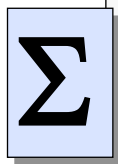




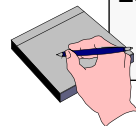
Symbole und Zeichen

**Ausblick:**

sagt, was kommt.

**Zwischensumme:**

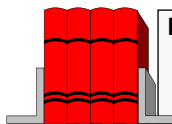
hebt für einzelne
Themen hervor,
was uns wichtig ist.

**Zentraler Begriff:**

wird hier eingeführt
oder erläutert.

**Vertiefung:**

hier werden
Hintergründe etwas
genauer beleuchtet.

**Literaturliste:**

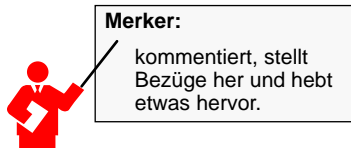
Grundlage für den
nachfolgenden Teil
mit Bewertung.

© <Autor>

Literaturreferenz:

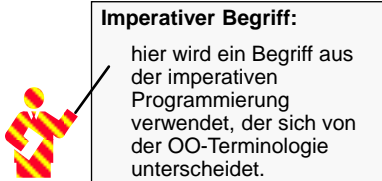
auf Bücher aus der
Literaturliste.

Symbole und Zeichen (II)



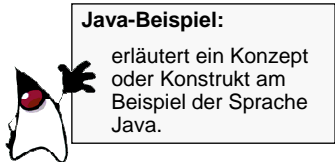
Merker:

kommentiert, stellt Bezüge her und hebt etwas hervor.



Imperativer Begriff:

hier wird ein Begriff aus der imperativen Programmierung verwendet, der sich von der OO-Terminologie unterscheidet.



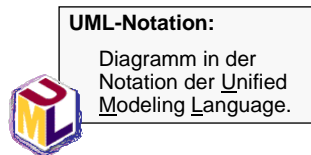
Java-Beispiel:

erläutert ein Konzept oder Konstrukt am Beispiel der Sprache Java.



Negativbeispiel:

warnt vor häufigen, aber bedenklichen Konstruktionen.



UML-Notation:

Diagramm in der Notation der Unified Modeling Language.



Softwaretechnik:

diskutiert Erfahrungen und Prinzipien.

Grundlegende Lehrbücher

David J. Barnes, Michael Kölling: **Java lernen mit BlueJ – Eine Einführung in die objektorientierte Programmierung**, 3. Aufl., Pearson Studium, 2006. (deutsche Übersetzung von: **Objects First with Java - A Practical Introduction using BlueJ**, 3. Aufl., Pearson Education, 2006.)
[Der aktuelle „Objects First“ Ansatz mit BlueJ. Gut geeignet zum Selbststudium.]

Cornelia Heinisch, Frank Müller, Joachim Goll: **Java als erste Programmiersprache**, 5. überarb. u. erw. Aufl., Teubner, Stuttgart, 2007.
[Eine gute konventionelle Einführung in Java.]

Reinhard Schiedermeier: **Programmieren mit Java**, 2. Aufl., Pearson Studium, 2010.
[Ebenfalls eine solide konventionelle Einführung in Java]

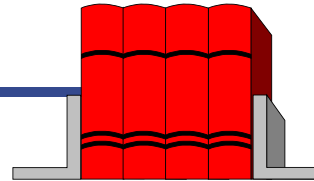
Mehr zu Java

- Reinhard Schiedermeier, Klaus Köhler: **Das Java-Praktikum**, dpunkt Verlag, 2008.
[Eine sehr nützliche Sammlung von Aufgaben zu Java.]
- Ken Arnold, James Gosling, David Holmes: **The Java Programming Language**, Fourth Edition, Addison-Wesley, 2005.
[Der Java-Klassiker. Knapp und ohne didaktischen Anspruch. Eher zum Einlesen für erfahrene Programmierer.]
- David Flanagan: **Java in a Nutshell**, 5. Aufl., O'Reilly Media, 2005.
[Der Java-Nachschlage-Klassiker. Kurz und knapp (auf 1224 Seiten) durch die wesentlichen Java-Bestandteile und -Packages.]
- Joshua Bloch: **Effective Java Programming Language Guide**, 2. Aufl., Addison-Wesley Longman, 2008.
[Die Fallstricke von Java ausführlich und sehr kompetent. Eher für Fortgeschrittene.]
- James Gosling, Bill Joy, Guy Steele: **The Java Language Specification**, Third Edition, Addison-Wesley, Juli 2005.
[Die offizielle Sprachdefinition. Für die, die es genau wissen wollen.]

Weitere Grundlagenwerke

- Peter Rechenberg, Gustav Pomberger, **Informatik-Handbuch**, Hanser-Verlag, 4., aktualis. u. erw. Aufl. 2006. 1251 S.
[Handbuch der wesentlichen Gebiete der Informatik.]
- Duden Informatik**, Dudenverlag, Ausgabe 2003.
[Grundbegriffe kurz und grundlegend definiert.]
- Grady Booch, James Rumbaugh, Ivar Jacobson, **The Unified Modeling Language Reference Manual**, Addison-Wesley, 2004.
[Die Referenz von den „Erfindern“ der UML]
- OMG Unified Modeling Language Specification**, Version 2.1.1, 2007. <http://www.omg.org/technology/documents/formal/uml.htm>
[Die aktuelle Standardversion der UML.]

Englischsprachig und weiterführend



Robert W. Sebesta, **Concepts of Programming Languages**, Addison-Wesley Educational Publishers, 8. Auflage, 2007.

[Gute und verständliche Einführung in die Definition von Programmiersprachen]

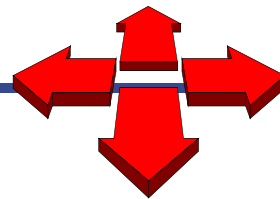
Bertrand Meyer: **Object-oriented Software Construction**. Second Edition. Prentice Hall, 1997.

[Der Klassiker unter den oo-Programmierbüchern (am Beispiel von Eiffel). Viele allgemeingültige und wertvolle Hinweise. Engagiert und bissig.]

Heinz Züllighoven et al.: **Object-Oriented Construction Handbook**. dpunkt-Verlag, 2004.

[Unser Diskussionsbeitrag. Für Fortgeschrittene (und die, die es werden wollen :-)]

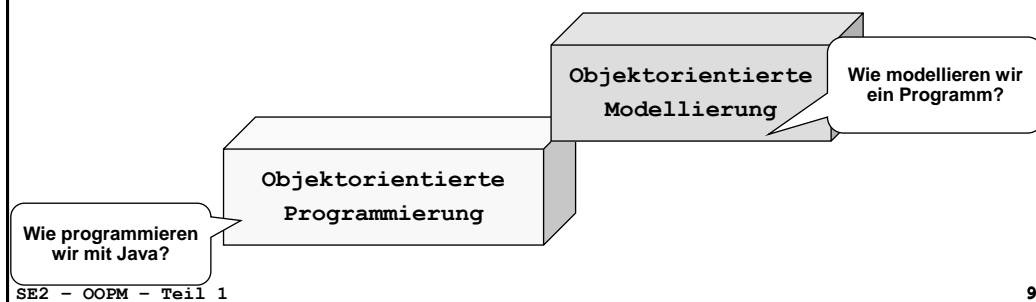
Einführung: Abstraktion und Modelle



- Aktivitäten bei der Softwareentwicklung
- Der allgemeine Modellbegriff
- Modelle als Abstraktionen
- Abstraktion als zentrales Hilfsmittel der Informatik
- Beziehungen zwischen Modell und Original

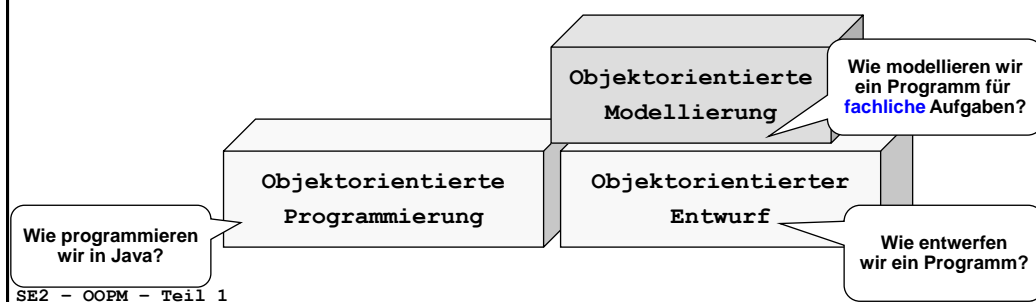
Objektorientierte Aktivitäten in SE1

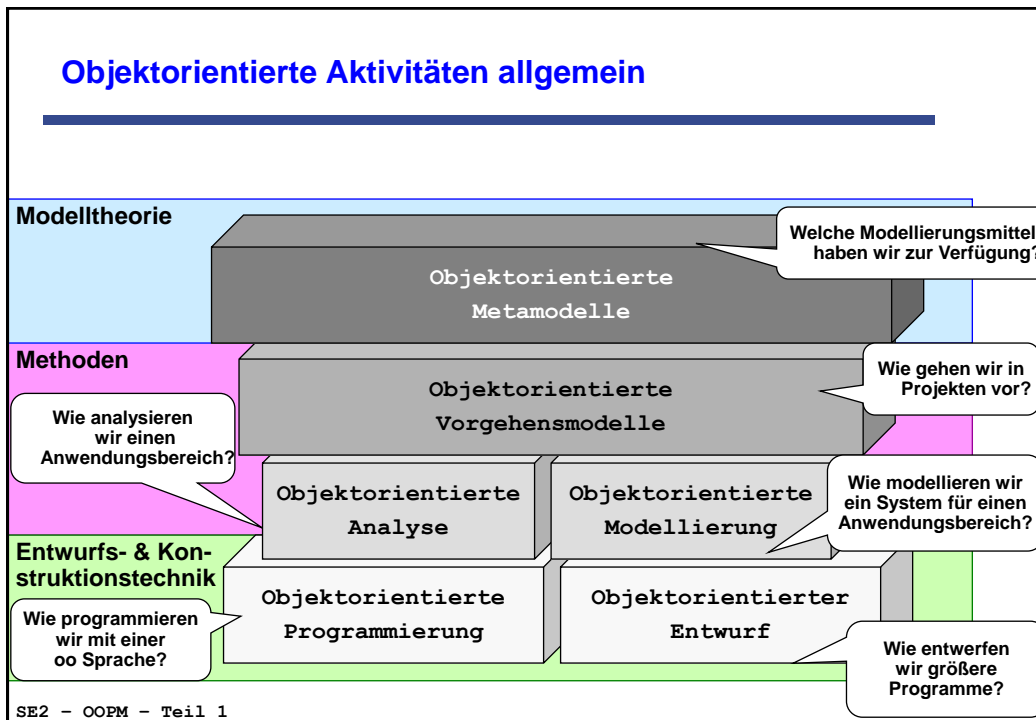
Fokus in SE1: Objektorientierte **Programmierung**



Objektorientierte Aktivitäten in SE2

SE2: Objektorientierte **Programmierung** und **Modellierung**





Modelle als zentraler Gegenstand der Softwareentwicklung



Modelle sind in der Wissenschaft und im Ingenieurwesen von zentraler Bedeutung:

- „Von einem **Modell** spricht man oftmals als Gegenstand wissenschaftlicher Methodik und meint damit, dass eine **zu untersuchende Realität** durch bestimmte Erklärungsgrößen im Rahmen einer wissenschaftlich handhabbaren Theorie abgebildet wird.“
- Da im Allgemeinen nicht alle Aspekte der untersuchten Realität in Modellen abbildbar sind, wird Modellbildung oftmals als **Reduktion, Konstruktion** oder **Abstraktion** bezeichnet.“

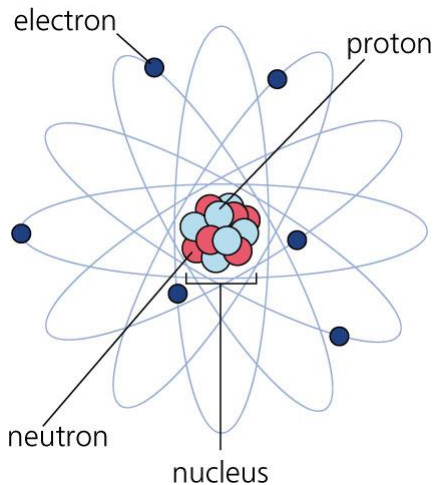
nach Wikipedia

- “Scientists spend a great deal of time building, testing, comparing and revising models, and much journal space is dedicated to introducing, applying and interpreting these valuable tools. In short, **models are one of the principal instruments of modern science.**”

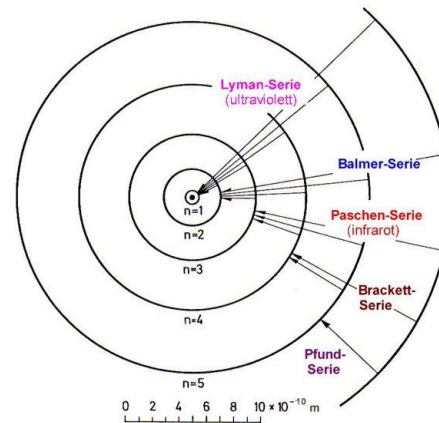
nach Stanford Encyclopedia of Philosophy

Beispiel für wissenschaftliche Modelle: Atommodelle

Atommodell nach Rutherford

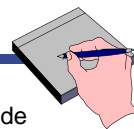


Atommodell nach Bohr



$$F_E = \frac{1}{4\pi\epsilon_0} \frac{e^2}{r^2} = ma_n = \frac{mv^2}{r}$$

Ein Modell als abstrahierende Repräsentation

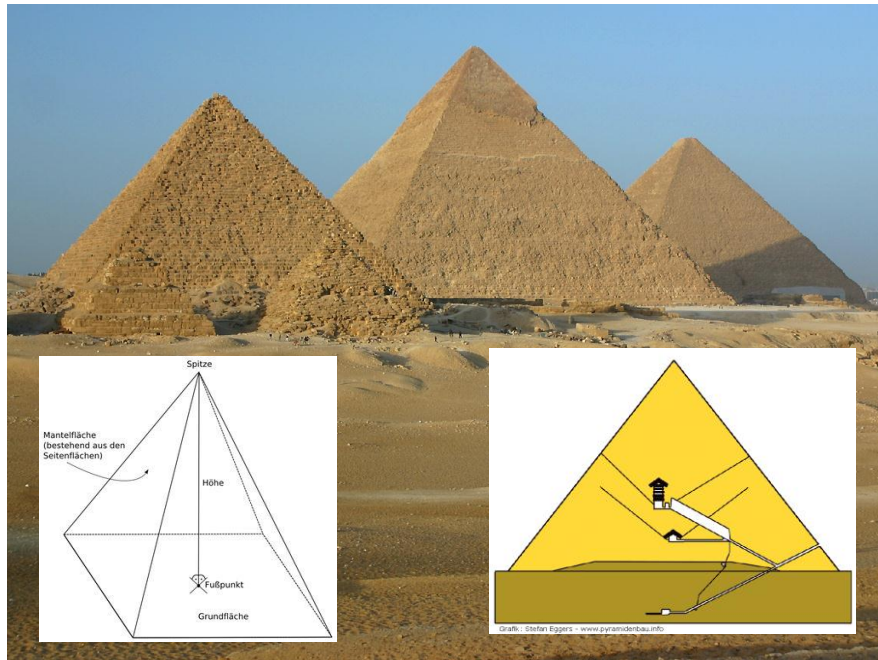


Es gibt verschiedene Modellbegriffe. In der Informatik wird häufig der folgende Ansatz von **Stachowiak** verwendet:

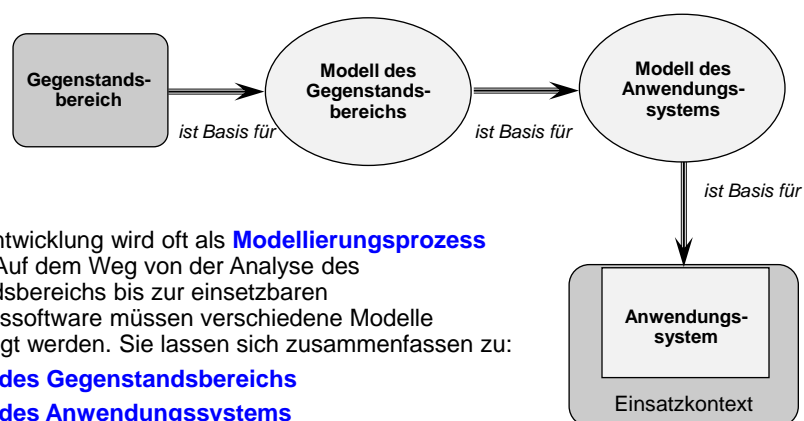
Ein Modell ist gekennzeichnet durch:

- **Abbildung**
Ein Modell ist immer ein Abbild von etwas, eine Repräsentation **natürlicher oder künstlicher Originale**, die selbst wieder Modelle sein können.
- **Verkürzung**
Ein Modell erfasst **nicht alle Attribute** des Originals, sondern nur diejenigen, die dem Modellschaffer oder Modellnutzer relevant erscheinen.
- **Pragmatismus**
Ein Modell ist einem Original aus pragmatischen Gründen zugeordnet. Die Zuordnung wird durch die Fragen **Für wen?, Warum? und Wozu?** begründet. Ein Modell wird von jemandem innerhalb einer bestimmten Zeitspanne und zu einem bestimmten Zweck für ein Original eingesetzt. Das Modell wird somit interpretiert.

Modell als Abbildung, Verkürzung, Mittel zum Zweck



Modelle der Softwareentwicklung

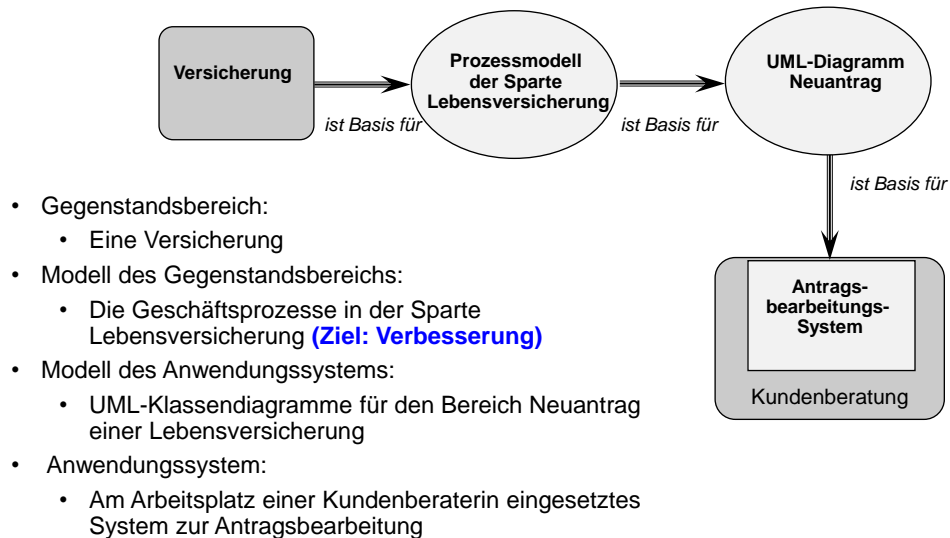


- Software-Entwicklung wird oft als **Modellierungsprozess** betrachtet. Auf dem Weg von der Analyse des Gegenstandsbereichs bis zur einsetzbaren Anwendungssoftware müssen verschiedene Modelle berücksichtigt werden. Sie lassen sich zusammenfassen zu:
 - **Modell des Gegenstandsbereichs**
 - **Modell 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.



Beispiel für die Modelle der Softwareentwicklung



SE2 – OOPM – Teil 1

17

Modellierung bedeutet Abstraktion



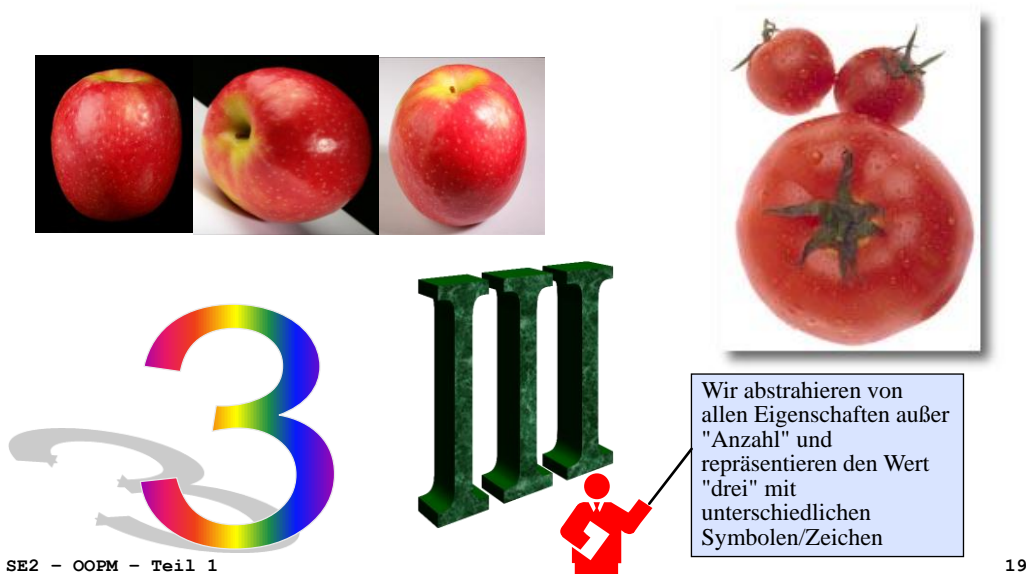
- **Abstraktion** ist das stärkste menschliche Hilfsmittel, um komplexe Dinge und Ereignisse zu verstehen.
- Der Schlüssel zur Abstraktion liegt darin, ausgewählte **gemeinsame Eigenschaften** von Gegenständen, Situationen und Prozessen in der Realität zu **erkennen** und die **Unterschiede** zu **ignorieren**.
- Wenn wir die relevanten gemeinsamen Eigenschaften erkannt haben, können wir zukünftige Ereignisse vorhersagen und kontrollieren.
- Oft verwenden wir Zeichen und Bilder, um diese relevanten Eigenschaften zu repräsentieren.
- In der Wissenschaft werden die Repräsentationen manipuliert, um zu Aussagen über zukünftige oder mögliche Ereignisse und Ergebnisse in der Realität zu kommen.

SE2 – OOPM – Teil 1

© C.A.R. Hoare, 1972

18

Beispiele für Abstraktion und Repräsentation: Zahlen



Wir abstrahieren von allen Eigenschaften außer "Anzahl" und repräsentieren den Wert "drei" mit unterschiedlichen Symbolen/Zeichen

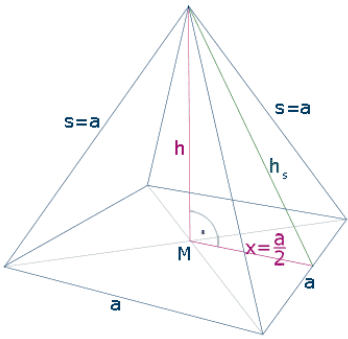
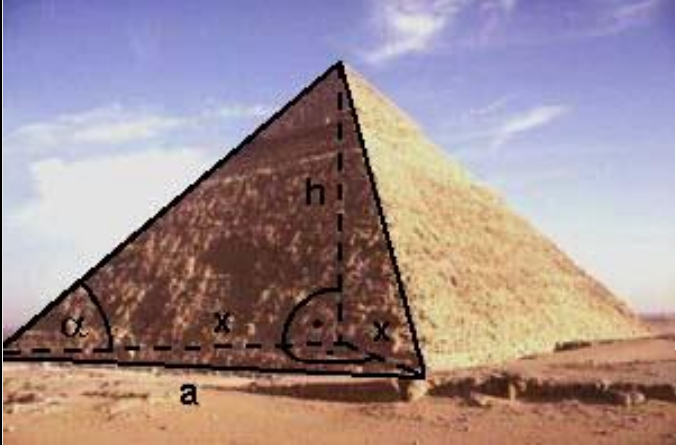
SE2 – OOPM – Teil 1

19

Von der Abstraktion zur Modell-Manipulation

- **Abstraktion:**
Identifiziere die relevanten gemeinsamen Eigenschaften von Gegenständen, Situationen und Prozessen in der Realität und ignoriere die Unterschiede.
- **Repräsentation:**
Wähle geeignete Symbole für diese Abstraktion, die Grundlage der notwendigen Kommunikation sein können.
- **Manipulation:**
Wähle die Regeln zur Transformation der Repräsentation, so dass ähnliche Veränderungen der Repräsentation auf mögliche Veränderungen in der Realität übertragen werden können.
- **Axiomatisierung/Formalisierung:**
Identifiziere formale Gesetzmäßigkeiten, mit denen sich nachweisen lässt, dass die Manipulation der Repräsentation und die Regeln für die Veränderungen in der Realität korrekt sind.

Beispiel für Manipulation von Repräsentationen bei Modellen

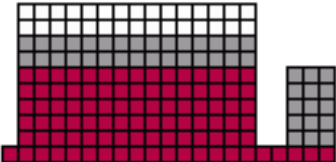



Um die Höhe der realen Pyramide zu berechnen, repräsentieren wir sie als Dreiecke, für die wir entsprechende Rechenvorschriften kennen.

SE2 – OOPM – Teil 1

21

Abstraktion und Modellbildung am Beispiel Kinosaal



0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Wir abstrahieren vom Kinosaal die Anzahl der Reihen und Sitze.

Wir repräsentieren diese Abstraktion als binäre Matrix.

Wir setzen belegte Plätze in der Matrix entsprechend auf 1.

Wir berechnen Preise anhand eines algorithmischen Terms.

Verkaufspreis = Anzahl * (NettoPreis + MWSt)

SE2 – OOPM – Teil 1

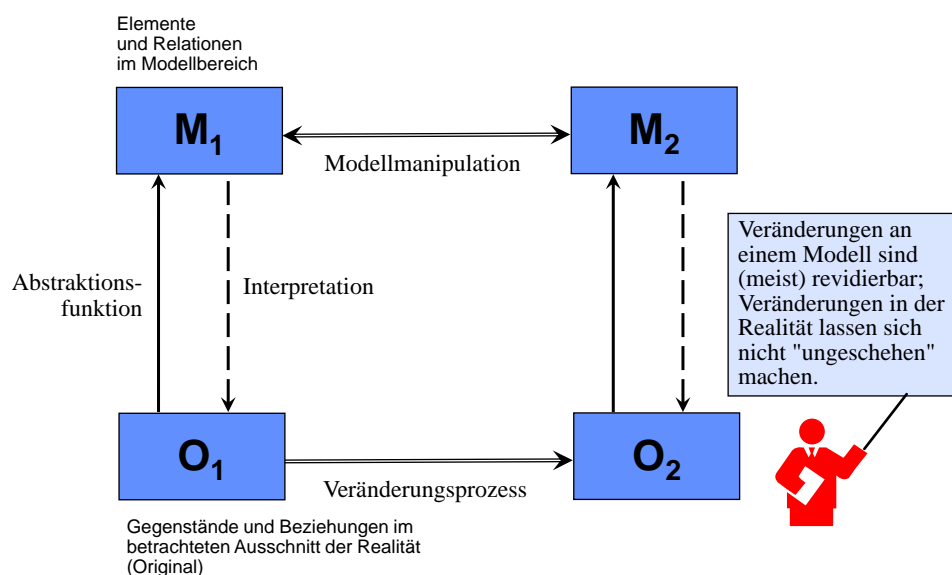
Modellieren erfordert Abstraktion vom Original und Interpretation des Modells

Ein **Modell** zeichnet sich durch **Abstraktion** vom Original für einen bestimmten **Zweck** aus:

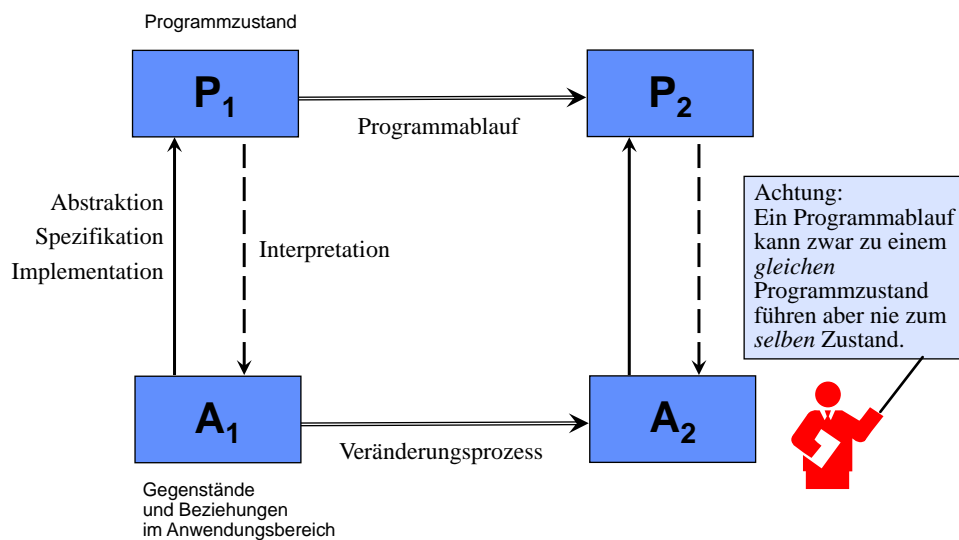
- Modelle können Zusammenhänge in der realen Situation erklären.
- Sie können zukünftige, mögliche oder sinnvolle Veränderungen des Originals zeigen (vorhersagen).
- Sie ermöglichen durch Manipulation einer Repräsentation gefahrloses oder seiteneffektfreies „Probearbeiten“.

Die meisten Zwecke kann ein Modell nur dann erfüllen, wenn wir seinen (veränderten) Zustand wieder auf das Original beziehen, d.h. wenn wir es **interpretieren**.

Beziehungen zwischen Original und Modell



Software als Modell des Anwendungsbereichs



SE2 – OOPM – Teil 1

25

Software-Modelle interagieren mit der Umwelt



- **Software-Modelle** sind in technischen Anlagen über Sensoren und Aktuatoren mit der Umwelt verbunden.
- Sie beeinflussen die Umwelt unmittelbar.

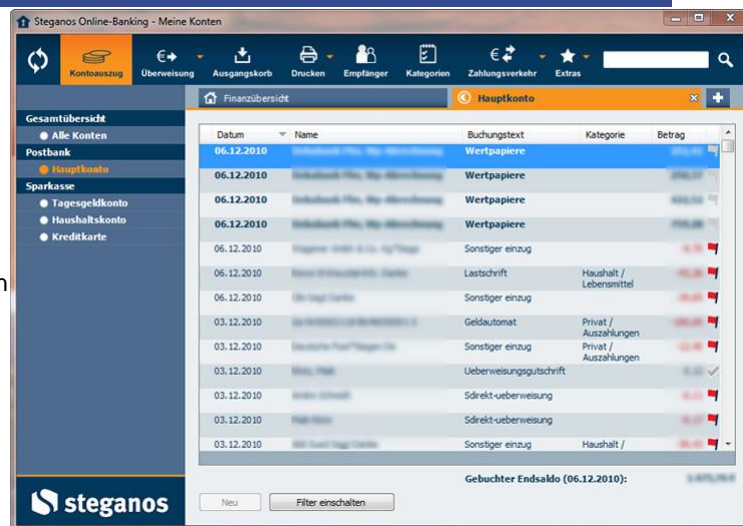
SE2 – OOPM – Teil 1

26

Teil 1: Abstraktion, Vertragsmodell, Fehlerbehandlung, Polymorphie

Software-Modelle sind Teil einer virtualisierten Umwelt

- **Software-Modelle** sind Teil unserer Umwelt geworden.
- Virtuelle Gegenstände beeinflussen unseren Alltag.



SE2 – OOPM – Teil 1

27

Software-Modelle verändern unsere Wahrnehmung der Umwelt

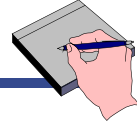
- **Software-Modelle** verändern unsere Wahrnehmung.
- Sie zeigen uns Dinge, die wir sonst nicht sehen oder wissen.



SE2 – OOPM – Teil 1

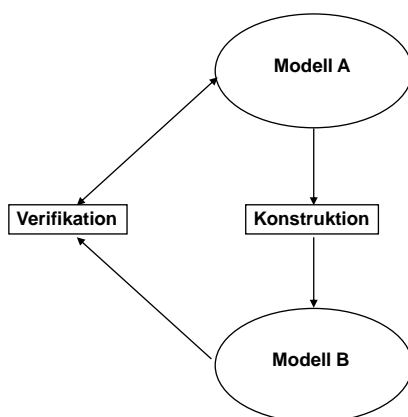
8

Anforderungen an die Modellierung von Software



- **Software muss valide sein**, d.h. sie muss für die beteiligten Personen ihren Zweck im Anwendungsbereich erfüllen.
Dazu müssen wir
 - Die angemessenen Abstraktionen und Repräsentationen wählen:
 - verständlich für die Beteiligten (Benutzer, Entwickler)
 - **weiterentwickelbar**
 - **benutzbar**
 - **interpretierbar**
 - Eine angemessene Formalisierung wählen:
 - **korrekt**, d.h. es erfüllt die vorgegebene Spezifikation
 - **verständlich**
 - **konsistent**
 - Eine angemessene Implementation wählen:
 - **effektiv**, d.h. das vorgegebene Ziel wird erreicht
 - **effizient**, d.h. mit möglichst geringem Aufwand (Laufzeit und Speicherbedarf)

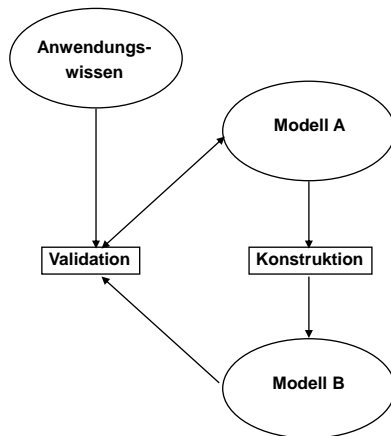
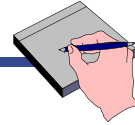
Verifikation von Modellen



Verifikation überprüft die Korrektheit und Konsistenz eines Modells der Software-Entwicklung bezogen auf sich selbst und die vorhergehenden Modelle.

Idealerweise bedeutet das, die Korrektheit der Transformation zwischen zwei Modellen nachzuprüfen. Wenn diese Modelle mit den Mitteln eines formalen Kalküls beschrieben sind, dann ist Verifikation ein formaler Prozess, der lokal, d.h. ohne Kenntnis anderer Modelle oder des Kontextes, durchgeführt werden kann.

Validation und Bewertung von Modellen



Validation ist die Bewertung von Modellen auf der Grundlage von Messungen und menschlicher Beurteilung. Wir stellen hierbei die Frage, ob ein Modell B (die Implementation), das aus einem Modell A (der Spezifikation) entwickelt worden ist, noch die Anforderungen der beteiligten Personen erfüllt.

Eine Überprüfung dieser Art kann kein formaler Prozess sein, da sie wesentlich auf menschlicher Bewertung und Einschätzung beruht, die wiederum untrennbar mit der Kenntnis der Anwendungssituation verbunden sind.

Abstraktion und Modellierung bei der Programmierung (1)

- Bisher betrachtet:
 - **Prozessabstraktion**
 - *Modell*: Schnittstelle mit Signatur
 - *Abstraktion*: Von Operationen/Methoden ist nur die Signatur bekannt; von der konkreten Implementation der Prozesse wird abstrahiert.
 - **Datenabstraktion**
 - *Modell*: Zustandsfelder und Zugriffsoperationen
 - *Abstraktion*: Werte von Feldern sind nur über Zugriffsoperationen zugänglich; vom konkreten Typ wird abstrahiert.
 - **Objektorientierte Abstraktion**
 - *Modell*: Objekte mit Methoden
 - *Abstraktion*: Objekte sind nur über sichtbare Dienstleistungen zugänglich. Von Zustandsfeldern und Implementationen wird abstrahiert.

Abstraktion und Modellierung bei der Programmierung (2)

- Die nächsten Schritte:
 - **Klassenhierarchien:**
 - Wir organisieren Klassen als Konzepte/Begriffe auf unterschiedlichen Ebenen von Abstraktionen.
 - **Unterschiedliche Granularität:**
 - Wir organisieren unsere Entwurfs- und Konstruktionselemente in geschachtelten, unterschiedlich großen Einheiten (Klassen, Packages, Teilsysteme).

Zusammenfassung



- SE2 thematisiert zusätzlich zur **Programmierung** explizit auch die **Modellierung** in der Softwareentwicklung.
- Modellierung bedeutet **Modelle** bilden:
 - Ein Modell ist eine **Verkürzung** eines **Originals** und hat immer eine **Pragmatik**, dient also einem Zweck.
- Verkürzen bedeutet **Abstraktion**, also herausarbeiten von Relevantem und Ignorieren von Irrelevantem (bezogen auf einen Zweck).
- Gutes Abstrahieren ist **anspruchsvoll**.
- Modelle ermöglichen **Probehandeln**, ohne das Original zu verändern. Heutzutage haben Softwaremodelle jedoch großen Einfluss auf die Realität.
- Modelle sollten **validiert** werden: Menschen sollten regelmäßig beurteilen, ob ein Modell seinen Zweck erfüllt.