

SE 2. Vorlesung

18.10.2016

* Software Systeme

1. Was kennzeichnet ein System?

* ist von der Außenwelt abgetrennt

* erfüllt eine Funktion → ABLAUF-STRUKTUR

* kann Teilsysteme enthalten

* Ein System enthält Komponenten / auch ein Subsystem enthält Komponenten → AUFBAU-STRUKTUR

* enthält Komponenten die in Beziehung zueinander stehen

* befindet sich zum def. Zeitpunkt in einem definierten Zustand.

* über Schnittstellen (Verbindungsstellen) ist der System mit seiner relevanten Umwelt (Kontext) verbunden

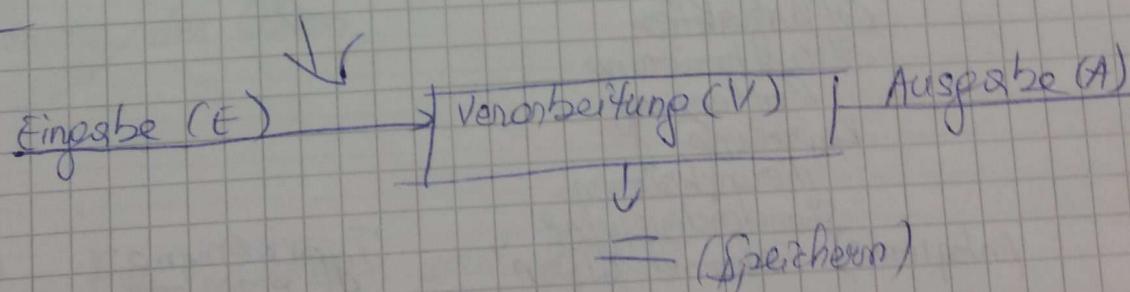
→ XDI 3633 Blatt 1: System

die Definition

→ Seite 9

Prozess

→ Seite 10



2. Warum sind Software-Systeme ganz spezielle Systeme?
- ↳ nicht materiell (immateriell)
 - ↳ hat keine natürliche Dekalität
 - ↳ sie benötigen einen Datenträger
 - Werkstoff: SPRACHE

um etwas zu lie-wirken benötigen wir
ein DV-System bestehend aus HWX-SW

~
Seite 14 → Datenmodell (z.B. Entity Relationship Modell)
Kom kann ein komplexes System nur nur mit einem
Modell verstehen oder darstellen

3. Welchen Unterschied gibt es zwischen Modellen und Prototypen
bei der Software Entwicklung
Modelle bildensaf; Prototypen sind lauffähig

4

Grundprinzipien der SW-Entwicklung

Abstraktion
Strukturierung
Zerlegung
Kapselung
Hierarchisierung
Standardisierung
(integrierte) Dokumentation
Typisierung
Persistenz

3. Verarbeitung

28.10.2016

- Was kennzeichnet qualitativ hochwertige Software-Systeme?
 - übersichtlich → Oberfläche
→ Code
 - perfomant
 - hoher Nutzen (d.h. die Anforderungen lösen)
 - einstieger freundlich → intuitiv bedienbar
 - wenig Sicherheitslücken → sicher
 - flexibel → anpassbar, erweiterbar, veränderbar
 - hoher Grad an Dokumentierung ausreichend
 - ↳ für Entwickler zur Weiterentwicklung
 - modifizierbar

für die Benutzung

- Benutzerfreundlichkeit

Wirtschaftlichkeit

Funktionsfüllung: Übereinstimmung zwischen den geplanten und realisierten funktionalen Anforderungen

z.B. im Pflichtenheft notiert

System - Test

Benutzbarkeit (Benutzerfreundlichkeit, Usability)

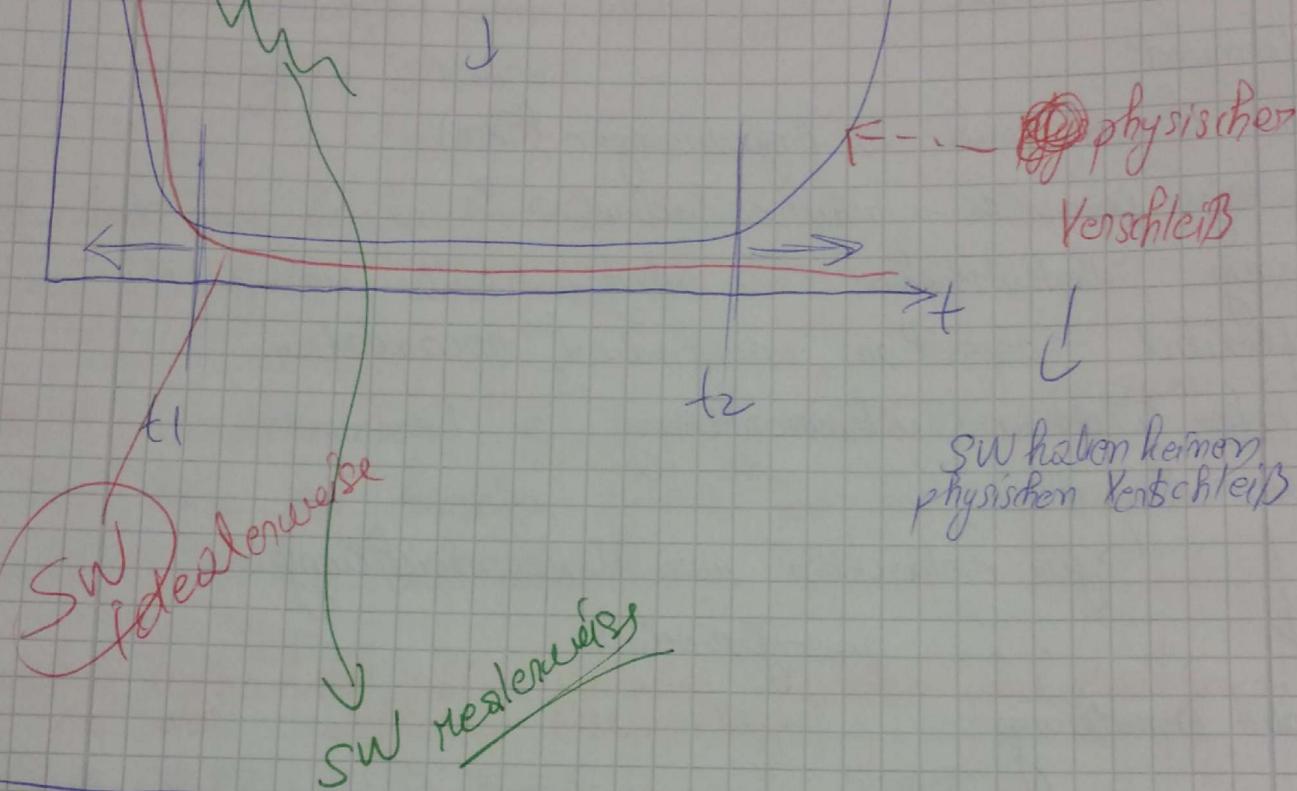
Effizient

- Antwortzeiten
- Transferzeiten
- Transportzeiten
- Speichervolumen

Zuverlässigkeit und Sicherheit unterscheiden

(Ausfallrate) $\rightarrow ?$ Badewannenkurve

(HW)



Kompatibilität $\hat{=}$ Anschließbarkeit

objektif ist kompatibel zu MS Word

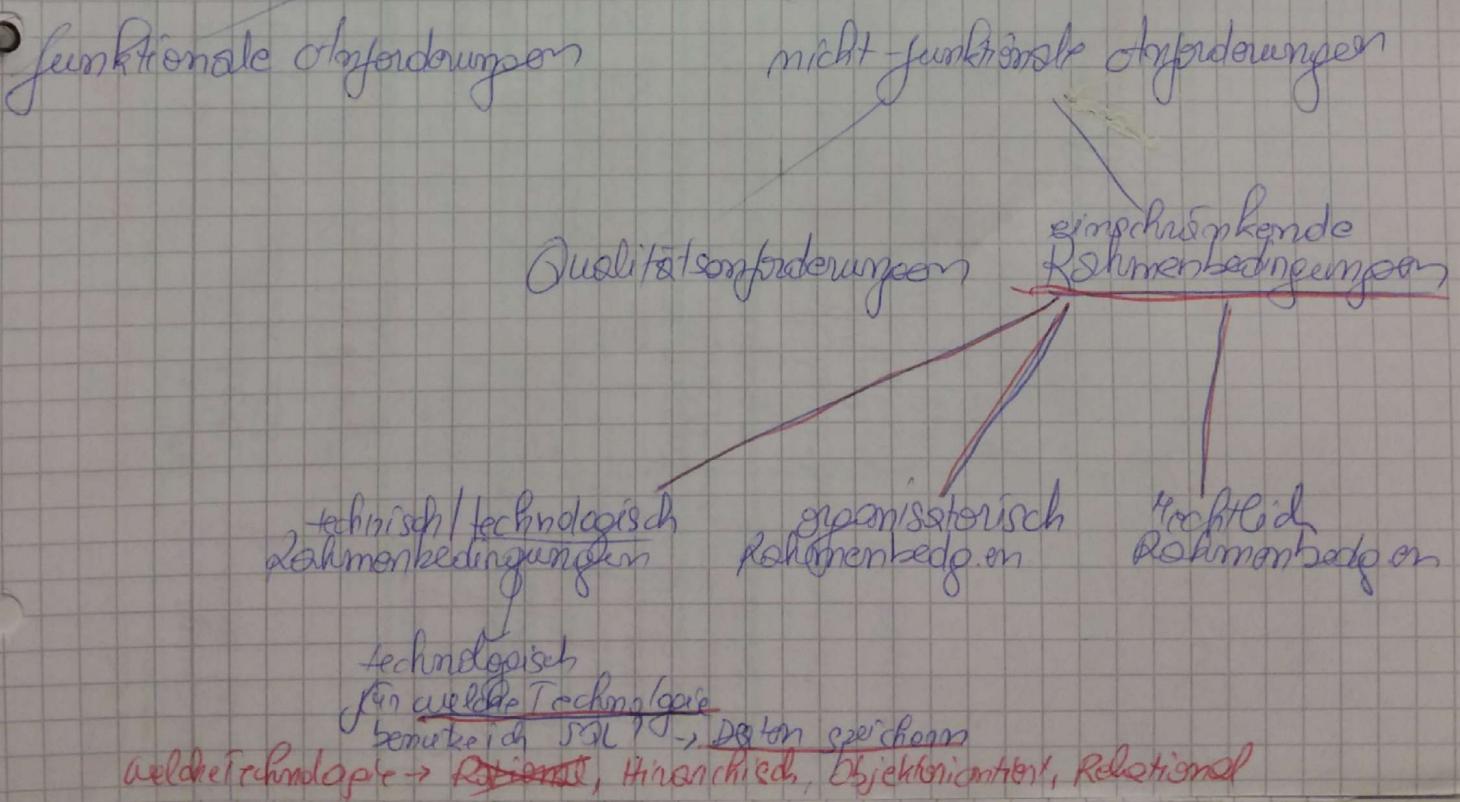
Software Engineering 4. Vorlesung

01.11.2016

Das SW-System

- muss benutzerfreundlich sein
 - Knopf muss links & auf dem Bildschirm anschwellen
 - intuitiv bedienbar sein
ohne Fehlzüge bedienbar.

Anforderungen



1) Das SW-S muss es erlauben, Personen anzumelden

funktional

2)

funktional

3)

funktional

4)

→

operational
Rahmenbedingungen

5)

funktional

6)

Qualitätsanforderungen

7) → F/R - Ht.

8) → F/R - Ht , R_{rechtlich}

(Leistungsmerkmale synonym funktl. Anforderungen)

↓
richtiger Begriff

SC 5 Vorbereitung der Anforderungsanalyse
 1 Nur ein Berichten die Risiken der Anforderungsanalyse
 2 Wie kann den Risiken begegnen werden

3 Womit beginnt die Anforderungsanalyse?

Risiko

Vorbeidung des Risikos

Anforderung ist nicht eindeutig

- Regeln im detaillierten Beschreibung vorgelagert einhalten (z.B. im Satzkonzept / Prädikat / Objekt)

Qualitätsanforderung

Eindeutigkeit → eindeutig

Anforderung ist nicht realisierbar

- überprüfen mit experimentellen Prototypen

Realisierbarkeit → realisierbar

Anforderung ist nicht korrekt

- grafische Darstellung (z.B. UML)

- Prototyp (Oberflächendesignprototyp) → Kommunikation Korrektheit → korrekt.

Beschreibung der Anforderung ist zu detailliert

- grafische Darstellung unter Nutzen den Abstraktionsgrad, d.h.

Angemessenheit → angemessen

Anforderungen sind nicht vollständig

- Validieren → Prototyp

- grafische Darstellung

Kommunikation Vollständigkeit → vollständig

Anforderungen sind widersprüchlich

- aufdecken von Beziehungen zwischen Anforderungen

(Beziehungen z.B. zwischen Funktionalen Anforderungen, Q-Anforderungen und Rahmenbedingungen aufdecken

→ Konsistenz von Anforderungen)

Konsistenz → konsistent

(Widerspruchsfreiheit)

Anforderungen sind zu umfangreich

- testen prüfen

- durch grafische Darstellungen zur Erklärung

Minimalität → minimal

Anforderungen sind nicht strukturiert

- Strukturierter

und damit nicht klar

Strukturiertheit → strukturiert

Validierung : Baue ich das richtige Haus?

Vergleichung : Baue ich das Haus richtig? ↗ völlig
verschieden,
Dinge!

Ermittlungstechniken

Ermittlungstechnik

antefakt - basierte Ermittlungstechnik

- z.B. - Eingabekarten, Ausgabekarten
- Prozessbeschreibung

- Befragungstechnik

{ - Audioaufzeichnung, Videoaufzeichnung

- Protokollaufzeichnungen

- Erkundungstechnik (Betrachtungen, Perspektivwechsel)

- Anwendungsfallmodellierung

→ Essenzbildung (ich hole die Essenz aus)

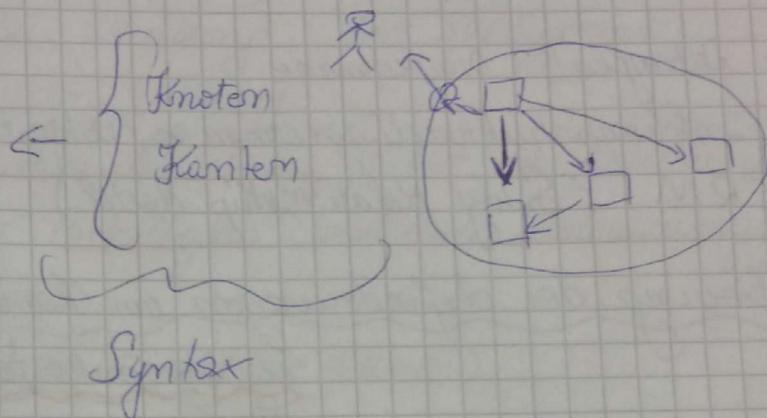
6. Vorlesung

15. 11. 2016

Wie werden ermittelte Anforderungen beschreiben? ①

- textlich
- grafisch
- Salzschablonen nach Chris Rupp

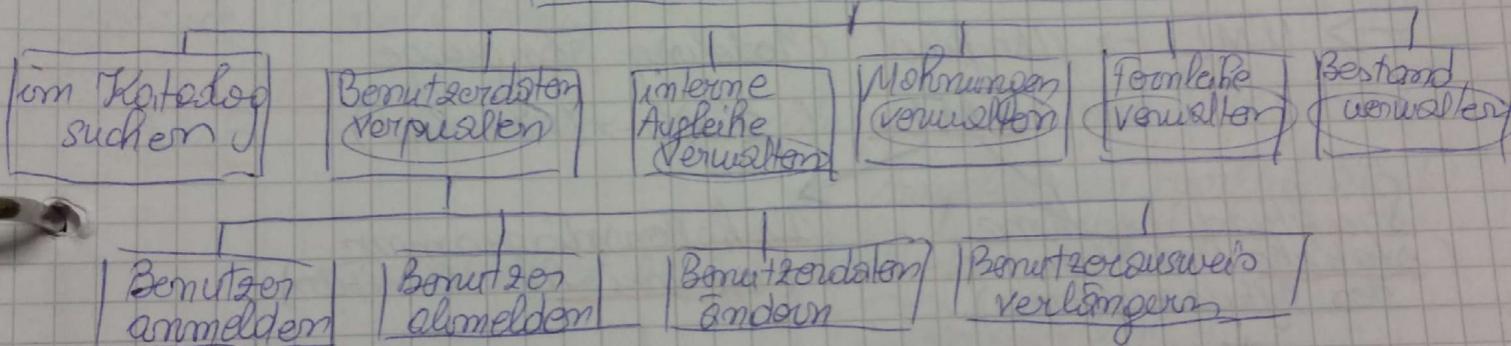
Struktur-
diagramm



1. Modellierung von funktionalen Anforderungen

1. Funktionsstruktur → Hierarchiebaum

Bibliothek nutzen und verwalten



Knoten: Funktion / Teilfunktion von anderen (Teil-)Funktionen unabhängig

Kanten: enthält - Beziehung

Semantik: Gesamtfunktion wird in Teilfunktion zerlegt.

Verwalten heißt: Einfügen, ändern, löschen

↓
VERB

2. Modellierung von Anforderungen im Kontext der strukturierten Analyse

→ strukturiertes Design

→ strukturierte Programmierung
(z.B. C!)

→ Datenfluss-Diagramme

→ Funktionsstrukturdiagramme / Datenstrukturdiagramme

→ ERM (Entity Relationship Model)

3. Modellierung der Anforderungen im Kontext der objektorientierten Analyse

→ objektorientiertes Design

→ objektorientierte Programmierung
(z.B. Java)

→ UML → Unified Modeling Language

UML

(2.0) -> Standard

Strukturdiagramme
(statische Struktur
des Softwaresystems)

Verhaltensdiagramme
(dynamische Struktur)

Klassendiagramm
Paketdiagramm
Verteilungsdiagramm
Objektdiagramm
Komponentendiagramm
Kompositionstrukturdiagramm

Anwendungsfalldiagramm
Aktivitätsdiagramm
Listendiagramm
Sequenzdiagramm
Kommunikationsdiagramm
Interaktionsübersichtsdiagramm
Zeitverlaufsdigramm

www.uml.org

(2)

Das Anwendungsfall-Diagramm

Semantik: Das Anwendungsfall-Diagramm (AWF-Diagramm) stellt die funktionalen Anforderungen aus der Sicht der Anwender dar. Die funktionalen Anforderungen werden im Beziehung gesetzt zu den Beteiligten & im Kontext (Beteiligt an der Nutzung des SW-Systems)

Beteiligte → Akteure

Die Kommunikationsbeziehung bildet die Beziehungen zwischen den Beteiligten und den Anwendungsfällen ab.

Knoten:

- Akteur
- Anwendungsfälle (AWF)

Kanten:

- zwischen Akteur + AWF : Kommunikationsbeziehung
- zwischen Akteuren + Akteur: Generalisierung / Spezialisierung
- zwischen AWF + AWF: erweitert, enthält

Topologie: netzförmig

BEGRIFFE:

AWF = abstrakte Abbildung einer vom SW-System angebotenen Funktionalität.

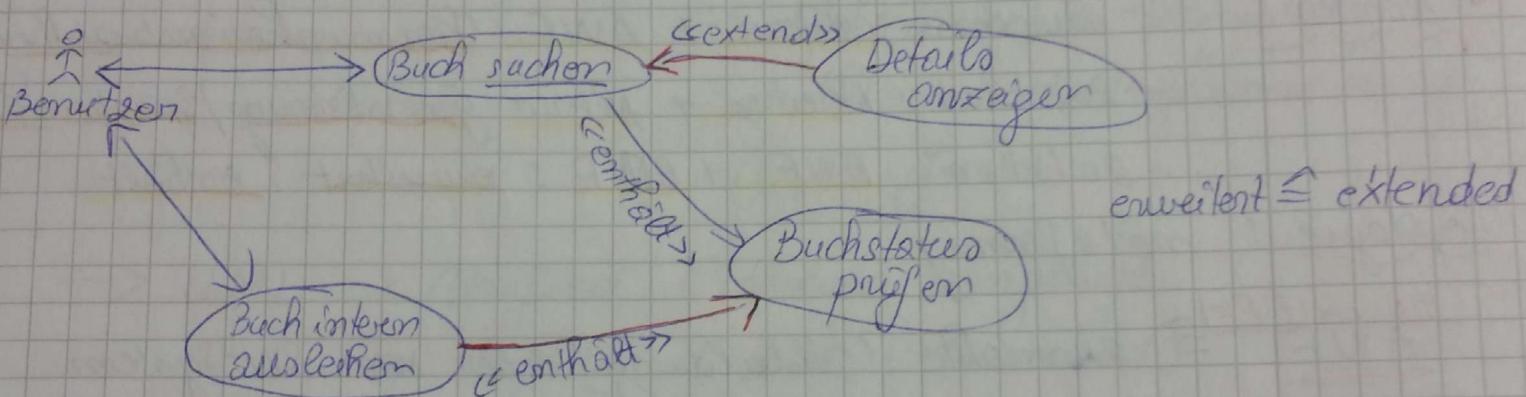
- abstrakter AWF / konkreter AWF
einstanzierbar
- wird ausgelöst durch ein Ereignis
 - datengetriebener Auslöser → z.B. Anmeldewunsch liegt vor
 - zeitlicher Ereignis
- liefert i. d. R. ein nach außen sichtbares Ergebnis
→ BSP → Benutzerausweis.
- wird beschrieben mit
 - textlich → Aktivitätsdiagramm
 - grafisch → Zustandsdiagramm

Glikteur: = abstrakte Abbildung einer externen Instanz, die mit dem System Kommuniziert, z.B. Benutzer, Mitarbeiter der Bibliothek,

- rollenorientiert
 - aktiver Akteur - - passiver Akteur.
 - Anwenden
 - Reelles Ereignis
 - System
 - z.B. Tiefstalobbe Fahrplan

Beziehungen:

- Akteur-AWF: abstrakte Abbildung für Ein-Gesamtdaten.
- AWF-AWF:



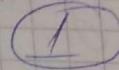
wenn Sie immer aktiv (den Satz) formulieren dann
ist die Richtung " \rightarrow " \Rightarrow

Aktiv Familien

optional

L) Details anzeigen, erweitert Buch suchen

Buch intern ausleihen \Rightarrow enthält Buchstatus prüfen



Subversion | Git → Bärbel

1. Welchen Sinn hat Konfigurationsverwaltung
Welche Beziehung besteht zwischen Konfigurations- und

Gefahr

- Eine falsche Komponente wird integriert
- gleichzeitig arbeiten mehrere Personen an einem Produkt
- Eine Komponente wird zum zweiten Mal entwickelt
- Es wird mit Änderungen begonnen, ohne das Altsystem zu sichern
- Ein fertiges System wird ausgeliefert und läuft dann Kunden weiter zu entwickeln und weiter davon gearbeitet

Begriffe:

• SW-Einheit, „ein Stück SW“ → atomar bearbeitbar.

unabhängig von anderen
SW

Verwendungs-

atomar $\hat{=}$ unabhängig von anderen

SW-Einheiten bearbeitbar, speicherbar,

Austauschbar.

- Wenn eine SW-Einheit wird, entsteht

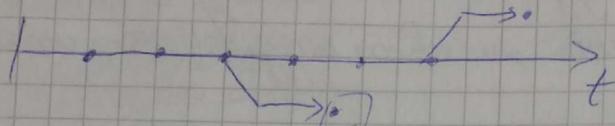
Wenn eine SW-Einheit geändert wird, entsteht eine neue Version dieser SW-Einheit.

d.h. SW-Einheiten haben einen Vorgänger und einen Nachfolger (i.R.) → Versionen stehen in einer zeitlichen Beziehung

IEEE

Achtung Es gibt einen Unterschied zwischen Version und Revision.

- Varianten stehen nicht in zeitlichem Bezug sondern neben einander (gleichzeitig)



- Konfigurieren: Menge von SW-Einheiten, die für einen definierten Zweck zusammengestellt ist und die natürlich auch zusammenpassen.
 - Gängend von den Anforderungen definiert das Gesuchlverfahren, welche SW-Einheit in die Konfiguration aufzunehmen ist.

- Baseline: Konfiguration, die stabil ist.

- Release: Baseline, die an Kunden ausgeliefert wird

Konfigurationsverwaltung = Rolle oder Organisationseinheit im Entwicklungsprozess mit folgenden Aufgaben:

- Identifizierung der SW-Einheiten.
- Bei Bedarf Bereitgestellt werden der SW-Einheiten.
- Änderungen an SW-Einheiten müssen überwacht + dokumentiert werden.
- Rekonstruktion älterer Versionen von SW-Einheitskonfigurationen.

Realisierung dieser Aufgaben?

(2)

→ Arbeitsorganisation; d.h. Regeln.

1. Alle Entwickler arbeiten in einer Umgebung

2. Das Konfigurationsmanagement hat das Mandat bei

↳ der Bereitstellung von SW-Einheiten/
Werkzeuggestützt. ↳ SW-Systemen.

3. Ein abgeleitetes Arbeitsregelunis muss eines Entwickler
muss zyklisch verwaltet werden.

→ → SW-Einheit wird:
- eindeutig identifiziert
- zur Veränderung bereit gestellt.
- Änderungen speichern und
anordnen
(-sablonen, ändern, zurück liefern)

4. Veraltete Versionen, Konfigurationen werden archiviert.

5. Neuerungen werden als Varianten verwaltet.

6. Das SW-System, das an den Kunden ausgeliefert wird
wird mit dem Werkzeug zur Konfigurationsverwaltung
zusammengestellt und präzis dokumentiert.

7. Nur geprüfte SW-Einheiten werden für alle
Konfiguration freigelassen.

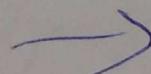
Realisierung dieser Aufgaben

→ Arbeitsorganisation, d.h. Regeln

→ technische Realisierung

e.g. werkzeuggestützt mit effizientem Speicherverfahren → Repository

2 Philosophien: zentrale/dezentrale Versionsverwaltung
~~alle Daten „in Ruhe“~~



2 Philosophien : zentrale / dezentrale Versionsverwaltung

- alle Daten "in Ruhe"
sind auf ~~verschieden~~ einem
Server gespeichert
- jeder Entwickler arbeitet
auf einer Kopie der
entsprechenden Daten
z.B. Subversion

→ Entwickler - u.
Serverumgebung
sind nicht
getrennt.

Belegarkeit

Gruppenbeleg

Gruppenstärke 3-5 Personen

PVL

Abgabe 20.01.2017

(Spätester Termin Abgabe früher möglich)

Vorlesung 8

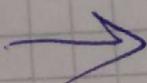
Die weitere Zerlegung eines AWF führt nicht (mehr)

zu voneinander unabhängigen AWF sondern zu
voneinander abhängigen Aktionenn AWF \equiv Aktivität (enthält voneinander abhängige
Aktionen)

Aktivitätsdiagramm

Das Aktivitätsdiagramm→ modelliert die Aktionen, die an einer Aktivität (=AWF)
gekapselt sind.D.h. ein Aktivitätsdiagramm beschreibt einen
AlgorithmusDie einzelnen Aktionen der Aktivität folgen
sequentiell aufeinander oder sind
bedingungssabhängig oder zyklisch gesteuert.

Einzelne Aktionen können verschiedene

Verantwortungsbereichen superordnet werden
(entnommen aus PAP, Petrinetzen, Datenflussdiag)

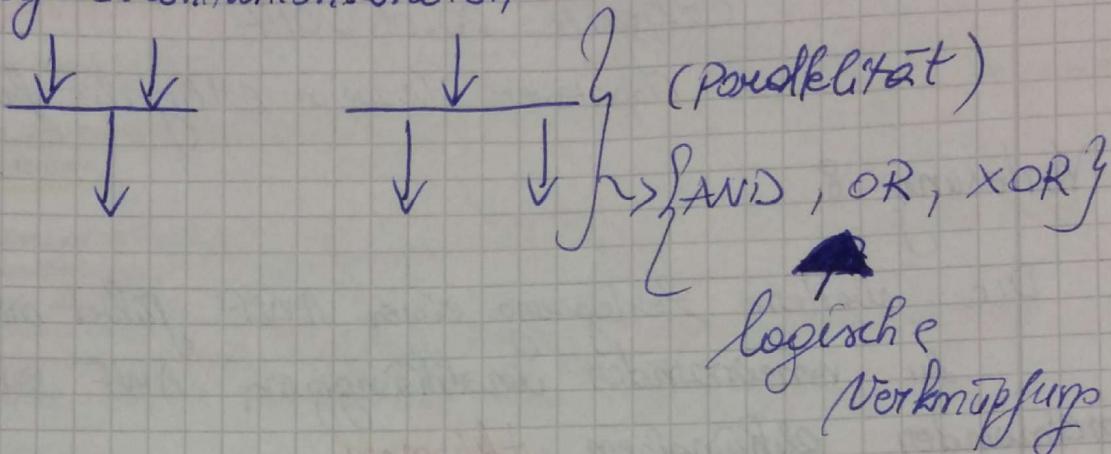
Knoten

* Aktion

- * Start - / Endzustand
- * Entscheidungsknoten

$\Leftarrow \Rightarrow$ (... hat mindestens
2 Ausgänge)

Synchronisationsknoten

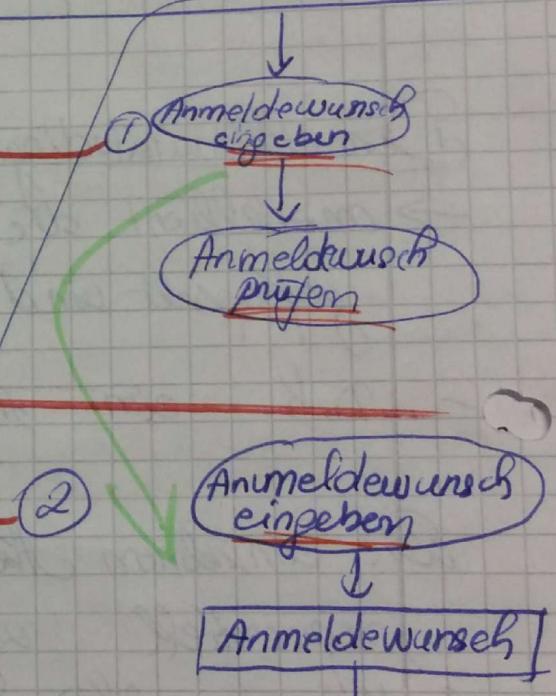


und viele weitere Knotendemente

Kanten im Aktivitätsdiagramm

* Transition

Objektfluss

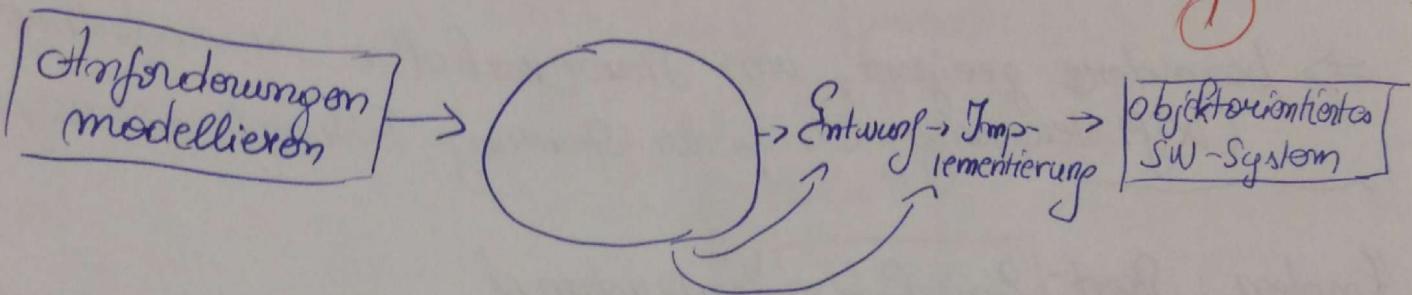


Topologie: netzförmig (nicht für diesen
bestimmten Fall)

Regel für die Syntax dieses Diagramms

→ - Eine Aktion wird mit einem Verb bezeichnet!

①



Welche Diagramme nutzt die UML zur Modellierung von Anforderungen?

- ✓ - AVF-Diagramm
- ✓ - Aktivitäts-Diagramm
- - Zustands-Diagramm
- - Klassendiagramm

Das Zustands-(Übergangs-)diagramm

Semantik: Das Zustandsdiagramm stellt die möglichen Zustände und deren Übergänge einer Betrachtungseinheit dar.

- ↳ Objekt
- ↳ (Teil-)System
- ↳ Prozess

Der Zustandswechsel ist durch das auslösende Ereignis ① und die eventuell noch außen sichtbare Reaktion ② beschrieben. Der Zustandswechsel kann zusätzlich ③ bedingungshängig sein.

Rahmenbedingungen: Die Betrachtungseinheit befindet sich zu einem Zeitpunkt t_1 im genau definierten Zustand.

- Es gibt genau einen Startzustand und einen oder mehrere Endzustände

⇒ besonders geeignet, um Steuerverhalten zu modellieren
(Kfzsteuerung, CD-Spieler-Steuerung, Fahrerinformationssystem)

Knoten: Start; Zwischen - Endzustand

- Entscheidungsknoten
- Terminator
- im Zusammenhang mit Regionen, gibt es Kreuzungen / Gabelungen
- ...

diese sind die eigentlichen Zustände.

Kanten: Zustandswechsel → Transition

↳ Ereignis [Bedingung] / Aktion (siehe Seite)

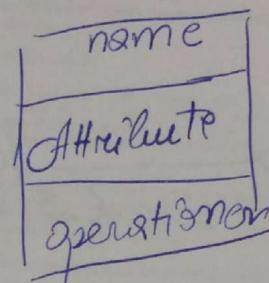
Topologie: netzförmig

Gliederung: zur Komplexitätsreduzierung → lassen sich Zustände zusammenfassen

Das Klassendiagramm

(2)

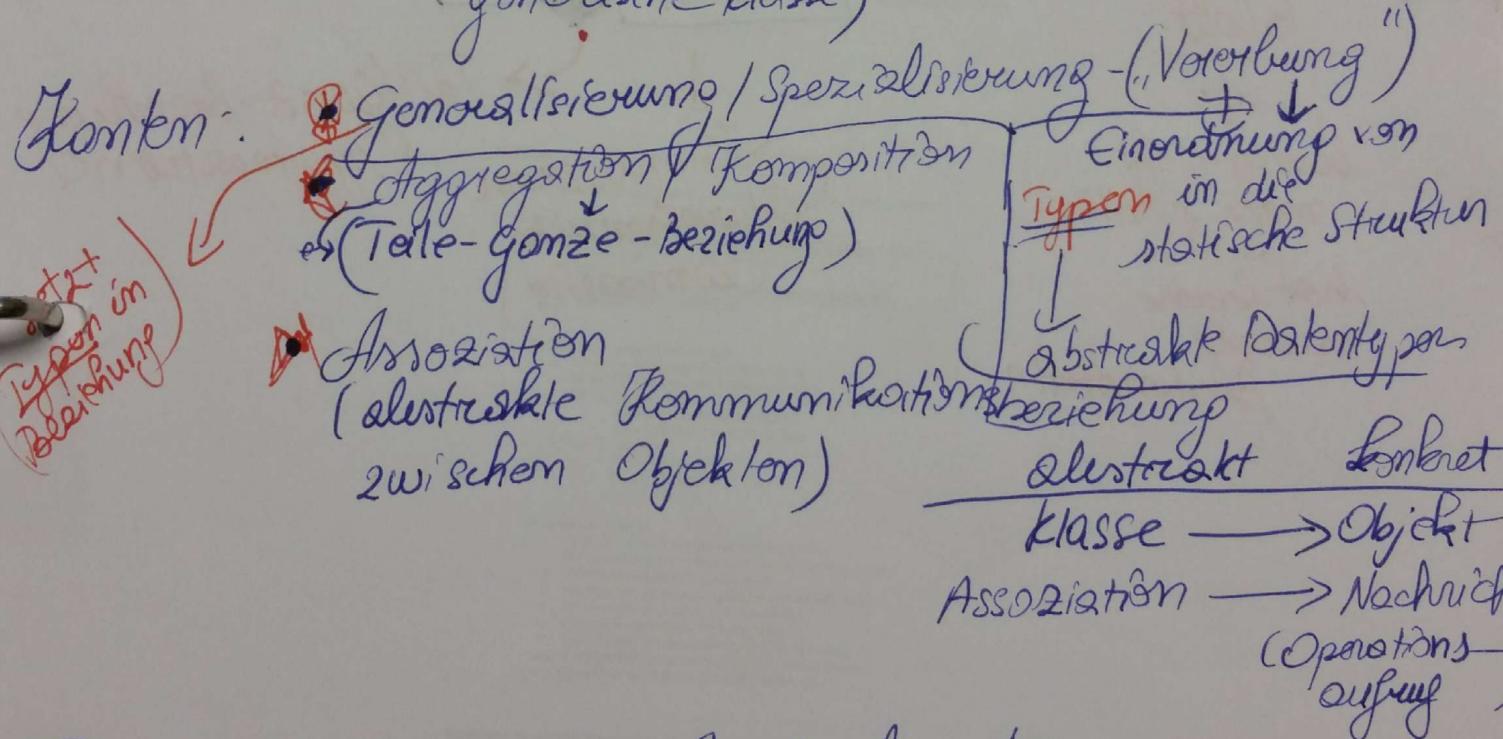
Klasse: Symbol



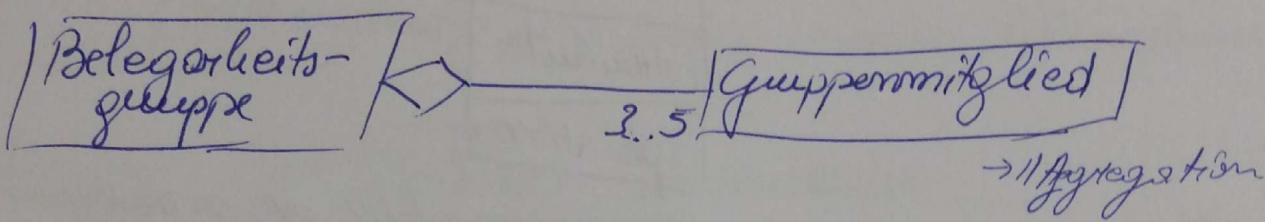
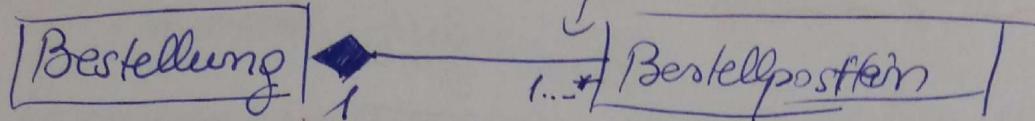
?

Semantik: Das Klassendiagramm stellt die einzelnen Klassen (mit oder ohne Details) und deren statische Beziehungen dar.

Knoten: - Klasse (abstrakte (nicht instanzierbar), konkrete (instanzierbar), Interface, generische Klasse)



Topologie: netzförmig mit eingelagerten Hierarchien.



Beispiel : Dateisystem \rightarrow Assoziation

