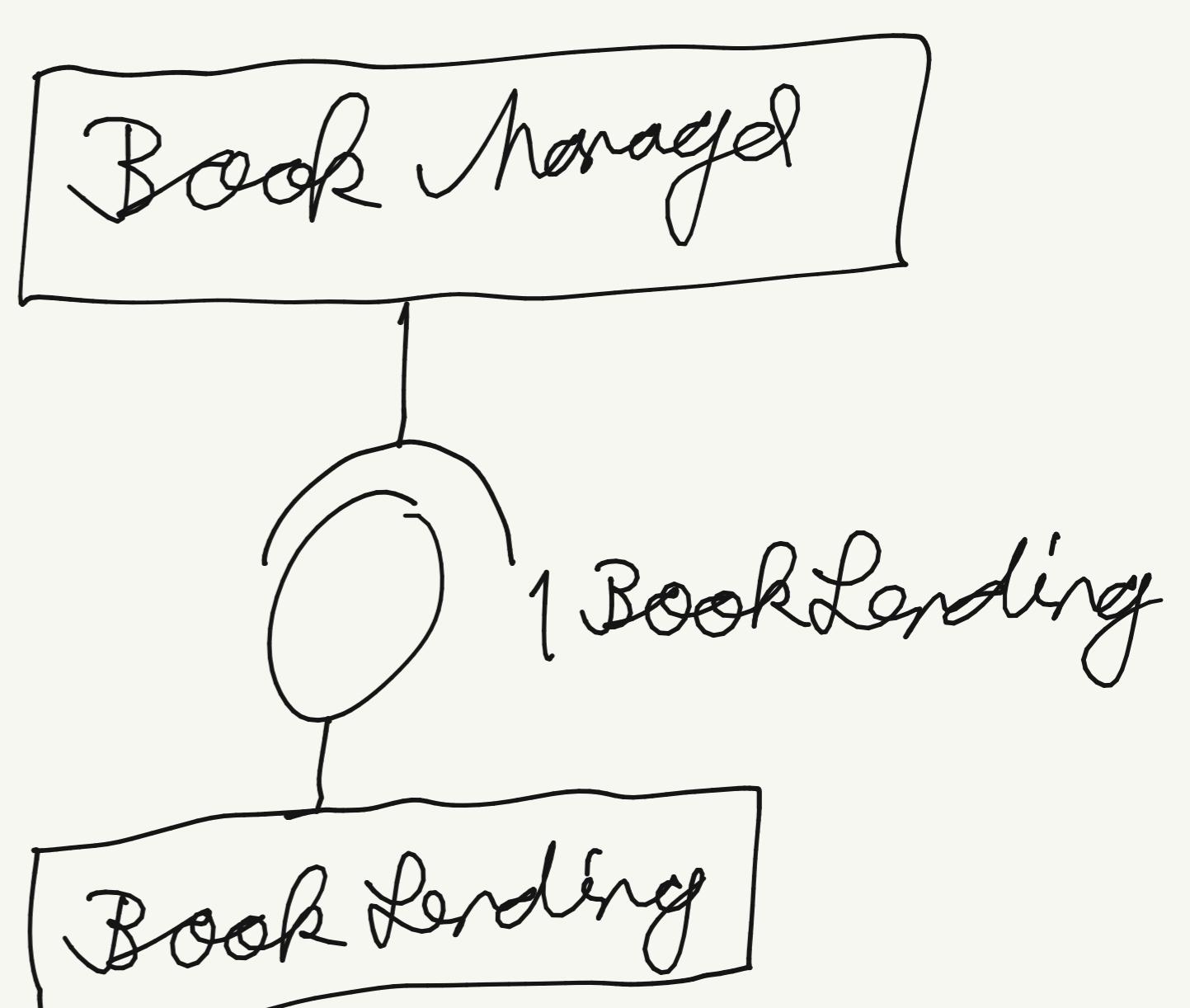


## UML Klassendiagramm

- Grafische Darstellung von Klassen, Schnittstellen, Attributen und Methoden sowie deren Beziehungen zueinander
- Mit Klassendiagrammen lassen sich unter anderem auch Datenbank-Strukturen und Domänenmodelle darstellen
- Wird häufig verwendet in:
  - Bausteinsicht / Module View
  - Conceptual View

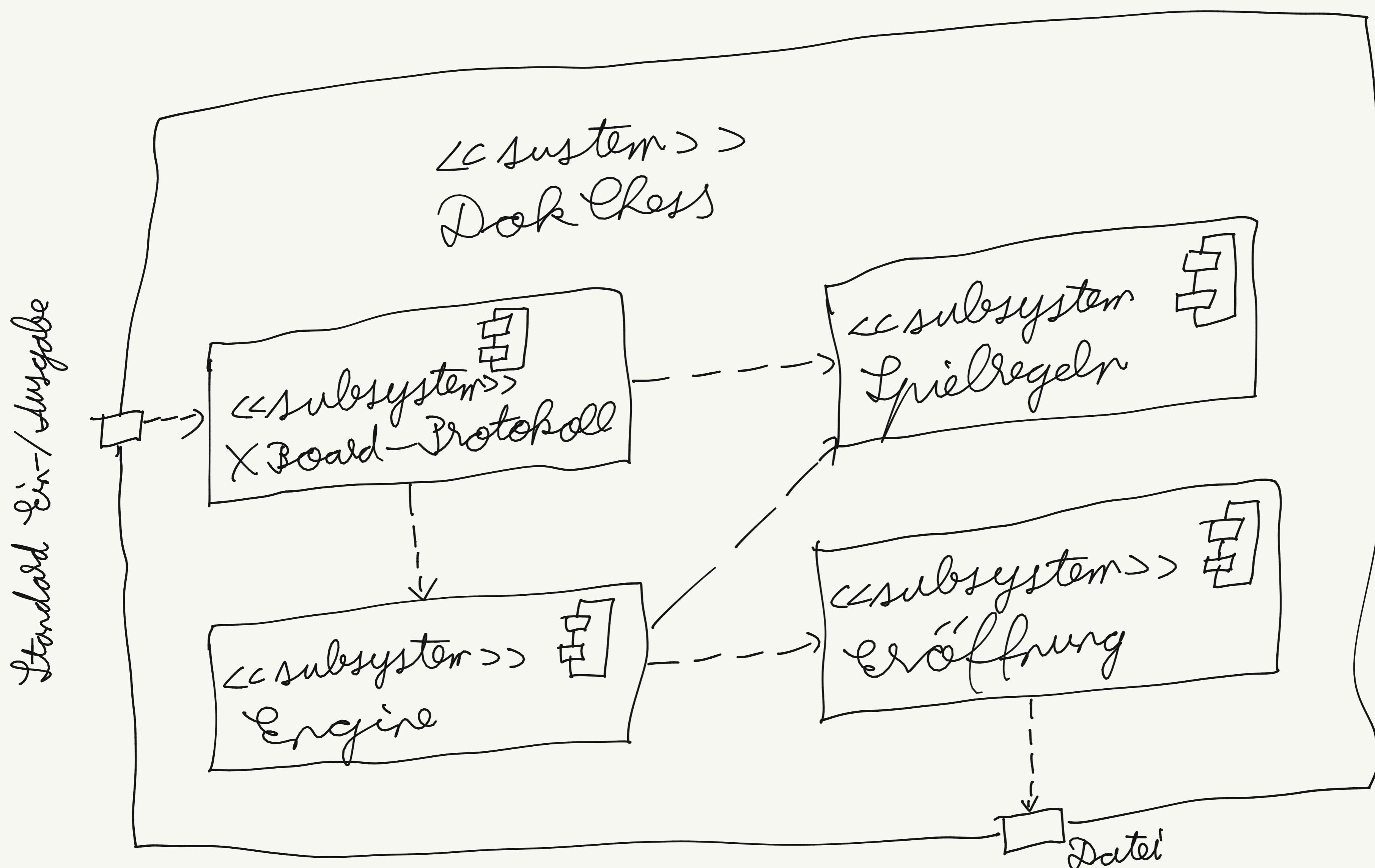
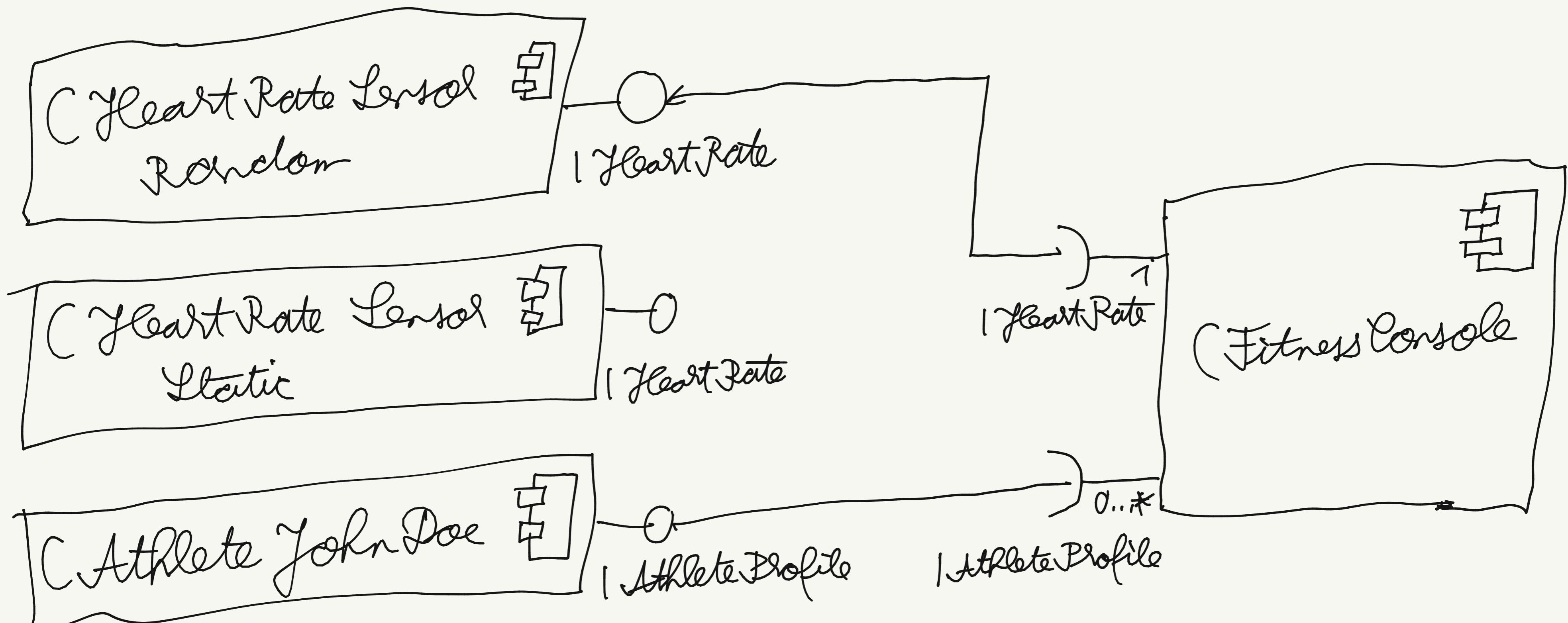


$\Leftrightarrow$



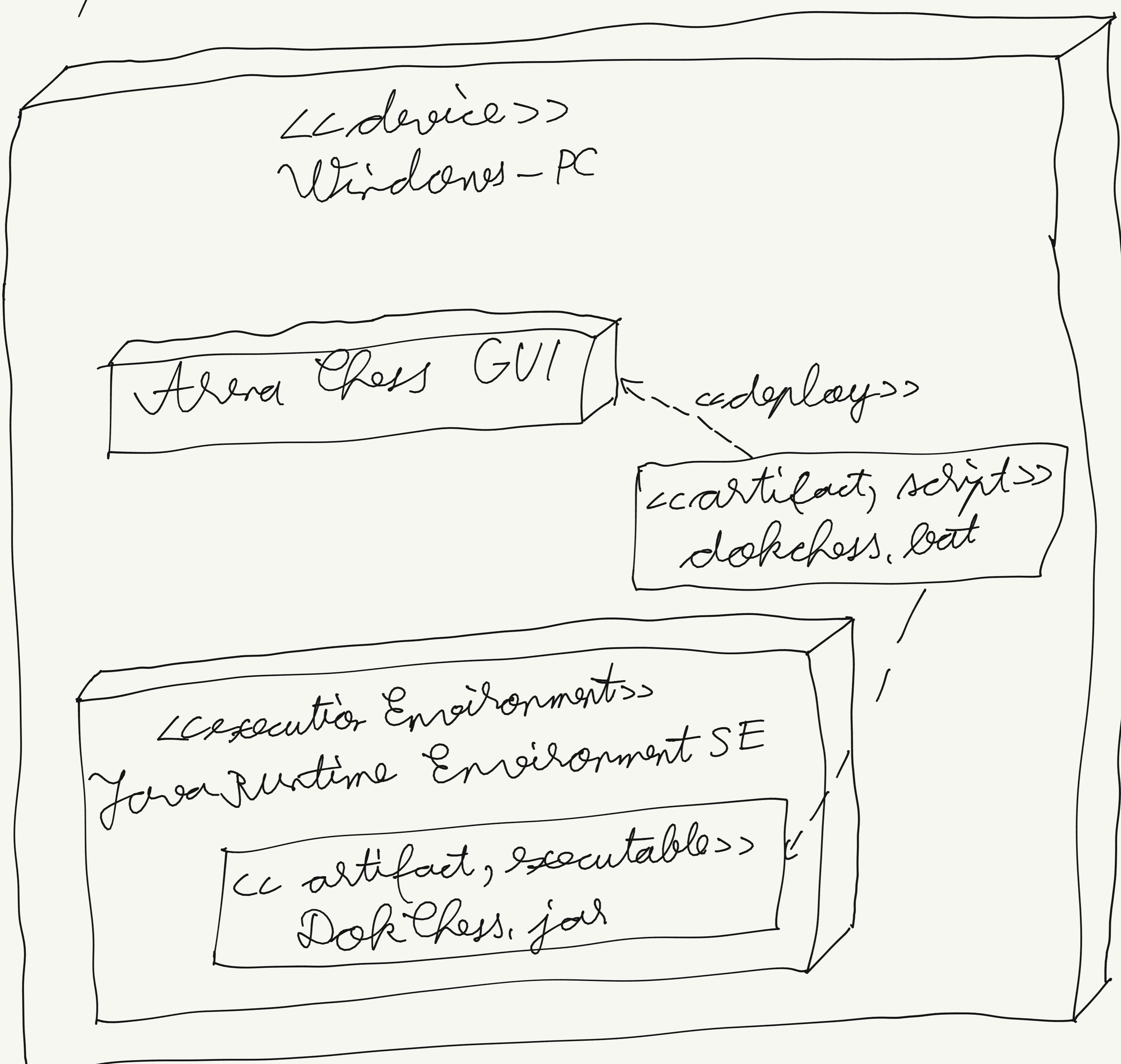
## UML Komponentendiagramm

- grafische Modellierung von Komponenten und Subkomponenten sowie deren Verbindung
- Wird häufig verwendet in:
  - Bausteinsicht
  - Module View
  - Kontextabgrenzung



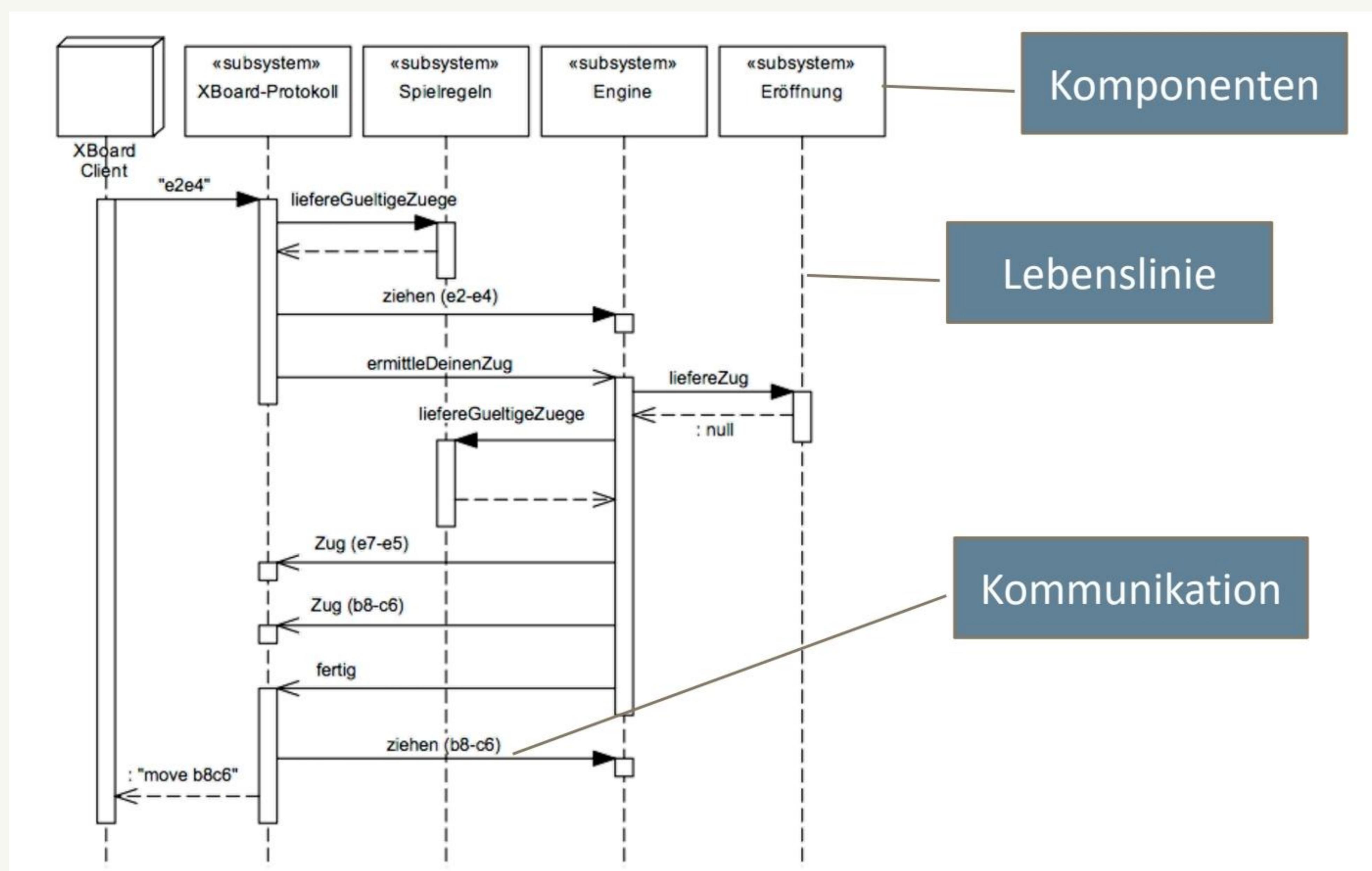
## UML Verteilungsdiagramm

- Grafische Modellierung der Verteilung von Komponenten auf Rechnerknoten



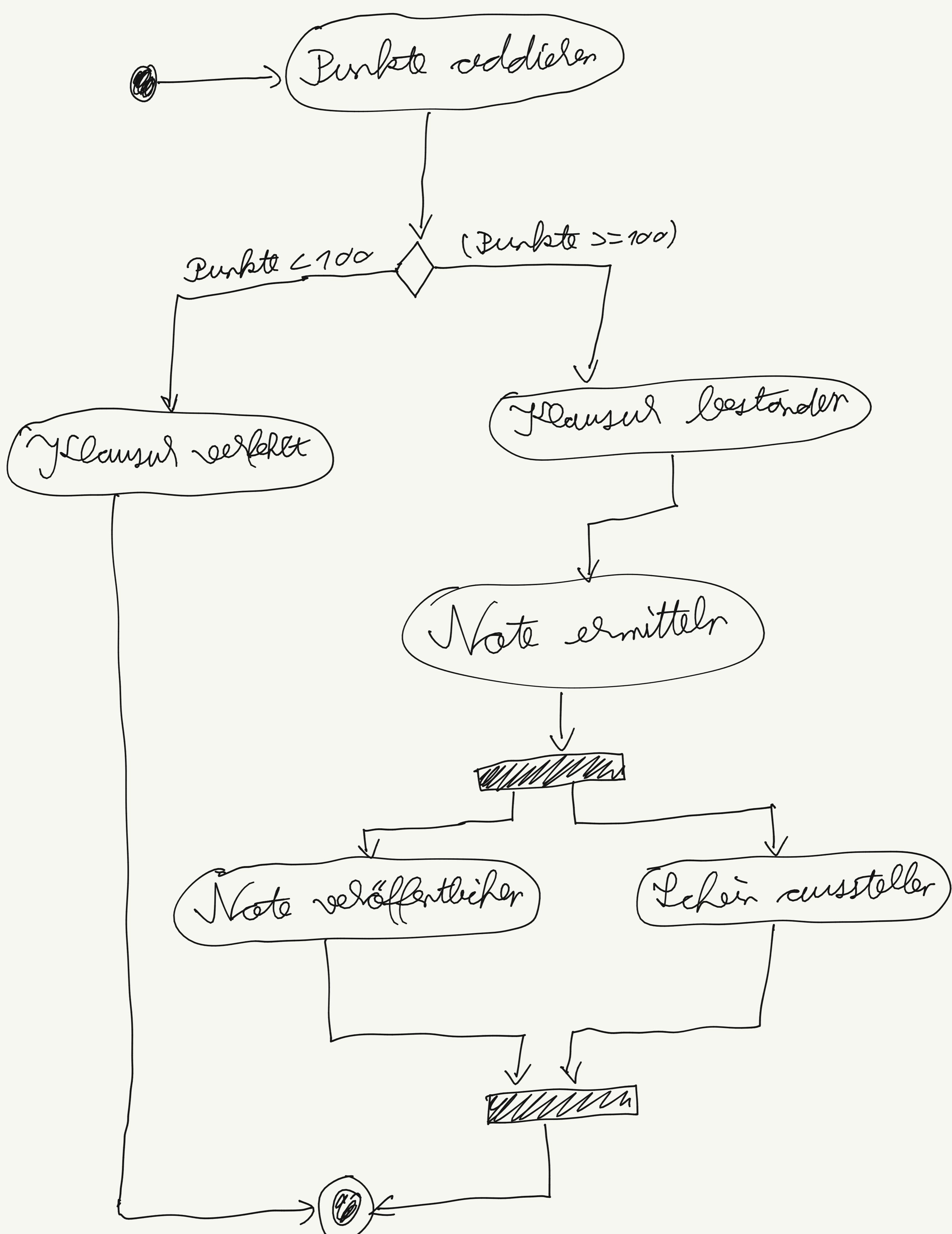
# UML Sequenzdiagramm

- Grafische Modellierung zur Beschreibung des Austauschs von Nachrichten zwischen Objekten
- Stellt in der Regel einen konkreten Systemablauf dar (im Gegensatz zu Aktivitätsdiagrammen)



## UML Aktivitätsdiagramm

- stellt die Vernetzung von Aktionen und deren Verbindungen mit Kontroll- und Datenflüsse dar



## Konsistenz zwischen den Diagrammen

- auf konsistente Benennung der Elemente zwischen Diagrammen achten
- auf Konsistenz der Schnittsteller achter
- auf konsistente Benennung der Elemente zwischen Diagrammen achter

# Dokumentation von Architekturentscheidungen

- Mit UML werden insbesondere Ist- und Sollzustand einer Architektur dokumentiert
- Hauptarbeit eines Architekten ist es, Architektur-relevanten Entscheidungen zu treffen
- Diese Entscheidungen können:
  - auf dem Ist-Zustand beruhen
  - auf einer Auswahl von möglicher Flößen und Soll-Zuständen beruhen
  - auf zahlreichen weiteren Faktoren beruhen

Diese Entscheidungen selbst müssen dokumentiert werden

- Dokumentation von Architekturentscheidungen
- Architekturentscheidungen zu dokumentieren, hilft dabei:
    - Entscheidungen team-übergreifend sichtbar zu machen
    - Entscheidungen in Team und im Unternehmen darzusetzen
    - Die Qualität von Architekturentscheidungen zu messen (Welche Entscheidungen hatten welche Auswirkungen?)
    - Die Entscheidung einer künftiger Ist-Architektur nachzu vollziehen
    - dem Team Grundlagen für detaillierte Entscheidungen zu geben
  - Feedback-Loops helfen, indem die getroffene Entscheidungen regelmäßig hinterfragt und bewertet werden zum Beispiel zu Beginn oder Ende eines Sprints

## Architectural Decision Records

- Bewährte Möglichkeit, Architekturentscheidungen zu dokumentieren sind Architectural Decision Records (ADR)

- Beispiel eines solchen Records:

Titel Prägnanter Titel der Entscheidung	ID	Status (z.B. vorgeschlagen, entschieden, umgesetzt)	Beschreibung & Kontext	Alternativen	Begründung
Native iOS Mobile-App	ADR-1	vorgeschlagen	Das UI wird nativ auf iOS implementiert.	Native iOS oder Web-App	Bessere UX, bessere Plattform-Integration.  Es wird in Kauf genommen, dass doppelter Aufwand für Android und iOS entsteht.
...					

Zusätzlich können unter anderem Informationen wie Namen des Entscheider oder das Datum der Entscheidung hintergelegt werden

# Allgemeine Aspekte zur Dokumentation

Weitere übliche Dokumententypen:

- Zentrale Architekturbeschreibung (arc42)
- Architekturüberblick
- Dokumentübersicht
- Übersichtspräsentator
- Architekturstapete
- Handbuch zur Dokumentation
- Technische Informationen
- Dokumentation von externer Lieferanten

Praxisregeln zur Dokumentation:

- Schreiben aus Sicht des Lesers
- Unnötige Wiederholungen vermeiden
- Mehrdeutigkeit vermeiden
- Standardisierte Organisationsstrukturen bzw. Templates und Satz-Schablonen
- Begründen Sie wesentliche Entscheidungen schriftlich
- Überprüfung der Gebrauchstauglichkeit
- Übersichtliche Diagramme
- Regelmäßige Aktualisierungen
- Eine Architekturbeschreibung sollte so einfach wie möglich sein, aber umfangreich wie notwendig
- „You know you've achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away“.