Martin Glinz

Requirements Engineering I

Kapitel 7

Objektorientierte Spezifikation

Universität Zürich
Institut für Informatik

7.1 Grundlagen

Idee

- Diejenigen Elemente des Anwendungsbereichs identifizieren, von denen das zu spezifizierende System Informationen kennen und verarbeiten muss
- Diese Elemente auf Objekte, deren Eigenschaften und Beziehungen abbilden
- Die Anforderungen anhand des resultierenden Strukturmodells gliedern und darstellen

Abstraktion

- Konkrete Objekte eignen sich nicht zur Modellierung von Anforderungen
  - zu speziell
  - oft zum Zeitpunkt der Spezifikation gar nicht bekannt

Beispiel:
Bei der Spezifikation eines Personalinformationssystem müsste die konkrete Person Eva Müller, 36 Jahre alt, Dr. oec. publ., Leiterin Fertigung, verheiratet, ein Kind, ... als *Objekt* modelliert werden

- ➪ Abhilfe:
  - Modellierung mit Klassen → Klassenmodelle
  - Modellierung mit abstrakten Objekten → Objektmodelle

Definitionen

Objekt (object) – Ein individuell erkennbares, von anderen Objekten eindeutig unterscheidbares Element der „Realität“, d.h. des betrachteten Problem- oder Lösungsbereichs.

Beispiele: Anna Maier, Fritz Müller, Susanne Freitag

Klasse (class) – eine eindeutig benannte Einheit, welche eine Menge gleichartiger Objekte beschreibt.

Beispiele: Student, Professor

Abstraktes Objekt (abstract object) – eine abstrakte Repräsentation eines einzelnen konkreten Objekts (Singulärobjekt) oder einer Menge von konkreten Objekten (Objektmenge).

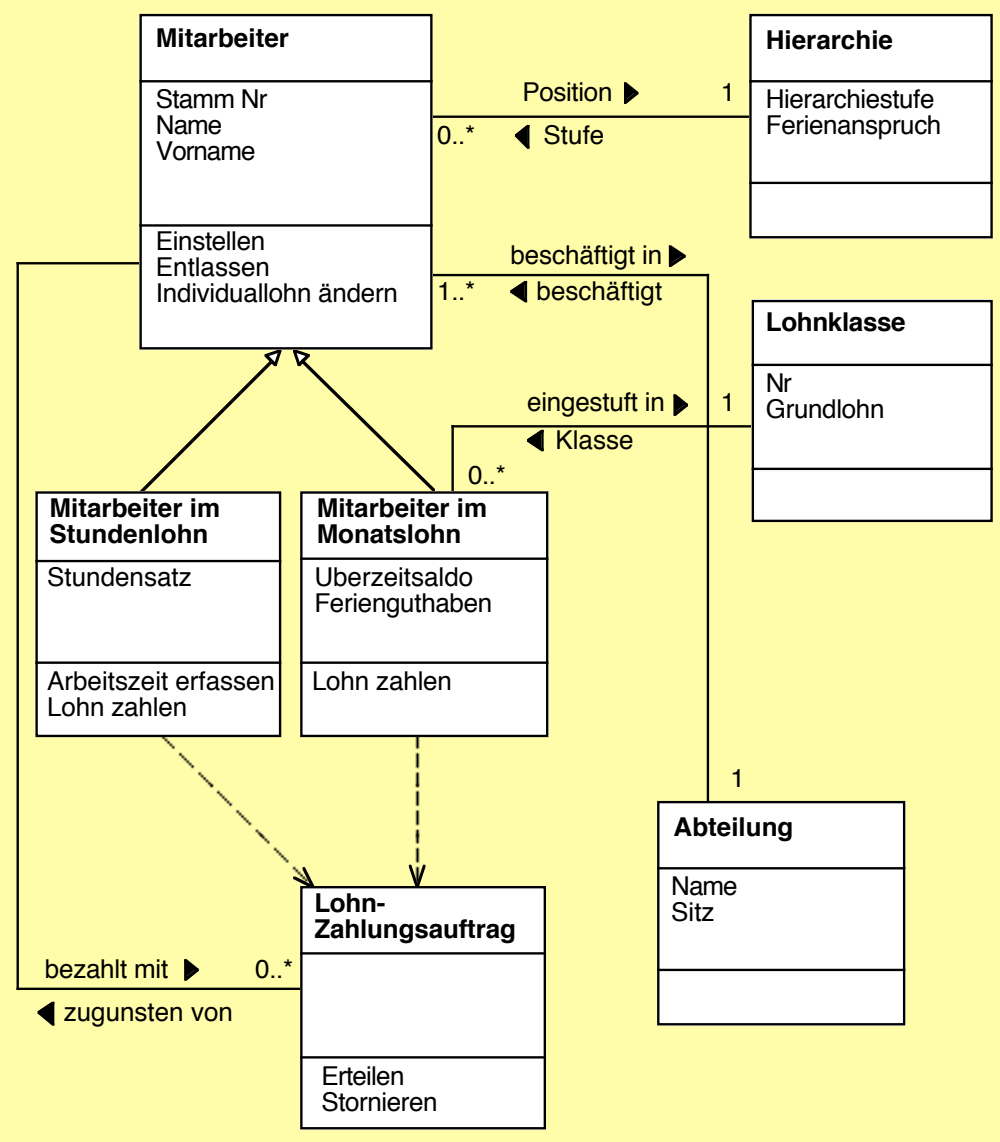
Beispiele: Dekan (Singulärobjekt), Tutor (Objektmenge)

Modellierte Elemente

- Zu jeder Klasse (in Klassenmodellen) bzw. zu jedem abstrakten Objekt (in Objektmodellen) werden modelliert:
- lokale Eigenschaften als Attribute
- von den Objekten der Klasse bzw. vom abstrakten Objekt angebotene Dienstleistungen als Operationen
- Zusammenhänge zwischen Objekten/Klassen durch Beziehungen
- Das Verhalten von Objekten muss in separaten Verhaltensmodellen spezifiziert werden
- Schnittstellen zum Kontext und Funktionalität aus Benutzersicht sind nicht adäquat modellierbar

7.2 Klassenmodelle

- Eignen sich vor allem für eine an den Daten eines Systems orientierte Anforderungsspezifikation
- Insbesondere für datenbankzentrische Systeme
- Als Sprache werden heute in der Regel UML Klassen-diagramme verwendet



7.3 Objektmodelle

Motivation

Zu spezifizieren sei ein Einsatzleitsystem für Not- und Rettungsdienste (Polizei, Feuerwehr, Sanität...). Ein solches System muss u.a.

- die Behandlung der hereinkommenden Ereignisse unterstützen
- alle Ereignisse archivieren

- In der Einsatzunterstützung brauchen wir
  - die hereingekommenen Ereignisse
  - das aktuell vom Einsatzleiter bearbeitete Ereignis
  - die behandelten Ereignisse
- Im Archiv brauchen wir
  - die Ereignisgeschichte

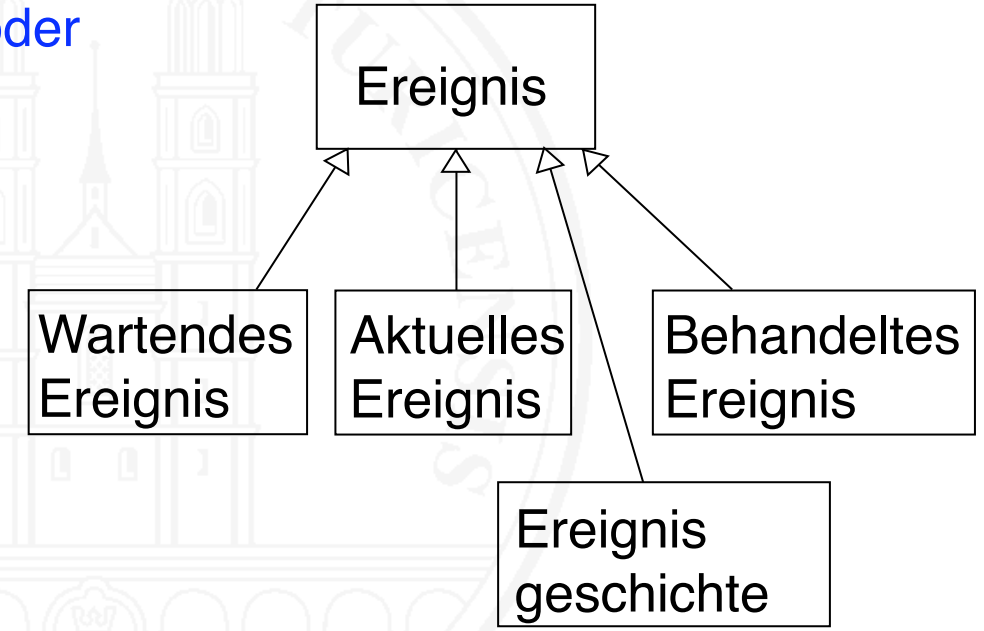
Motivation für Objektmodelle – 2

In einem Klassenmodell muss das wie folgt modelliert werden:

entweder

Schlecht: modelliert essenzielle Teile des Problems nicht

oder



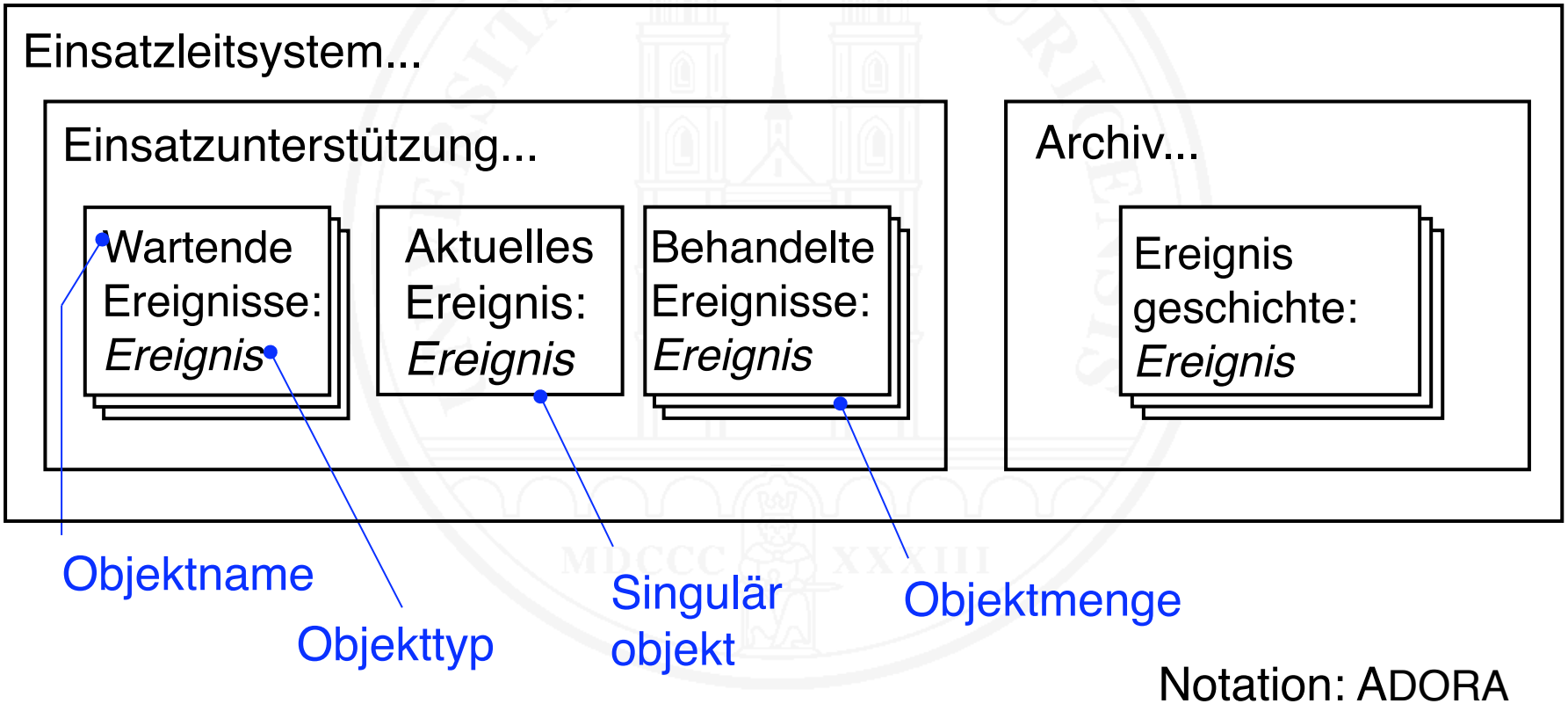
Unnatürlich: Unterklassen sind strukturell identisch

Motivation für Objektmodelle – 3

- Klassenmodelle funktionieren nicht
  - wenn verschiedene Objekte der gleichen Klasse zu unterscheiden sind
  - wenn Zusammenarbeit zwischen Objekten zu modellieren ist
- Klassenmodelle sind nicht hierarchisch zerlegbar
  - Was geschieht, wenn verschiedene Objekte einer Klasse zu verschiedenen Teilen eines Systems gehören?
  - Was ist die genaue Semantik wenn eine Klasse andere Klassen enthält?
- Die Verwendung von Unterklassen ist eine Notlösung

Modellierung mit abstrakten Objekten

- Ein Objektmodell mit abstrakten Objekten ermöglicht eine problemadäquate Modellierung



Notation: ADORA

ADORA

- ADORA ist eine Sprache und das zugehörige Werkzeug zur objektorientieren Spezifikation softwareintensiver Systeme (Glinz et al. 2002)
- Grundkonzepte
  - Modellierung mit abstrakten Objekten
  - Hierarchische Modelldekomposition
  - Integration von Objekt, Verhaltens- und Interaktionsmodellierung
  - Visualisierung im Kontext mit Sichtenbildung
  - Anpassbarer Formalitätsgrad der Darstellung
- In der Forschungsgruppe Requirements Engineering an der Universität Zürich entwickelt
  - Zentraler Gegenstand unserer aktuellen Forschungsarbeit
  - Forschung jenseits von UML

Abstrakte Objekte in UML

- Ab UML 2.0 ist die Modellierung mit abstrakten Objekten auch mit UML möglich
- Abstrakte Objekte werden als Strukturklassen* oder als Komponenten modelliert
- Der zugehörige Diagrammtyp heißt Kompositstrukturdiagramm bzw. Komponentendiagramm

*obwohl es faktisch keine Klassen, sondern abstrakte Objekte sind

Aufgabe: Klassen vs. abstrakte Objekte

Zu spezifizieren sei eine dezentrale Heizungssteuerung, welche aus einem zentralen Heizungssteuergerät und einem Raumsteuergerät in jedem beheizten Raum besteht.

- Das zentrale Steuergerät soll über ein Bedienfeld verfügen, das aus einer Tastatur, einer LCD-Anzeige sowie einem Ein- und einem Aus-Knopf besteht.
- Das Raumsteuergerät soll über ein Bedienfeld verfügen, das aus aus einer LCD-Anzeige sowie fünf Knöpfen für Ein, Aus, Plus, Minus und Eingeben verfügt.

Modellieren Sie dieses Problem

a. mit Klassen

b. mit abstrakten Objekten

7.4 Methodik der objektorientierten Spezifikation

- Analyse von Informationsobjekten des Anwendungsbereichs
- Analyse von Glossaren, wo vorhanden
- Objektanalyse von Texten
- Auswertung einer Ereignis-Reaktionsanalyse nach notwendigen Daten

- Siehe auch
  - Kapitel 4
  - Vorlesung Informatik IIa: Modellierung, Kapitel 8

Objektorientierte Spezifikation: Vor- und Nachteile

+ Gut geeignet zur Beschreibung der Systemstruktur
+ Unterstützt Lokalität von Daten und Einkapselung von Eigenschaften
+ Erlaubt strukturähnliche Implementierungen
+ Systemdekomposition möglich

– Funktionalität aus Benutzersicht nicht adäquat modellierbar
– Verhaltensmodelle nur schwach integriert
– Dekomposition häufig nicht unterstützt

Literatur

Booch, G. (1994). *Object Oriented Analysis and Design with Applications*. Second Edition. Redwood City, Ca.: Benjamin/Cummings.

Coad, P., E. Yourdon (1991a). *Object-Oriented Analysis*. 2nd edition. Englewood Cliffs, N.J.: Prentice Hall. [auf Deutsch: Objektorientierte Analyse, 1994 im gleichen Verlag]

Glinz, M., S. Berner, S. Joos (2002). Object-Oriented Modeling With ADORA. *Information Systems* **27**, 6. 425-444.

Jacobson, I., M. Christerson, P. Jonsson, G. Övergaard (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Amsterdam; Reading, Mass. [u.a.]: Addison-Wesley.

Joos, S., S. Berner, M. Arnold, M. Glinz (1997). Hierarchische Zerlegung in objektorientierten Spezifikationsmodellen. *Softwaretechnik-Trends*, **17**, 1 (Feb. 1997). 29-37.

Oestereich, B. (1998). *Objektorientierte Softwareentwicklung*. R. München: Oldenbourg.