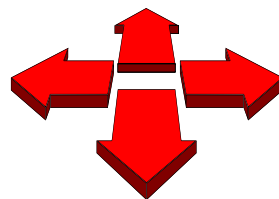




Objektorientierte Modellierung – fachlich und technisch

- Einordnung objektorientierter Methoden
- Modelle der Softwareentwicklung
- Fachliche Modellierung
- Ein Bibliotheksbeispiel
 - ~~strukturorientiert~~
 - verhaltensorientiert
- Technische Modellierung



Literaturhinweise

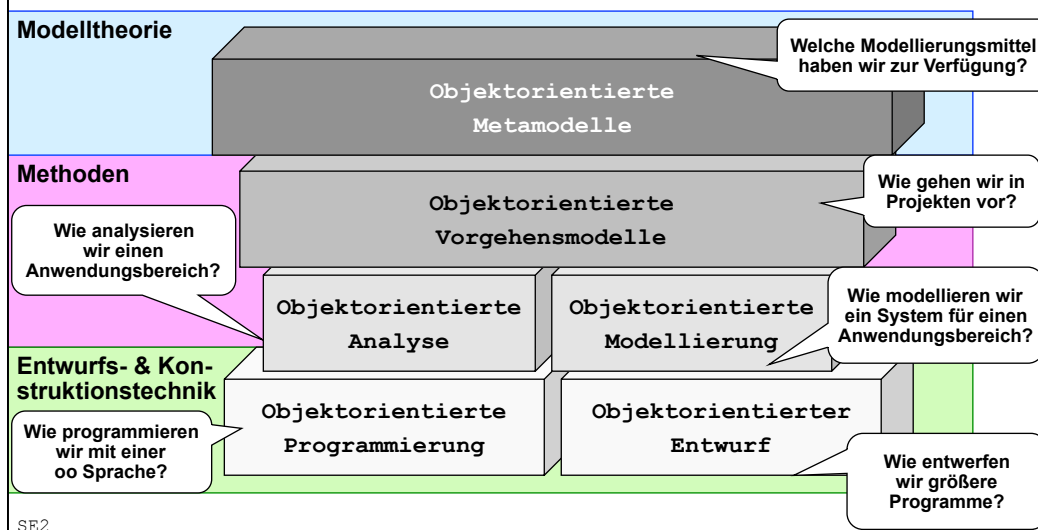
Heinz **Züllighoven** et al.: *Object-Oriented Construction Handbook*.
dpunkt-Verlag, 2004.
[Einige Begriffe und Konzepte für diesen Teil.]

Martin **Hitz**, Gerti **Kappel**, Elisabeth **Kapsammer**, Werner
Retschitzegger: *UML @ Work. Objektorientierte Modellierung mit
UML 2. 3.*, aktualisierte und überarbeitete Auflage, dpunkt.verlag,
2005.
[Mehr zu den UML-Modellen dieses Teils. Siehe auch die
Materialien zu SE2]

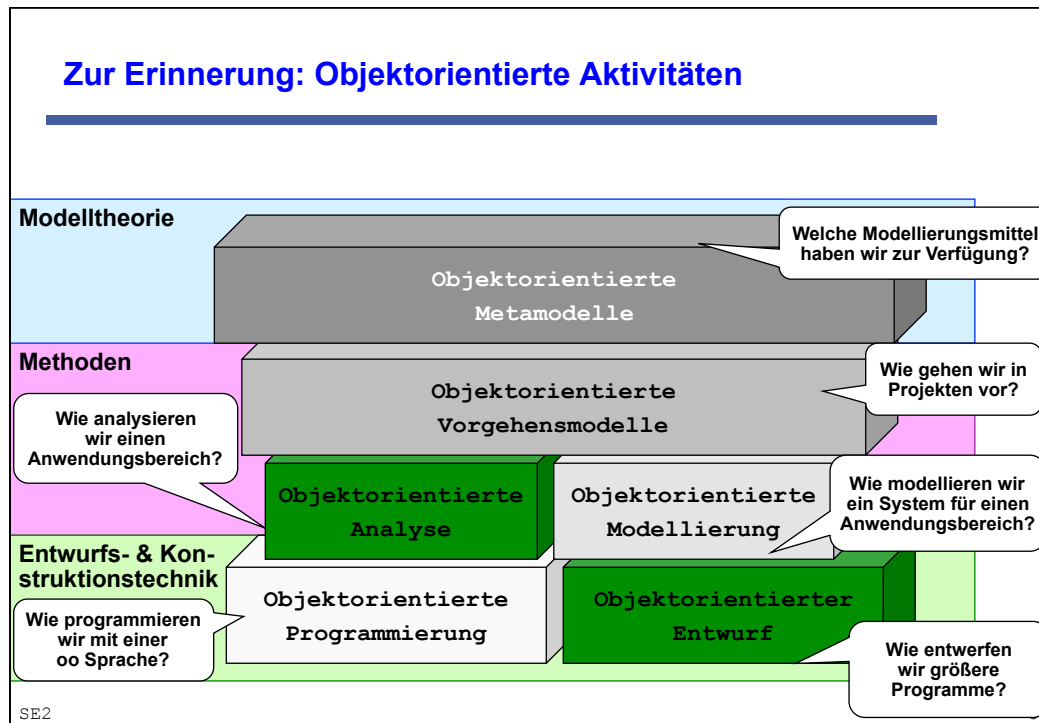
SE2

3

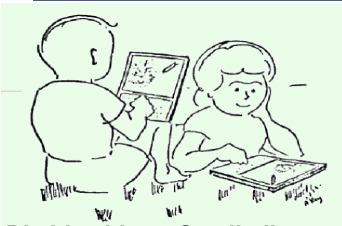
Zur Erinnerung: Objektorientierte Aktivitäten



SE2



Objektorientierte Methoden – das Urgestein



Die Idee hinter Smalltalk:
Software repräsentiert Arbeitsmaterial

Alan Kay:
"Computing is Simulation"
"Do not automate the work you are engaged in, only the materials."

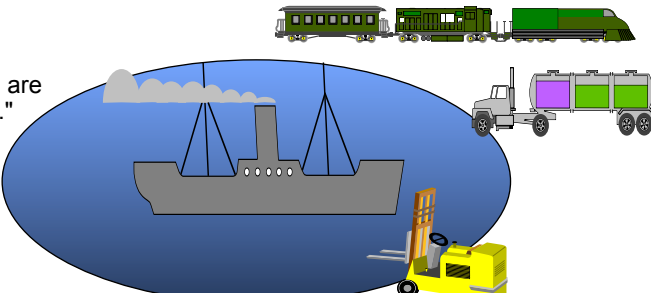
- Reactive engine (1969)
- Personal dynamic media
- The Dynabook

Die Motivation von Simula:
Objekte simulieren Dinge

Ole-Johan Dahl, Kristen Nygaard:

Anwendungen im Bereich Simulation und Operations Research

- Simula I (1964)
- Simula 67



SE2 6

Objektorientierte Methoden – die Anfänge

- Seit Simula (1967) wurden immer wieder Papiere zur objektorientierten Programmierung veröffentlicht. Die Themen "Analyse" und "Entwurf" wurden nur am Rande behandelt.
- Objektorientierte Methoden, die sich mit dem gesamten Software-Entwicklungsprozess beschäftigten, wurden erst in den 80er-Jahren vorgestellt.
- Forschung und Praxis wurden geprägt von:
 - Grady Booch: *Object-Oriented Design* (1982)
 - Bertrand Meyer: *Object-Oriented Software Construction* (1988)
 - Sally Schlaer, Stephen J. Mellor: *Object-Oriented Systems Analysis* (1988)
 - Peter Coad: *Object-Oriented Analysis and Object-Oriented Design* (1990/1991)
 - Rebecca Wirfs-Brock et al.: *Designing Object-Oriented Software* (1990)
 - James Rumbaugh et al.: *Object-Oriented Modeling and Design* (1991)
 - Ivar Jacobson: *Object-Oriented Software Engineering* (1991)

SE2

7

Objektorientierte Methoden – der UML-Einschnitt

- Die *UML* reduzierte die Vielfalt der Methodenlandschaft
 - Grady Booch, James Rumbaugh: *Unified Method* (1994)
 - Booch, Jacobson, Rumbaugh: *UML- the Unified Modeling Language*
 - Jacobson, Booch, Rumbaugh: *The Unified Software Development Process (UP)*
 - Mary Loomis, Jim Odell et al. (im Auftrag der Object Management Group - OMG): *UML 1.0* (1997)
- Aktuell erscheinen im wesentlichen Arbeiten zu einzelnen Aspekten der Objektorientierung wie
- Agile Vorgehensweisen
 - Entwurfsmuster
 - Testen und Qualitätssicherung (Metriken)

SE2

8

Struktur- und anwendungsorientierte Herangehensweise

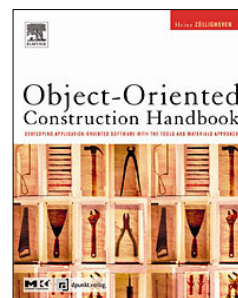
- Aus unserer Sicht gibt es zwei grundlegende Herangehensweisen bei den objektorientierten Methoden:
 - **Anwendungsorientiert:**
Der Einsatzkontext wird in seiner fachlichen Dynamik aus Sicht der Anwender betrachtet. Der mögliche und sinnvolle Umgang mit den fachlichen Gegenständen in Arbeitsprozessen wird analysiert.
 - **Strukturorientiert:**
Der Einsatzkontext wird eher technisch und strukturell betrachtet. Die fachlichen Gegenstände werden auf ihre relevanten Daten untersucht, die im IT-System gespeichert werden sollen.
- Anwendungs- oder strukturorientierte Vorgehensweisen bestimmen Analyse, Modellierung und Konstruktion. Sie führen zu unterschiedlichen Softwaresystemen mit verschiedenen Benutzungsmodellen.
- Wir stellen im Folgenden im Wesentlichen eine anwendungsorientierte Vorgehensweise vor.
- Dabei betrachten wir vor allem interaktive Anwendungssoftware.

SE2

9

Der Werkzeug & Material-Ansatz: eine anwendungsorientierte OO-Methode

- Der **Werkzeug & Material-Ansatz (WAM)** wurde über viele Jahre in Forschung und Praxis entwickelt und ist eine wesentliche methodische Grundlage des Arbeitsbereichs Softwaretechnik.
- Er gibt eine Anleitung zur Entwicklung vor allem interaktiver Softwaresysteme.
- Grundprinzipien sind
 - Anwendungsorientierung
 - Strukturähnlichkeit zwischen Gegenstandsbereich und Anwendungssystem



Züllighoven, H., et. al.,
[Object-Oriented Construction Handbook](http://www.dpunkt.de/buch/3-89864-254-2.html),
 dpunkt.verlag/Copublication with Morgan-Kaufmann,
 Oktober 2004, 544 Seiten,
<http://www.dpunkt.de/buch/3-89864-254-2.html>

SE2

10

Anwendungsorientierung von Software

- **Anwendungsorientierung** ist aus unserer Sicht der Schlüssel zu einer qualitativ hochwertigen Softwareentwicklung.
- Denn Mitarbeiter eines Unternehmens sollen ihre **Aufgaben mithilfe der Software einfach und angemessen erledigen** können.
- Frage:
 - Wie muss Anwendungssoftware gestaltet werden, um **Menschen bei ihren Aufgaben am Arbeitsplatz** optimal zu **unterstützen**?

SE2

11

Anwendungsorientierung und der WAM-Ansatz

- Der **WAM-Ansatz** ist vor allem auf Software ausgerichtet, die von **Anwendern bei ihrer täglichen Arbeit** eingesetzt wird.
- Software soll den Anwendern helfen, **ihre verschiedenen Aufgaben zu erledigen** und sie flexibel bei den komplexen Geschäftsprozessen unterstützen, in die sie eingebunden sind.
- **Aufgaben und Prozesse verändern sich häufig** und gute **Software** muss entsprechend **anpassungsfähig** sein.

SE2

12

Was sind Merkmale der Anwendungsorientierung?

- a) Entwickler müssen die Aufgaben der Anwender verstehen
- b) Anwender müssen die Aufgaben der Entwickler verstehen
- c) Die Auswahl der Programmiersprache und der Technologie ist zentral
- d) Der Zweck eines Programms ist entscheidend
- e) Anwendungssoftware soll die Aufgaben und Arbeitsprozesse unterstützen
- f) Aufgaben und Arbeitsprozesse müssen an die neuen Möglichkeiten der Software angepasst werden.

SE2

13

Was sind Merkmale der Anwendungsorientierung?

- a) Entwickler müssen die Aufgaben der Anwender verstehen
- b) Anwender müssen die Aufgaben der Entwickler verstehen
- c) Die Auswahl der Programmiersprache und der Technologie ist zentral
- d) Der Zweck eines Programms ist entscheidend
- e) Anwendungssoftware soll die Aufgaben und Arbeitsprozesse unterstützen
- f) Aufgaben und Arbeitsprozesse müssen an die neuen Möglichkeiten der Software angepasst werden.

SE2

14

Anwendungsorientierte Analyse des Gegenstandsbereichs

- **Anwendungsorientierung** bedeutet, dass die **Entwickler** die **Ziele** und **Zwecke** der Aktivitäten im Anwendungsbereich verstehen müssen; dazu analysieren sie die **fachlichen Aufgaben**.
- Um die **fachlichen Aufgaben** zu identifizieren und zu verstehen, analysieren die Entwickler die **Prozesse**, durch die die fachlichen Aufgaben erledigt werden.
- Um die Prozesse zu verstehen, betrachten die Entwickler die Art und Weise, wie im Rahmen dieser Prozesse mit **Gegenständen gearbeitet** wird.
- Die Analyse betrachtet den **einzelnen Arbeitsplatz** und die **Kooperation zwischen Arbeitsplätzen**.

SE2

15

Begriffe der anwendungsorientierten Analyse



Der **Anwendungsbereich** ist meist eine Organisation oder eine Abteilung.



Aufgaben werden in **Arbeitsprozessen** von verschiedenen **Personen** erledigt.



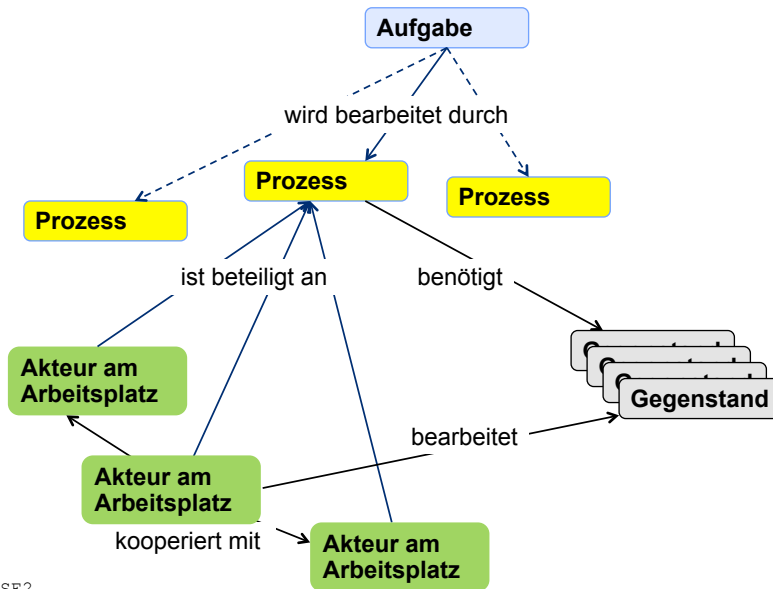
Personen *bearbeiten* am **Arbeitsplatz** die entsprechenden **Unterlagen** oder **Materialien**. Dabei kooperieren sie.



Arbeitsmaterial wird mit geeigneten **Mitteln** bearbeitet.

16

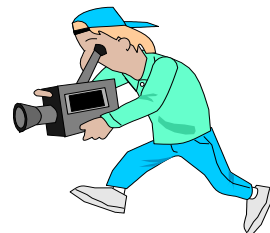
Anwendungsorientierte Zusammenhänge im Anwendungsbereich



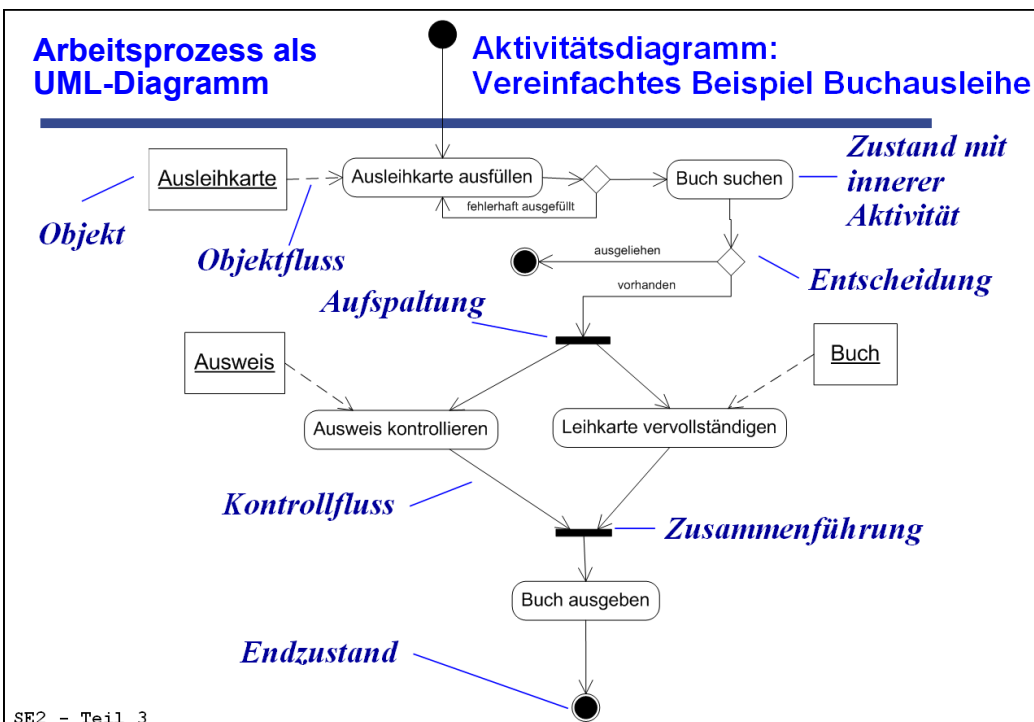
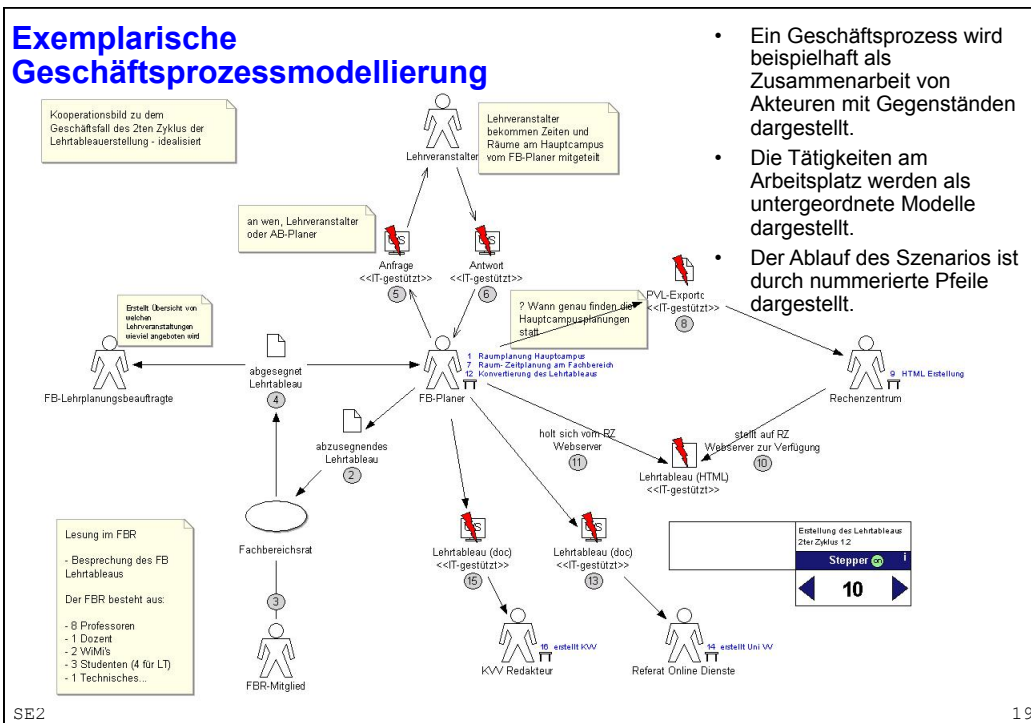
Hilfsmittel bei der Analyse des Anwendungsbereichs



- Interviews mit Anwendungsexperten
- Exemplarische Geschäftsprozessanalyse
- Rollenspiele
- „Teilnehmende Beobachtung“ (Hospitieren) vor Ort
- Ethnographische evtl. videogestützte Analysen



Teil 2: Modellierung und Entwurf interaktiver Softwaresysteme: Muster, Regeln, Rahmenwerke



Was gehört zur anwendungsorientierten Analyse?

- a) Die Entwickler analysieren die Arbeitsprozesse der Anwender.
- b) Die Anwender beschreiben ihre Arbeitsprozesse in einem eigenen Dokument.
- c) Die Begriffe des Anwendungsbereichs sind Grundlage der Entwicklung.
- d) Die Begriffe des Anwendungsbereichs müssen durch die Entwickler vereinheitlicht werden.
- e) Arbeitsprozesse werden von den Entwicklern in Modellen wie der UML beschrieben und dann mit den Anwendern diskutiert.
- f) Arbeitsprozesse werden als Kooperation zwischen Akteuren mit den verschiedenen Materialien beschrieben.
- g) Aus den Arbeitsprozessen werden die unterschiedlichen Datenflüsse herausgezogen und beschrieben.

SE2

21

Was gehört zur anwendungsorientierten Analyse?

- a) Die Entwickler analysieren die Arbeitsprozesse der Anwender.
- b) Die Anwender beschreiben ihre Arbeitsprozesse in einem eigenen Dokument.
- c) Die Begriffe des Anwendungsbereichs sind Grundlage der Entwicklung.
- d) Die Begriffe des Anwendungsbereichs müssen durch die Entwickler vereinheitlicht werden.
- e) Arbeitsprozesse werden von den Entwicklern in Modellen wie der UML beschrieben und dann mit den Anwendern diskutiert.
- f) Arbeitsprozesse werden als Kooperation zwischen Akteuren mit den verschiedenen Materialien beschrieben.
- g) Aus den Arbeitsprozessen werden die unterschiedlichen Datenflüsse herausgezogen und beschrieben.

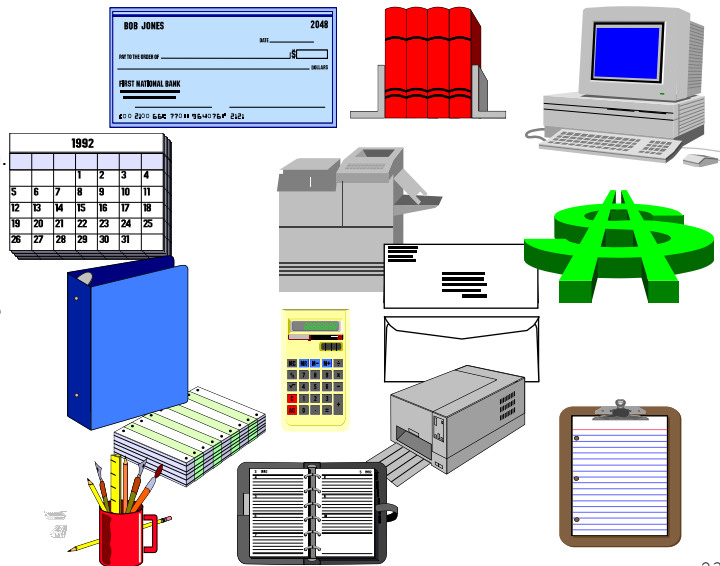
SE2

22

WAM: Fachliche Gegenstände als Ausgangspunkt

- Wir orientieren wir uns an den Gegenständen, die zur Erledigung der fachlichen Aufgaben verwendet werden.
- Diese Gegenstände sind ein guter Ausgangspunkt zur Beantwortung der Frage:

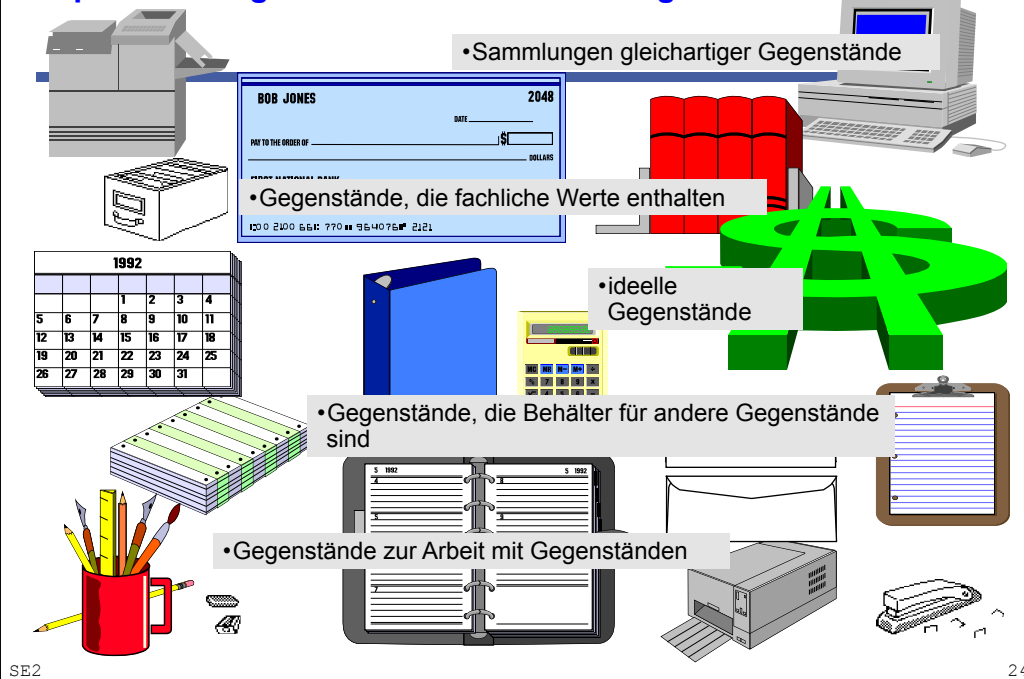
Wie können Arbeitsprozesse am Arbeitsplatz sinnvoll unterstützt werden?



SE2

23

Beispiele für Gegenstände der Modellierung



SE2

24

Entwurf nach WAM: Strukturähnlichkeit

- Fachliche Begriffe und Gegenstände sowie die relevanten Geschäftsprozesse bestimmen den Entwurf der Software.
- Das anwendungsfachliche Begriffsgebäude soll sich in den Elementen der Anwendungssoftware und in der Struktur der Softwarearchitektur widerspiegeln.
- Dadurch finden Anwender die Gegenstände ihrer Arbeit und die Begriffe ihrer Fachsprache im Anwendungssystem wieder. Sie können ihre Arbeit anhand ihrer Erfahrung organisieren.
- Die Entwickler werden beim Entwurf und der Weiterentwicklung der Software unterstützt. Sie werden angeleitet, Architekturen fachlich zu strukturieren und die Softwareelemente bei fachlichen und softwaretechnischen Änderungen mit Anwendungskonzepten in Beziehung zu setzen.

SE2

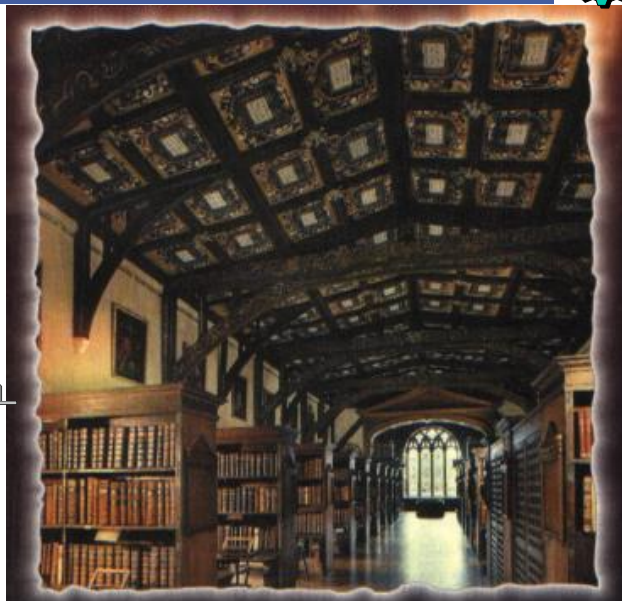
25

Das Bibliotheksbeispiel

Thema: Eine Bibliothek

- Strukturorientiertes Beispiel
- Anwendungsorientiertes Beispiel

– Budde, Kühlenkamp, Sylla, Züllighoven: *Bib – ein Bibliographie-System*. In: H.J. Hoffmann (Hrsg.): *Smalltalk verstehen und anwenden*. München, Hanser, 1987.
– Das anwendungsorientierte Bibliotheksbeispiel wurde über viele Jahre bei SWT in LVs (z.B. STE) verwendet.



SE2

26

Ein strukturorientiertes Bibliothekssystem

- **Die Aufgabenstellung:**
Ein Bibliothekssystem ist ein Hilfsmittel, um Schriften (z.B. Bücher oder Artikel) zu verwalten. Diese werden durch bibliographische Einträge charakterisiert. Jeder bibliographische Eintrag enthält Daten, die eine Schrift identifizieren.
- **Vorgehensweise bei Analyse und Entwurf:**
 - Wie kann das System konstruiert werden?
 - Welche relevanten fachlichen Daten sollen im System verwaltet werden?



SE2

27

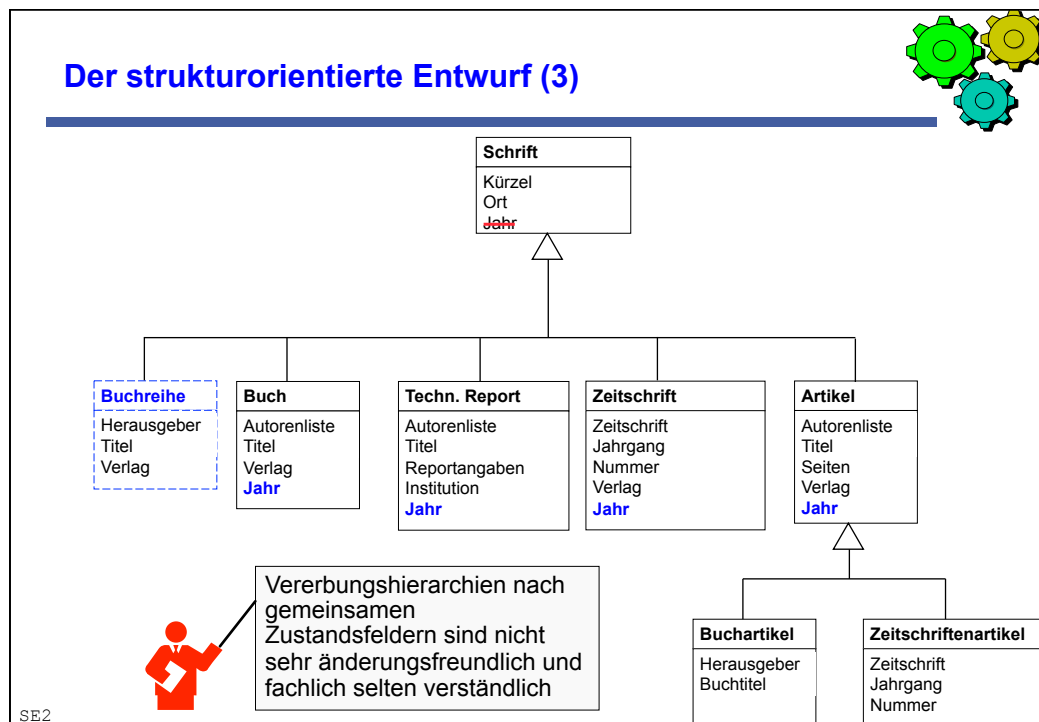
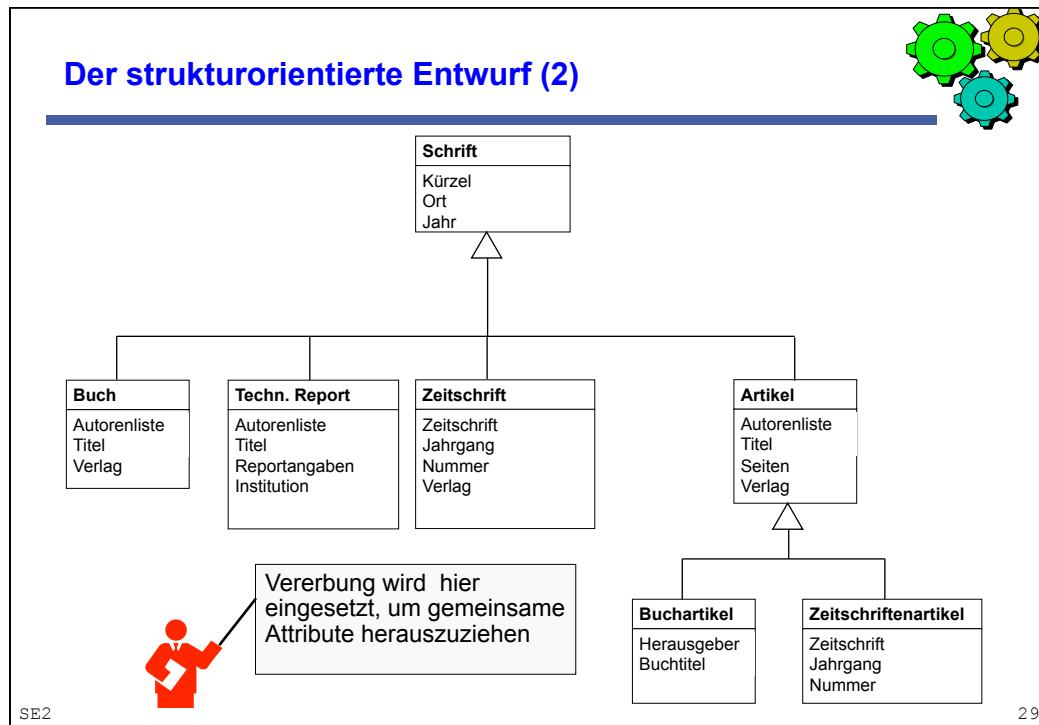
Der strukturorientierte Entwurf (1)

Buch	Techn. Report	Zeitschrift
Kürzel Autorenliste Titel Verlag Ort Jahr	Kürzel Autorenliste Titel Reportangaben Institution Ort Jahr	Kürzel Zeitschrift Jahrgang Nummer Verlag Ort Jahr
Buchartikel	Zeitschriftenartikel	
Kürzel Autorenliste Titel Seiten Herausgeber Buchtitel Verlag Ort Jahr	Kürzel Autorenliste Titel Seiten Zeitschrift Jahrgang Nummer Verlag Ort Jahr	

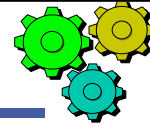
- **Leitfrage bei der strukturorientierten Analyse:**
Welche Daten charakterisieren die Gegenstände eines Bibliothekssystems?

SE2

28



Das anwendungsorientierte Bibliothekssystem



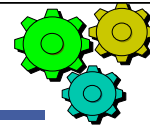
- **Die Aufgabenstellung:**
Die Aufgabe der Bibliothek ist das Sammeln, Erschließen und Benutzbarmachen von Literatur für die Angehörigen eines Fachbereichs. Die Arbeit mit der Bibliothek soll rationalisiert werden. Der Verwaltungsaufwand ist so weit wie möglich zu reduzieren. Dabei sind Routineaufgaben zu minimieren.
- **Vorgehensweise bei Analyse und Entwurf:**
 - Was sind die Aufgaben in einer Bibliothek?
 - Welche Aufgaben sollen durch das Anwendungssystem unterstützt werden?



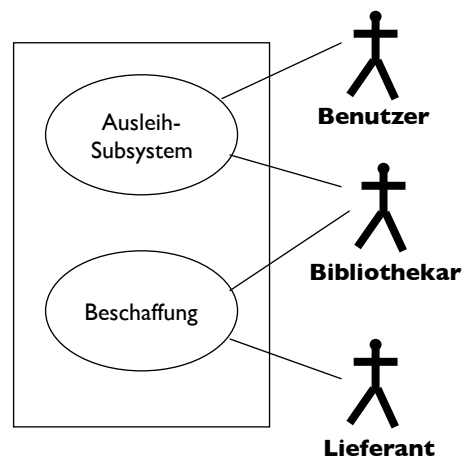
SE2

31

Use Case: Beispiel Bibliothek

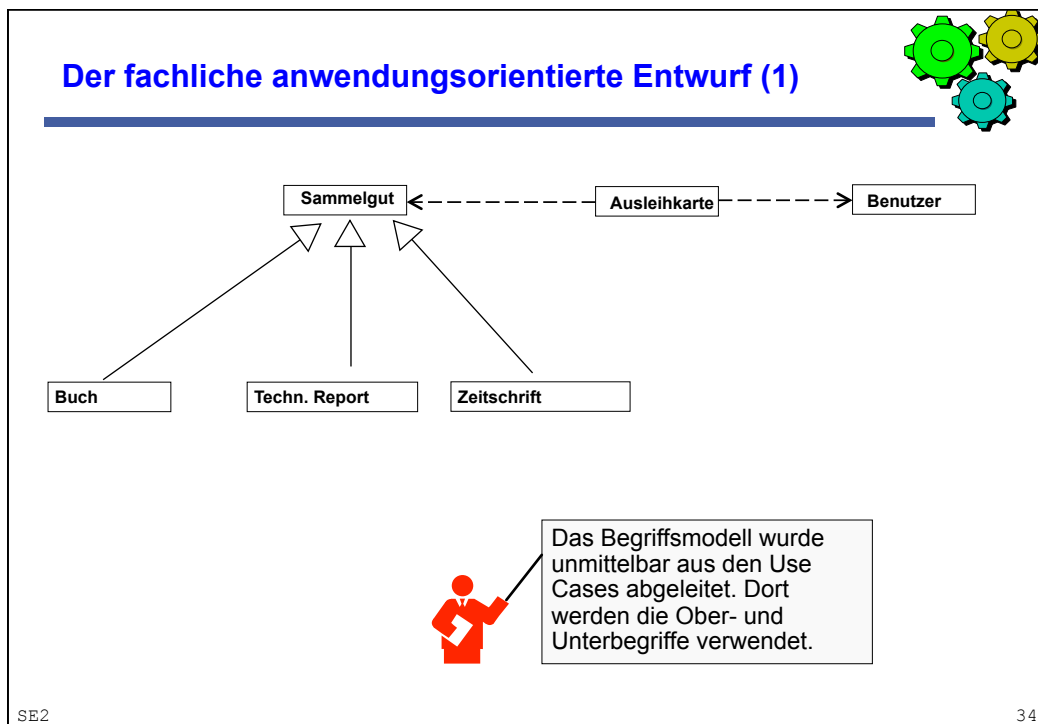
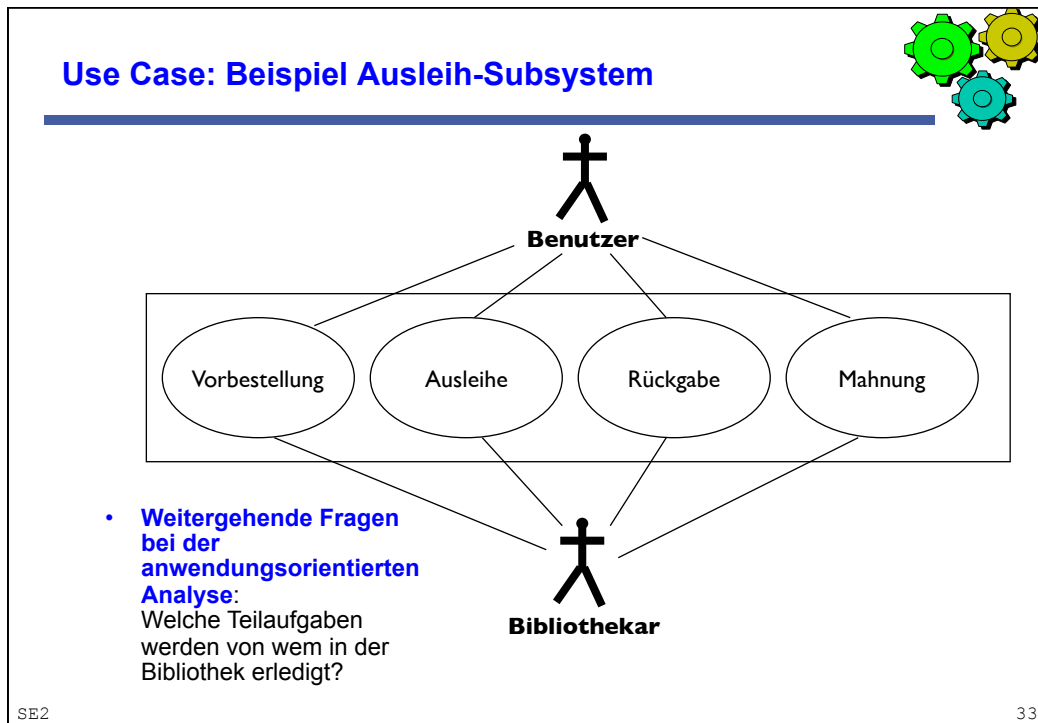


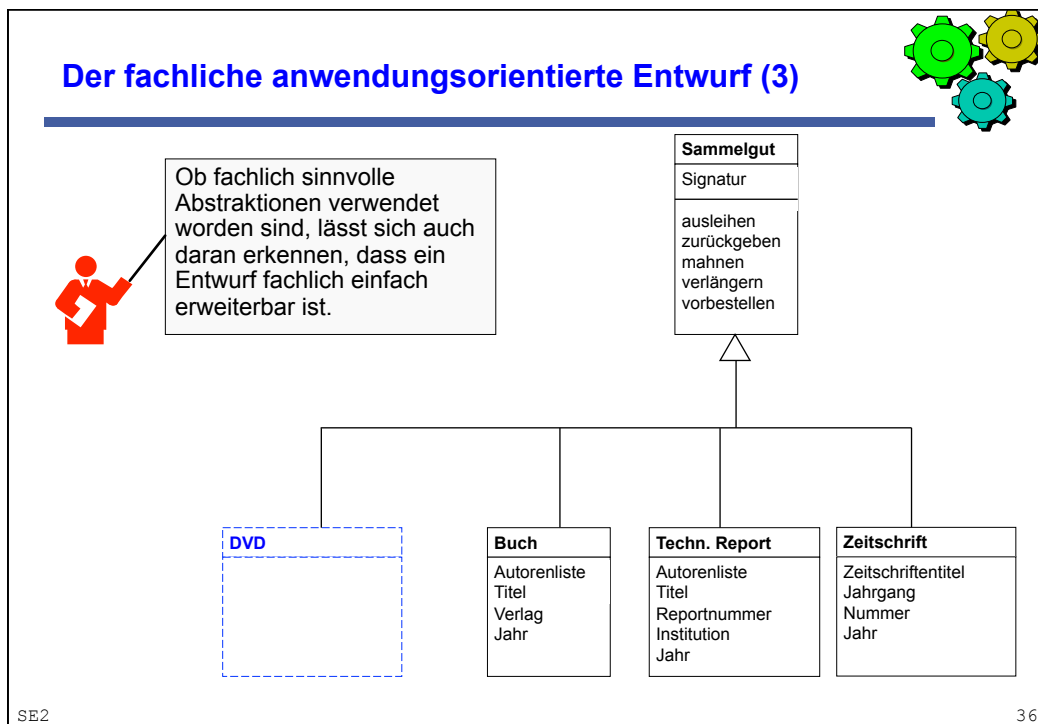
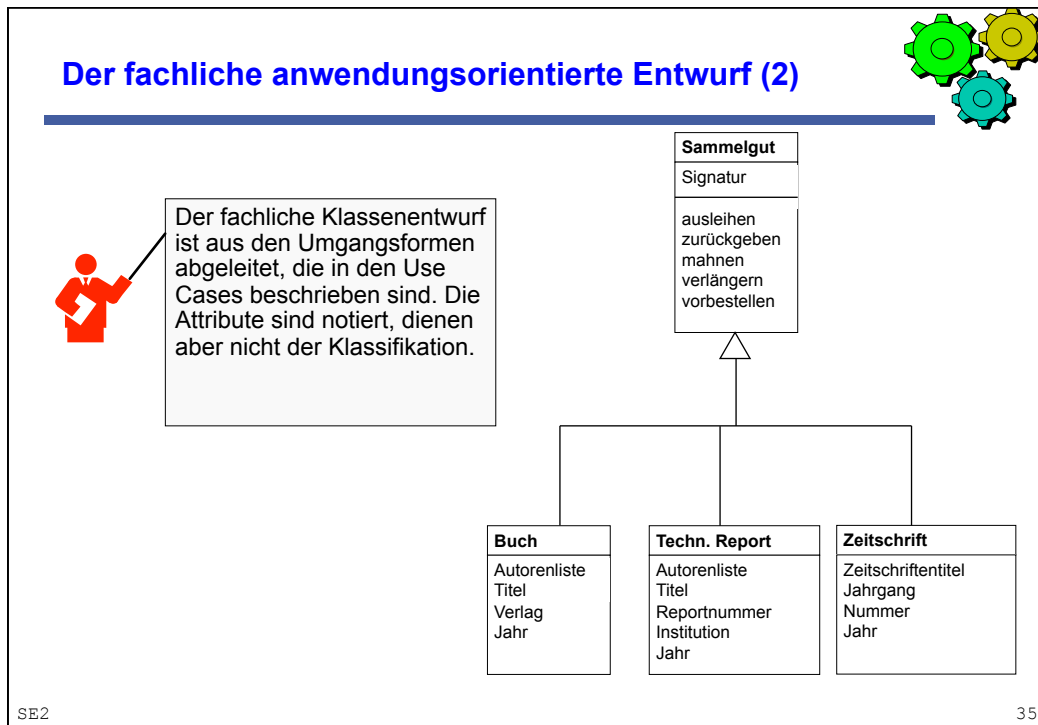
- **Leitfrage bei der anwendungsorientierten Analyse:**
Wer ist an welchen Aufgaben in der Bibliothek beteiligt?



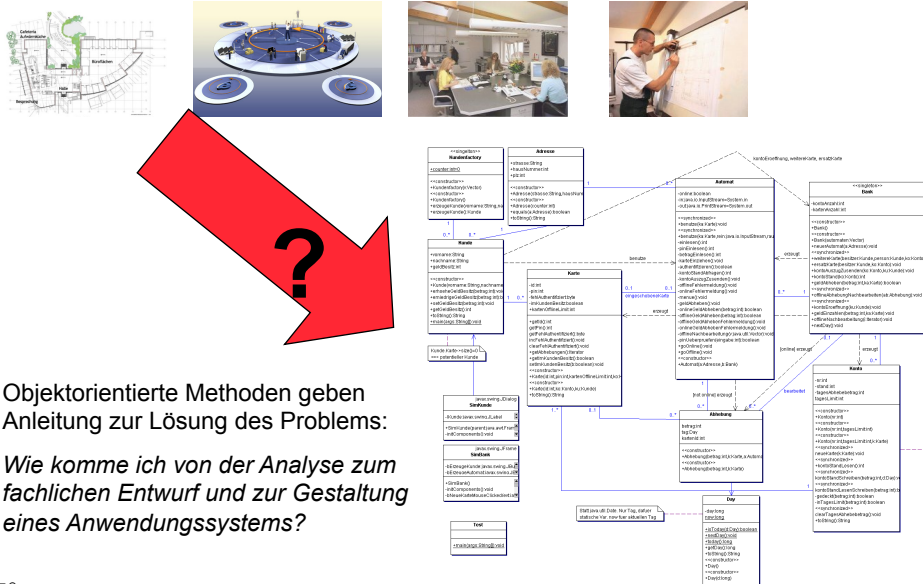
SE2

32





Vom fachlichen Entwurf zur Gestaltung des Anwendungssystems



SE2

37

Vom fachlichen Entwurf zur Gestaltung des Anwendungssystems



SE2

Entwurfsmetaphern

Eine **Entwurfsmetapher** überträgt einen „handgreiflichen“ bekannten Gegenstand (z.B. Papierkorb oder Ordner) in das Benutzungsmodell von Anwendungssoftware.

Eine Entwurfsmetapher **strukturiert die Wahrnehmung** der Oberflächenelemente einer Anwendung und liefert Begriffe, um darüber zu reden. Sie leitet die **Vorstellung und Kommunikation** über das, was fachlich analysiert, modelliert und technisch realisiert werden soll.

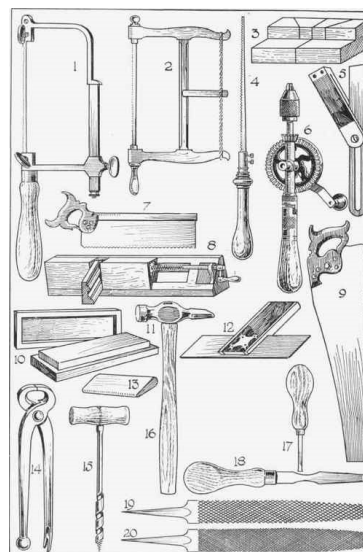
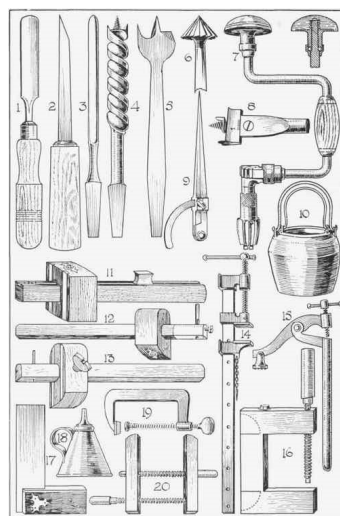
Eine Entwurfsmetapher soll die Handhabung und Funktionalität von Elementen des **Benutzungsmodells** für die Beteiligten verständlicher machen.

Zentrale Entwurfsmetaphern des **WAM-Ansatzes** sind **Werkzeug, Material, Automat, Behälter und Arbeitsplatz**. Sie haben immer auch eine technische Konstruktionsanleitung und Entwurfsmustern.

SE2

39

Annäherung an Werkzeug: Was ist ein Werkzeug?



SE2

40

Annäherung an Werkzeug: Der Arbeitsvorgang definiert, was ein Werkzeug ist



SE2



41

WAM: Menschen bearbeiten Material mit Werkzeugen

- **Arbeiten** bedeutet oft, dass Menschen einen Arbeitsgegenstand mit geeigneten Werkzeugen bearbeiten. Dies gilt nicht nur im Handwerk.
- **Materialien** sind also Gegenstände, die im Rahmen einer Aufgabe Teil des Arbeitsergebnisses werden. Sie werden durch Werkzeuge und Automaten bearbeitet und verkörpert fachliche Konzepte. Viele Eigenschaften vorhandener Arbeitsgegenstände lassen sich sinnvoll auf Softwarematerialien übertragen.



SE2

- **Werkzeuge** sind Gegenstände, mit denen Menschen im Rahmen einer Aufgabe Materialien verändern oder sondieren. Sie eignen sich meist für verschiedene Zwecke und unterschiedliche Materialien. Sie müssen geeignet gehandhabt werden. Werkzeuge vergegenständlichen wiederkehrende Arbeitshandlungen. Eine direkte Abbildung von (manuellen) Werkzeugen in Software ist selten sinnvoll.

42

Was sind Merkmal des Entwurfs nach WAM?

- a) Die Gegenstände im Arbeitsprozess spielen eine wichtige Rolle.
- b) Wichtig sind die Daten, die in der Arbeit verwendet werden.
- c) Die Begriffe des Anwendungsbereichs sollen hauptsächlich in den Menüs verwendet werden.
- d) Die Begriffe des Anwendungsbereichs sollen in den Programmen verwendet werden.
- e) Anwendungssoftware soll nach interaktiver Oberfläche (GUI), Fachlogik auf dem Server und Datenbank strukturiert werden.
- f) Ein Gegenstand kann anhand objektiver Merkmale als Werkzeug identifiziert werden.
- g) Werkzeuge und Materialien sind wesentliche Elemente jedes Arbeitsprozesses.

SE2

43

Was sind Merkmal des Entwurfs nach WAM?

- a) Die Gegenstände im Arbeitsprozess spielen eine wichtige Rolle.
- b) Wichtig sind die Daten, die in der Arbeit verwendet werden.
- c) Die Begriffe des Anwendungsbereichs sollen hauptsächlich in den Menüs verwendet werden.
- d) Die Begriffe des Anwendungsbereichs sollen in den Programmen verwendet werden.
- e) Anwendungssoftware soll nach interaktiver Oberfläche (GUI), Fachlogik auf dem Server und Datenbank strukturiert werden.
- f) Ein Gegenstand kann anhand objektiver Merkmale als Werkzeug identifiziert werden.
- g) Werkzeuge und Materialien sind wesentliche Elemente jedes Arbeitsprozesses.

SE2

44

Beispiel: der Pausenplaner mit Material Pausenplan

neuer Pausenplan > - Planungstabelle

Pausenplan
Bearbeiten
Werkzeuge
Optionen
Hilfe

vollständig
korrekt

Pausenplan

	01.07.2005	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
Kleiner Schulhof	Sarah	Sarah	Nathalie	Karla	Markus	
1. Hälfte	Mike	Knut	Rosa	Marion	Vera	
	Daniela	Timo	Linda	Daniela	Knut	
Kleiner Schulhof	Markus	Malte	Karla	Nathalie	Joachim	
2. Hälfte	Rosa	Richard	Richard	Vera	Shani	
	Vera	Richard	Timo	David	Timo	
Großer Schulhof	Marc	Klara	Siglinde	Kai	Klara	
1. Hälfte	Niels	Agathe	David	Agathe	Mike	
	Agathe		Niels		Rosa	
Großer Schulhof	Joachim	Marc	Sigfrid	Malte	Sigfrid	
2. Hälfte	Shani	Daniela	Agathe	Knut	George	
	George	David	Mike	George	Richard	
Spätaussicht						
	Karla	Kai	Oliver	Marc	Sigfrid	Nathalie
		Joachim	Malte	Klara	Siglinde	
Kleiner Schulhof	Oliver	Markus	Oliver	Markus	Sarah	
Vertretung	Richard	George	Shani	Rosa	Marion	
Großer Schulhof	Siglinde	Karla	Kai	Marc	Malte	
Vertretung	David	Linda	Vera	Niels	Daniela	

Erstkräfte

	Bezei.	Ausl.	Aufs.	Vertr.	Spät
Joachim	100	0	0	0	
Kai	75%	1	0	0	
Karla	100	0	0	0	
Klara	75%	0	1	0	
Malte	100	0	0	0	
Marc	100	0	1	-1	
Markus	80%	0	0	1	
Nathalie	75%	0	1	0	
Oliver	60%	2	0	0	
Rosa	75%	0	0	1	

Zweitkräfte

	Bezei.	Ausl.	Aufs.	Vertr.	Spät
Agathe	100	-2	2	---	
Daniela	100	-1	1	---	
David	100	-1	1	---	
George	100	-1	1	---	
Knut	100	0	0	---	
Linda	50%	1	1	---	
Marion	50%	1	1	---	
Mike	100	-2	2	---	
Niels	100	0	0	---	

Pausenplandetails

Rahmenbedingung:

☐

Recycler

Aufsichtskraftdetails

SE2

45

Beispiel: Der Portmonitor stellt unterschiedliche Werkzeuge und Materialien am Arbeitsplatz bereit

The screenshot shows the HPA Port Monitor software interface. The main window displays a map of a harbor area with various structures and water bodies. A red box highlights a specific area labeled 'Kaiserspeicher'. The right sidebar shows a list of objects with their descriptions, locations, and types. The bottom section shows a list of documents and a preview of the selected document, 'Kaiserspeicher.jpg'.

SE2

46

Zusammenfassung & Ausblick



- Es gibt verschiedene Methoden der objektorientierten Modellierung und Konstruktion; **UML** und **UP** sind die kommerziell derzeit dominierenden Ansätze.
- Wir haben mit dem **Werkzeug & Material-Ansatz** eine anwendungsorientierte Variante vorgestellt.
- Dabei bilden die **Aufgaben im Einsatzkontext** und die dabei verwendeten **Gegenstände** den Mittelpunkt.
- **Anwendungsorientierte Ansätze** führen fachlich und technisch zu Entwürfen, die sich von strukturorientierten Ansätzen unterscheiden.
- Für die **einfache Benutzung** ist die Darstellung von bekannten Gegenständen sehr hilfreich.
- Zu einem anwendungsorientierten Ansatz passen das **Vertragsmodell** und ein **fachlich orientiertes Zustandsmodell**.

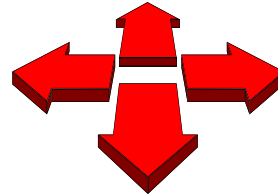
SE2

47



Strukturierung interaktiver Anwendungssysteme

- Ausgangspunkt: Interaktive Softwaresysteme
- Motivation für Entwurfsregeln: Softwarequalität
- Konstruktionsregeln für einfache interaktive Systeme



SE2

49

Interaktive Softwaresysteme

- Interaktive Softwaresysteme sind gekennzeichnet durch eine **häufige Interaktion** mit der Benutzerin / dem Benutzer.
- Moderne interaktive Systeme bieten eine **grafische Benutzungs-schnittstelle**.
- Der Rechenaufwand auf Seite des Computers für die eigentliche Antwort ist gering; aufwändig ist meist eher die Berechnung ihrer **Darstellung**.



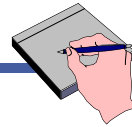
- Im Gegensatz dazu: eine aufwändige Verarbeitung durch den Computer anstoßen, der dann völlig selbstständig rechnen kann.
 - Beispiele: Umwandeln von Video-Dateien von einem Format in ein anderes, Klimasimulationen, Genom-Sequenzierung.



SE2

50

„Reiche“ und „schlanke“ Systeme



- Interaktive Systeme sind seit etlichen Jahren selbstverständlich. Als grundlegende Differenzierung unterscheiden wir
 - **Desktop-Systeme** (auch engl.: **Rich-Clients**): Die Verarbeitung der Anweisungen des Benutzers erfolgt primär auf dem Rechner (Desktop, Laptop, mobiles Gerät), mit dem der Benutzer direkt interagiert.
 - **Web-Systeme** (auch engl.: **Thin-Clients**): Die Verarbeitung erfolgt überwiegend auf einem Server (Web-Server), der Rechner vor Ort dient primär dazu, die Interaktion (meist in einem Browser) zu vermitteln und zu visualisieren.
- Die Grenzen zwischen diesen beiden grundsätzlichen Ansätzen verschwimmen inzwischen immer mehr.

SE2

51

Beispiele für Rich- und Thin-Clients

- | | |
|---|---|
| <ul style="list-style-type: none">• Bekannte Desktop-Systeme:<ul style="list-style-type: none">• MS Office, OpenOffice• Photoshop• iTunes• Eclipse• Umfangreicher Download• Meist explizites Update auf neue Versionen | <ul style="list-style-type: none">• Bekannte Web-Anwendungen:<ul style="list-style-type: none">• Web-Mail• Facebook (auch über Apps)• Google Docs• XING, LinkedIn• Meist nur ein Browser auf Client-Seite notwendig• Update transparent auf dem Server |
|---|---|

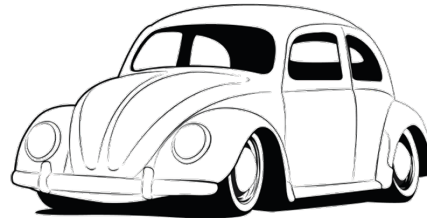
Wir fokussieren in SE2 auf **interaktive Desktop-Systeme**, die Konstruktion von Web-Systemen ist ein fortgeschrittenes Thema für weiterführende Veranstaltungen.

SE2

52

Einfache interaktive Softwaresysteme

- Im Folgenden stellen wir einen Satz an **Entwurfsregeln** vor, die die Konstruktion von **einfachen interaktiven Softwaresystemen** erleichtern.
- Diese Systeme sind **einfach**, weil in ihnen wichtige Eigenschaften von vielen Softwaresystemen **nicht berücksichtigt** sind:
 - **keine Persistenz**: die SE2-Systeme arbeiten **ohne Datenbanken**.
 - **keine Verteilung**: die SE2-Systeme sind **Einzelplatzsysteme**.
- Trotz dieser Vereinfachungen ist die Konstruktion von interaktiven Softwaresystemen immer noch **anspruchsvoll**.
- So anspruchsvoll, dass wir einige **Konstruktionsregeln** für SE2 aufgestellt haben, die ihre Erstellung erleichtern sollen.



SE2

53

Wirklich einfache Software: Qualität egal

„There is software that's easy to write.
That would be software that is small,
written by one person over a short
period of time, used by that one person,
used once and then thrown away.
Everything else is difficult.“



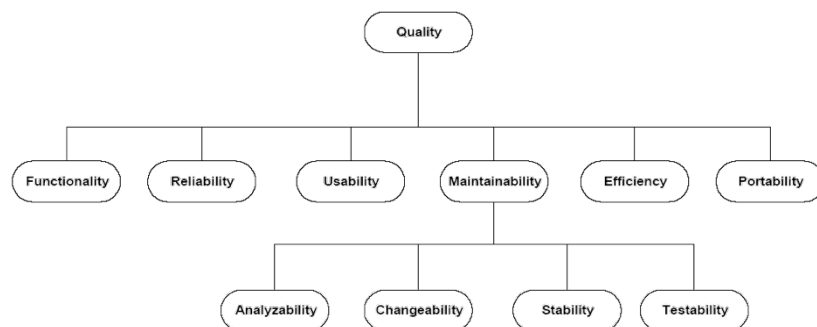
© Deacon, Object-Oriented Analysis and Design, Addison-Wesley 2005

SE2

54

Motivation für Entwurfsregeln: Qualität von Software

- Wir streben nach **möglichst hoher Qualität** bei der Softwareerstellung.
- Die Qualität von Software kann nach B. Meyer differenziert werden in
 - **Äußere bzw. externe Qualität** (Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz) und
 - **Innere bzw. interne Qualität** (Verständlichkeit, Wartbarkeit, Modularität).
- In der Benutzung zählt nur die äußere Qualität, die aber über interne Qualität erreicht und gesichert wird.



SE2

55

Hintergrund: verschiedene Qualitätsbegriffe

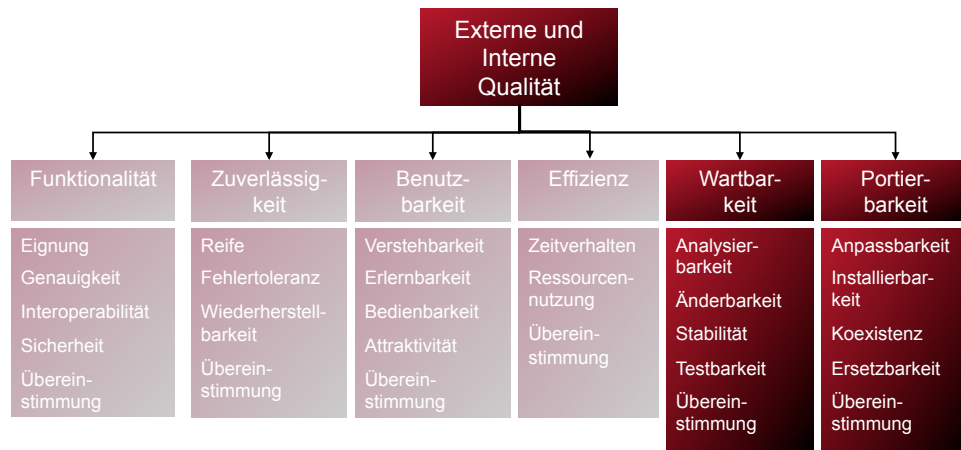
- **Der transzendente Ansatz**
 - Qualität ist universell erkennbar, absolut, einzigartig; sie kann nicht gemessen werden, sie lässt sich nur durch Erfahrung bewerten.
- **Der produktbezogene Ansatz**
 - Objektiv bewertbare Eigenschaften eines Produktes werden spezifiziert und vermessen; Produkte können somit in Rangordnungen gelistet werden.
- **Der benutzerbezogene Ansatz**
 - Der Benutzer legt die Qualität fest; je besser ein Produkt Bedürfnisse befriedigt, desto höher die Qualität; Q. ist somit nicht objektiv bewertbar.
- **Der prozessbezogene Ansatz**
 - Qualität wird durch den Erstellungsprozess bestimmt; ein exakt spezifizierter, detailliert kontrollierter Prozess führt zu einem hochwertigen Produkt.
- **Der Kosten/Nutzen-bezogene Ansatz**
 - Qualität ist eine Funktion von Kosten und Nutzen; Q. heißt, einen bestimmten Nutzen zu einem akzeptablen Preis zu erhalten.

SE2

© Balzert, Lehrbuch der Softwaretechnik

56

Externe und interne Qualität in der ISO 9126



SE2

© Uni Cottbus

57

Zusätzliche Motivation: Software verändert sich

- Software ist keine Prosa, die einmal geschrieben wird und dann unverändert bleibt.
- Software wird **erweitert, korrigiert, gewartet, portiert, adaptiert**, ...
- Diese Arbeit wird von **unterschiedlichen Personen** vorgenommen (manchmal über Jahrzehnte).
- Für Software gibt es deshalb nur zwei Optionen:
 - Entweder sie wird gewartet.
 - Oder sie stirbt.
- **Software, die nicht gewartet werden kann, wird sterben!**



Siehe auch: „Software Aging“ von David Parnas, Keynote auf der ICSE 1994

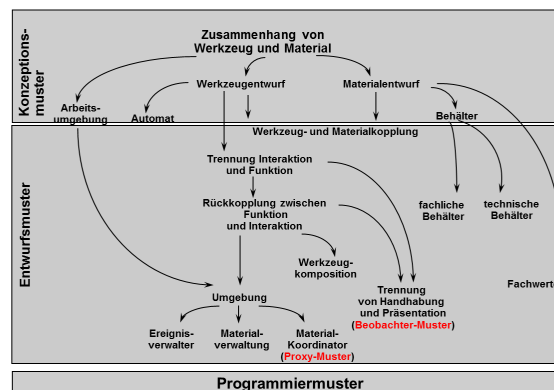
© Barnes, Kölling

SE2

58

Umfangreiche Entwurfsregeln: der WAM-Ansatz

- Der **WAM-Ansatz** umfasst neben einem detaillierten Vorgehensmodell und etlichen Dokumenttypen für die objektorientierte Analyse und Modellierung auch einen umfangreichen Satz an Konstruktionshinweisen für interaktive Systeme.
- Diese Hinweise halten jahrelange Konstruktions-erfahrung aus professionellen Entwicklungsprojekten in Form von Mustern fest, die im WAM-Ansatz in **Konzeptions-**, **Entwurfs-** und **Programmiermuster** differenziert werden.



SE2

59

Überschaubare Regeln: SE2-Entwurfsregeln

- Für eine Lehrveranstaltung im zweiten Semester ist der WAM-Ansatz in allen seinen Ausprägungen deutlich zu umfangreich; die Grundideen des Ansatzes sind jedoch auch für kleine Systeme bereits anwendbar.
- Im Folgenden beschreiben wir deshalb einige an diesem Ansatz orientierte **Entwurfs- und Konstruktionsregeln**, die das Erstellen interaktiver Anwendungen erleichtern sollen.
- Obwohl diese Regeln vom **WAM-Ansatz** motiviert sind, sind sie eigenständig nutzbar und anwendbar; eine umfassende Kenntnis des WAM-Ansatzes ist **keine Voraussetzung** für ihre Nutzung.
- Wir weisen darauf hin, dass Konstruktionsregeln immer die **Gefahr** bergen, sklavisch befolgt zu werden (wie auch Quelltextkonventionen). Dies ist insbesondere deshalb problematisch, weil Regeln selten alle möglichen Situationen abdecken können; diese Regeln sollen in erster Linie dazu dienen, **Ungeübten** die softwaretechnische Konstruktion zu erleichtern.

SE2

60

Exkurs: Stufen beim Lernen

- **Alistair Cockburn** beschreibt in seinem Buch „Agile Software Development“ **drei Stufen in der persönlichen Entwicklung auf dem Weg zu kompetenter Softwareentwicklung** und orientiert sich dabei an asiatischen Kampfsportarten wie dem **Aikido**; diese Stufen finden aber auch beispielweise Anwendung beim japanischen Brettspiel **Go**.
- Die Stufen sind
 - **Shu** – *beschützen, verteidigen, einhalten, befolgen*
 - Das Lernen der Form / **Kopieren**
 - **Ha** – *zerreißen, durchbrechen*
 - Das Überschreiten der Form / **Abweichen**
 - **Ri** – *sich entfernen, trennen, abschneiden*
 - Eigene Wege finden / **Freie Verwendung**
- In SE2 ist der Fokus auf dem **Erlernen guter Form**; in der konkreten Anwendung wird es immer wieder Bedarf für Abweichung geben; letztlich sollen die Regeln aufgrund reicher eigener Erfahrung überflüssig werden.

守	shu – embracing the kata
破	ha – diverging from the kata
離	ri – discarding the kata

SE2

61

Grundlegend: Trenne Fachlogik und Technik



- **Anwendungsfachliche Klassen**, die vor allem die Fachlogik modellieren, sollten deutlich von rein **technisch motivierten Klassen** unterscheidbar sein.



SE2

62

Modellierung anwendungsfachlicher Klassen

- Nah am Anwendungsbereich
- Abstraktionen der jeweiligen Domäne
- Mit den Anwendern diskutierbar
- Oft sehr spezifisch, in jedem Projekt anders
- Wiederverwendung nur möglich, wenn domänenspezifische Klassen bereits vorliegen.
- Andere häufig verwendete Begriffe: **Domain Objects**, **Business Objects**

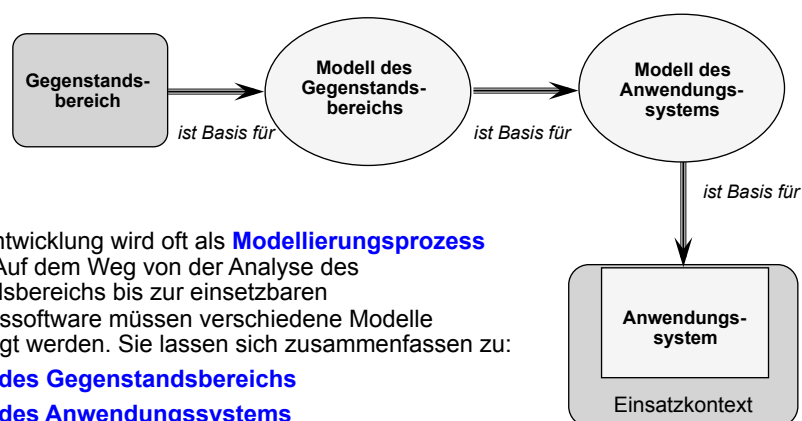
Aber: innerhalb
eines Projektes oft
das Stabilste!



SE2

63

Wiederholung: Modelle in der Softwareentwicklung



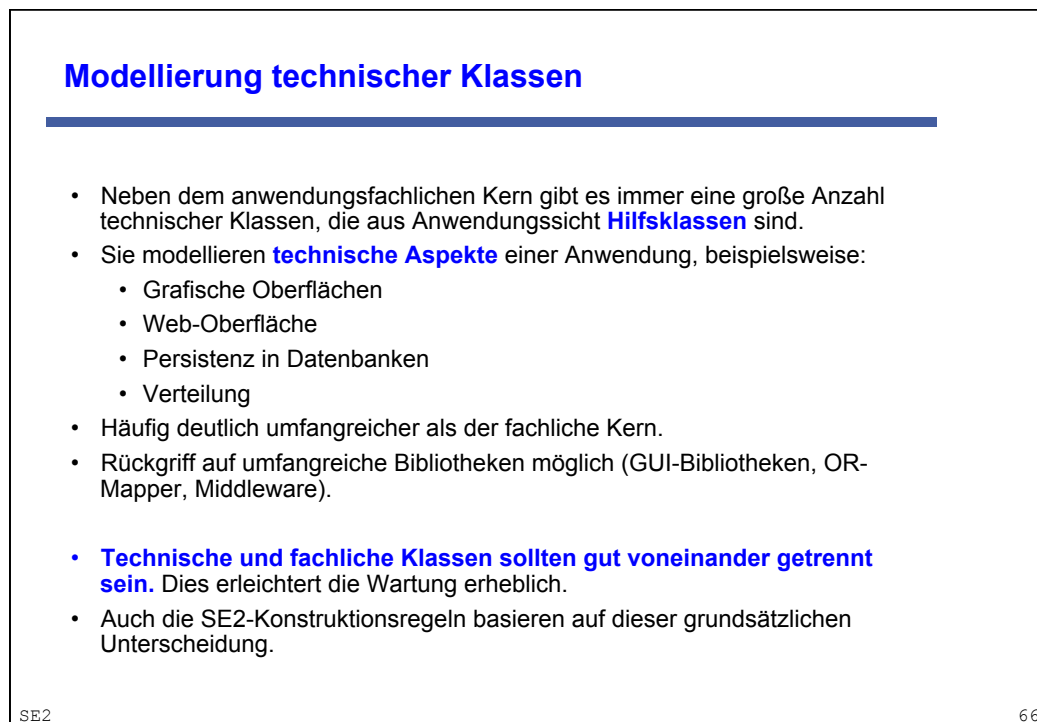
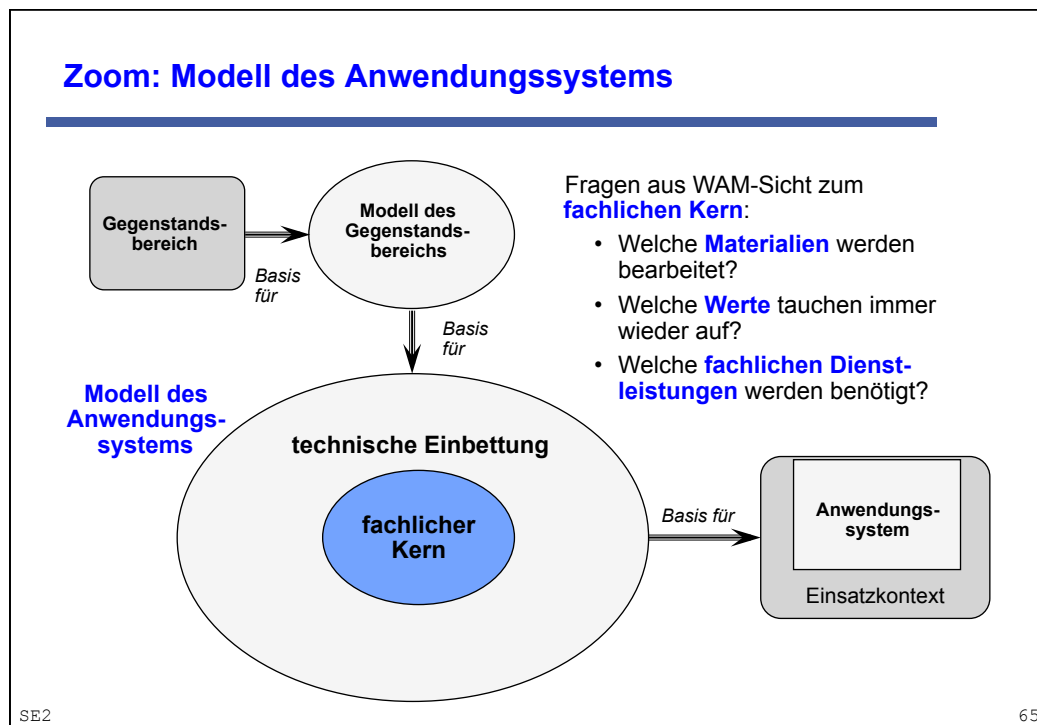
- Software-Entwicklung wird oft als **Modellierungsprozess** betrachtet. Auf dem Weg von der Analyse des Gegenstandsbereichs bis zur einsetzbaren Anwendungssoftware müssen verschiedene Modelle berücksichtigt werden. Sie lassen sich zusammenfassen zu:
 - **Modell des Gegenstandsbereichs**
 - **Modell des Anwendungssystems**
- Objektorientierte Methoden unterscheiden sich auch darin, ob und wie sie diese Modelle erstellen.



Der Quellcode ist Teil des Modells des Anwendungssystems; das ablaufende Programm ist Teil des Anwendungssystems.

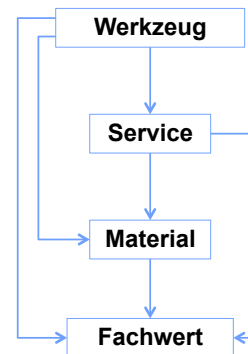
SE2

64



Die SE2-Entwurfsregeln

- Die **SE2-Entwurfsregeln** benennen vier **Elementtypen**, aus denen sich ein interaktives System zusammensetzt:
 - Materialien** realisieren veränderliche, anwendungsfachliche Gegenstände.
 - Fachwerte** sind anwendungsfachliche Werte; sie sind unveränderlich.
 - Werkzeuge** bieten eine grafische Benutzungsschnittstelle und ermöglichen das interaktive Bearbeiten von Materialien.
 - Services** bieten materialübergreifend fachliche Dienstleistungen an, die systemweit zur Verfügung stehen sollen.



Die Pfeile zeigen die **erlaubten Benutzt-Beziehungen** zwischen den Elementtypen. Jeder Elementtyp kann außerdem Elemente vom eigenen Typ benutzen (hier nicht dargestellt).

SE2

67

Allgemeines zu den Entwurfsregeln

- In seiner programmtechnischen Umsetzung kann ein Elementtyp aus mehreren Klassen bestehen. Dies gilt insbesondere für Werkzeuge.
- Für jeden Elementtyp (Werkzeug, Material, Service, Fachwert) sollte es ein **eigenes Java-Paket** geben. Darunter kann es bei Bedarf noch eine weitere Aufteilung geben, z.B. im Werkzeug-Paket für verschiedene Werkzeuge.
- Die Elementtypen Material, Fachwert und Service werden durch **fachliche Klassen** modelliert, die somit auch eine **fachlich motivierte Schnittstelle** haben; Werkzeuge hingegen bieten ihre fachliche „Schnittstelle“ interaktiv gegenüber dem Benutzer an, die Schnittstellen der implementierenden (technisch motivierten) Klassen sind eher technisch geprägt.
- Für alle fachlichen Klassen gilt:
 - Sie sichern ihre Konsistenz über das Vertragsmodell.
 - Es gibt eine zugehörige Testklasse.

SE2

68

Materialien



- Materialien modellieren **anwendungsfachliche „Gegenstände“**, wie CD, DVD, Videospiel, Kunde. Die modellierten Gegenstände müssen im Anwendungsbereich nicht tatsächlich gegenständlich sein; auch abstrakte Dinge, die fachlich relevant sind, werden als Material modelliert (Bsp.: Versicherungspolice, Konto, Vertragsentwurf, Partitur, Film).
- Materialien werden im Arbeitsprozess mit Werkzeugen erzeugt, bearbeitet und beseitigt. Sie vergegenständlichen **Arbeitsergebnisse**.
- Ein Material hat **ausschließlich fachliche Aufgaben**. Das heißt, dass ein Material keinerlei technische Aufgaben übernimmt, die beispielsweise die Persistenz betreffen oder das UI-Framework (in SE2: Swing).
- Materialien **kennen ausschließlich** andere **Materialien** und **Fachwerte**.
- Jeder fachlich relevante Materialtyp wird durch **eine eigene Klasse** modelliert!
- Für jedes einzelne Material im Anwendungsbereich gibt es nur genau ein passendes Material-Exemplar einer solchen Klasse im Softwaresystem.

SE2

69

Fachwerte



- Bei der Modellierung eines Anwendungsbereichs gibt es immer auch Begriffe, die eher wertartige Dinge beschreiben, wie **Kontonummer** oder **Geldbetrag**.
- Wir beschreiben solche Begriffe über **Fachwerte**.
- Fachwerte sind fachlich motivierte **Werte**. Werte sind ein allgemeineres Konzept, das beispielsweise auch Zahlen und Zeichenketten umfasst.
 - Ein Wert ist **unveränderlich**.
 - Wir beschreiben Werte programmiertechnisch über **Werttypen**.
 - Werttypen sind besondere Typen mit einer **unveränderlichen Wertemenge**; Werte werden somit konzeptuell nicht erzeugt, sondern bei Bedarf aus der Wertemenge ausgewählt.
- Fachwerte bilden die Grundkonstanten in einem Anwendungssystem.
- Wir werden uns Werttypen in einer der nächsten Vorlesungen ausführlich ansehen.

SE2

70

(Fachliche) Services



- Services bieten **fachliche Dienstleistungen** an, die systemweit zur Verfügung stehen sollen. Sie dienen in der Regel dafür, **materialübergreifende** Operationen anzubieten.
- Im Gegensatz zu Materialien gibt es von jedem Service **nur jeweils ein Exemplar** (Beispiel: Kundenstamm).
- Services können **Materialien verwalten**. Sie kapseln dabei aus Sicht der Anwendung häufig auch die **Persistenz** von Materialien (also ihre dauerhafte Speicherung).
- Services liefern **Referenzen auf Original-Materialien**, nicht auf Kopien.
- Services muss es **nicht in jedem interaktiven System** geben.
- Services werden von Werkzeugen benutzt, kennen diese aber nicht. Services **kennen nur andere Services, Materialien und Fachwerte**.
- Services werden an zentraler Stelle erzeugt und „verdrahtet“, beispielsweise in einer Startup-Klasse, und den Werkzeugen bei Bedarf als Konstruktorparameter übergeben.

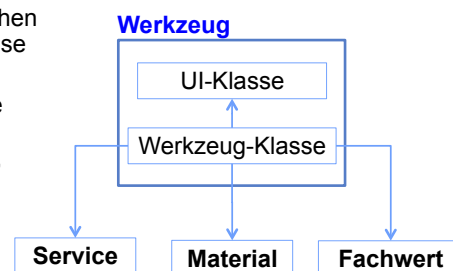
SE2

71

Werkzeuge



- Werkzeuge dienen zur **Benutzerinteraktion**. Mit einem Werkzeug können Benutzerinnen und Benutzer über dessen **grafische Schnittstelle** interaktiv Materialien ansehen und bearbeiten.
- Ein Werkzeug übernimmt **genau eine fachliche Aufgabe**, die in einem kurzen Satz gut beschreibbar sein sollte.
- In SE2 zerlegen wir ein Werkzeug immer in eine **Werkzeug-Klasse** und eine **UI-Klasse**:
 - Die Werkzeug-Klasse **vermittelt** zwischen der grafischen Schnittstelle der UI-Klasse und den fachlichen Klassen.
 - Die UI-Klasse eines Werkzeugs hat die Aufgaben, die **GUI-Komponenten** der **grafischen Schnittstelle** zu erzeugen, zu layouten und zu verwalten.



SE2

72

Werkzeugkonstruktion: Erste Schritte



- Relevante Entwurfsregeln bis zu diesem Punkt:
 - Die Werkzeug-Klasse erhält ihr **Material** als Konstruktorparameter, über Setter (wenn das Material austauschbar sein soll) oder holt es sich über Services.
 - Der Werkzeug-Klasse werden benötigte **Services als Konstruktorparameter** übergeben.
 - Die Werkzeug-Klasse **erzeugt** ein Exemplar ihrer **UI-Klasse** im eigenen Konstruktor.
- Die UI-Klasse eines Werkzeugs hat die Aufgaben, die **GUI-Komponenten** zu **erzeugen**, zu layouten und zu **verwalten**.
- Bevor wir diese Unterteilung weiter betrachten können, benötigen wir einiges an Grundlagenwissen über **Entwurfsmuster** und **grafische Benutzungsschnittstellen**.

SE2

73

Vorläufige Zusammenfassung



- Die **SE2-Entwurfsregeln** für interaktive Anwendungen (insbes. Rich-Clients) geben konstruktive Hinweise für die Strukturierung von Softwaresystemen mit einer grafischen Oberfläche.
- Sie definieren vier Elementtypen:
 - **Materialien** – veränderbare fachliche Objekte, die üblicherweise Arbeitsergebnisse modellieren;
 - **Fachwerte** – unveränderliche fachliche Abstraktionen;
 - **Services**, die systemweit fachliche Dienstleistungen anbieten und häufig Materialien verwalten;
 - **Werkzeuge** zur interaktiven Bearbeitung von Materialien.
- Der Entwurf von Materialien, Fachwerten und Services ist primär fachlich anspruchsvoll, während die Konstruktion von Werkzeugen vor allem technisch anspruchsvoll ist.

SE2

74