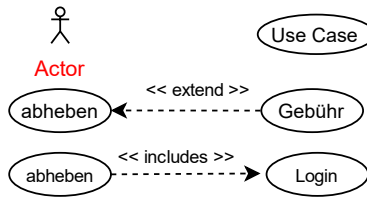


SOFTWARE ENGINEERING / UML - DIAGRAMS / SYMBOLS

USE CASE DIAGRAM

Behaviour Diagramm



Use Case

extend = wird verwendet, um optionale oder zusätzliche Funktionen darzustellen, die unter bestimmten Bedingungen zum Tragen kommen, aber **nicht zwingend notwendig** sind

includes = dient dazu, **große Use Cases zu vereinfachen**, indem wiederkehrende oder gemeinsame Verhaltensweisen in weitere Use Cases ausgelagert werden. Um doppelte Logik zu vermeiden. ("Login" wird für die use cases 1. Geld abheben und 2. Geld einzahlen und 3. Geld überweisen benötigt)

ACTIVITY DIAGRAM

Behaviour Diagramm

Start Node

End Node



Merge Node

MEHRERE Wege zu **EINEM** zusammenführen neues Angebot einholen

Fork Node

Teilung in mehrere gleichzeitige(parallele) Abläufe

Join Node

synchronisiert (sammelt)mehrere Abläufe

[Bedingung 1]

Decision Node

EIN Weg teilt sich aufgrund unterschiedlicher Bedingungen zu **MEHREREN** Wegen

[Bedingung 2]

CLASS DIAGRAM

Strukture Diagramm

Zugriffsmodifikatoren:

Multiplizität:

+ public
- private
protected
~ package
Abstract

* = 0, 1, or mehr
1 = genau 1
2..4 = zw. 2 u. 4 (inkl.)
3..* = 3 oder mehr



Komposition (besteht - aus - Beziehung)

Eine Universität kann nicht ohne Räume existieren. Wenn Universität gelöscht wird, gibt es auch keinen Raum mehr.
Aggregation(ist Teil von - Beziehung)
Person ist Teil von Kurs. Aber beide existieren auch unabhängig von einander.



→ **Assoziation (starke Beziehung)**
dauerhafte Beziehung zwischen 2 Klassen. Multiplizität ! Ein Professor kann einen oder mehrere Kurse haben

- - - - - → **Abhängigkeit (schwache Beziehung)**
Diese Beziehung besteht nur während eine bestimmten Interaktion. Keine Multiplizität !

→ **Generalisierung / Vererbung**
Studen und Professor sind Spezialisierungen (Vererbungen) der allgemeinen Klasse Person

SEQUENZ DIAGRAM

Behaviour Diagramm

Akteuren oder Objekten die miteinander Kommunizieren.

Actor

Lifeline Creation

erstellt ein neues Objekt das temporär an der Kommunikation teilnimmt.

Lifeline Creation



Lifeline

jeder Teilnehmer hat eine Lebenslinie. Sie stellt den zeitlichen Verlauf der - von dieser Lebenslinien- gesendeten oder empfangenen Nachrichten dar.

Activation

zeigt an, wann ein Objekt aktiv ist, um Nachrichten zu senden oder zu empfangen.

Sie bleibt so lange aktiv, bis alle darin enthaltenen "Unteraktivierungen" (**Nested Messages**) andauern.

Message
Kommunikation zwischen Lifelines

synchron Message

Synchron Message

Sender wartet auf Antwort bevor er die aktuelle Nachricht abschließt und fortfährt.

Sender kann derweil andere Nachrichten bearbeiten. Ermäßigung kann ausgewählt werden während auf Antwort vom Tarifplan gewartet wird..

asynchrone Message

Asynchron Message

Sender schickt Nachricht ohne auf Antwort zu warten. Es können weitere Münzen eingeworfen werden.

* Message

*** (Iteration) Message**

Nachricht wird mehrmals wiederholt, bis Bedingung erfüllt ist. Betrag ist ausreichend.

Datenfluss

Datenfluss (Rückgabewert)

X (Lifeline Destruction)

Ende der Beteiligung des Objekts an der Interaktion

SRS / THE SOFTWARE REQUIREMENT SPECIFICATION

(User and SystemRequirements)

RUPP's METHODE

- **Prozess identifizieren** = nutze Prozesswörter (drucken, speichern, ..) = Das System speichert Daten

- **Aktivität charakterisieren** = unterscheide zwischen

Systemaktivität: Das System druckt

Benutzerinteraktion: Das System ermöglicht dem Nutzer, eine Datei zu speichern

Schnittstelle: Das System empfängt Daten von der Bibliothek.

- **Bestimme Grad der Verbindlichkeit:**

shall/must: Verpflichtend. = Das System **solll** die Datei speichern.

should: Wünschenswert, aber nicht zwingend

will/could: Absicht oder zukünftige Möglichkeit.

Won't = optional - wird im aktuellen Release nicht inkludiert

MOSCOW METHODE

funktional Anforderungen

(werden im **USE-CASE DIAGRAMM** dargestellt)

= beschreiben das Verhalten und die Struktur des Systems

nicht funktionale Anforderungen

Zuverlässigkeit, Reaktionszeit, Speicheranforderungen, Performance, Sicherheit, Skalierbarkeit, Wartbarkeit, etc.
= können oft mehrere funktionale Anforderungen auslösen
= sollten in messbaren Größen definiert werden.