Formale Modellierung in der Softwareentwicklung

Modul 9 (v1.0)

Kanonikvorlesung: Foundations of Computing

Heiko Mantel

MAIS, TU Darmstadt, WS10/11

Motivation

Was haben Sie in den Modulen 0, 2 und 3 erlernt?

- Modellierung: Szenario und Anforderungen an dieses Szenario
- Methodik: formale Modellierung mit mathematischen Konzepten
- Szenario: Denksportaufgabe
- Szenario: Arbeitskreise der EU

Informationssysteme spielten in diesen Szenarien keine Rolle

Was haben Sie in den Modulen 5, 6, 7, 8 erlernt?

- Szenario: Semantik einer imperativen Programmiersprache

Was hat formale Modellierung mit Softwareentwicklung zu tun?

- Dieser Zusammenhang wird in diesem Modul herausgearbeitet.

Übersicht: Modul 9

Software Engineering

- Einführung
- Software Prozesse
- Aktivitäten
- Prozessmodelle

Formale Modellierung

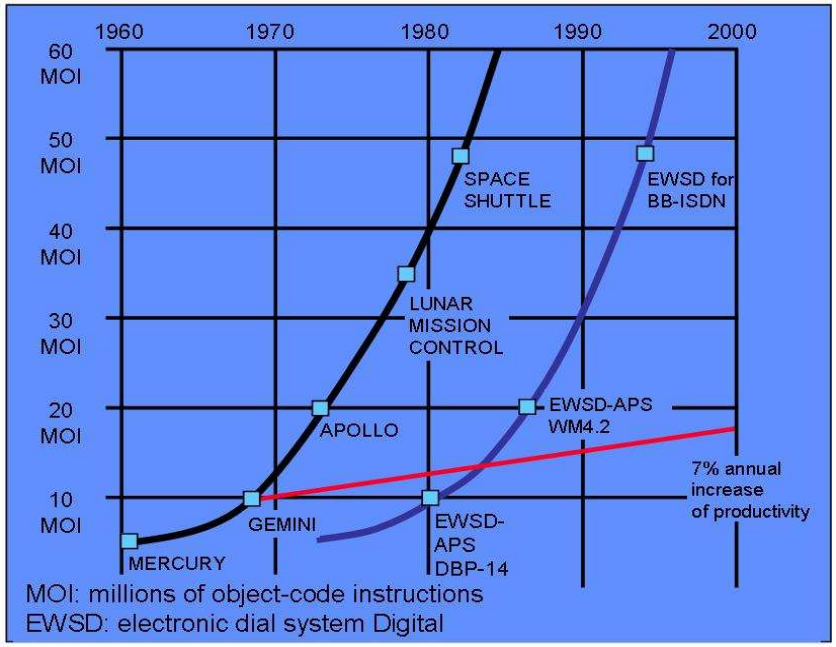
- Modellierung
- Spezifikationssprachen
- Formale Softwareentwicklung
- Formale Verifikation
- Werkzeugunterstützung

Evaluierung

Software Engineering (1)

Warum braucht man Software Engineering?

Anstieg der Größe und Komplexität von Programmen



Quelle: H Balzert. Lehrbuch der Software-Technik, 2. Auflage, Spektrum, 2000.

Beispiele:

- Software im amerikanischen Raumfahrtprogramm (schwarz)
- Software in Siemens Telefonanlagen (blau)

Software Engineering (2)

Was ist Software eigentlich?

Software beinhaltet Programme und dazugehörige Dokumentation.

Was ist Software Engineering eigentlich?

Software Engineering ist eine Ingenieurdisziplin, die sich mit allen Aspekten der Softwareerstellung beschäftigt.

- beinhaltet die Programmierung
- beinhaltet auch das Erstellen der Dokumentation

Welche Aspekte bestimmen die Qualität von Software?

- funktionale Aspekte, z.B. Ein-/Ausgabeverhalten
- nicht funktionale Aspekte, z.B. Benutzbarkeit, Effizienz, Sicherheit, Wartbarkeit und Zuverlässigkeit

"Software engineers shall ensure that their products and related modifications meet the highest professional standards possible." **[Software Engineering Code of Ethics and Professional Practice]**

Software Engineering (3)

Was ist mit Dokumentation gemeint?

Es gibt zwei Arten von Dokumentation:

- **Benutzerdokumentation, z.B.**
  - Beschreibung der Funktionalität
  - Anleitung zum Verwenden des Programms
  - Beschreibung der Systemvoraussetzungen
- **Entwicklungsdokumentation, z.B.**
  - Beschreibung der Systemanforderungen
    - Lastenheft und Pflichtenheft
  - Protokollierung von Entwurfsentscheidungen
  - Beschreibung der internen Funktionsweise des Systems
  - Beschreibung der Systemarchitektur
  - Testfälle und Testergebnisse

Die Aufgabe von Softwareentwicklern besteht zu einem großen Teil in der Erstellung von Entwicklungsdokumentation.

Software Engineering (4)

Beispiele für Unterschiede zwischen Programmierung und SE

- Die Anforderungen werden früh in der Entwicklung bestimmt.
  - detaillierte und methodische Anforderungsanalyse
  - präzise Dokumentation aller Anforderungen in einer möglichst präzisen und verständlichen Form
    - Spätere Änderungen der Anforderungen müssen erkennbar sein.
- Die Komplexität wird reduziert.
  - Verwendung von Teile-und-herrsche und schrittweiser Verfeinerung
  - Dokumentation aller Entwicklungsschritte und -entscheidungen
- Die Anforderungen werden sorgfältig und methodisch validiert
  - Unterscheidung von Validierung und Verifikation (spätere Folie)
  - Verwendung unterschiedlich aufwendiger Methoden je nach Kritikalität des Systems oder der Systemkomponente

Software-Prozesse

Was ist ein Software-Prozess?

- eine Menge von Aktivitäten deren Ziel ist, Software zu entwickeln

Beispiele für Aktivitäten

- Spezifikation, Programmentwicklung, Validierung und Verifikation

Was ist ein Prozessmodell?

- Ein Prozessmodell bietet eine vereinfachte Sicht auf einen Software-Prozess, z.B. aus einer bestimmten Perspektive.
- Ein Prozessmodell kann Aktivitäten verfeinern.
- Ein Prozessmodell kann Aktivitäten in eine Reihenfolge bringen.

Beispiele für Prozessmodelle

- Wasserfallmodell, V-Modell, Spiralmodell

Die Qualität eines Produkts wird zu einem großen Teil durch den Prozess bestimmt, mit dem das Produkt entwickelt wird.

Aktivität: Spezifikation (1)

Relevante Aspekte bei der Spezifikation eines IT Systems

- Welche Dienste sollten vom System angeboten werden?
- Unter welchen Rahmenbedingungen soll das System arbeiten?
  - z.B. beschränkte Ressourcen auf mobilen Geräten
- Welche Randbedingungen sind bei der Entwicklung zu beachten?
  - z.B. begrenztes Budget für die Entwicklung

Machbarkeitsstudien und Marktanalysen

- Sind die Anforderungen durch verfügbare Technologie erfüllbar?
- Wird sich die Entwicklung des Systems wirtschaftlich rentieren?

Anforderungsanalyse

- Bestimmen der Anforderungen, z.B. durch Gespräche mit Kunden
- Einsatz von Prototypen und existierenden Systemen ist hilfreich

Anforderungsdefinition

- Die Anforderungen werden in Dokumenten festgehalten.

Aktivität: Spezifikation (2)

Validierung der Anforderungen beinhaltet z.B.

- Sind die Anforderungen präzise beschrieben?
- Sind die Anforderungen miteinander konsistent?
- Sind die Anforderungen vollständig?
- Sind die Anforderungen realistisch?

Modelle werden bei der Spezifikation eingesetzt, um z.B.

- **die Anforderungen zu definieren oder**
- **die Systemumgebung zu beschreiben.**

Aktivität: Programmentwicklung

Relevante Aspekte bei der Programmentwicklung sind z.B.

- Entwurf und Implementierung

Architekturentwurf

- Spezifikation der Komponenten des Systems
  - Spezifikation der Schnittstellen und der angebotenen Dienste
- Spezifikation der Beziehungen zwischen den Komponenten

Entwurf der Algorithmen und Datenstrukturen

- Umsetzung in der Implementierung

Entwurf der Komponenten

- Das Vorgehen entspricht dem Entwurf des Gesamtsystems.

Modelle können z.B. zur Beschreibung der Architektur, der Datenstrukturen und der Algorithmen eingesetzt werden.

Formale Modellierung und Verifikation von Beziehungen zwischen Modellen ist möglich.

Aktivität: Validierung und Verifikation

Was ist der Unterschied zwischen Validierung und Verifikation?

Validierung

- Entspricht das IT System den Wünschen?
- lässt sich nicht vollständig formal durchführen

Verifikation

- Erfüllt das IT System die Spezifikation?

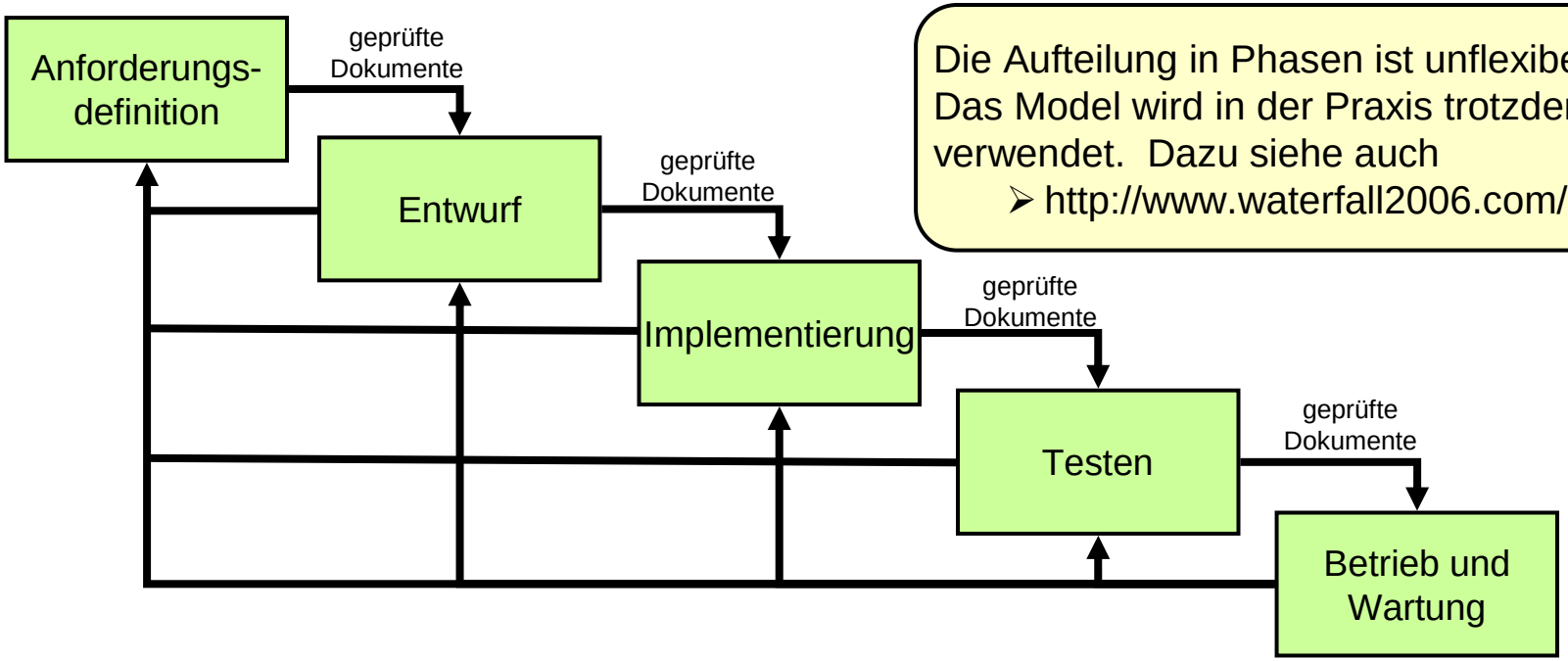
Techniken zur Verifikation

- Manuelle Prüfmethoden, z.B. Inspektionen oder Walkthrough
- Statische Analyse, z.B. Typüberprüfung oder formale Verifikation
- Testen, z.B. kontrollfluss- oder datenflussorienterte Verfahren

Formale Modelle bilden die Basis z.B. für

- **die automatische Generierung von Testfällen**
- **eine formale Verifikation durch mathematische Beweise**
  - **Erfüllt die Implementierung die formale Spezifikation?**

Prozessmodell: das Wasserfallmodel



Die Aufteilung in Phasen ist unflexibel. Das Model wird in der Praxis trotzdem verwendet. Dazu siehe auch

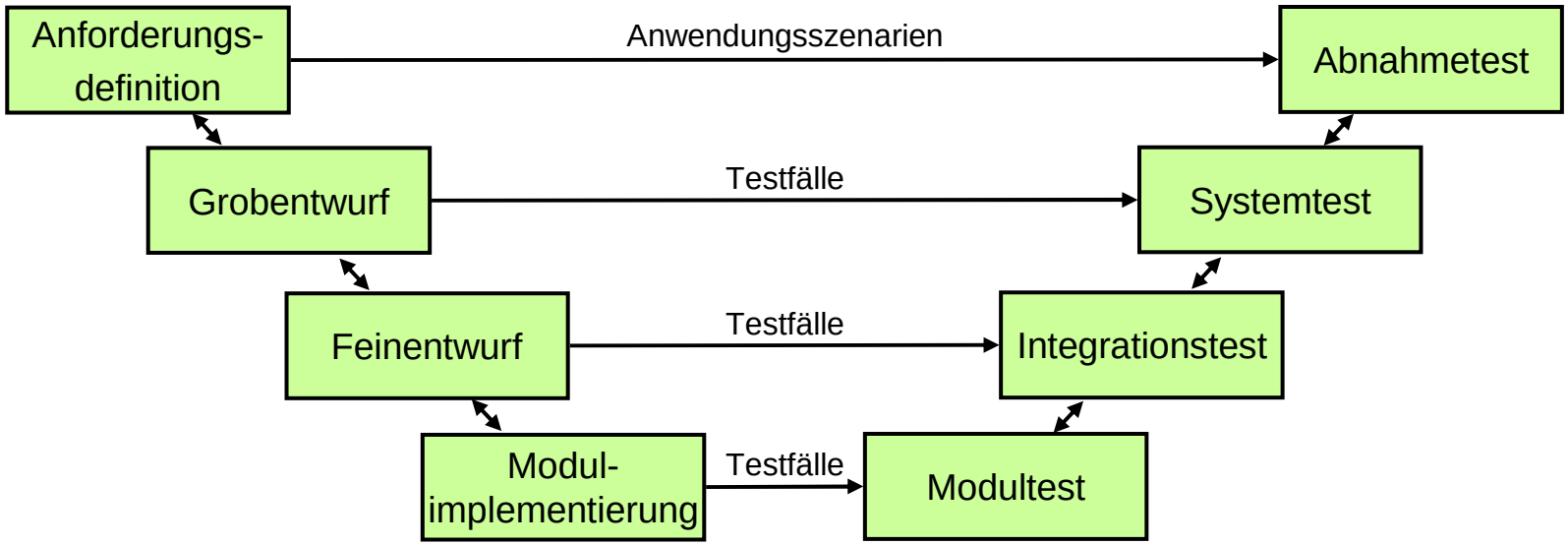
- http://www.waterfall2006.com/

Ein dokumentengetriebenes Prozessmodell

- Jede Aktivität ist in der richtigen Reihenfolge und in der volle Breite vollständig durchzuführen. Der Ablauf ist sequentiell, wobei eine Aktivität beendet sein muss, bevor die nächste beginnen kann.
- Am Ende jeder Aktivität steht ein fertiges Dokument.
- Die Auftraggeberbeteiligung ist nur in der Definitionsphase vorgesehen. Die anschließende Entwicklung erfolgt ohne Beteiligung des Auftraggebers.

Prozessmodell: das “V-Modell”

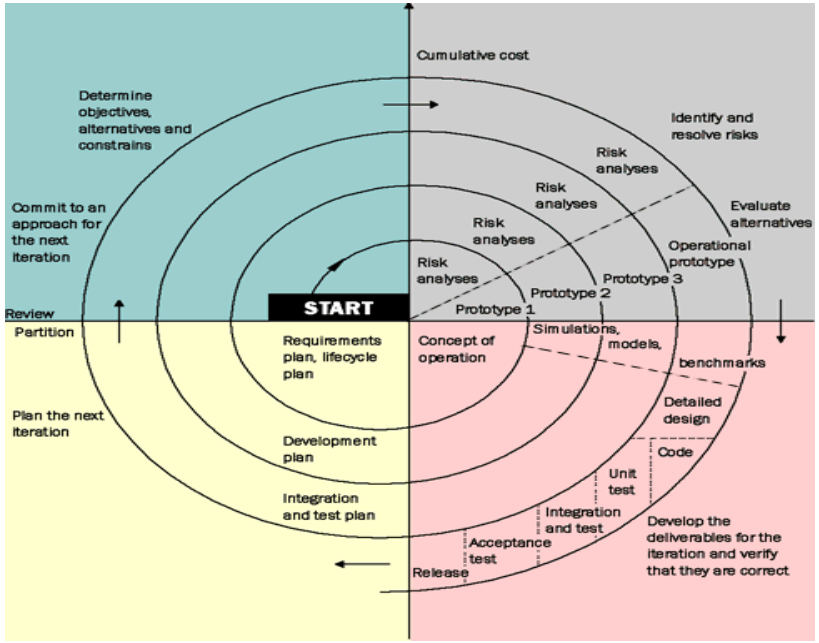
http://www.informatik.uni-bremen.de/gdpa/



Ein dokumentengetriebenes Prozessmodell

- definiert Aktivitäten und Produkte
  - Produkte haben Zustand (geplant, in Bearbeitung, vorgelegt, akzeptiert).
  - Aktivitäten können ein Produkt erzeugen oder es verändern.
- Verifikation und Validierung sind integraler Bestandteil
  - **Verifikation**: Erfüllt ein System seine Anforderungen?
  - **Validierung**: Erfüllt ein System seinen Zweck?

Prozessmodell: das Spiralmodell



Source: http://www.elanman.org/teaching/gmu/swe620-infs622/Graphics/spiral_model.gif

Schrittweise Entwicklung in Schleifen

- Vier Sektoren: Zielsetzung, Risikoabschätzung und –reduktion, Entwicklung und Validierung, Planung
- Risiko wird im Prozessmodell explizit berücksichtigt.
- In einer Schleife kann das Vorgehen in Abhängigkeit vom Risiko gewählt werden.

Übersicht: Modul 9

Software Engineering

- Einführung
- Software Prozesse
- Aktivitäten
- Prozessmodelle

Formale Modellierung

- Modellierung
- Spezifikationssprachen
- Formale Softwareentwicklung
- Formale Verifikation
- Werkzeugunterstützung

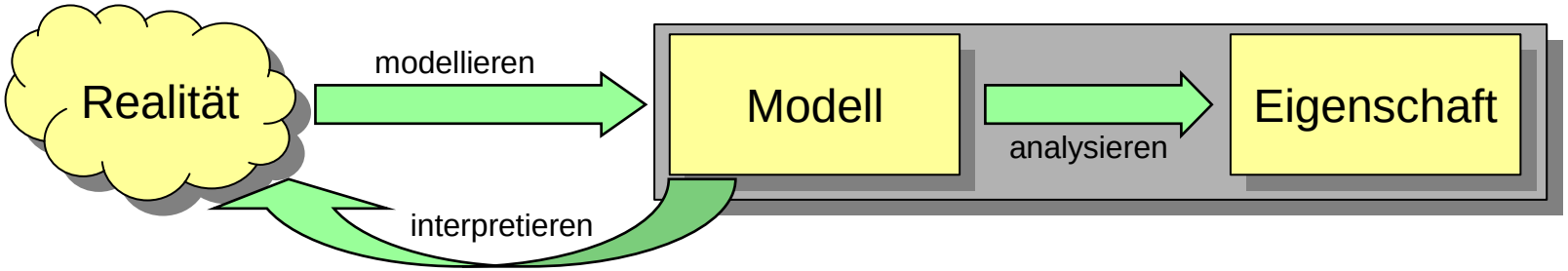
Evaluierung

Modellierung (1)

Was ist ein Modell eines Systems?

eine Beschreibung des Systems

- aus einer gegebenen Perspektive und
- auf einer gegebenen Abstraktionsstufe



Ein Modell ist absichtlicht nicht originalgetreu, um bestimmte Aspekte der Realität deutlicher hervorzuheben. Daher können nicht alle Eigenschaften des Modells der Realität entsprechen.

Alle relevanten Eigenschaften sollten der Realität entsprechen.

Modellierung (2)

Beispiele für mögliche Perspektiven:

- Perspektive einer bestimmten Rolle, z.B. der Rolle Auftraggeber
- Ein-/Ausgabeverhalten (Black-box Sicht)
- verhaltensorientierte Sicht
- Datenfluss und Kontrollfluss
- Architektur

Beispiele für mögliche Abstraktionsstufen:

- Anforderungssicht
- Abstraktionsstufen werden oft relativ zueinander als „hoch“ bzw. als „niedrig“ bezeichnet, wobei es durchaus mehr als zwei Abstraktionsstufen geben kann.

Prinzipiell kann man Modellierung während jeder Aktivität in einer Software Entwicklung einsetzen.

Modelle können z.B. eingesetzt werden, um das Verständnis des Systems oder seiner Anforderungen zu verbessern.

Spezifikationssprachen (1)

Eine Spezifikationssprache ist

- eine formale Notation zur Spezifikation von Modellen.

Die Syntax einer Spezifikationssprache definiert,

- welche Ausdrücke als Spezifikationen zulässig sind.

Eine Spezifikation beschreibt

- ein Modell oder eine Menge von Modellen.

Die Semantik einer Spezifikationssprache definiert,

- welches Modell/welche Modelle eine Spezifikation beschreibt.
- Die Semantik ist eine Abbildung in die Sprache der Mathematik.

Eine Spezifikation heißt inkonsistent,

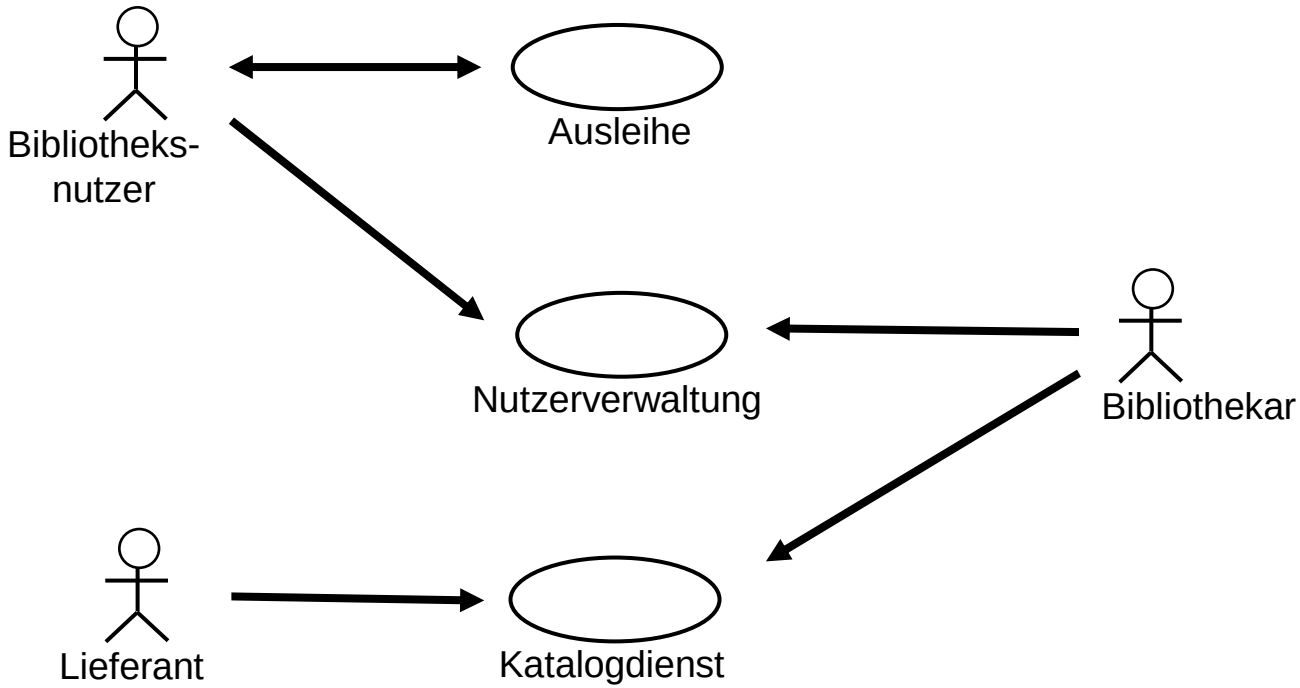
- wenn sie die leere Menge von Modellen beschreibt.

Beispiele für graphische Spezifikationssprachen

- Use cases, Aktionsfolgen, Datenfluss- und Klassendiagramme, …

Notation: Use Cases

Beispiel für eine Spezifikation

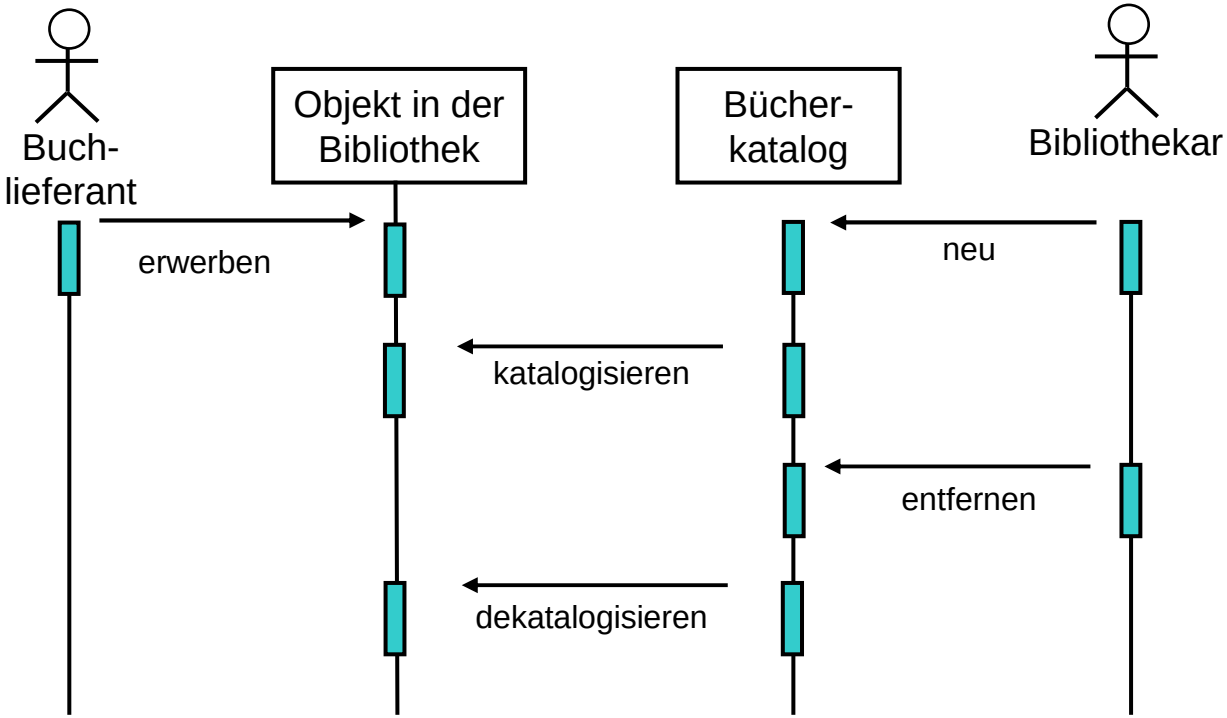


Quelle: I. Sommerville, Software Engineering, 6th edition, 2001, Addison-Wesley

Ein gerichteter Graph beschreibt die Akteure (Strichmännchen) und die Klassen von möglichen Interaktionen (Ellipsen).

Notation: Aktionsfolgen

Beispiel für eine Spezifikation

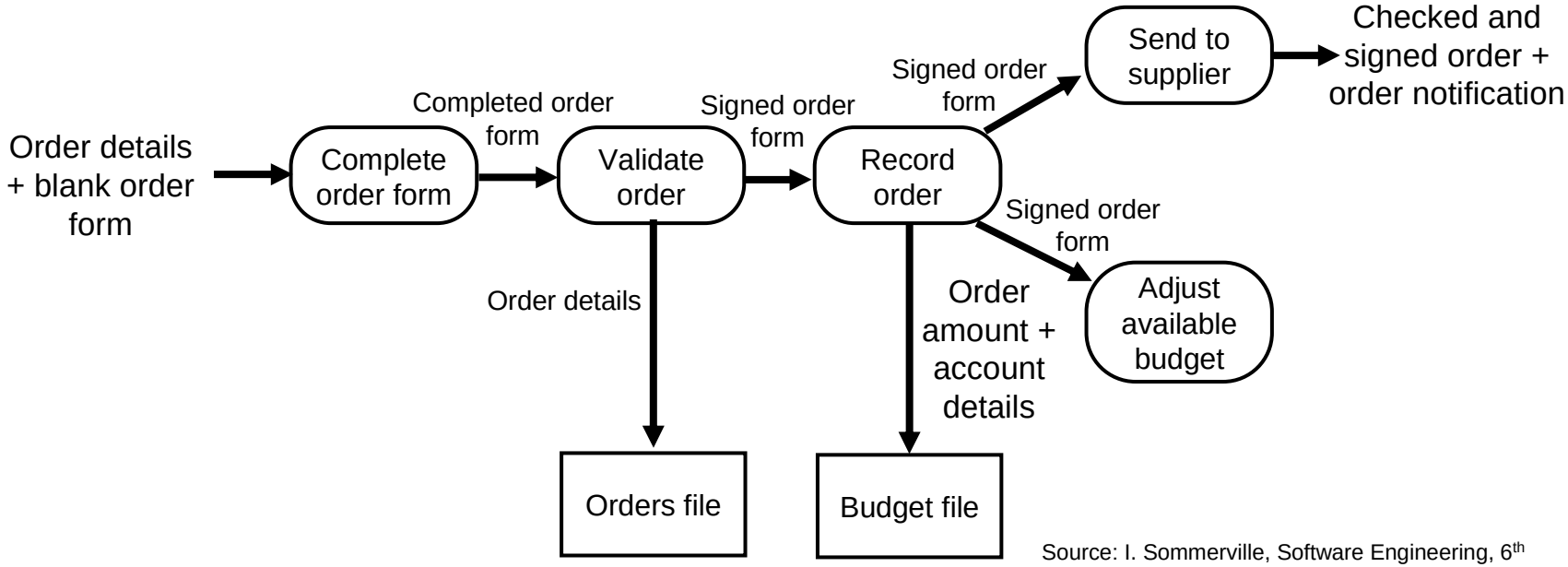


Quelle: I. Sommerville, Software Engineering, 6th edition, 2001, Addison-Wesley

Ein Diagramm beschreibt die Akteure, die Objekte im System, die Operationen und deren kausale Abhängigkeiten (Reihenfolge).

Notation: Datenflussdiagramm

Beispiel für eine Spezifikation



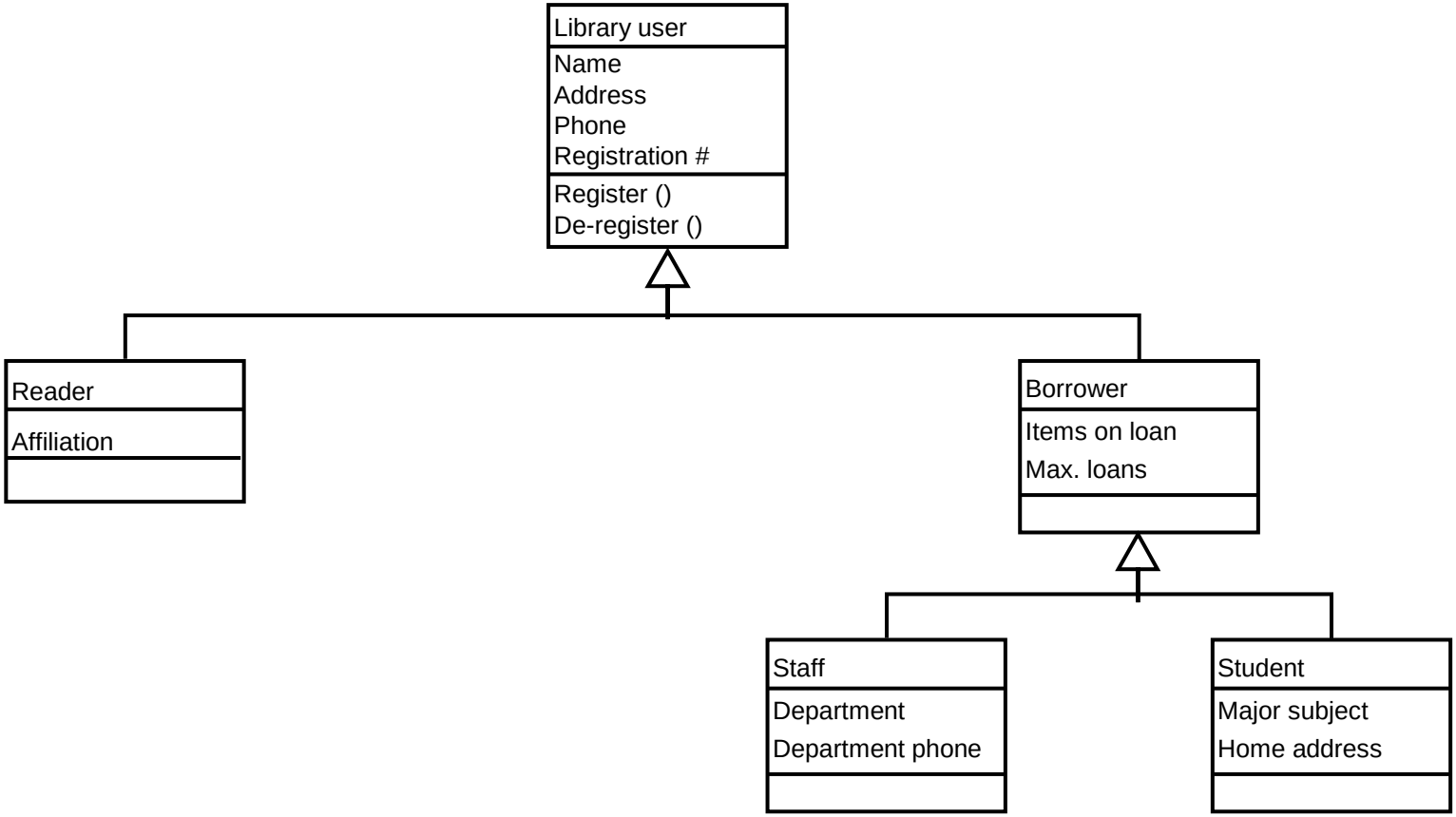
Source: I. Sommerville, Software Engineering, 6th edition, 2001, Addison-Wesley

Ein Diagramm beschreibt den Datenfluss zwischen Entitäten.

- ❑ abgerundete Rechtecke: Berechnungsschritte
- ❑ beschriftete Pfeile: Datenfluss
- ❑ Rechtecke: Datenspeicher oder Datenquellen

Notation: Klassendiagramm

Beispiel für eine Spezifikation



Quelle: I. Sommerville, Software Engineering, 6th edition, 2001, Addison-Wesley

Spezifikationssprachen (2)

Welche Klassen von Spezifikationssprachen gibt es?

- Es gibt semi-formale und formale Spezifikationssprachen.
- Die Syntax ist in beiden Klassen jeweils eine formale Notation zur Beschreibung von Modellen.

Semi-formale Spezifikationssprachen

- **Semantik:** Es gibt syntaktisch zulässige Ausdrücke, für die nicht eindeutig definiert ist, welche Modelle diese spezifizieren.

Formale Spezifikationssprachen

- **Semantik:** Für jeden syntaktisch zulässigen Ausdruck ist eindeutig definiert, welches Modell bzw. welche Menge von Modellen dieser spezifiziert.

Vorteil von formalen Spezifikationssprachen

- Die Bedeutung von Spezifikationen ist eindeutig definiert.

Spezifikationssprachen (3)

Formale Spezifikationen

können Bestandteil der Entwicklungsdokumentation sein.

Eine Entwicklungsdokumentation kann sowohl

- informelle Beschreibungen als auch
- semi-formale Spezifikationen als auch
- formale Spezifikationen enthalten.

Formale Methoden

können selektiv eingesetzt werden, indem man z.B.

- sehr kritische Komponenten und Anforderungen formal spezifiziert,
- kritische Komponenten und Anforderungen semi-formal spezifiziert,
- weniger kritische Komponenten und Anforderungen informell beschreibt.

Beispiele für formale Spezifikationssprachen

- CSP, CCS, PI-Kalkül, TLA, CTL, LTL, Z, CASL, …

Einsatz von Formalen Methoden

Formale Spezifikation von Anforderungen

- dient zur Kommunikation zwischen Auftraggebern und Entwicklern.
- Vermeidung von Missverständnissen
  - Formale Spezifikationssprachen haben eine eindeutige Semantik, wodurch Mehrdeutigkeiten vermieden werden.

Formale Spezifikation des Systems während der Entwicklung

- dient vor allem zur Kommunikation zwischen Entwicklern
- Vermeidung von Missverständnissen
  - Formale Spezifikationssprachen haben eine eindeutige Semantik, wodurch Mehrdeutigkeiten vermieden werden.
- Generierung von Programmteilen aus Spezifikationen ist möglich.

Formale Verifikation

- dient zur Überprüfung von Systemeigenschaften
- Durch mathematische Beweise werden Fehler vermieden.

Formale Softwareentwicklung

Zwei Ansätze, um Software vollständig formal zu entwickeln.

Transformationsbasierter Ansatz

- Startpunkt: formale Spezifikation der Anforderungen
- Die Spezifikation wird durch Anwendung von Transformationen schrittweise modifiziert.
  - Jeder Ansatz beinhaltet eine Bibliothek von Transformationen.
  - Für jede Transformation ist klar, welche Eigenschaften unter ihr erhalten bleiben.
- Zielpunkt: eine Spezifikation, die einem Programm entspricht

Beispiel: TAS [http://www.informatik.uni-bremen.de/~cxl/tas/]

Erfinde-und-Verifiziere Ansatz (invent-and-verify approach)

- Startpunkt: formale Spezifikation der Anforderungen
- Bei jedem Entwicklungsschritt wird
  - eine neue, konkretere Spezifikation erfunden und verifiziert, dass alle Eigenschaften, die von Interesse sind, erhalten wurden.
- Zielpunkt: eine Spezifikation, die einem Programm entspricht

Beispiel: VSE [http://www.dfki.de/vse/projects/vse.html]

Formale Verifikation (1)

Wo kann formale Verifikation eingesetzt werden?

- auf der Ebene von Code
  - sowohl in Programmiersprachen als auch in Maschinensprachen
- auf der Ebene von Modellen
  - Nachweis, dass ein Modell gegebene Eigenschaften erfüllt
  - Nachweis, dass Modelle in einer gegebenen Beziehung stehen

Beispiel für Verifikation auf Codeebene (Hoare Logik)

Hoare-Tripel: **{P} C {Q}**

- **P**: die Vorbedingung (eine prädikatenlogische Formel)
- **C**: das Programm (in einer imperativen Programmiersprache)
- **Q**: die Nachbedingung (eine prädikatenlogische Formel)

Gilt **{x=11} x:=x+2 {x=13}** ?

Gilt **{x>y} while x>0 do x:=x-1 {x>y}** ?

Gilt **{x<y;y>0} while x>0 do x:=x+1 {x>y}** ?

Formale Verifikation (2)

Was ist für formale Verifikation nötig?

- Kalküle mit denen man mathematische Beweise führen kann
  - Beispiele: Hoare Logik, Refinement Calculus

Wie kann man formale Verifikation erleichtern?

- durch Automatisierung der Beweisüberprüfung
  - Sind alle Schritte im Beweis im Sinne des Kalküls zulässig?
- durch Automatisierung der Beweissuche
  - mit automatischen oder halbautomatischen Theorembeweisern

Mehr zu formaler Verifikation und Model Checking siehe FGDI 3!

Werkzeugunterstützung

Werkzeuge werden eingesetzt z.B.

- um die Verwendung von formalen Methoden zu erleichtern oder
- um die Qualität der Verwendung zu erhöhen.

In welchen Bereichen ist Werkzeugunterstützung hilfreich?

- Editieren von formalen Spezifikation
- Überprüfen, ob eine Spezifikation gemäß der Syntax gültig ist
- Editieren und Konstruieren von formalen Beweisen
- Unterstützung bei der Beweissuche
- Verwaltung von Spezifikationen und Beweisen
  - Welche Beweise werden durch eine gegebene Änderung der Spezifikation ungültig?

Beispiele für Werkzeuge für formale Methoden

- B-tool, ISABELLE, PVS, VSE, ..

Übersicht: Modul 9

Software Engineering

- Einführung
- Software Prozesse
- Aktivitäten
- Prozessmodelle

Formale Modellierung

- Modellierung
- Spezifikationssprachen
- Formale Softwareentwicklung
- Formale Verifikation
- Werkzeugunterstützung

Evaluierung

Evaluierung von Software

Überprüfung der Qualität anhand vorgegebener Kriterien

- Das Ergebnis der Überprüfung sollte nachvollziehbar sein.
- Die Kriterien sollten sinnvoll und allgemein anerkannt sein.

Was wären überzeugende Kriterien für Qualität?

- Die Anforderungsspezifikation spielt eine zentrale Rolle.
  - Spiegelt die Spezifikation die realen Anforderungen wieder?
  - Genügt das System den spezifizierten Anforderungen?
- Formale Verifikation kann eingesetzt werden, um sicherzustellen, dass die Software allen spezifizierten Anforderungen genügt.
  - Formale Verifikation ist allerdings anspruchsvoll. (Expertise?)
  - Formale Verifikation ist auch zeitintensiv. (Budget?)

Evaluationskriterien

Kriterienwerke zur Evaluierung

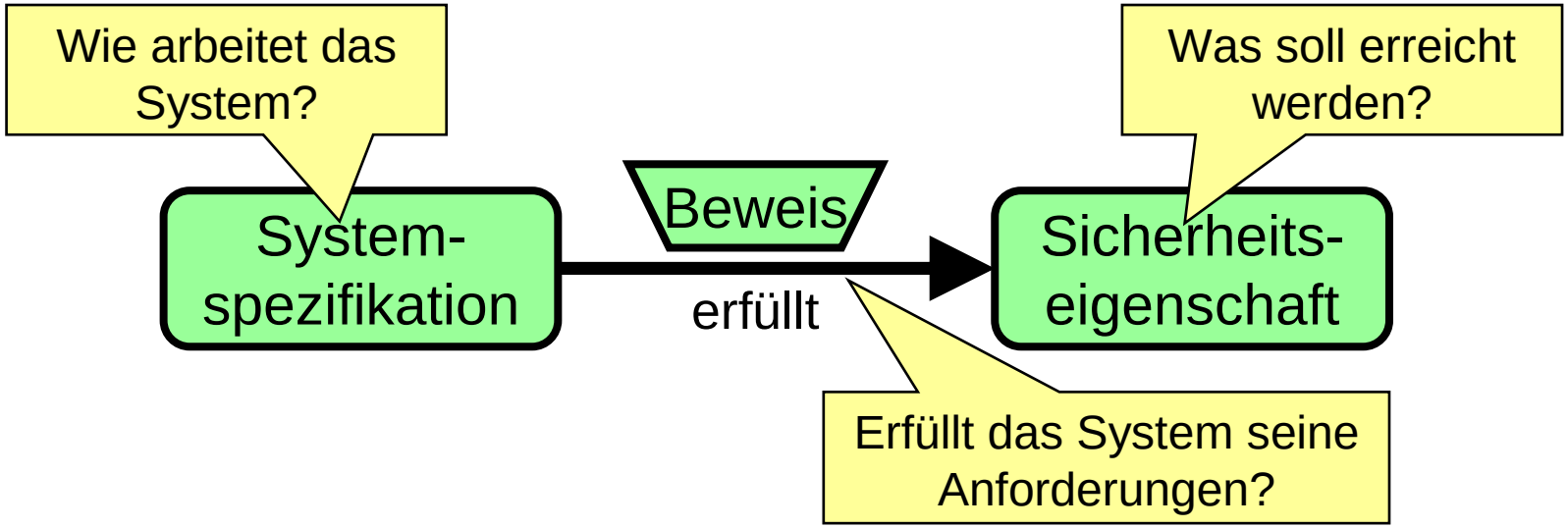
- Beispiele:
  - TCSEC, Orange Book (1983), USA
  - ITSEC (1991), F, D, NL, UK
  - CTPEC (1993), Kanada
  - Common Criteria (1999), CA, F, D, NL, UK, USA
    - aktuell Version 3.1 (2006)
- Eine vollständige formale Verifikation wird bisher nicht gefordert.
- Auf hohen Evaluierungsstufen wird die Verwendung von formalen Methoden zur Modellierung gefordert, z.B. ab E4 in ITSEC oder ab EAL 5 in Common Criteria v2.3: formale Sicherheitsmodelle.

Warum sollte man diese Kriterienwerke verwenden?

- Einheitlichkeit von Evaluationen
- Internationale Abkommen garantieren die Vergleichbarkeit von Evaluationen in unterschiedlichen Mitgliedsländern.

Formale Sicherheitsmodelle

Typische Struktur:



Beachte:

- Die Spezifikation des Systems (Verhalten, Architektur, ..) ist klar von der Spezifikation der Anforderungen getrennt.
- Obiges Diagramm eignet sich für die Spezifikation auf einer gegebenen Abstraktionsebene.

Übersicht: Modul 9

Software Engineering

- Einführung
- Software Prozesse
- Aktivitäten
- Prozessmodelle

Formale Modellierung

- Modellierung
- Spezifikationssprachen
- Formale Softwareentwicklung
- Formale Verifikation
- Werkzeugunterstützung

Evaluierung

Einsatz Formaler Methoden

Wie setzt man formale Modelle im Software Engineering ein?

als Bestandteil von Dokumenten in der Entwicklung

- zur Vermeidung von Mehrdeutigkeiten in Spezifikationen
- als Basis für die formale Verifikation von kritischen Anforderungen
- als Basis für die Generierung von Testfällen
- um den Anforderungen von Evaluationskriterien zu genügen

als Bestandteil des Denkens bei der Entwicklung

- Überprüfung: Weiß ich wirklich präzise z.B.
  - Was die Anforderungen an das System sind?
  - Für welche Systemarchitektur das Team sich entschieden hat?
- als Grundlage für die Kommunikation mit anderen
  - Präzisierung der Aspekte, die man kommunizieren will
    - z.B. für konzeptionell schwierige Aspekte (z.B. IT Sicherheit) oder komplexe Systeme (z.B. nebenläufige oder verteilte Systeme)

Rückblick auf Modul 9

Einige wesentliche Lernziele dieses Moduls

- Wie ist der Zusammenhang zwischen Modellierung und strukturierte Softwareentwicklung?
  - Bei welchen Aktivitäten kann Modellierung einsetzen?
  - Wie kann man Modelle einsetzen?
- Was ist eine Spezifikationssprache?
  - Unterscheidung: Syntax und Semantik
  - Unterscheidung: formal und semi-formal
- Wie kann man formale Methoden einsetzen?
  - Selektiver Einsatz formaler Spezifikationssprachen ist möglich.
  - Prinzipiell ist sogar eine vollständig formale Entwicklung von Software möglich.
- Wie wird sicherheitskritische Software evaluiert?
  - Kurzeinführung in Kriterienwerke für die Evaluierung

Literatur

Allgemeine Einführungen in die Softwaretechnik:

H Balzert.
Lehrbuch der Software-Technik, 2. Auflage, Spektrum, 2000

Sommerville.
Software Engineering, 6th edition, 2001, Addison-Wesley

Common Criteria (Bespiel für ein Kritienwerk)

- www.commoncriteriaportal.org