Softwareanforderungsanalyse

Modellierung der Funktionen des Systems

Burkhardt Renz

Institut für SoftwareArchitektur der Technischen Hochschule Mittelhessen

Wintersemester 2015/16

Übersicht

- Aufgaben des Modells der Funktionen
- Sicht auf die Funktionalität
- Operationalisierung von Zielen
- Darstellung von Funktionen im Modell
- Heuristische Regeln

Modellierung der Funktionen des Systems

- Sicht auf die Funktionalität
  - Welche Dienste stellt das System bereit?
  - Wie tragen sie zum Erreichen von Zielen bei?
- Funktionale Sicht wird benötigt für
  - Beschreibung von Leistungen für Anwender, externe Systeme etc.
  - Verfolgbarkeit der Realisierung von Zielen in den Funktionen des Systems
  - Input für Design und Entwicklung
  - Basis für Definition von Testfällen, insbesondere für Black-Box-Tests
  - Basis für die Dokumentation für Benutzer
- Darstellung: Operationalisierungsdiagramm, das zeigt, wie Funktionen zum Erreichen von Zielen beitragen und Objekte im Objektmodell dazu als Input/Output verwenden

Übersicht

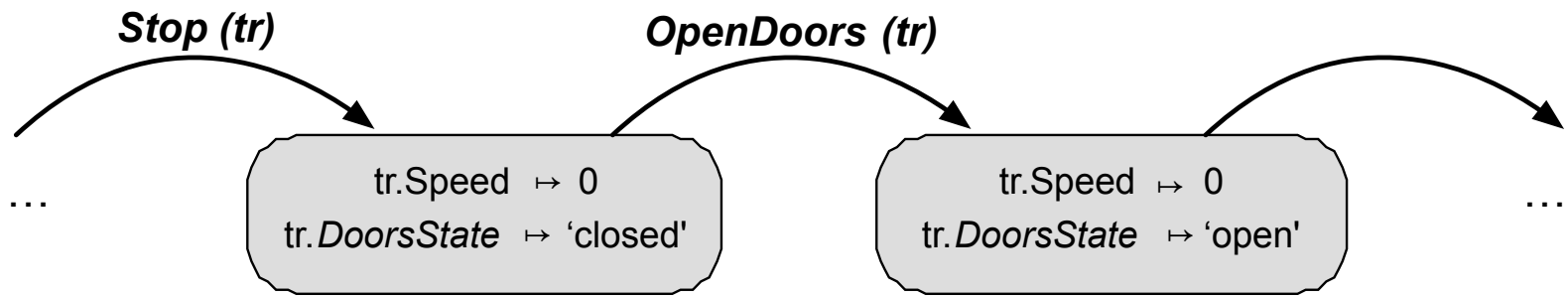
- Aufgaben des Modells der Funktionen
- Sicht auf die Funktionalität
  - Was sind Funktionen?
  - Charakterisierung von Funktionen
- Operationalisierung von Zielen
- Darstellung von Funktionen im Modell
- Heuristische Regeln

Was sind Funktionen?

- Funktionen verändern den Zustand des Systems
- Ihre Signatur besteht aus den Eingabevariablen und den Ausgabevariablen
- Man kann sie also auffassen als Menge von Paaren von Zuständen *vorher* – *nachher*

$$Fn \subseteq InputState \times OutputState$$

- Die Ausführung der Funktion entspricht dann genau einem Übergang von einem *InputState* in einen *OutputState*



Quelle: Lamsweerde S. 423

Eigenschaften von Funktionen

Eigenschaften

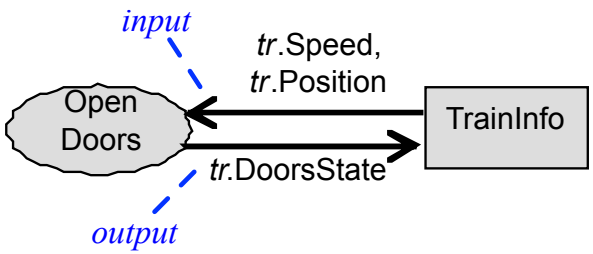
- Funktionen müssen Ziele des Zielemodells operationalisieren
- Funktionen sind im Modell atomar
- Akteure agieren nebenläufig und auch ein einzelner Akteur kann mehrere Funktionen parallel ausführen

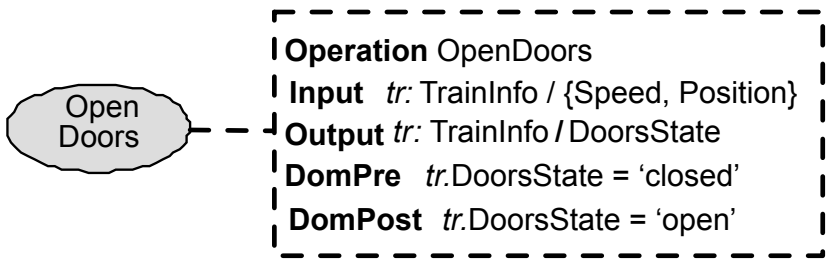
Kategorien von Funktionen

- Softwarefunktionen ausgeführt durch Softwarekomponenten des Systems, oft auch Dienste oder Services genannt
- Funktionen der Umgebung des Systems, ausgeführt durch menschliche Akteure, Geräte oder externe Softwaresysteme, oft auch Tasks genannt

Charakterisierung von Funktionen

- Annotierte Grundeigenschaften: *Name*, *Def*, *Category*
- Signatur
  textuell oder graphisch
- Vor- und Nachbedingungen in Bezug auf das Anwendungsgebiet
  beschreiben (*deskriptiv*) den Zustand des Systems vor und nach der Ausführung und Funktion





Quelle: Lamsweerde S. 425, 426

Ausführender der Funktion

- Ein Akteur führt eine Funktion aus, wenn Instanzen des Akteurs die Ausführung anstoßen können. Dazu muss gelten
- Ein ausführender Akteur muss die Eingabe- und Ausgabevariablen der Funktion beobachten bzw. steuern
- Eine Funktion kann nur von exakt einem Akteur (dessen Instanzen) ausgeführt werden

Übersicht

- Aufgaben des Modells der Funktionen
- Sicht auf die Funktionalität
- Operationalisierung von Zielen
  - Vor-, Nach- und Triggerbedingungen für die Zielerreichung
  - Verhalten von Akteuren
  - Operationalisierung von Zielen und Erfüllungsargument
- Darstellung von Funktionen im Modell
- Heuristische Regeln

Operationalisierung von Zielen

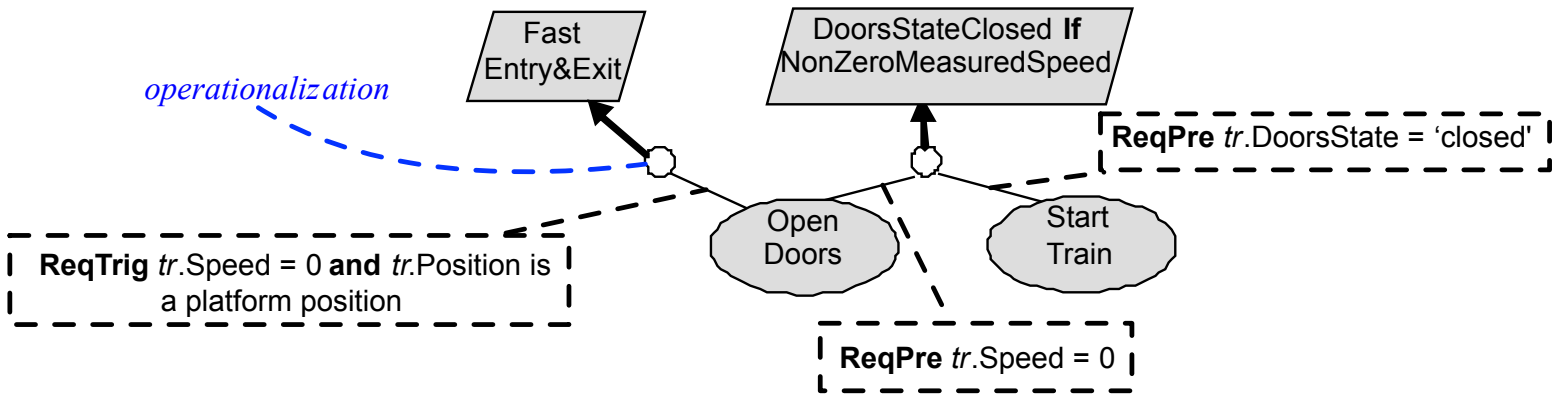
- Ziele wurden im Zielemodell so weit verfeinert, dass für ihre Erreichung genau ein Akteur verantwortlich ist
- Akteure können Funktionen ausführen
- Man spricht von der Operationalisierung eines Ziels um zu bezeichnen, dass die Funktionen, die der für das Ziel verantwortliche Akteur ausführen kann, dazu führen, dass das Ziel erreicht wird
- graphisch wird das dargestellt durch eine Verbindung zwischen dem Ziel und der Funktion

Präskriptive Bedingungen für Zielerreichung

Erforderliche Bedingungen (*required conditions*) sollen sicherstellen, dass das gewünschte Ziel erreicht wird

- `ReqPre`: Vorbedingung, die die Eingabevariablen erfüllen müssen, damit die Funktion ausgeführt werden *kann*
- `ReqTrig`: Triggerbedingung bedeutet, dass die Funktion ausgeführt werden *muss*, wenn die Bedingung eintritt
- `ReqPost`: gibt an, was nach der Ausführung der Funktion garantiert ist

Beispiel für Operationalisierung eines Ziels



Quelle: Lamsweerde S. 428

Das Beispiel textuell

```
Operation OpenDoors
  Def Software operation controlling the opening of
      all doors of a train;
  Input tr: TrainInfo;
  Output tr: TrainInfo/DoorsState;
  DomPre tr.DoorsState = 'closed';
  DomPost tr.DoorsState = 'open';
  ReqPre for DoorsStateClosedIfNonZeroMeasuredSpeed:
      tr.Speed = 0;
  ReqPre for SafeEntry&Exit:
      tr.Position is a platform position;
  ReqTrig for FastEntry&Exit:
      tr.Position is a platform position and tr.Speed = 0;
```

Verhalten von Akteuren

- Zur Erreichung der Ziele, für die ein Agent verantwortlich ist, muss er die Funktionen ausführen die das Ziel operationalisieren und zwar entsprechend der Vor-, Nach- und Triggerbedingungen
- Ein Akteur, der eine Funktion sofort ausführt, wenn alle ihre Vorbedingungen erfüllt sind, wird eifrig (*eager*) genannt
- Ein Akteur, der eine Funktion erst ausführt, wenn eine Triggerbedingung das verlangt, wird verzögert (*lazy*) genannt
- Mehrere Triggerbedingungen können im selben Systemzustand zutreffen – die dadurch ausgelösten Funktionen werden dann *nebenläufig* ausgeführt

Operationalisierung von Zielen und Erfüllungsargument

Ein Ziel G ist korrekt operationalisiert in die Funktionen $F_1, \ldots, F_n$ wenn die Spezifikationen der Funktionen $Spec(F_1), \ldots, Spec(F_n)$ folgendes erfüllen:

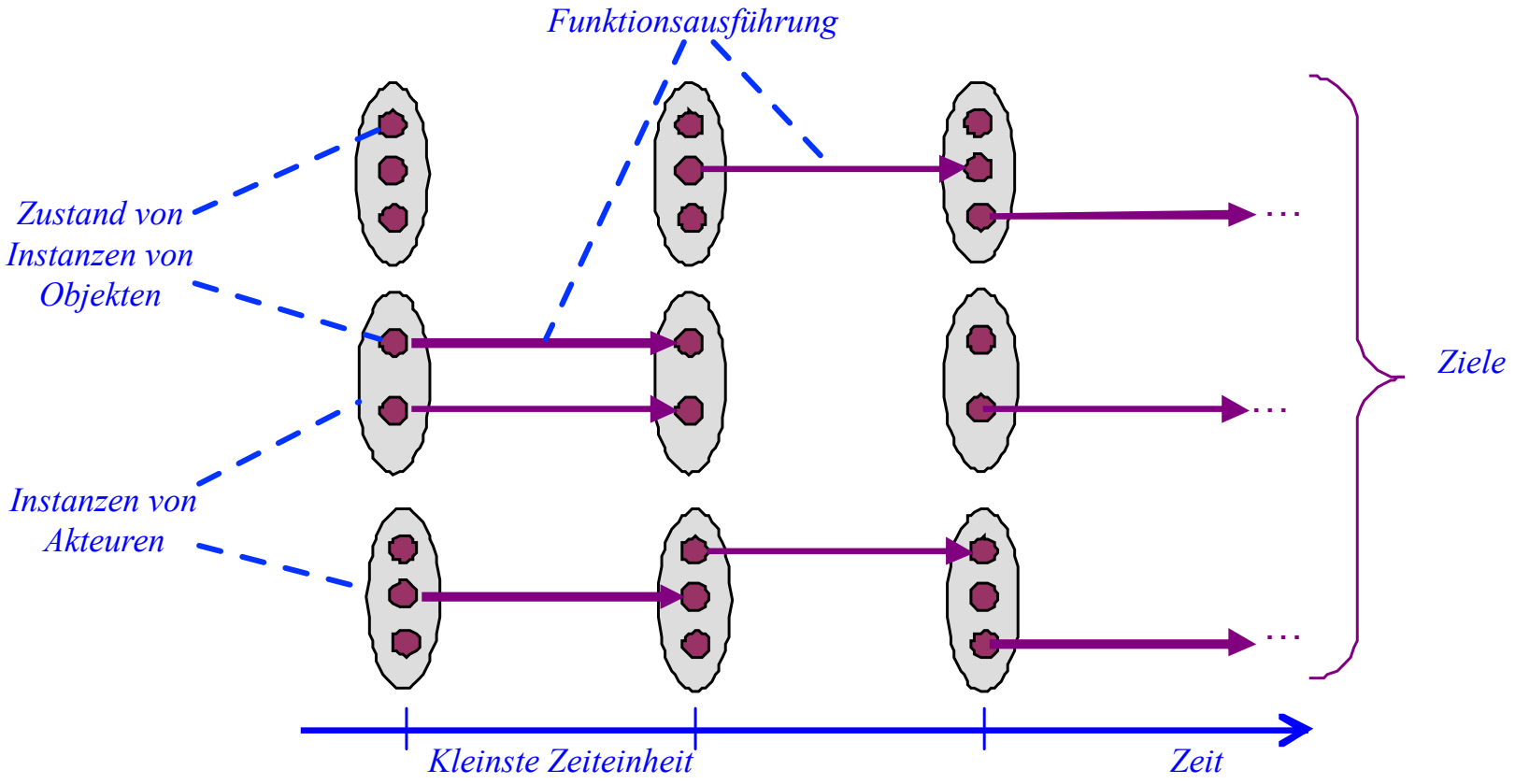
1. Vollständigkeit, d.h.
   $\{Spec(F_1), \ldots, Spec(F_n)\} \models G$
2. Konsistenz d.h.
   $\{Spec(F_1), \ldots, Spec(F_n)\} \not\models \bot$
3. Minimalität, d.h.
   Für jede echte Teilmenge $S \subset \{Spec(F_1), \ldots, Spec(F_n)\}$ gilt $S \not\models G$

Das Erfüllungsargument besteht im Nachweis, dass $\{Spec(F_1), \ldots, Spec(F_n)\} \models G$ gilt

Übersicht

- Aufgaben des Modells der Funktionen
- Sicht auf die Funktionalität
- Operationalisierung von Zielen
- Darstellung von Funktionen im Modell
  - Ziele, Akteure, Objekte und Funktionen
  - Operationalisierungsdiagramm
  - Das Anwendungsfalldiagramm der UML
- Heuristische Regeln

Wie hängen Ziele, Akteure, Objekte und Funktionen zusammen?



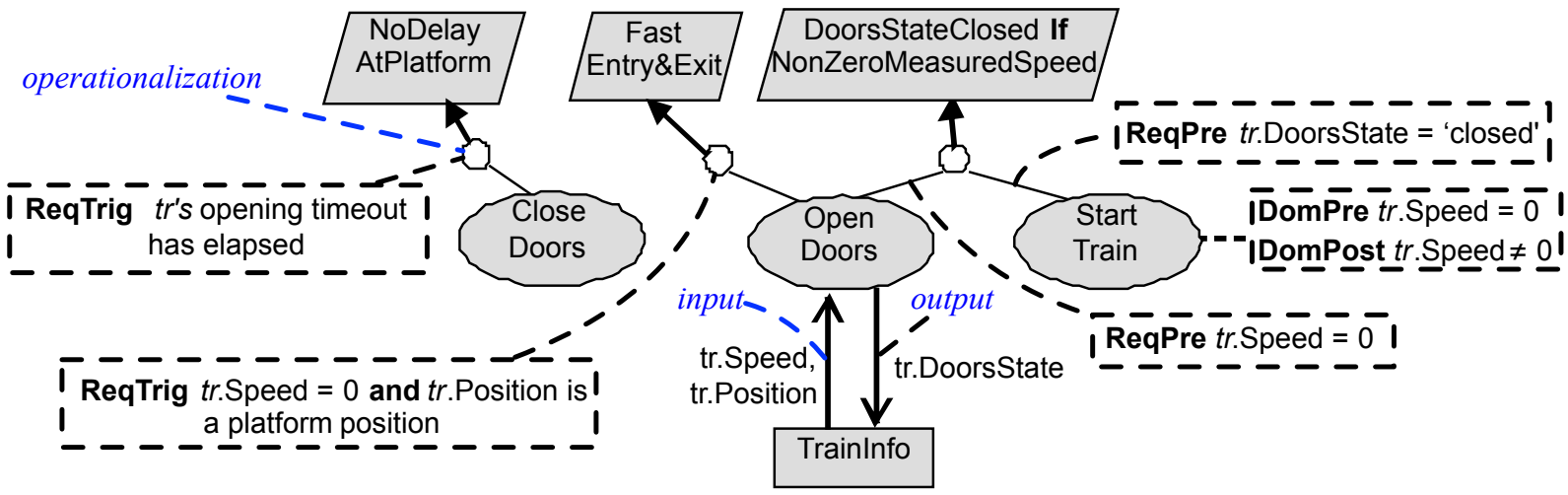
Quelle: Lamsweerde S.434

Operationalisierungsdiagramm

Das Operationalisierungsdiagramm zeigt in einem annotierten Graph, durch welche Funktionen Ziele erreicht werden:

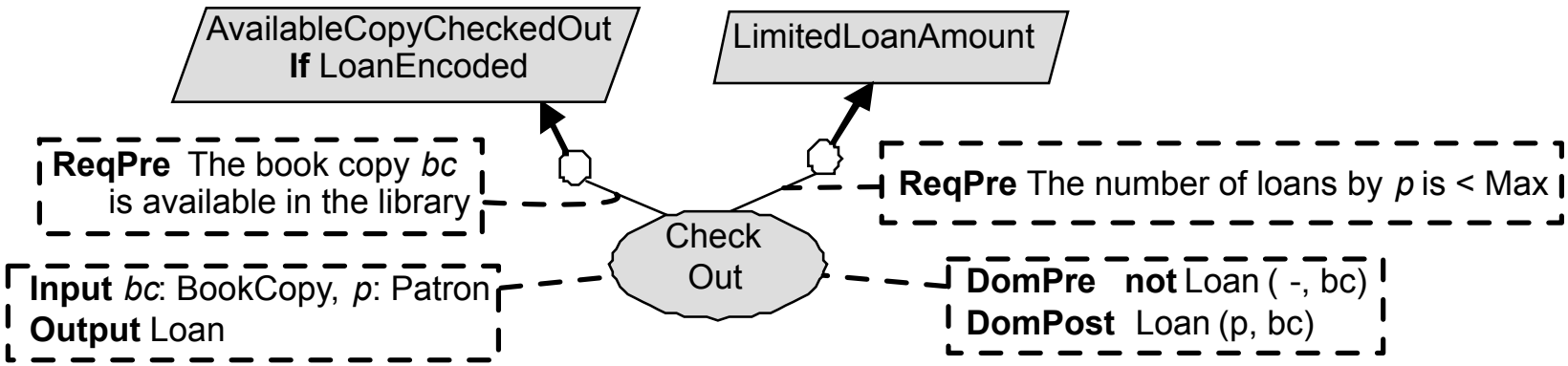
- Funktionen, Ziele und Objekte sind die Knoten
- Operationalisierung verbindet Funktionen und Ziele
- Input/Output-Beziehungen verbinden Funktionen mit Objekten

Operationalisierungsdiagramm – Beispiel Zugsteuerung



Quelle: Lamsweerde S.435

Operationalisierungsdiagramm – Beispiel Bibliothek



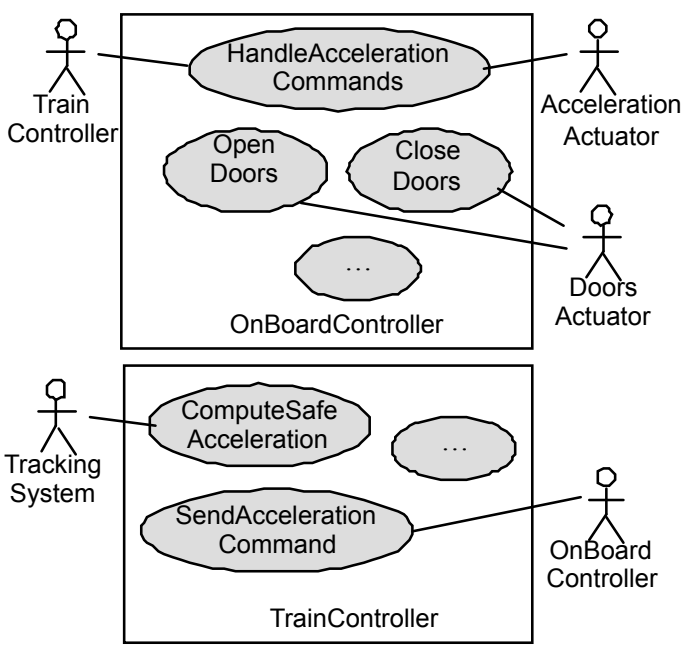
Quelle: Lamsweerde S.436

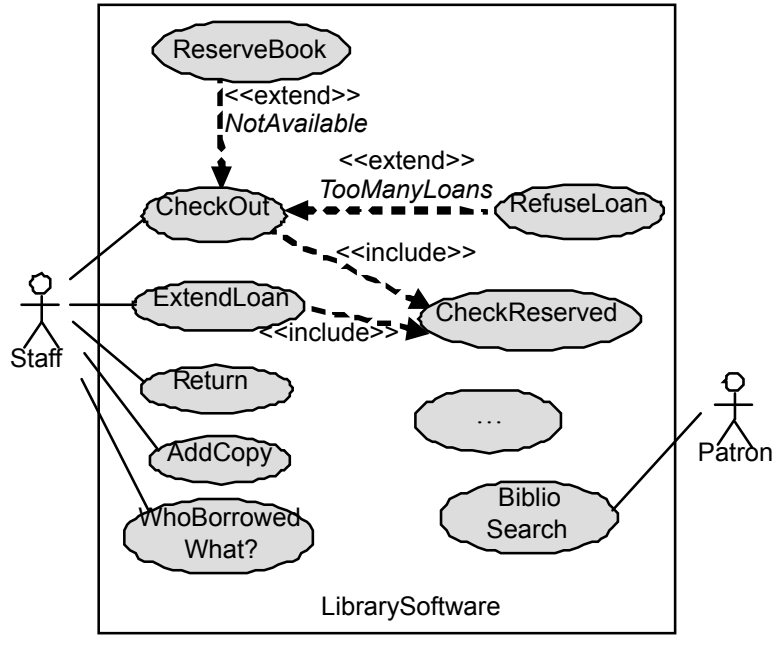
Das Anwendungsfalldiagramm der UML

Das Anwendungsfalldiagramm der UML kann aus den Operationalisierungsdiagrammen generiert werden:

- Für jeden Akteur sind alle Funktionen, die er ausführen kann seine Anwendungsfälle
- Andere Akteure sind beteiligt, wenn sie Attribute von Objekten beobachten oder steuern, die in der Ausführung der Funktion vorkommen

Funktionen und UML Anwendungsfalldiagramm – Beispiel





Quelle: Lamsweerde S.436

Übersicht

- Aufgaben des Modells der Funktionen
- Sicht auf die Funktionalität
- Operationalisierung von Zielen
- Darstellung von Funktionen im Modell
- Heuristische Regeln

Vorgehen beim Entwicklen der Operationalisierungsdiagramme

1. Funktionen identifizieren mit `DomPre` und `DomPost`
2. Identifizieren und Zuordnen der Input- und Output-Variablen
3. Präzise Spezifikation der Funktionen mit Vor-, Nach- und Triggerbedingungen
4. Operationalisierungsbeziehungen zu den Zielen im Zielemodell herstellen

Heuristiken

- Analyse von Zielen:
  Ist das Ziel vom Typ `Achieve` $\rightarrow$ Funktion, die den gewünschten Zustand erzeugt
  Ist das Ziel vom Type `Maintain` $\rightarrow$ Funktion, die die Bedingungen sicherstellt
- Szenarien durchspielen und daraus Funktionen entwickeln
- Zusammenhang zwischen Zielen, Objekten und Akteuren untersuchen, um die verbindenden Funktionen zu finden