Softwareanforderungsanalyse

Spezifizieren und Dokumentieren von Anforderungen

Burkhardt Renz

Institut für SoftwareArchitektur der Technischen Hochschule Mittelhessen

Wintersemester 2015/16

Übersicht

Was spezifizieren und dokumentieren?

- Gliederung für Anforderungsspezifikation
- Problem Frames

Wie spezifizieren und dokumentieren?

- Natürliche Sprache
- Modellbasierte Darstellung
- Formale Spezifikation

Übersicht

- Was spezifizieren und dokumentieren?
  - Aufbau und Gliederung der Anforderungsspezifikation
  - Problem Frames
- Spezifikation in natürlicher Sprache
- Modellbasierte Spezifikationen
- Formale Spezifikationen

Gliederung der Anforderungsspezifikation

- ISO/IEC/IEEE Standard 29148:2011
  Systems and software engineering – Life cycle processes – Requirements engineering
  enthält in Kapitel 8.4 Beispiel für die Gliederung des Software requirements specification document
- Im deutschen Sprachraum oft Unterscheidung
  - Lastenheft
    „Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragsnehmers innerhalb eines Auftrags“
  - Pflichtenheft
    „. . . vom Auftragnehmer erarbeitete Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenhefts“

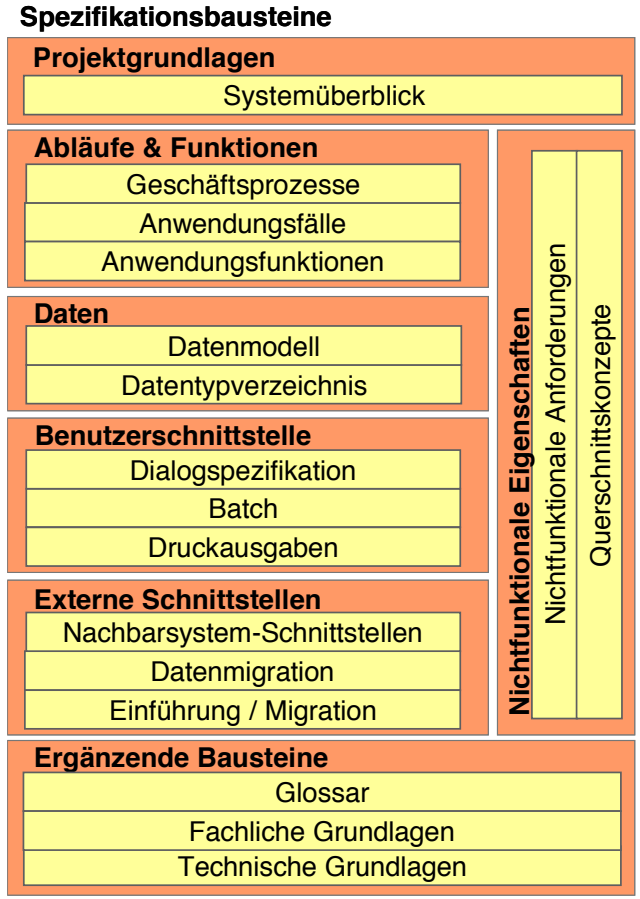
Mustergliederung nach ISO 29148:2011 I

1. Einführung
   - 1.1 Anlass und Ziele
   - 1.2 Einsatzbereich
   - 1.3 Produktübersicht
     - 1.3.1 Kontext
     - 1.3.2 Funktionen des Produkts
     - 1.3.3 Art der Benutzer
     - 1.3.4 Annahmen und Einschränkungen
2. Referenzen
3. Einzelanforderungen
   - 3.1 Externe Schnittstellen
   - 3.2 Funktionen
   - 3.3 Anforderungen an die Benutzbarkeit
   - 3.4 Anforderungen an die Leistungsfähigkeit
   - 3.5 Anforderungen bzgl. des logische Datenmodells
   - 3.6 Entwurfsbedingungen und -einschränkungen
   - 3.7 Weitere Qualitätsmerkmale
   - 3.8 Wartungs- und Supportinformationen

Mustergliederung nach ISO 29148:2011 II

4. Verifikation (bzgl. aller Unterpunkte von 3.)
5. Anhang
   - 5.1 Annahmen und Abhängigkeiten
   - 5.2 Akronyme und Abkürzungen

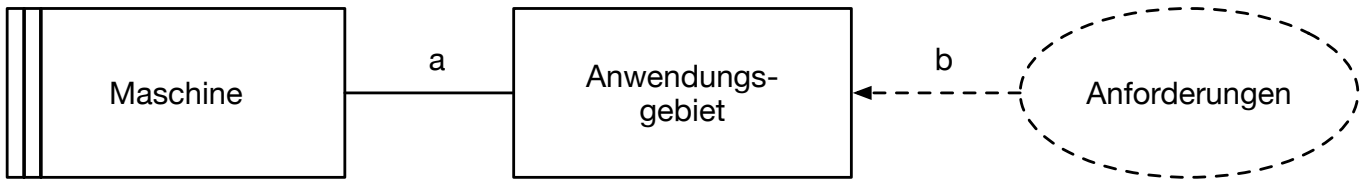
Spezifikationsbausteine von sd&m



Quelle: Andreas Birk *Anforderungsspezifikationen in großen IT-Projekten*, in: Jahrestagung der GI-Fachgruppe Requirements Engineering, Kaiserslautern 2004.

Problem Frames

Die Grundidee



Grundlegende Problem Frames

`http://homepages.thm.de/~hg11260/mat/saa-pf-bh.pdf`

Übersicht

- Was spezifizieren und dokumentieren?
- Spezifikation in natürlicher Sprache
  - Sprachliche Regeln
  - Blaupausen für die Formulierung von Anforderungen
  - Glossar
- Modellbasierte Spezifikationen
- Formale Spezifikationen

Spezifikation in natürlicher Sprache

- wird am häufigsten für Spezifikation von Anforderungen verwendet, warum?
- ausdrucksmächtig
- für jeden ohne Spezialkenntnisse schreibbar und lesbar
- aber:
  - inhärent mehrdeutig
  - fehlerträchtig
  - schwierig zu prüfen
  - je umfangreicher desto unübersichtlicher

Verbesserung der Qualität natürlichsprachlicher Spezifikationen

Man kann die Qualität von Anforderungsspezifikationen in natürlicher Sprache verbessern durch

- geeignete Gliederung und Strukturierung des Dokuments
- Sprachliche Regeln für Formulierungen bis hin zu
- Blaupausen für die Formulierung von Einzelanforderungen
- klare Definition und Verwendung von Bezeichnungen durch ein Glossar

Sprachliche Regeln: Struktur der Sätze

- Vollständige Satzstruktur bilden
- Im Aktiv formulieren
- Anforderungen in Hauptsätzen, Nebensätze nur zur Erläuterung etc.
- Bei Vergleichen Bezugspunkt angeben
- Bei Alternativen und/oder Fallunterscheidungen alle Möglichkeiten berücksichtigen

Sprachliche Regeln: Vage Bezeichnungen vermeiden

- Unspezifische Nomen durch präzise Angaben ersetzen
  nicht: „die Daten werden. . . “
  sondern: „die Daten des aktuellen Auftrags. . . “
- Verben, die Prozesse, Funktionen oder Abläufe beschreiben, präzise definieren
  „Daten werden übertragen“
  welche Daten? wohin übertragen?

Sprachliche Regeln: Nominalkonstruktionen

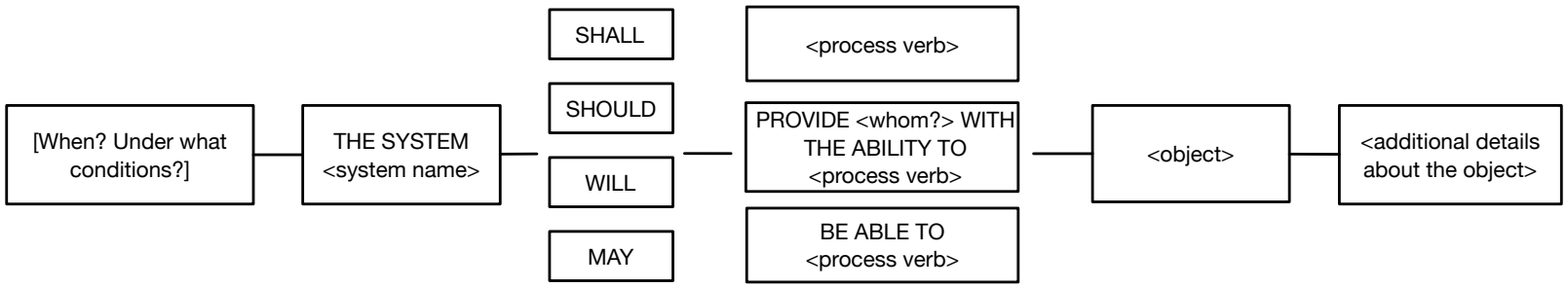
- Nominalkonstruktionen sind Nomen, die aus Verben gebildet werden
  z.B. „Initialisierung, Neustarten des Systems“
- Oft verbergen sich hinter Nominalkonstruktionen unvollständig spezifizierte Abläufe

Sprachliche Regeln: Quantoren und Ausschlüsse

- Allquantoren hinterfragen, weil sie in Umgangssprache oft implizit kontextabhängig verwendet werden
  z.B. „auf dem Dialog werden alle Daten angezeigt“
  z.B. „der Vorgang kann jederzeit abgebrochen werden“
- Existenzquantoren durch explizites Angeben des Exemplars präzise verwenden
  z.B. „Es gibt eine Einstellung für die Schriftgröße“
  welche? wo?
- Ausschlüsse nach Ausnahmen hinterfragen
  z.B. „Es ist nicht möglich, dass. . . “
  wirklich?

Blaupausen für die Formulierung von Anforderungen

Vorlage für Bildung präziser Formulierung von Anforderungen



Quelle: Klaus Pohl, Chris Rupp *Requirements Engineering Fundamentals*, Kapitel 5.2

Modus von Äußerungen

Michael Jackson kritisiert den Einsatz von Zeitformen (Tempus) wie in „shall“ und „will“, weil Zeiten keinen Modus angeben, sondern eher zweideutig sind.

Besser die präzise Unterscheidung:

- Optativ – Wunschform = drückt aus, was gewünscht wird zu erreichen, was *vorgeschrieben* wird, präskriptiv
- Indikativ – Wirklichkeitsform = drückt aus, was im Anwendungsgebiet *ist*, unabhängig von dem zu konstruierenden System, *beschreibend*, deskriptiv
- Definition – Festlegung einer präzisen Sprechweise für Konzepte etc.

Glossar

- Verzeichnis der Begriffe und Bezeichnungen des Anwendungsgebiets
- Präzise Definition der Begrifflichkeiten – hilft Mehrdeutigkeiten zu vermeiden
- erfordert genaues Verstehen des Anwendungsgebiets
- Basis der Kommunikation zwischen Fachexperten und Softwareentwicklern
- Regeln:
  - Hintergrund der Begriffsbildung festhalten
  - Agreement bei allen Beteiligten herstellen
  - Konsistente Struktur festlegen
  - Zentral zugänglich
  - Über die Laufzeit des Projekts gepflegt
  - Verwendung der Begriffe des Glossars ist verpflichtend

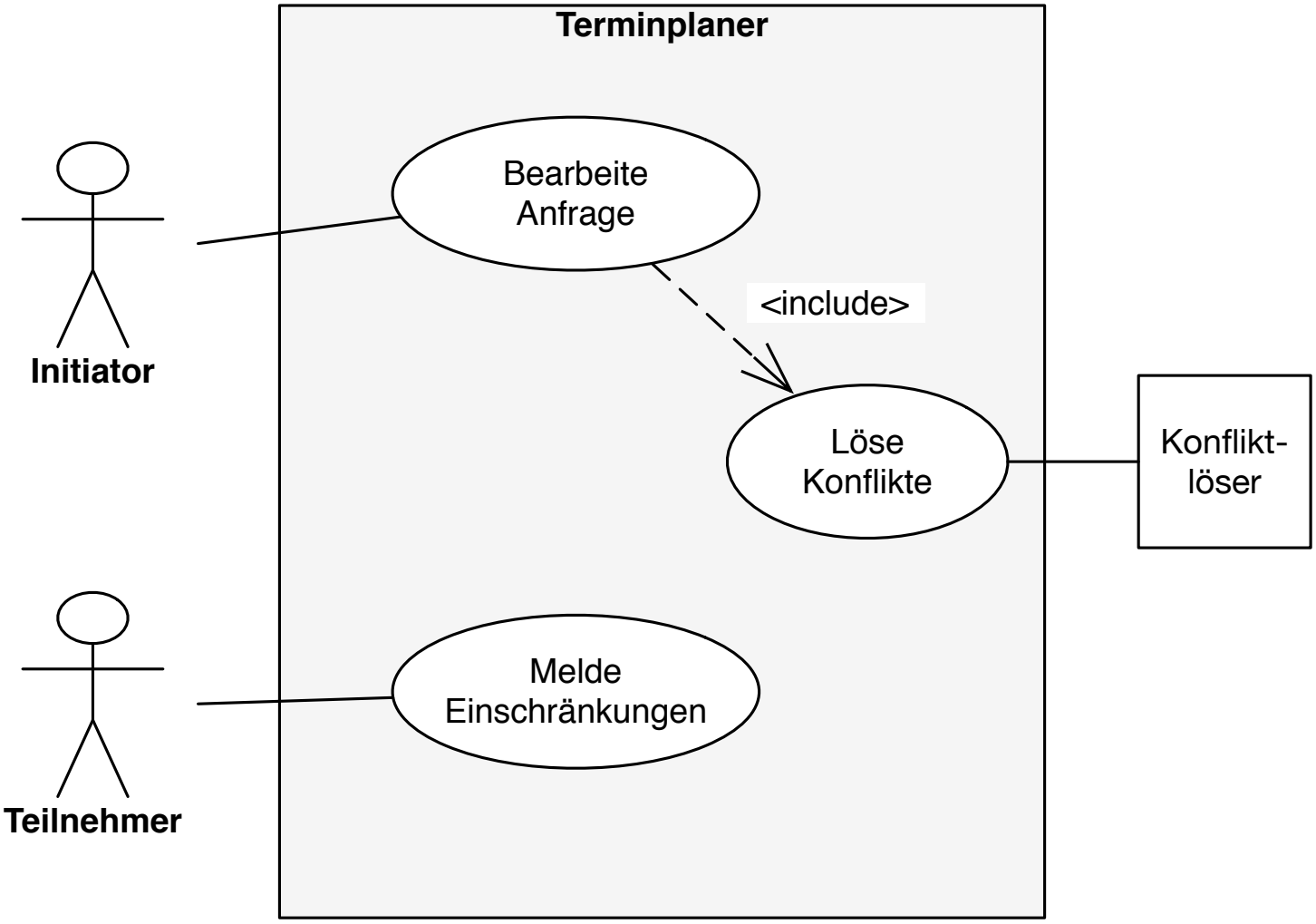
Übersicht

- Was spezifizieren und dokumentieren?
- Spezifikation in natürlicher Sprache
- Modellbasierte Spezifikationen
  - Anwendungsfälle
  - Drei Perspektiven
  - Datenperspektive
  - Perspektive der Funktionalität
  - Perspektive des Verhaltens
- Formale Spezifikationen

Use-Case-Diagramm

- Anwendungsfall: Interaktionssequenz eines Akteurs mit dem System zur Erreichung eines Zieles
- Use-Case-Diagramm: Übersicht über die Anwendungsfälle
- und ihre Beziehungen zur Systemumgebung
- sowie untereinander
- Nicht viel mehr als eine graphisch dargestellte Liste der Anwendungsfälle

Beispiel Use-Case-Diagramm



Spezifikation des Anwendungsfalls

- Oft werden Blaupausen für textuelle Darstellung verwendet, z.B. Alistair Cockburn: *Writing Effective Use Cases*
- Aktivitätendiagramm zur Darstellung der Interaktionen
- Sequenzdiagramm zur Darstellung der Interaktionen

Kritik der Ansatzes mit Anwendungsfällen

Use cases are a popular albeit fairly fuzzy form of operational specification. As their specification does not convey much, use cases are not really amenable to useful forms of analysis.
However, they provide an outline view of the operations that an agent has to perform; such a view may prove useful for elicitation and communication with stakeholders.

– van Lamsweerde S. 277

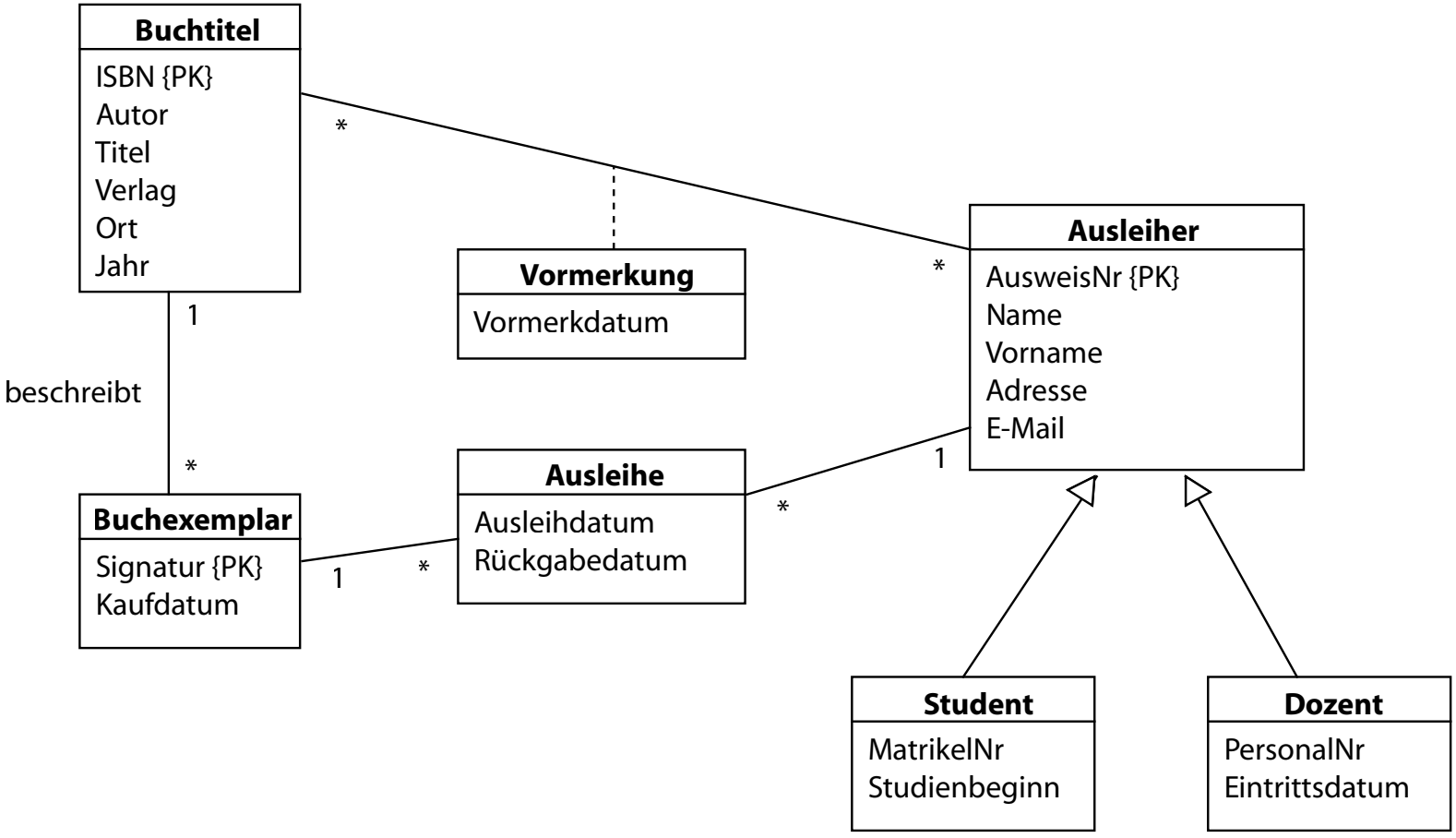
Drei Perspektiven für Anforderungen

- Datenperspektive
  Welche Daten benötigt man für die Aufgabe?
  Wie hängen sie zusammen?
  Welche Daten speichert und/oder liefert das System?
- Perspektive der Funktionalität
  Welche Transformation von Daten macht das System?
  Welche Aktionen in der Umgebung löst es aus?
- Perspektive des Verhaltens
  Welche Zustände hat das System?
  Wie reagiert es auf Ereignisse in bestimmten Zuständen?
  Welche Zustandsübergänge macht es dann?

Entity-Relationship-Modellierung

- Entitäten und Entitätstypen
- Attribute
- Beziehungen und Assoziationen (Beziehungstypen)
- Multiplizitäten
- Diagrammdarstellung in Chen-Notation oder UML-Notation

Beispiel: Bibliotheksverwaltung



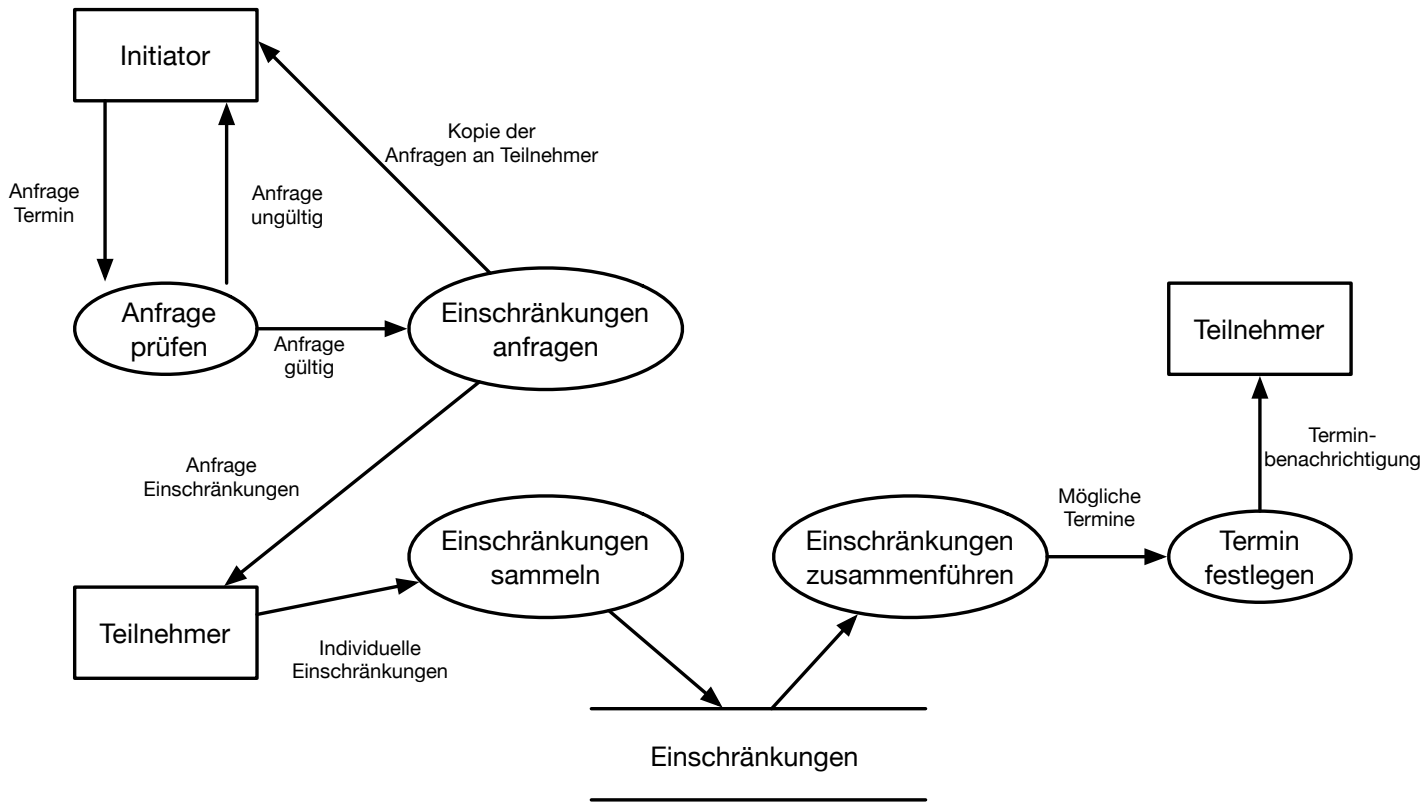
UML-Klassendiagramm

- Weiterentwicklung der Entity-Relationship-Modellierung
- im Wesentlichen kommen Methoden hinzu
- zugeschnitten (und basierend) auf objekt-orientierte Programmiersprachen – etwas andere Semantik als ER-Modelle

Strukturierte Analyse

- Spezifikationsmethode aus den 80er Jahren
- Aktivitätsdiagramm: Prozesse/Funktionen mit Input- und Output-Daten
- Datenflussdiagramm:
  - Datenspeicher
    dargestellt durch parallele Linien
  - Datenfluss
    dargestellt durch Pfeil
  - Prozess/Funktion
    dargestellt durch Oval
  - Akteur
    dargestellt durch Rechteck
- wird heute nicht mehr viel verwendet

Beispiel: Terminplaner



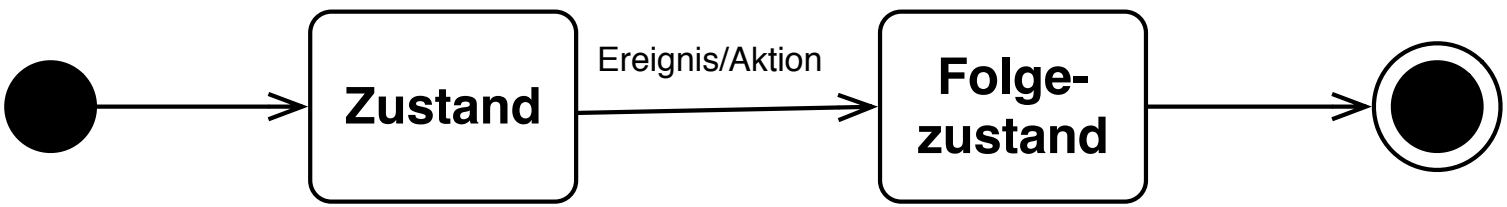
UML-Diagramme

- Aktivitätendiagramm
  Aktivitäten, Kontrollfluss, Entscheidungsknoten,
  Synchronisation nebenläufiger Ausführung, ...
- Sequenz- bzw. Interaktionsdiagramm
  Beteiligte Rollen, Lebenslinie, Nachrichten/Methodenaufruf,
  ...

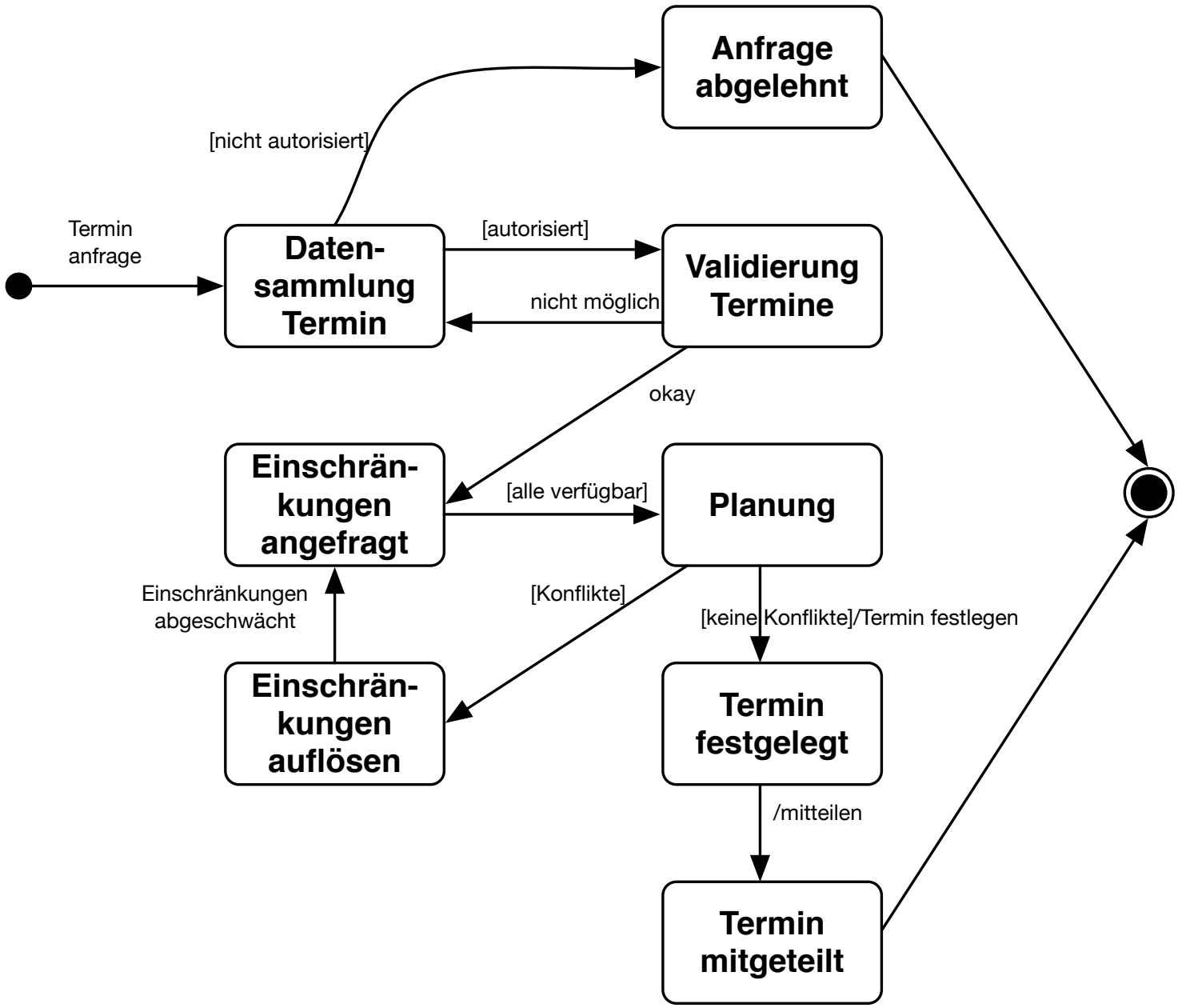
Statecharts bzw. UML-Zustandsdiagramm

- Statechart von David Harel 1984
- Zustandsdiagramm der UML

Prinzip



Beispiel: Terminplaner



Übersicht

- Was spezifizieren und dokumentieren?
- Spezifikation in natürlicher Sprache
- Modellbasierte Spezifikationen
- Formale Spezifikationen
  - Logik
  - Beispiel: Z
  - Beispiel: Alloy
  - Diskussion formaler Methoden

Grundlagen formaler Spezifikationen

- Aussagenlogik
  Atomare Aussagen, die wahr oder falsch sein können
  Komplexe Aussagen gebildet durch $\neg$, $\wedge, \vee, \rightarrow, \ldots$
- Prädikatenlogik
  Terme bezeichnen Objekte des Universums, Funktionen
  Prädikate sind Beziehungen der Objekte, die wahr oder falsch sein können
  Aussage gebildet durch aussagenlogische Operatoren sowie die Quantoren über Variablen $\forall\, x, \exists\, y$
- Temporale Logik
  Folge von Zuständen, in denen bestimmte atomare Aussagen wahr oder falsch sein können
  Temporale Operatoren $\Diamond$ später mal, $\Box$ immer, $\bigcirc$ im nächsten Zustand

Die Spezifikationssprache Z

- Z ist eine Sprache zur formalen Beschreibung mathematischer Sachverhalte
- dient der Beschreibung von Software und auch Hardware
- entwickelt an der Universität Oxford von Jean-Raymond Abrial
- 2002 durch ISO als Standard 13568 standardisiert
- Name kommmt von Ernst Zermelo, Axiome der Mengenlehre nach Zermelo-Fraenkel (ZF, ZFC)
- in der Praxis verbreiteste formale Spezifikationssprache
- Literatur: Jonathan Jacky *The Way of Z: Practical Programming with Formal Methods*, CUP 1997

Grundelemente von Z, 1

- Eine Z-Spezifikation besteht aus Mengen, Typen, Axiomen und Schemata
- Typen, wie $[Datum], \mathbb{N}$
- Mengen haben einen Typ, wie $Personen : \mathbb{P}\ Name$, $Zaehler : \mathbb{N}$
- Axiome definieren globale Variablen und Invarianten, wie

$$\begin{array}{|l} limit : \mathbb{N} \\ \hline limit \leq 65535 \end{array}$$

Grundelemente von Z, 2

Schemata bilden einen eigenen Namensraum und gliedern die Spezifikation, sie bestehen aus:

- Name
- Deklaration von Zustandsvariablen
- Invarianten
- Beziehungen
- Operationen, die Zustand ändern

Beispiel: Bibliotheksverwaltung

Bestand an Büchern

Bibliothek

$$
\begin{array}{l}
Bestand : \mathbb{P}\, Buch \\
Benutzer : \mathbb{P}\, Person \\
ausgeliehen : Buch \pfun Person \\
\hline
\mathrm{dom}\, ausgeliehen \subseteq Bestand \\
\mathrm{ran}\, ausgeliehen \subseteq Benutzer
\end{array}
$$

Beispiel: Bibliotheksverwaltung

Ausleihen von Büchern

Ausleihen

$\Delta Bibliothek$

$auszuleihendesBuch? : Buch$

$Ausleiher? : Person$

$auszuleihendesBuch? \in Bestand \setminus \mathrm{dom}\ ausgeliehen$

$Ausleiher? \in Benutzer$

$ausgeliehen' = ausgeliehen \cup \{(auszuleihendesBuch?, Ausleiher?)\}$

$Bestand' = Bestand$

$Benutzer' = Benutzer$

Alloy

- Spezifikationssprache entwickelt von Daniel Jackson MIT
- basiert auf Mengen, Relationen und relationaler Logik
- hat interaktiven Alloy-Analyzer, der Modelle zur Spezifikation findet und so interaktives Entwickeln einer Spezifikation erlaubt
- Alloy = Legierung, weil Verschmelzung von Z mit Model Checking
- verwendet oft für (kritische) Ausschnitte einer Spezifikation
- Literatur: Daniel Jackson *Software Abstractions: Logic, Language, and Analysis*, MIT Press 2012
- Literatur: Burkhardt Renz und Nils Asmussen *Kurze Einführung in Alloy*, THM 2014

Beispiel mit Alloy

Demo Zugsegmente und Sicherheitsrichtlinie (nach D. Jackson)

Diskussion: formale Methoden

Stärken

- Immer eindeutig (da Semantik formal definiert)
- Konsistenz formal prüfbar
- Erfüllung wichtiger Eigenschaften beweisbar und/oder automatisiert testbar
- Formale Verifikation von Programmen/Code möglich
- Modelle simulierbar/animierbar, z.B. Alloy

Schwächen

- sehr aufwändig
- Nachweis der Vollständigkeit schwierig
- Große Spezifikationen schwer zu verstehen, profunde Ausbildung notwendig
- Aspekte von Benutzerschnittstellen schwierig darstellbar