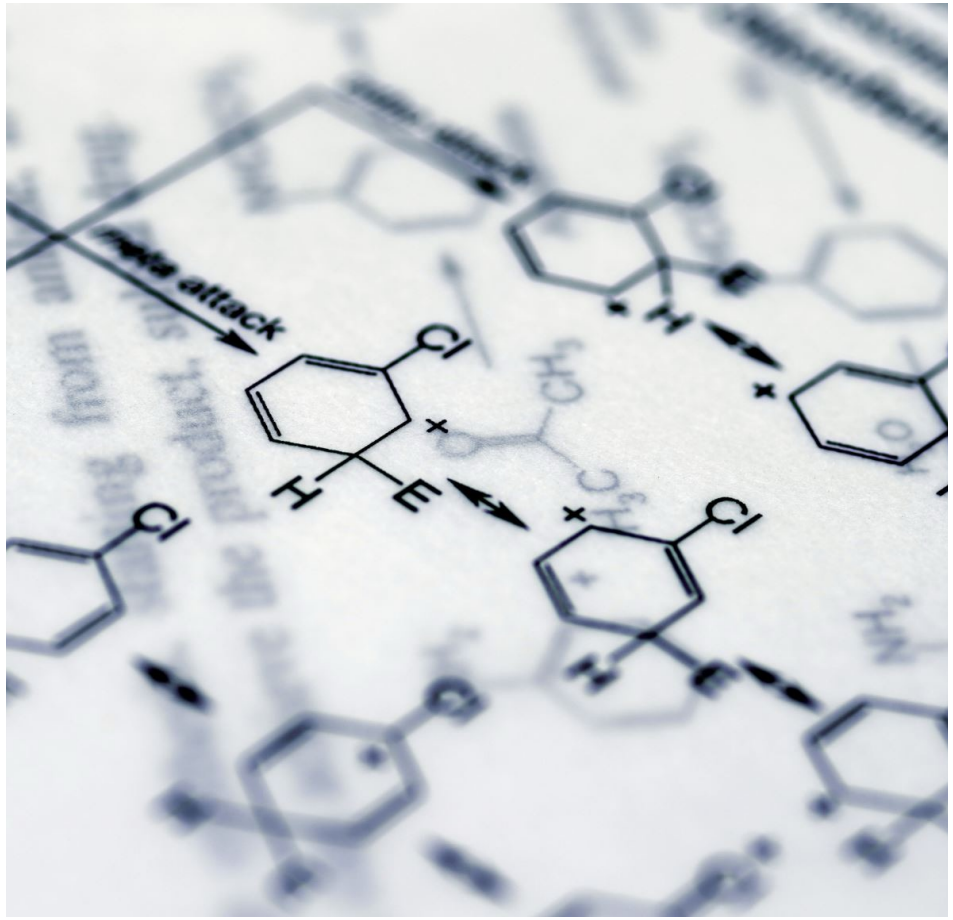
Themenbereiche der Vorlesung



Themenbereiche

A – Grundlegende Konzepte

B – Dokumentation und Kommunikation

C – Entwurf von Softwarearchitekturen

D – Architekturmuster

E – Qualität von Softwarearchitekturen

Dokumentation & Kommunikation

Themenbereich B

Inhalte

- Stakeholder
- Views & Viewpoints
- Sichtenmodelle
- UML – Unified Modeling Language
- Dokumentation von Architekturentscheidungen
- Allgemeine Aspekte zur Dokumentation

Kommunikation von Softwarearchitekturen Warum so wichtig?

- Verständnis und Zusammenarbeit
- Einheitliche Vision
- Risikominimierung
- Bessere Entscheidungsfindung
- Kunden- und Stakeholder-Engagement
- Dokumentation und Nachvollziehbarkeit

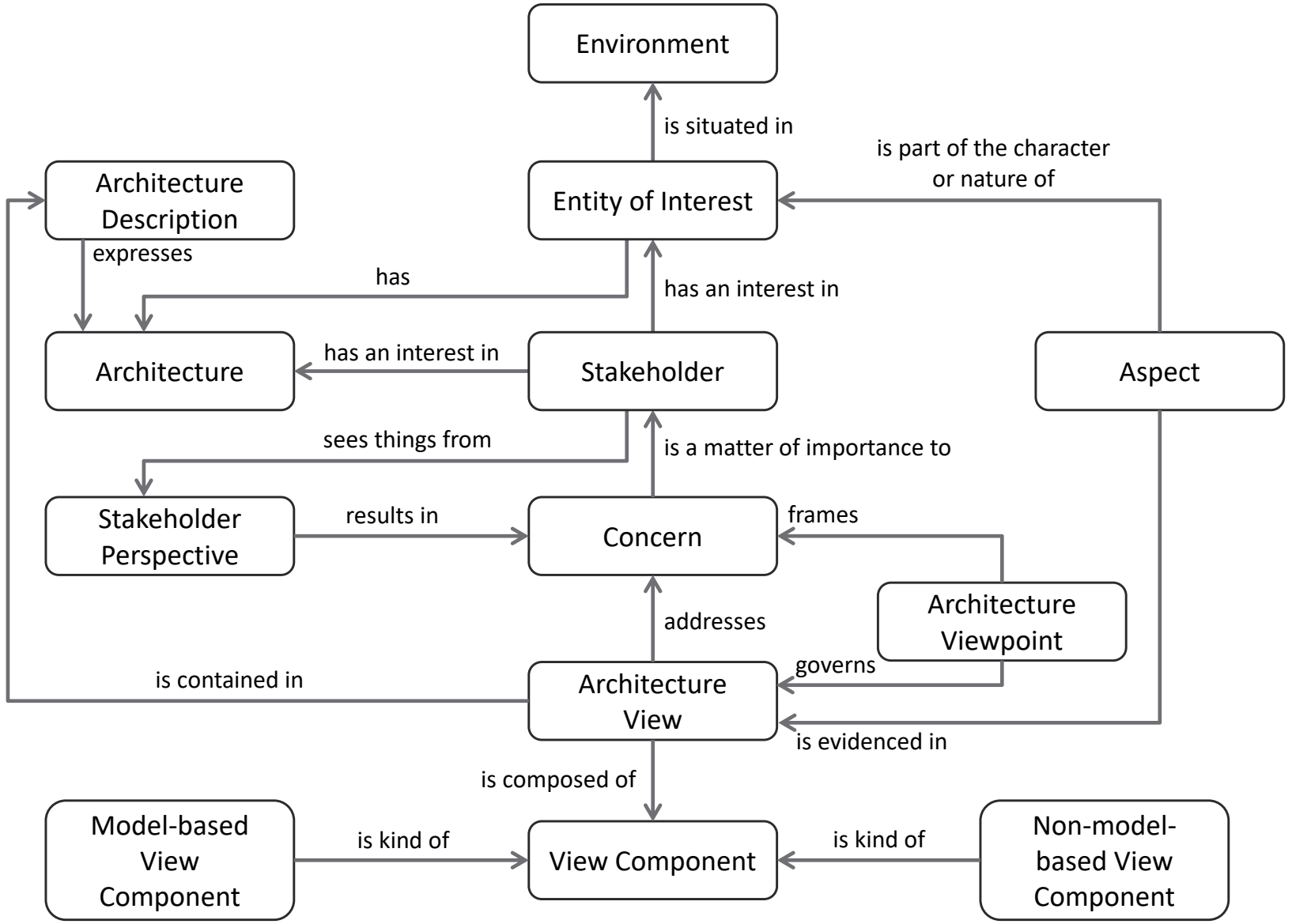
Arten der Kommunikation von Softwarearchitekturen

- Dokumentation
- Architekturdiagramme
- Präsentationen
- Prototypen und Proof-of-Concepts
- Code und Kommentare
- Workshops und Brainstorming-Sitzungen

ISO/IEC/IEEE 42010-2022

- “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 Architekturansichten, Architekturrahmenwerke, Architekturbeschreibungssprachen und Architekturbeschreibungsmethoden.
- Bietet Richtlinien für die Auswahl und Anpassung von Architekturansätzen und -methoden, um den spezifischen Anforderungen und Kontexten von Projekten gerecht zu werden.

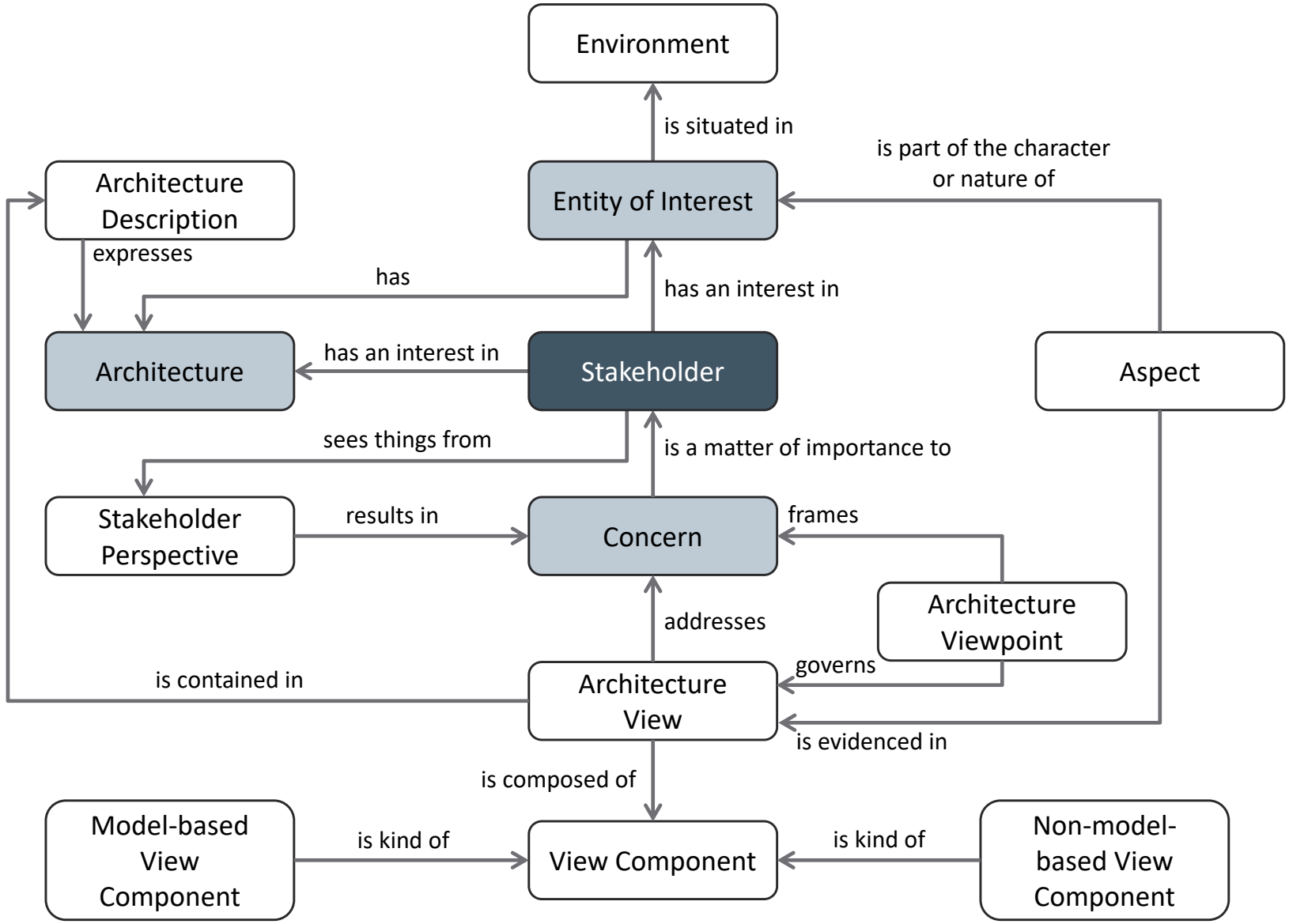
Konzeptionelles Modell zur Beschreibung von Softwarearchitekturen nach ISO/IEC/IEEE 42010-2022



ISO/IEC/IEEE 42010:2022 (second edition)
http://www.iso-architecture.org/42010/cm/

Stakeholder

Stakeholder nach ISO/IEC/IEEE 42010-2022

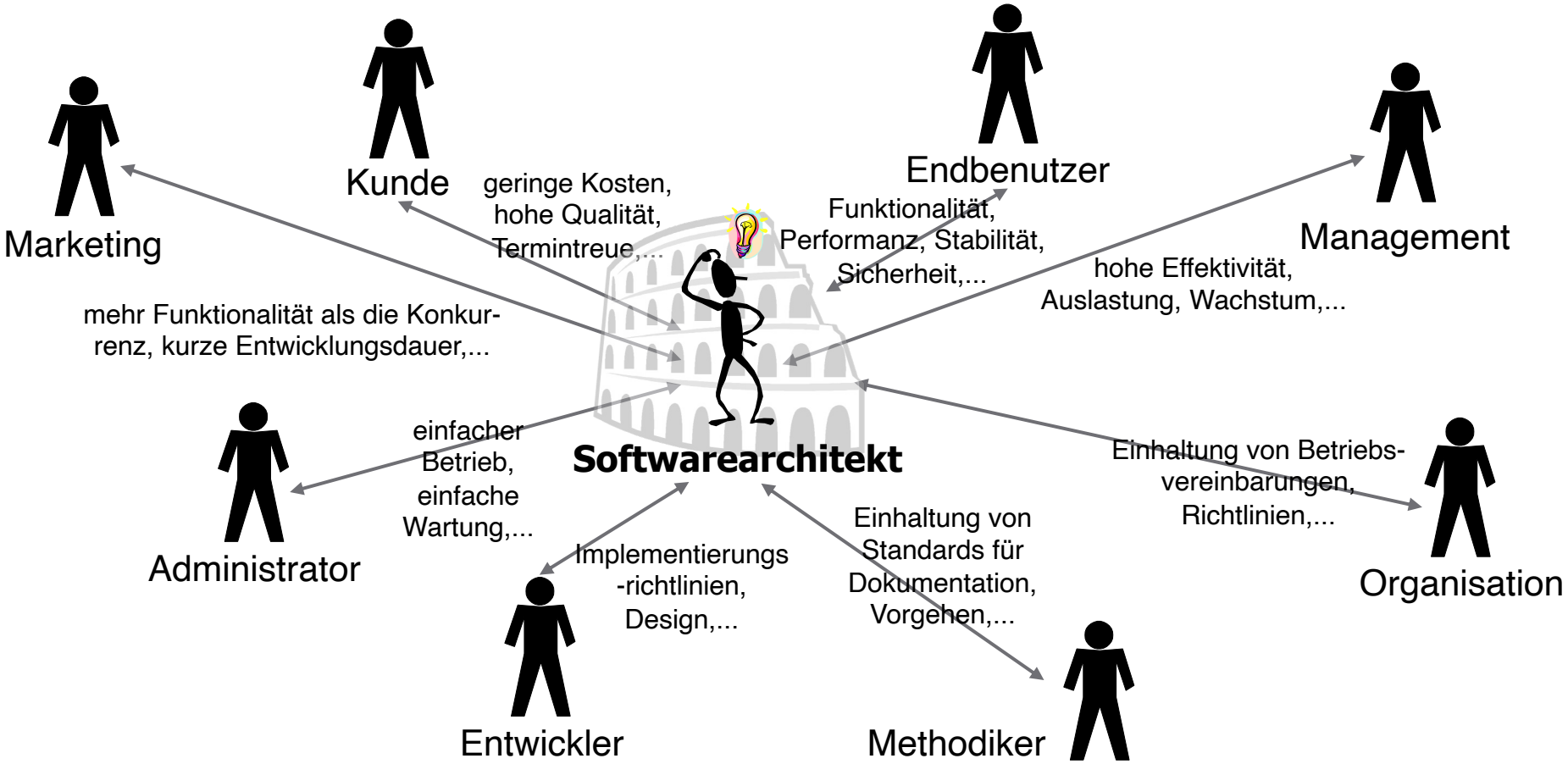


ISO/IEC/IEEE 42010:2022 (second edition)
http://www.iso-architecture.org/42010/cm/

Stakeholder

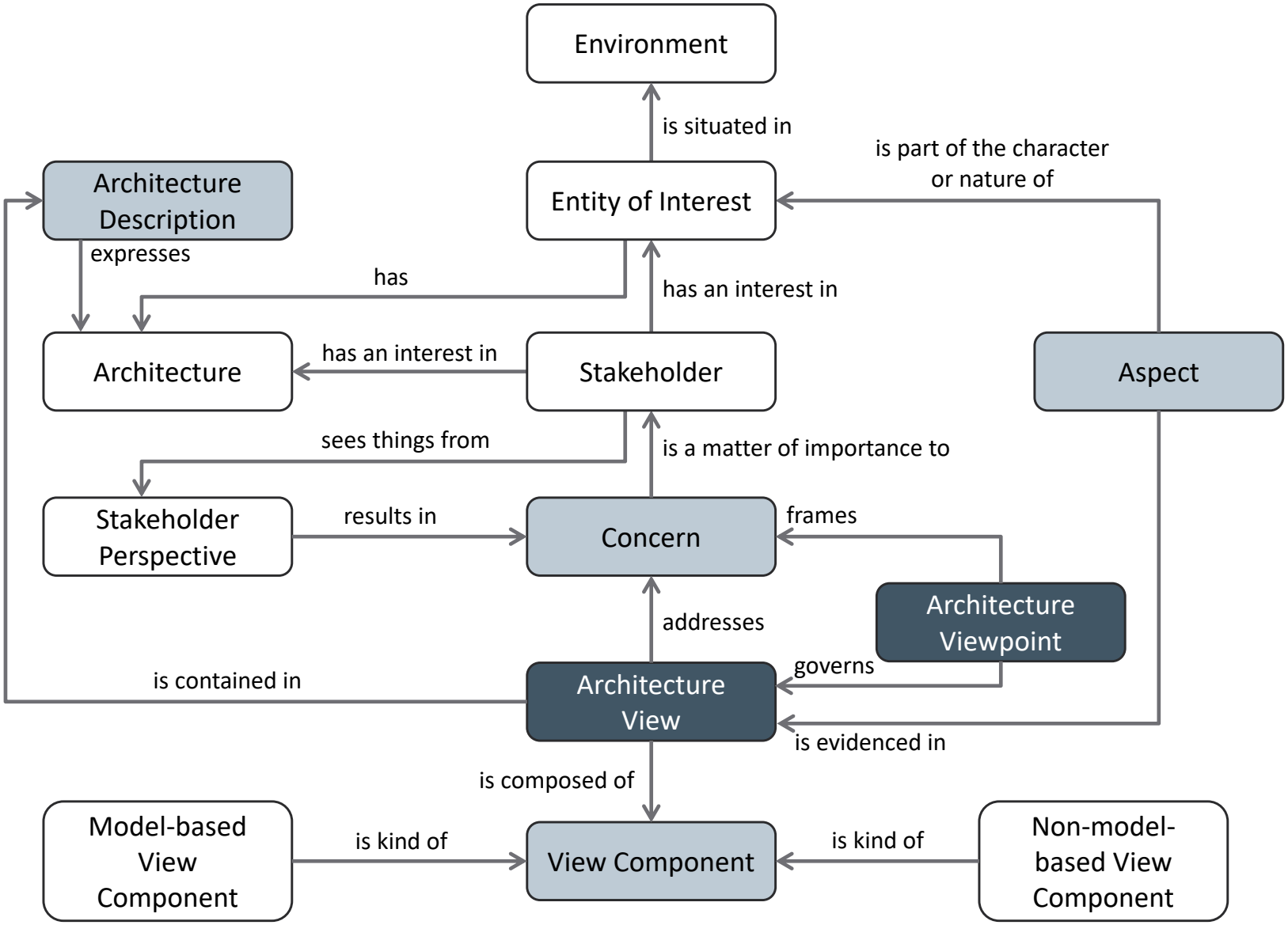
Recht, Anteil, Anspruch, anderes Interesse

> role, position, individual or organization having a right, share, claim or other interest in an architecture entity or its architecture that reflects their needs and expectations
>
> [ISO/IEC/IEEE 42020:2019]



Views & Viewpoints

Views & Viewpoints nach ISO/IEC/IEEE 42010-2022



AD = Architecture Description

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
    - Zum Beispiel Wartbarkeit oder Benutzerfreundlichkeit

- Rolle in der Architekturbeschreibung
  - Dienen als Grundlage für die Erstellung und Strukturierung der Architektursichten
  - Lenken den Fokus der Architekturarbeit 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, Wartbarkeits-Aspekt, Benutzerfreundlichkeits-Aspekt

Viewpoint nach ISO/IEC/IEEE 42010-2022

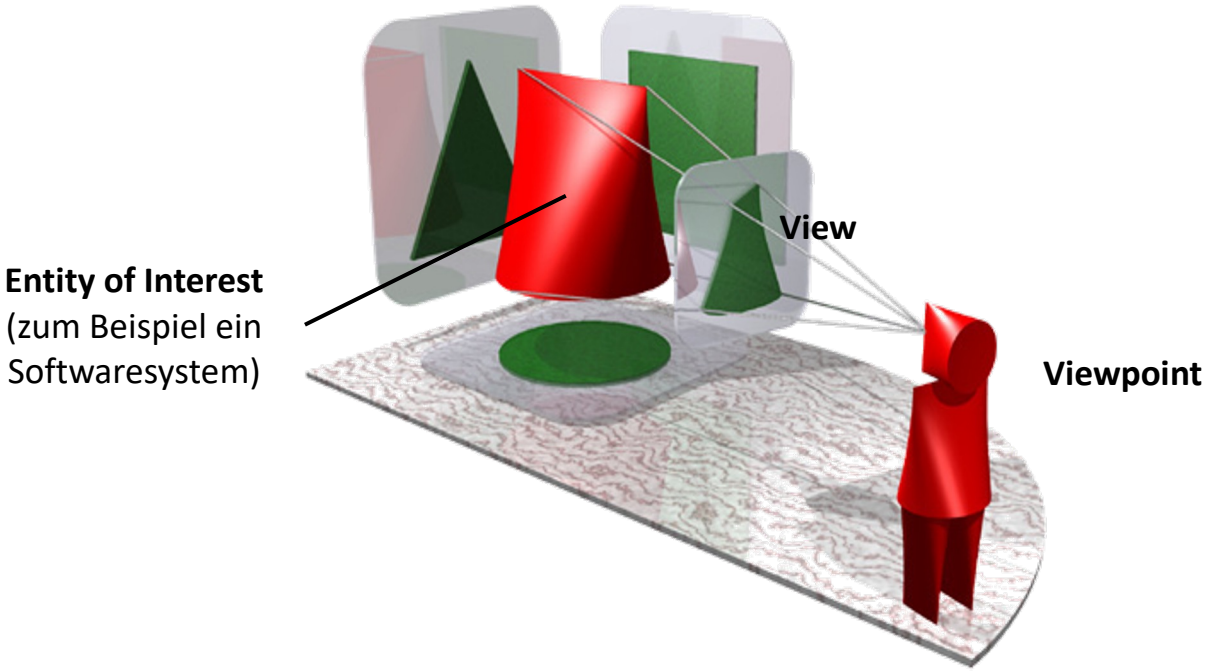
- Ein **Viewpoint** ist ein definierter Satz von Konventionen und Regeln zur Darstellung und Beschreibung bestimmter Concerns eines Systems. Verschiedene Viewpoints sind dafür konzipiert, die Interessen und Bedenken unterschiedlicher Stakeholder zu adressieren.
- Ein Viewpoint definiert unter anderem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

Architecture 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
- Liefert spezifische Einsichten in das System
- UML-Diagramme können zum Beispiel Architecture Views sein
  - Klassendiagramme, Komponentendiagramme, Sequenzdiagramme, …

Views und Viewpoints

Viewpoint: Standpunkt, Blickwinkel
View: Sicht



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

Architecture View & Architecture Viewpoint

Beispiele

Stakeholder	Viewpoint	Views	View Components
Administrator, Developer, IT Infrastructure	Deployment	UML Deployment Diagram	Server, Netzwerkknoten, Datenbanken
Developer, Tester	Statische Struktur	UML Komponenten-diagramm	Komponenten, Schnittstellen
Developer	Laufzeitsicht	UML Sequenzdiagramm, UML Aktivitätsdiagramm	Prozesse, Aktivitäten

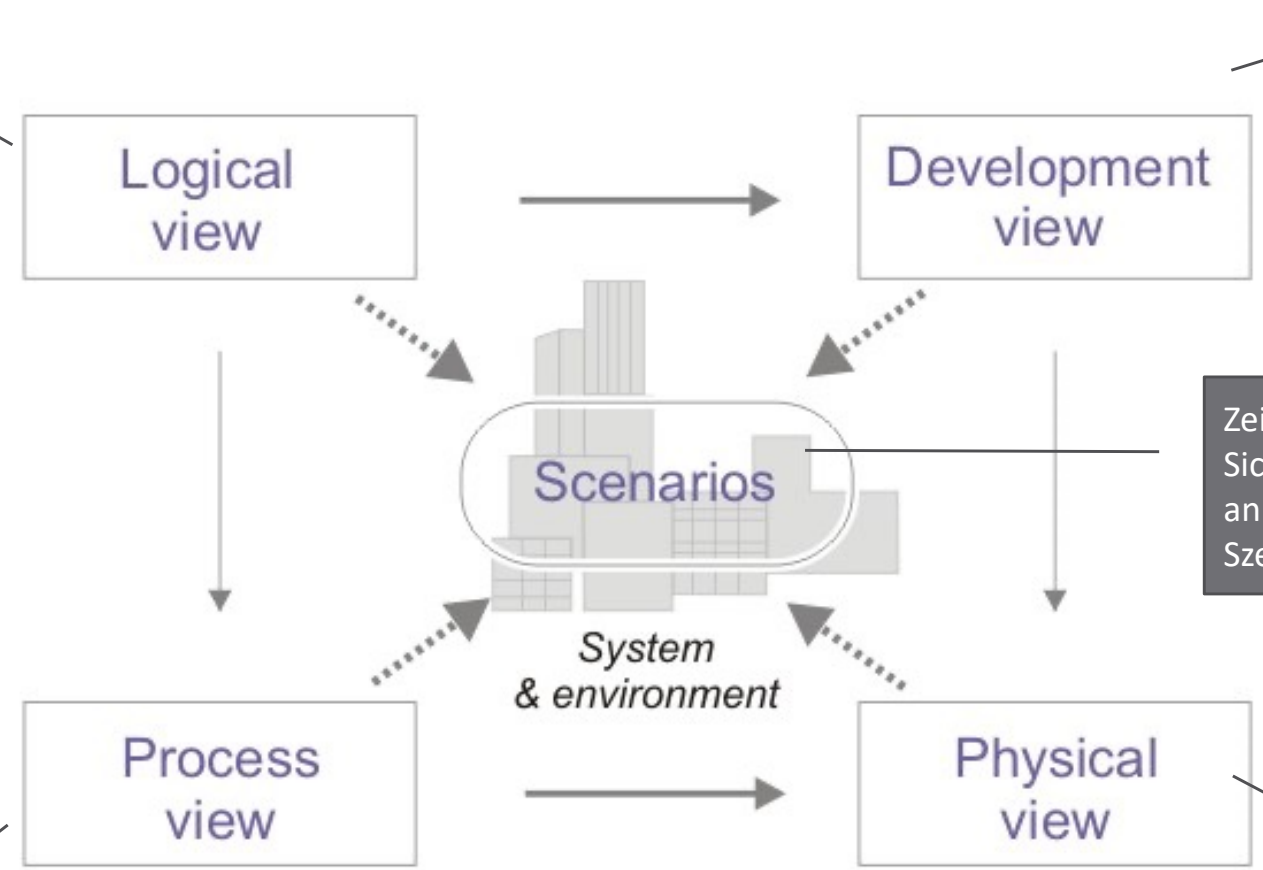
Sichtenmodelle

Sichtenmodelle

- Strukturierte Darstellungen eines Systems, die verschiedene Aspekte der Architektur aus unterschiedlichen Perspektiven oder Sichten zeigen.
- Umfassen eine Sammlung von verschiedenen Architektur-Sichten, die zusammen eine umfassende und kohärente Beschreibung eines Systems liefern.
- Die Norm ISO/IEC/IEEE 42010 bietet eine Grundlage zur Erstellung solcher Sichtenmodelle

4+1 Sichtenmodell nach Philippe Kruchten

Fokus auf die Strukturierung des Systems in Module und deren Interaktion



Struktur des Quellcodes sowie Details zum Build-Prozess und anderer Aspekte des Entwicklungsprozesses

Zeigt, die verschiedenen Sichten zusammenarbeiten anhand verschiedener Szenarien

Fokus darauf, wie die Module zur Laufzeit interagieren und wie Daten durch das System fließen

Physikalische Verteilung des Systems auf Hardware sowie Aspekte wie Netzwerktopologie

Siemens‘ Four View Model

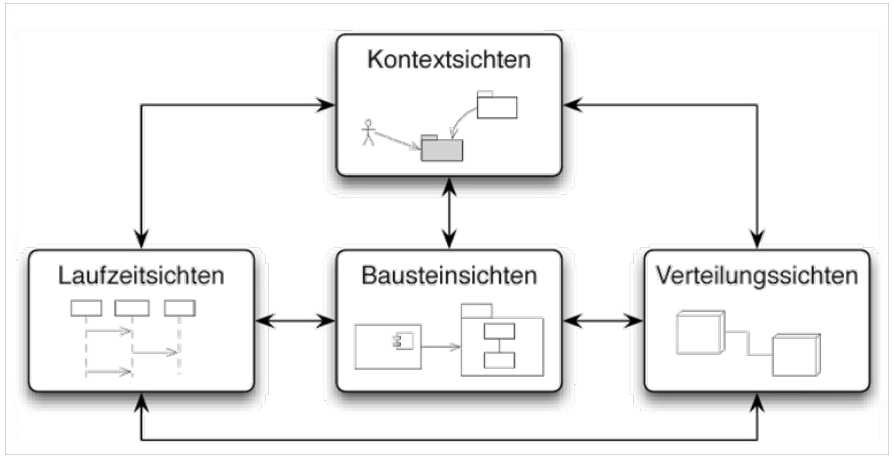
Entwickelt von Siemens Corporate Research zur umfassenden Beschreibung von Softwarearchitekturen

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

Sichtenmodell nach arc42

Viewpoints

- 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-Haube
8.4 User Experience
….

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
….

10. Qualitätsanforderungen
10.1 Qualitätsbaum
10.2 Qualitätsszenarien

11. Risiken und technische Schulden

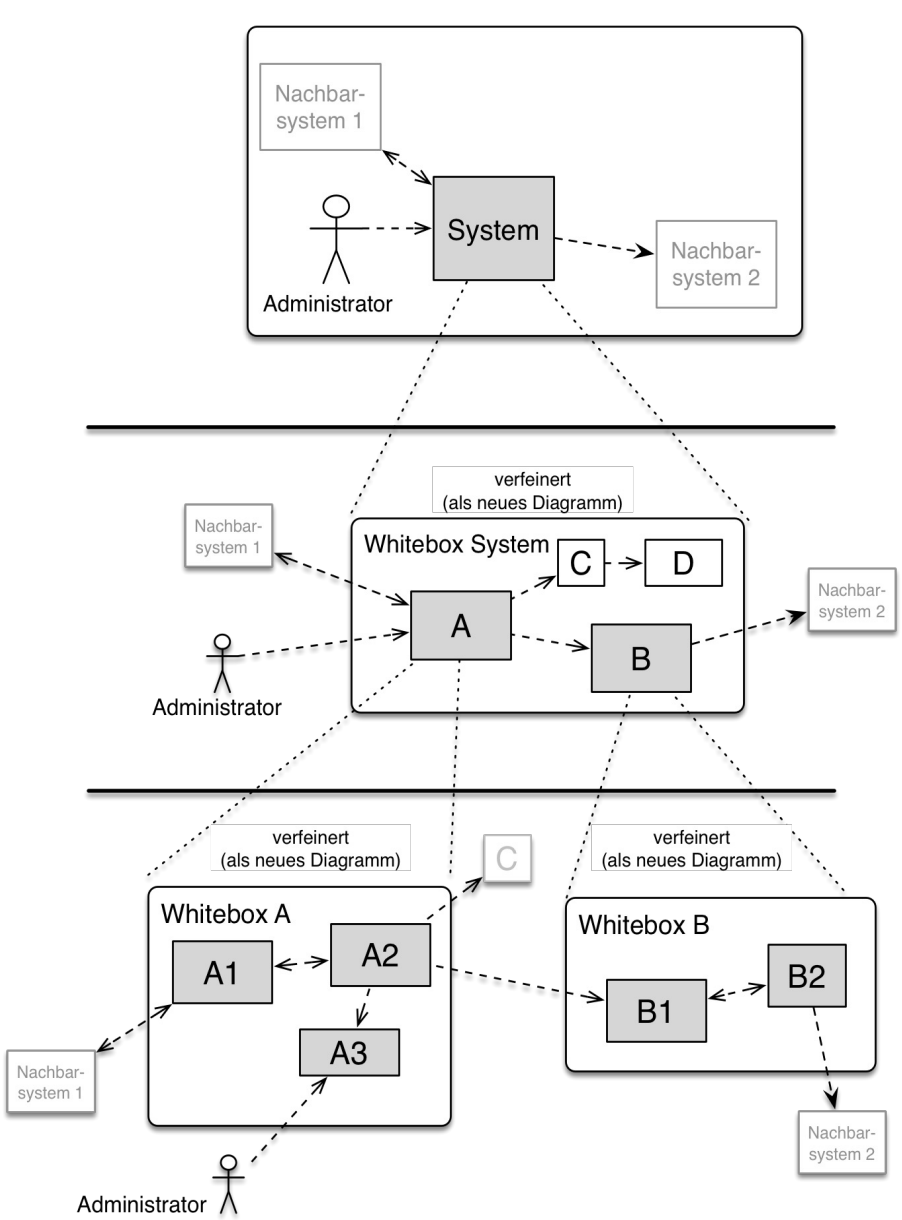
12. Glossar

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
    - Grobe Interaktionen und Schnittstellen
  - Mittlere Ebene
    - Aufteilung des Systems in Subsysteme/Komponenten
    - Interaktionen zwischen Subsystemen/Komponenten
    - Schnittstellen und Kommunikationswege
  - Mikroarchitektur
    - Detaillierte Beschreibung der Bausteine oder Komponenten
    - Interne Struktur und Abhängigkeiten
    - Schnittstellen und Kommunikationswege

Ermöglicht Zielgruppen-spezifische Dokumentation

Ebenen in arc42



arc42 – Beispiel DokChess

Quelle: https://www.dokchess.de

UML – Unified Modeling Language

Jede Sicht braucht geeignete Beschreibungstechniken

Mit einer Diagrammart lassen sich nicht alle Aspekte aller Sichten gleichermaßen abbilden. Für jeden Viewpoint werden unterschiedliche Views benötigt.

Mit der „Unified Modeling Language (UML)“ steht eine grafische Modellierungssprache zur Verfügung, welche zur Dokumentation von Softwaresystem verwendet werden kann.

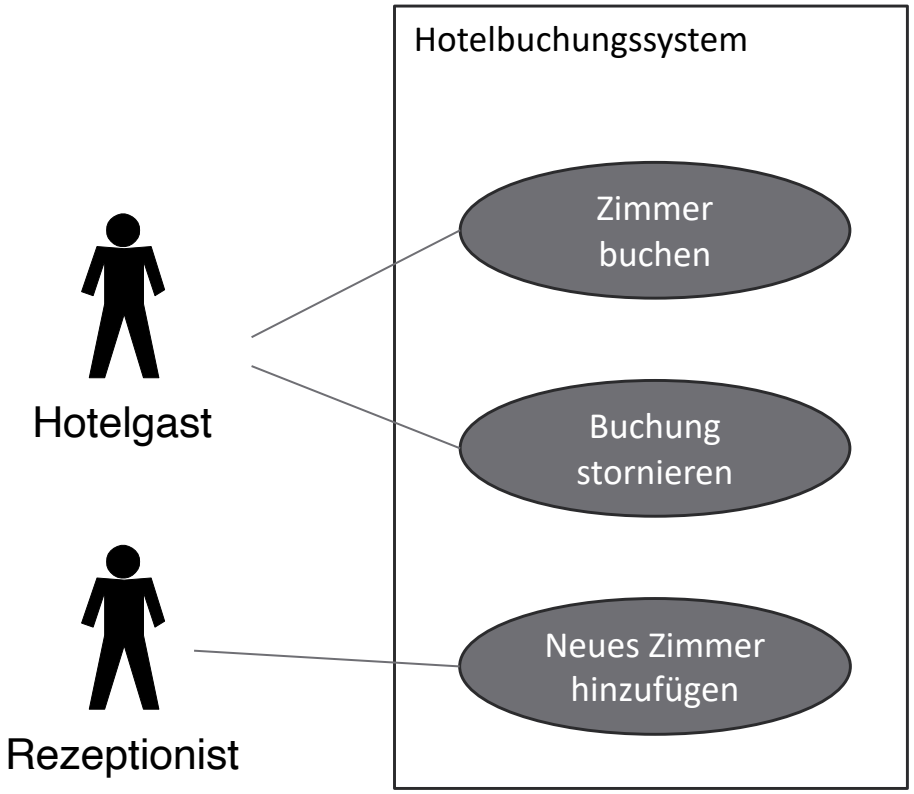
- Sprache zur Erstellung von Views

<u>Diagramme in UML lassen sich in zwei Hauptgruppen einteilen:</u>

- **Strukturdiagramme**
  Klassendiagramm, Komponentendiagramm, Verteilungsdiagramm, Kompositionsstrukturdiagramm, Paketdiagramm, Objektdiagramm, Profildiagramm

- **Verhaltensdiagramme**
  Aktivitätsdiagramm, Anwendungsfalldiagramm, Interaktionsübersichtsdiagramm, Kommunikationsdiagramm, Sequenzdiagramm, Zeitverlaufsdiagramm, Zustandsdiagramm

UML Anwendungsfalldiagramm

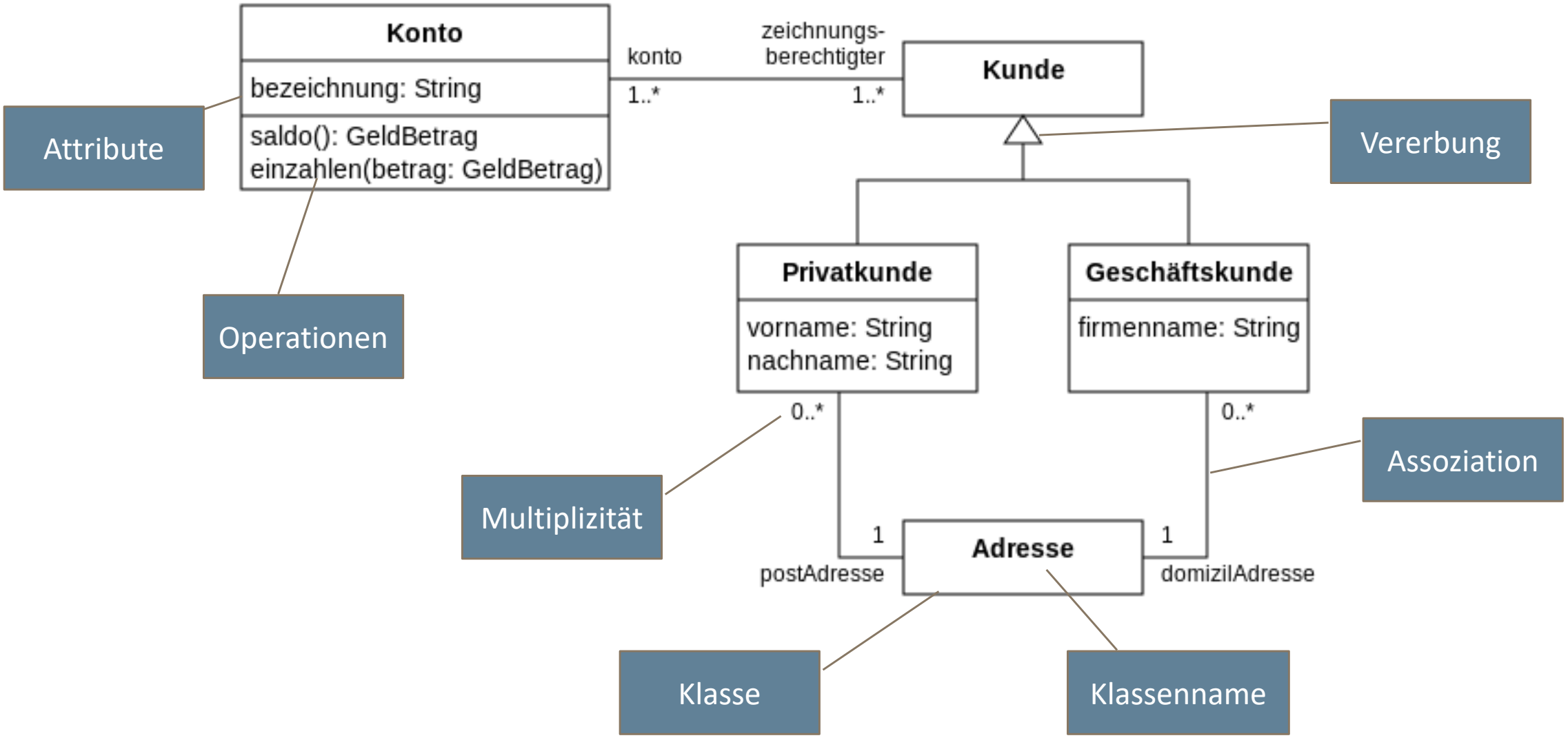


- Modellierung von funktionalen Anforderungen eines Systems aus Anwender-Perspektive
- Bestandteile
  - Systemgrenze, inklusive Name des Systems
  - Akteure (Piktogramme) / Externe Systeme
  - Anwendungsfälle (Ellipsen)
  - Beziehungen

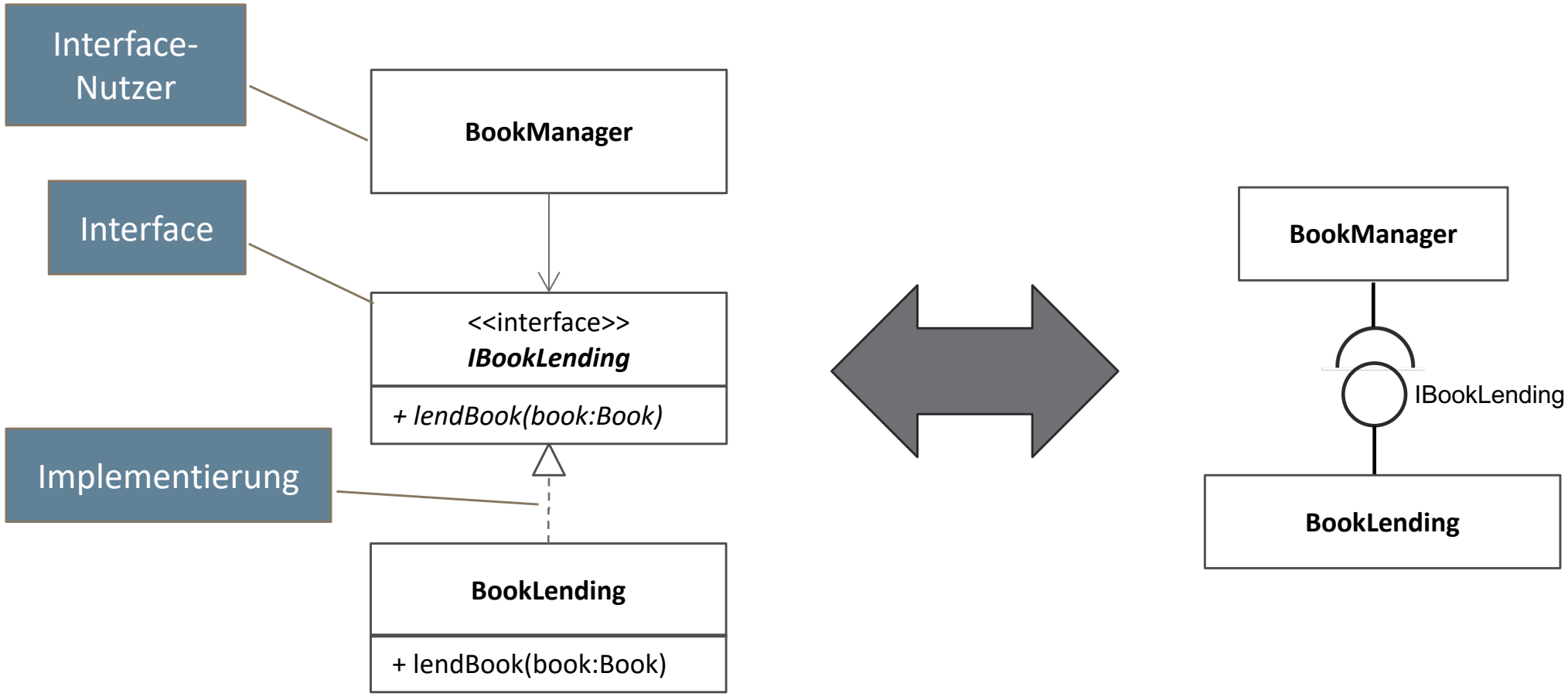
UML Klassendiagramm

- 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

UML Klassendiagramm

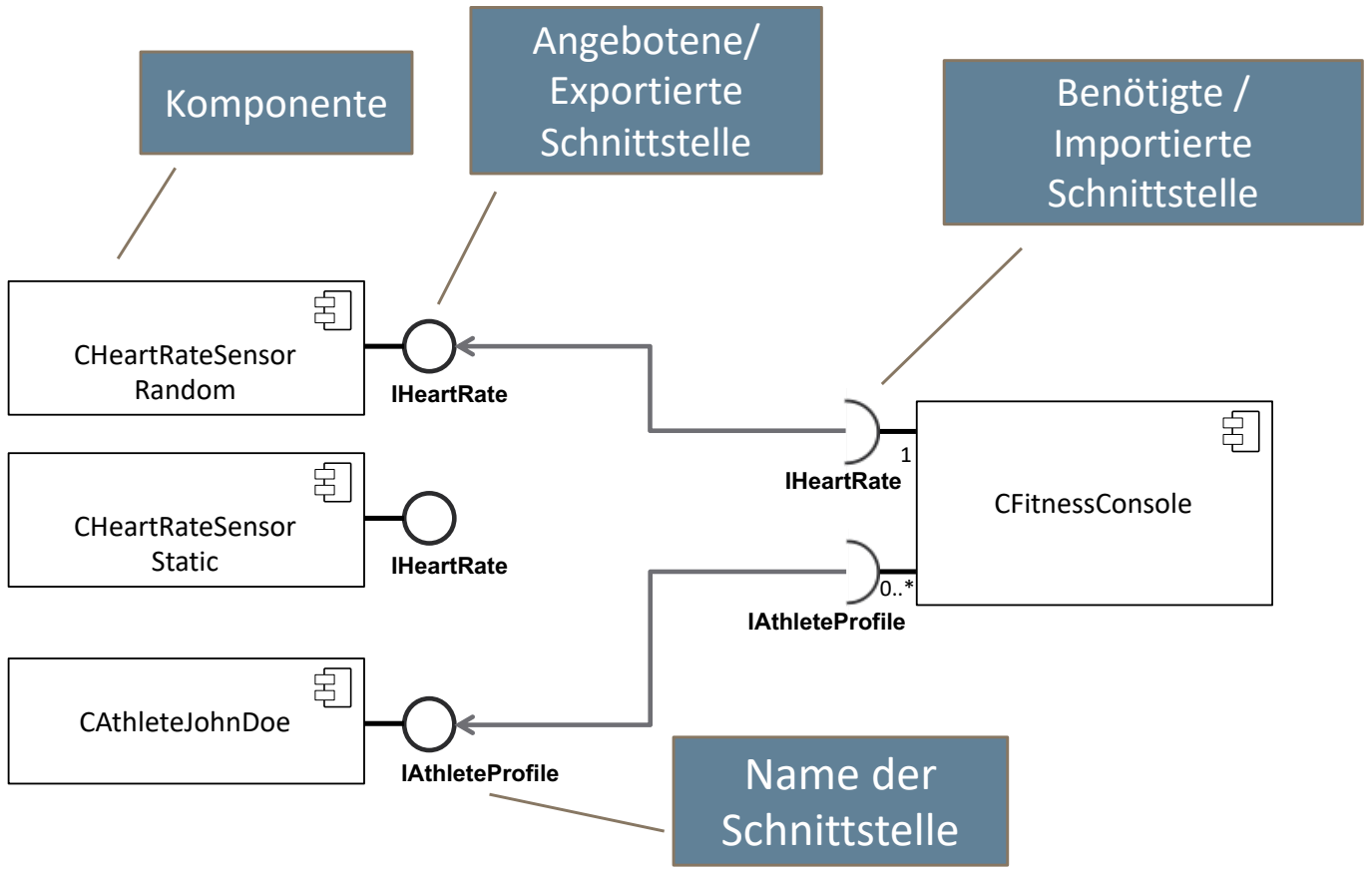


UML Klassendiagramm

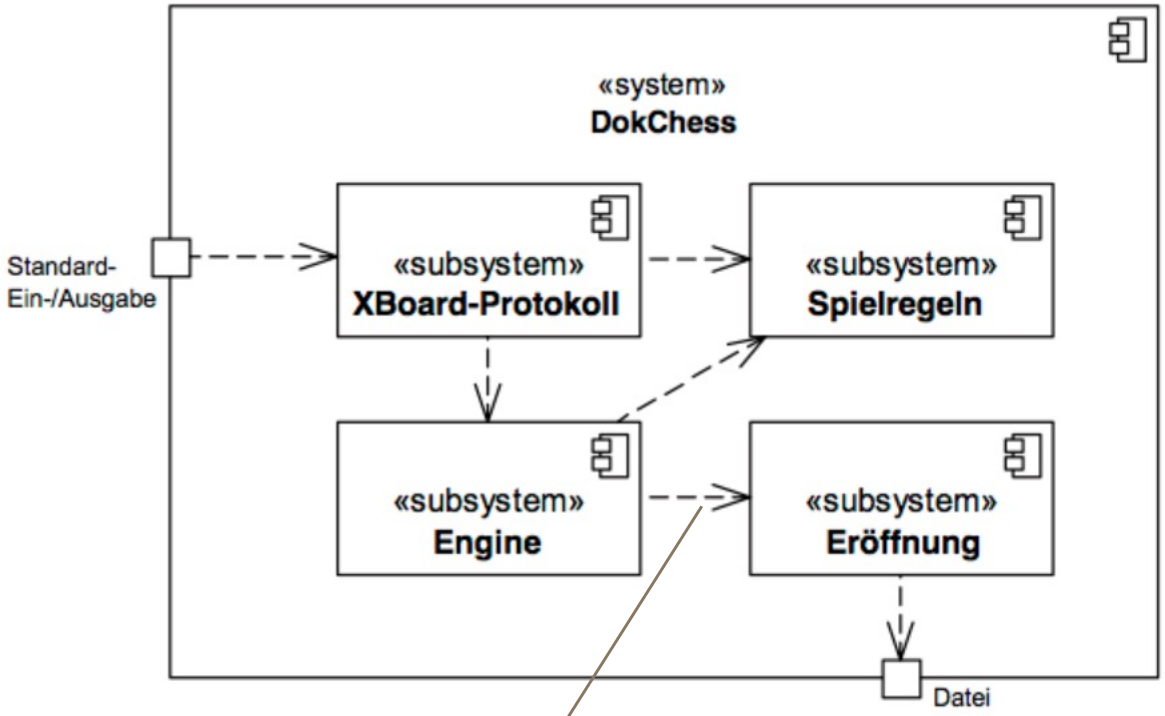


UML Komponentendiagramm

- Grafische Modellierung von Komponenten und Subkomponenten sowie deren Verdrahtung.
- Wird häufig verwendet in
  - Bausteinsicht
  - Module View
  - Kontextabgrenzung



UML Komponentendiagramm

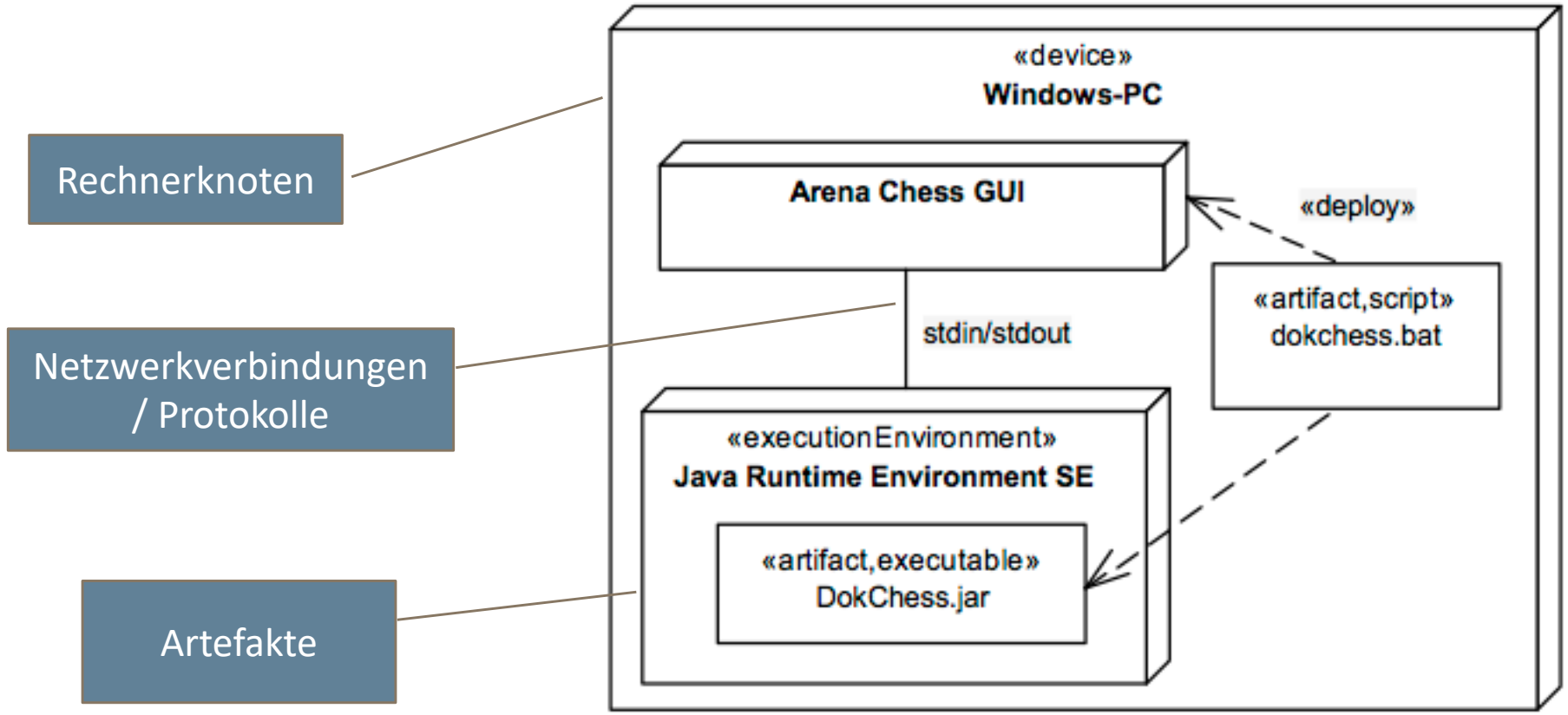


Komponente „Engine“ hängt von Komponente „Eröffnung“ ab

Quelle: arc42.de

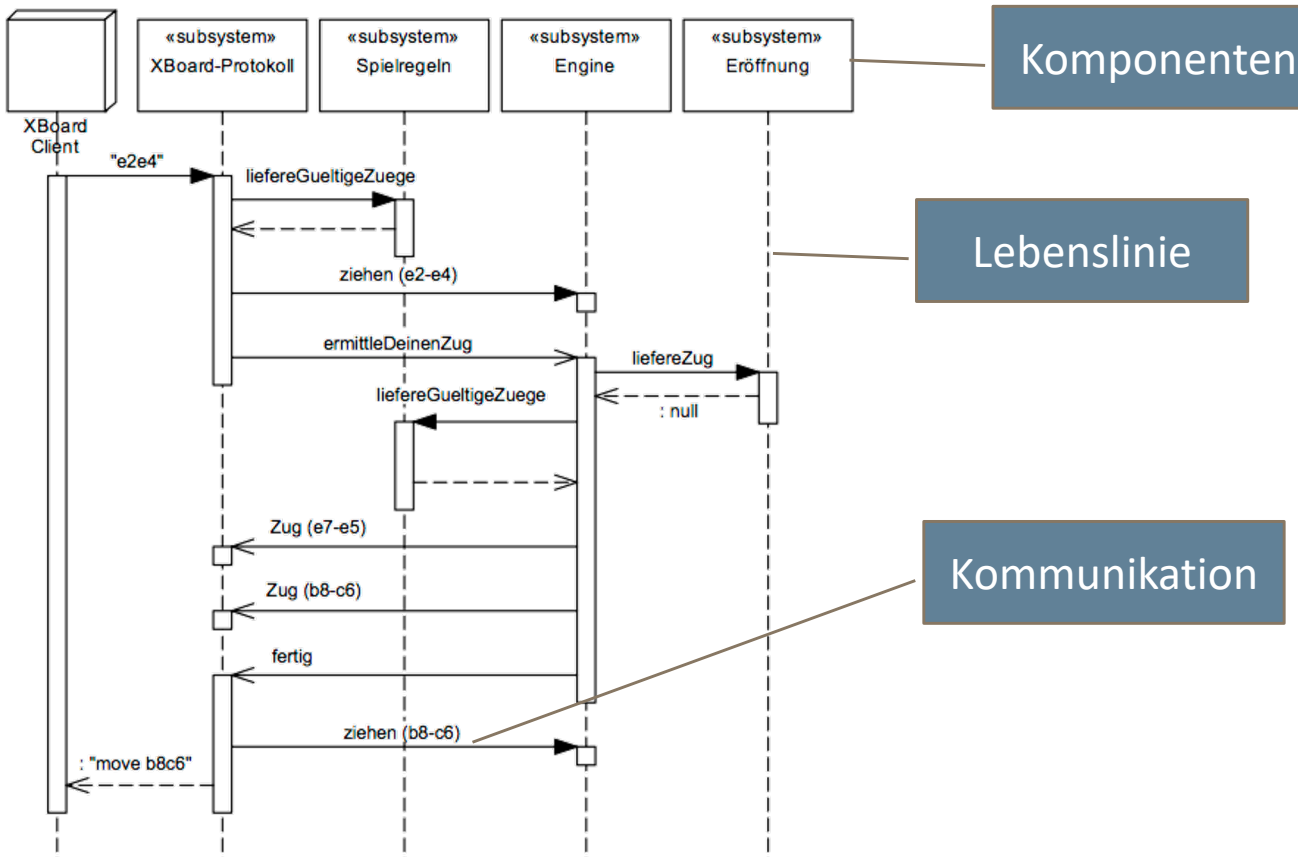
UML Verteilungsdiagramm

- Grafische Modellierung der Verteilung von Komponenten auf Rechnerknoten



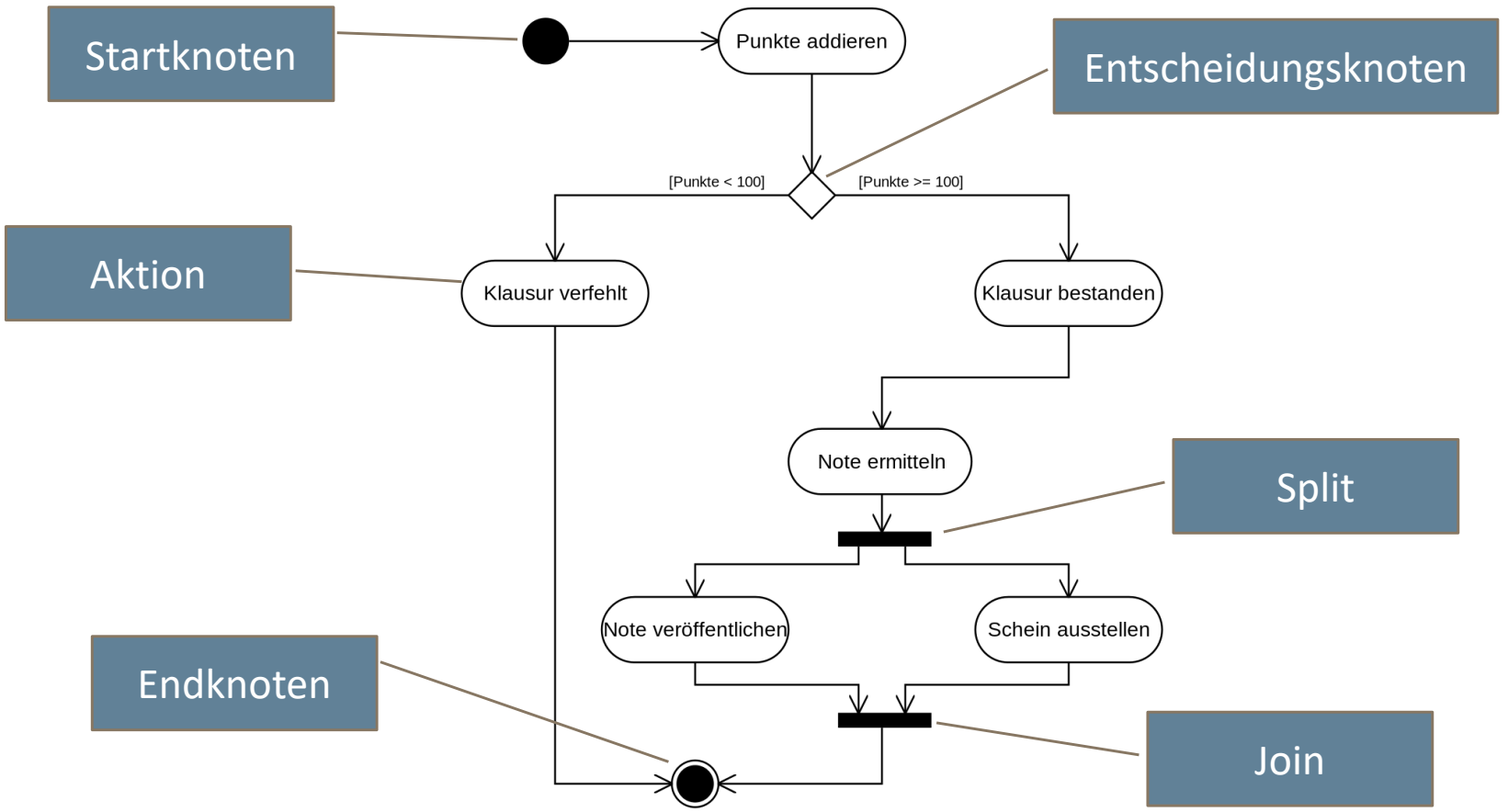
UML Sequenzdiagramm

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

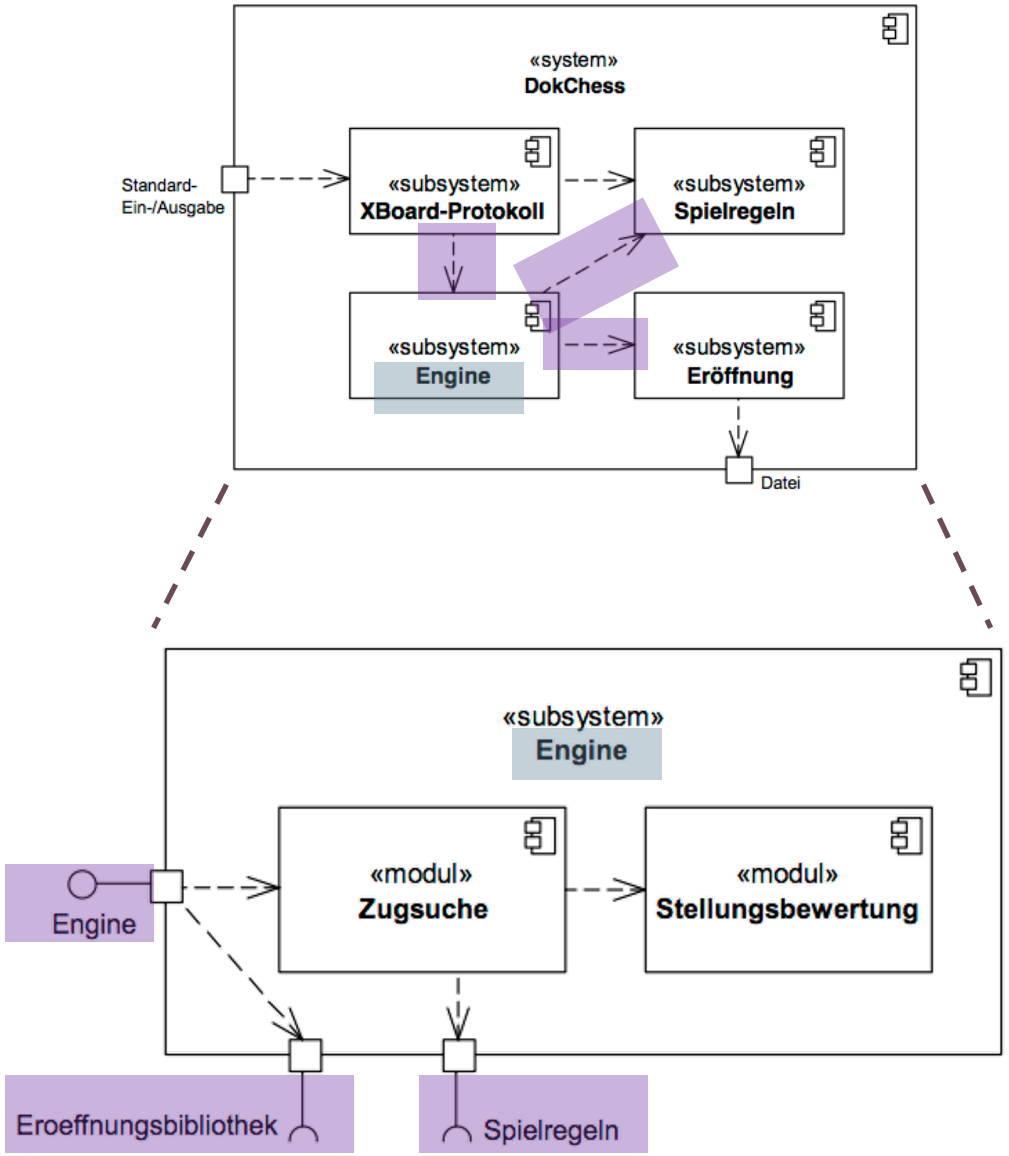


UML Aktivitätsdiagramm

- Stellt die Vernetzung von Aktionen und deren Verbindungen mit Kontroll- und Datenflüssen grafisch dar.

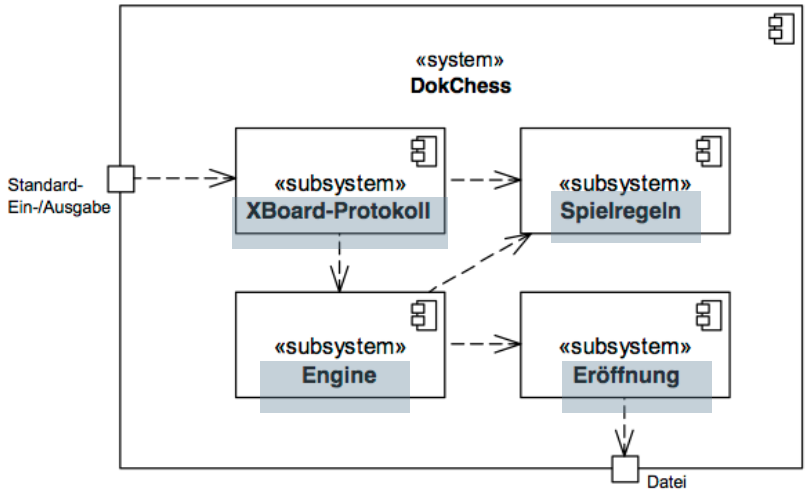


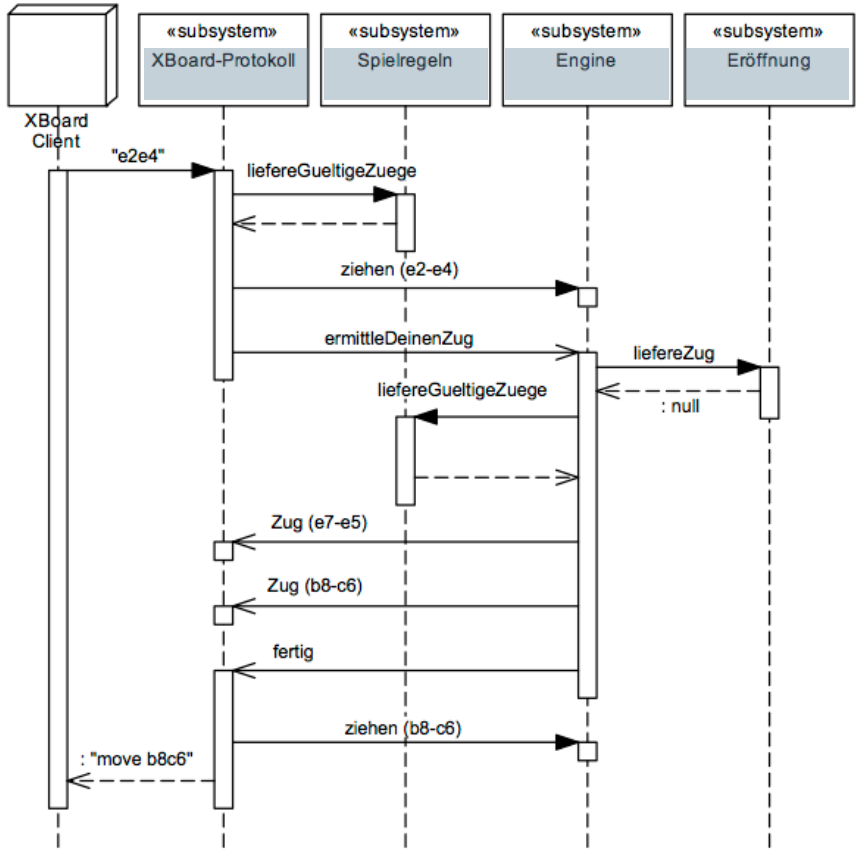
Konsistenz zwischen Diagrammen



- Auf konsistente Benennung der Elemente zwischen Diagrammen achten (Beispiel “Engine“)
- Auf Konsistenz der Schnittstellen achten

Konsistenz zwischen Diagrammen





- Auf konsistente Benennung der Elemente zwischen Diagrammen achten (Beispiel “Engine“, „Spielregeln“, „Eröffnung“, „Xboard-Protokoll“)

Dokumentation von Architekturentscheidungen

Architekturentscheidungen

- Mit UML werden insbesondere Ist- und Sollzustand einer Architektur dokumentiert
- **Hauptarbeit eines Architekten ist es, Architektur-relevante Entscheidungen zu treffen.**
- Diese Entscheidungen können
  - Auf dem Ist-Zustand beruhen
  - Auf einer Auswahl von möglichen Plänen 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 im Team und im Unternehmen durchzusetzen
  - Die Qualität von Architekturentscheidungen zu messen
    - Welche Entscheidungen hatten welche Auswirkungen?
  - Die Entstehung einer künftigen Ist-Architektur nachzuvollziehen
  - dem Team Guidelines für detailliertere Entscheidungen zu geben

- Feedback-Loops helfen, indem die getroffenen 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 (ADRs)*.
- 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 der Entscheider oder das Datum der Entscheidung hinterlegt werden.

Weitere ADR-Templates findet ihr unter: https://adr.github.io

Allgemeine Aspekte zur Dokumentation

Weitere übliche Dokumenttypen

- Zentrale Architekturbeschreibung (arc42)
- Architekturüberblick
- Dokumentübersicht
- Übersichtspräsentation
- Architekturtapete
- Handbuch zur Dokumentation
- Technische Informationen
- Dokumentation von externen Schnittstellen

Praxisregeln zur Dokumentation

- Schreiben aus Sicht des Lesers
- Unnötige Wiederholungen vermeiden
- Mehrdeutigkeit vermeiden
- Standardisierte Organisationsstruktur bzw. Templates und Satz-Schablonen
- Begründen Sie wesentliche Entscheidungen schriftlich
- Überprüfung der Gebrauchstauglichkeit
- Übersichtliche Diagramme
- Regelmäßige Aktualisierungen

Praxisregeln zur Dokumentation

- Eine Architekturbeschreibung sollte so einfach wie möglich sein, aber so 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.”

[nach Antoine de Saint-Exupery]