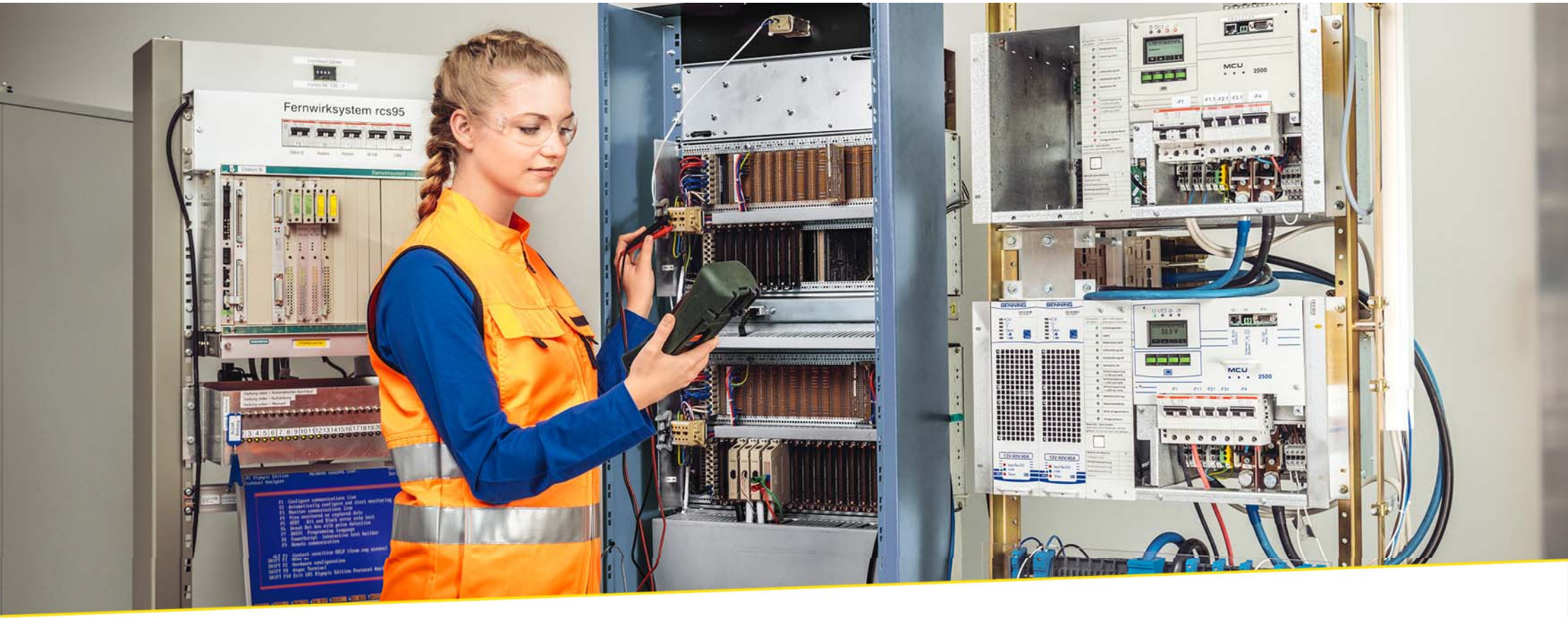
Software Architektur Grundlagen



SBB CFF FFS

bls

Rhätische Bahn
Ferrovia retica Viafier retica

VÖV UTP

Was ist Software Architektur?

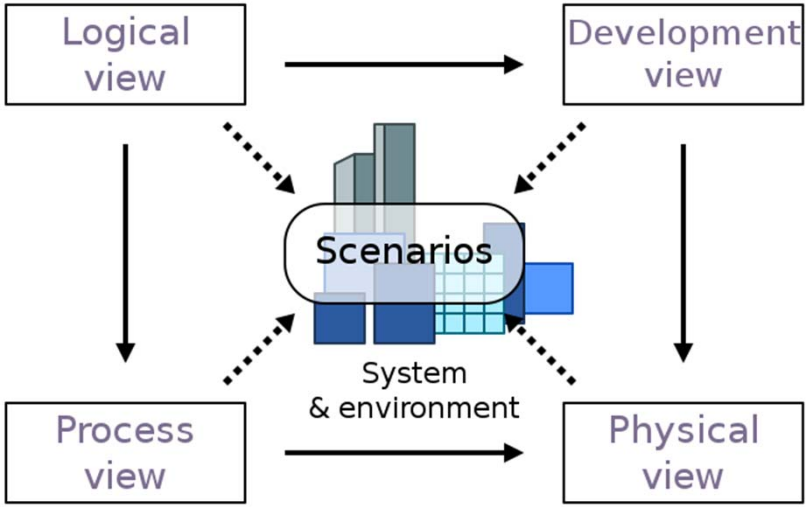
- Die Architektur einer Software definiert das System und die Interaktionen der Komponenten
- Strukturiert und visualisiert das System
- Konzeptionelle Abstraktion eines komplexen Systems
- Es gibt verschiedene Sichten für eine Software (4+1 Sichtenmodell):
- UML ist ein Bestandteil der Darstellung von Softwarearchitekturen

Wozu braucht es Software Architektur?

- Basis zur Kommunikation
  - Stakeholder können anhand der verschiedenen Visualisierungen kommunizieren
  - Kosten durch Missverständnisse können so verhindert werden
- Die frühesten Entscheidungen werden getroffen
  - Die wichtigsten Entscheidungen werden ganz am Anfang getroffen
  - Frühe Fehler können vermieden werden
- Übertragbarkeit des Modells
  - Wenn die Architektur gut beschrieben ist, kann das Modell für zukünftige Projekte verwendet werden

4+1 Sichtenmodell

- Logical View: Befasst sich mit der Funktionalität
  - Klassendiagramm, Kommunikationsdiagramm, Sequenzdiagramm
- Development View: Beschreibt die Entwicklersicht
  - Komponentendiagramm, Paketdiagramm
- Process View: Bildet die dynamischen Aspekte ab
  - Aktivitätsdiagramm
- Physical View: Befasst sich mit der Verteilung der Software auf Hardware
  - Verteilungsdiagramm
- Scenarios: Sicht für den «Kunden», beschreibt die Anwendungsfälle
  - Anwendungsfalldiagramm (Use-Case Diagramm)



https://en.wikipedia.org/wiki/4%2B1_architectural_view_model

Was macht eine gute Software Architektur aus?

Eine gute Software Architektur:

- Ist Robust und einfach zu warten
- Hat Konzepte die von allen Beteiligten verstanden werden
- Ist flexibel und erweiterbar
- Kann ändernde Anforderungen entgegen nehmen
- Ist Skalierbar
- Berücksichtigt die üblichen Best-Practices

Das Endprodukt einer guten Software Architektur:

- Ist Benutzerfreundlich
- Ist flexibel und adaptierbar
- Ist auch bei grossem Benutzerandrang robust
- Verhält sich so, wie der Benutzer arbeitet
- Weist keine Performanzschwächen auf
- Ist wartbar
- Ist zuverlässig

UML

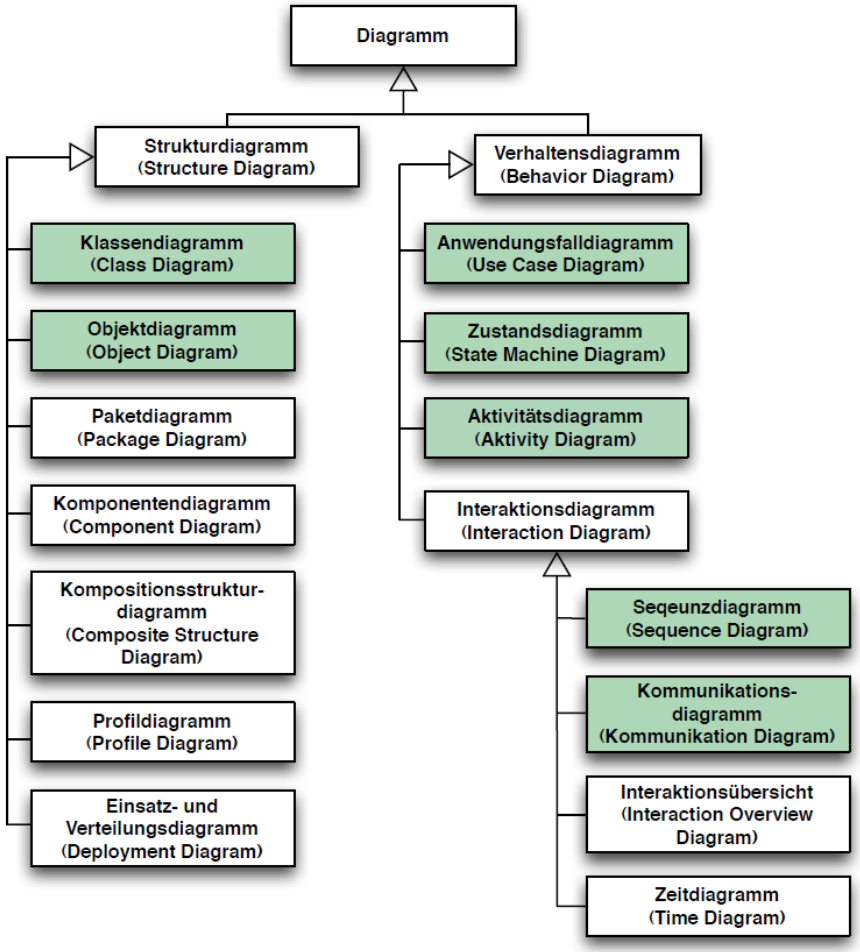
Was ist UML?

- Steht für Unified Modeling Language
- Wird zur Dokumentation von Software und Systemen verwendet
- Beinhaltet ein Set von Diagrammen

Welche Typen von UML Diagrammen gibt es?

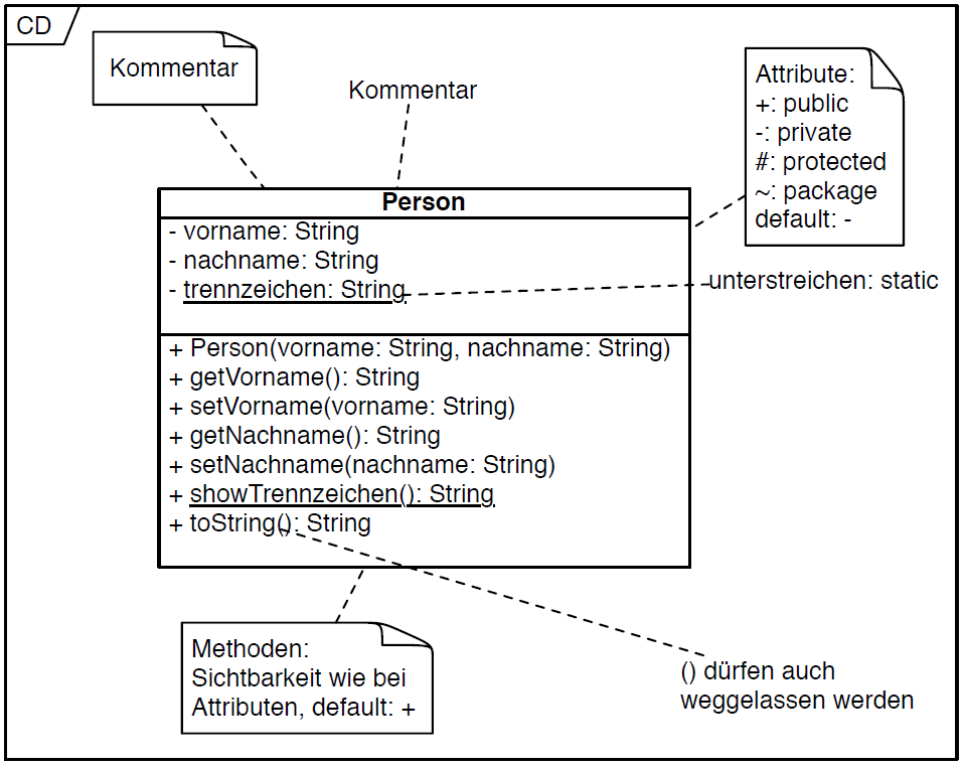
- Verhaltensmuster
  - Aktivitätsdiagramm
  - Anwendungsfalldiagramm (Use-Case Diagramm)
  - Interaktionsübersichtsdiagramm
  - Zustandsdiagramm
  - Sequenzdiagramm
  - Kommunikationsdiagramm
- Strukturelle Muster
  - Klassendiagramm
  - Objektdiagramm
  - Komponentendiagramm

UML Diagramme

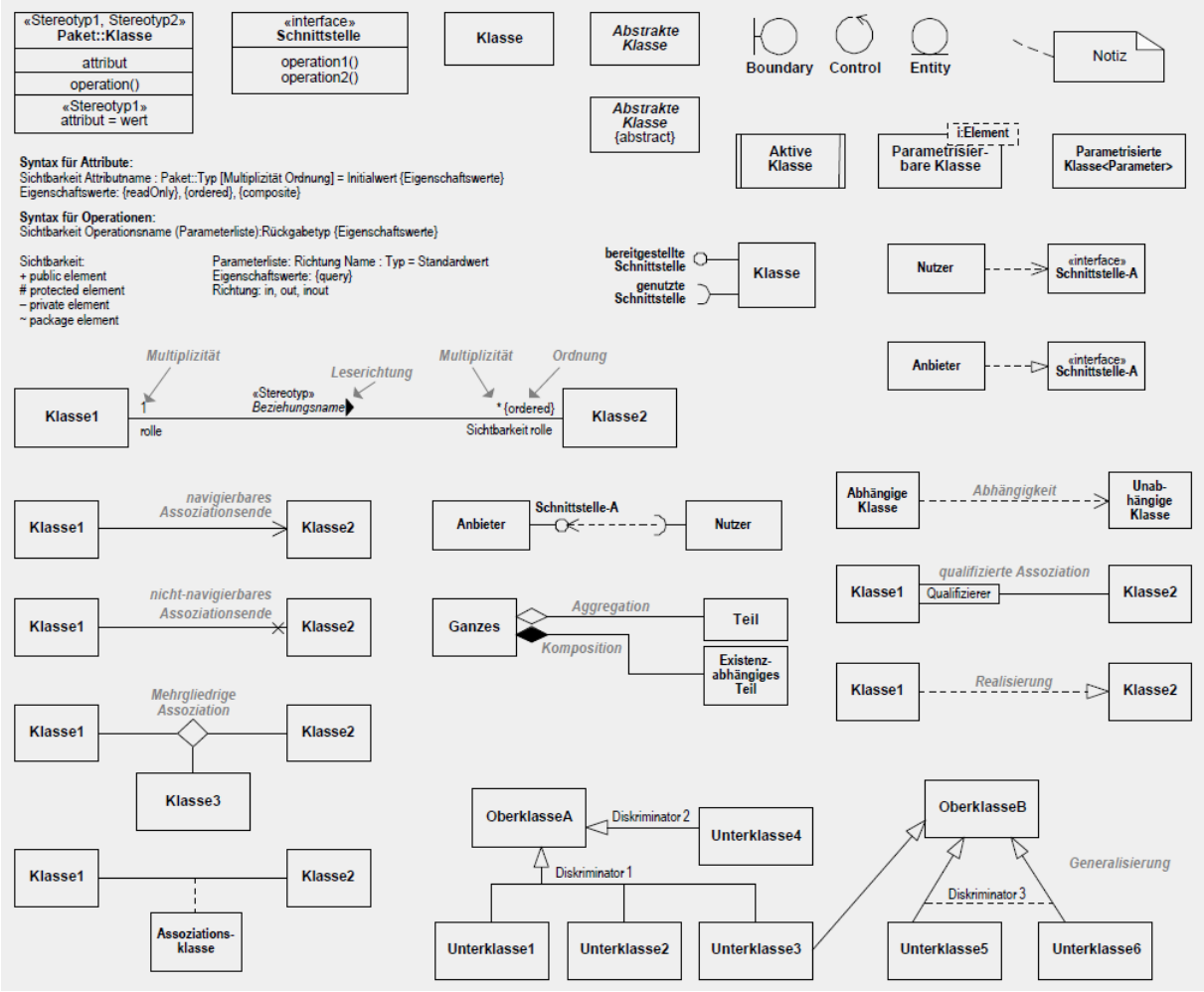


Klassendiagramm

Ein Klassendiagramm ist ein Strukturdiagramm der Unified Modeling Language (UML) zur grafischen Darstellung (Modellierung) von Klassen, Schnittstellen sowie deren Beziehungen. Eine Klasse ist in der Objektorientierung ein abstrakter Oberbegriff für die Beschreibung der gemeinsamen Struktur und des gemeinsamen Verhaltens von Objekten (Klassifizierung). Sie dient dazu Objekte zu abstrahieren. Im Zusammenspiel mit anderen Klassen ermöglichen sie die Modellierung eines abgegrenzten Systems in der objektorientierten Analyse und Entwurf.



Klassendiagramm



Klassendiagramm - Geheimnisprinzip

Vorteile:

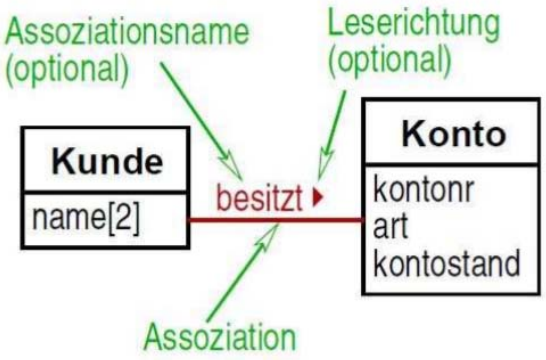
- Der Zustand eines Objektes kann nicht inkonsistent werden, ohne dass dieses Objekt ausdrücklich die Erlaubnis dazu erteilt
- Seiteneffekte vermindern sich, weil eine Änderung innerhalb der Kapsel keine Auswirkung auf die Schnittstellennutzer hat

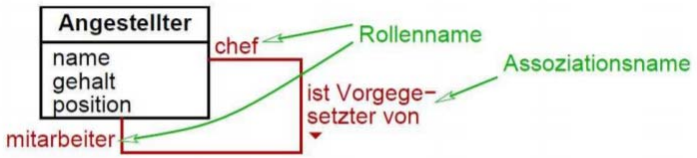
Nachteile:

- Der Implementierungsaufwand ist grösser.
- Die Performance sinkt, weil mehr Operationsaufrufe und Prüfungen benötigt werden

Assoziationen

- Zwischen Objekten/Klassen können Beziehungen bestehen
- Die Assoziation ist eine statische Beziehung, d.h. sie drückt aus, welche Objekte zu jederzeit miteinander verbunden sind, damit Abläufe funktionieren können (sie kann nicht darstellen, wie eine Datenänderung entsteht)
- Eine Assoziation kann auch reflexiv sein
- Neben Assoziationsnamen auch Rollennamen möglich





Assoziationen 2

Multiplizität von Assoziationen

- Assoziation sagt zunächst nur, dass ein Objekt andere Objektekennen kann
- Multiplizität legt fest, wie viele Objekte ein Objekt kennen kann (oder muss)
- Die Multiplizität wird am Ende der Assoziations-Linie notiert

Assoziationen 3

Aggregation:

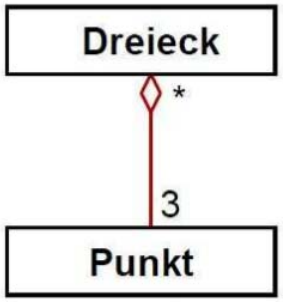
- Teile existieren selbständig und können (gleichzeitig) zu mehreren Aggregat-Objekten gehören

Komposition:

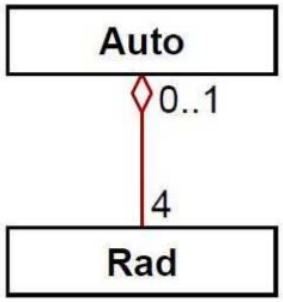
- starke Form der Aggregation
- Teil-Objekt gehört zu genau einem Komposit-Objekt
  - es kann nicht Teil verschiedener Komposit-Objekte sein
  - es kann nicht ohne sein Komposit-Objekt existieren
- Beim Erzeugen (Löschen) des Komposit-Objekts werden auch seine Teil-Objekte erzeugt (gelöscht)

Assoziationen 4

Aggregation

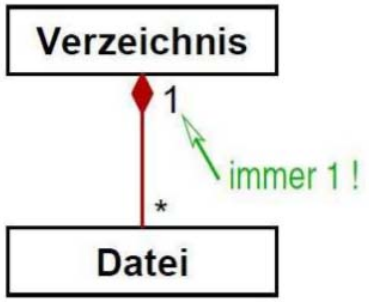


Ein Dreieck besteht immer aus 3 Punkten.
Ein Punkt ist Teil von beliebig vielen (incl. 0) Dreiecken.



Ein Auto besteht (u.a.) aus 4 Rädern.
Ein Rad ist Teil von höchstens einem Auto (es gibt auch Räder ohne Auto).

Komposition



Ein Verzeichnis besteht aus beliebig vielen Dateien
Dateien stehen immer in einem Verzeichnis. Wird dieses gelöscht, so auch alle enthaltenen Dateien.

Assoziationsklassen

- Manchmal hat auch eine Assoziation Eigenschaften und Verhalten
- Dann: Assoziation wird explizit als Klasse modelliert