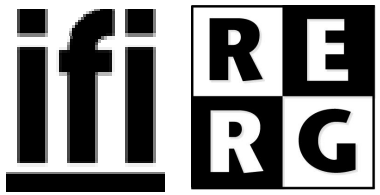
JUGS – Research in Progress Seminar

Objektorientierte Modellierung mit der Sprache ADORA

Silvio Meier

smeier@ifi.unizh.ch



Universität Zürich
Institut für Informatik
Requirements Engineering Research Group

Wer sind wir?

- Requirements Engineering Research Group – Universität Zürich
- ADORA – eines unserer Projekte
- Gegenwärtige und ehemalige ADORA-Projektmitglieder
  - Martin Glinz
  - Stefan Joos
  - Stefan Berner
  - Yong Xia
  - Nancy Schett
  - Christian Seybold
  - Silvio Meier
  - Verschiedene Studierende...

Zeitplan

- Einleitung
- Sprachkonzepte von ADORA
- Visualisierungskonzepte mit ADORA
- Ausblick
- Demo des ADORA Werkzeugs

ADORA in Kürze

- Analysis and Description of Requirements and Architecture
  - Eigenständige Modellierungssprache für objektorientierte Anforderungsspezifikationen und Architekturen
  - Andere Sprachkonzepte als bei UML und anderen konventionellen Sprachen
- Ziel: Defizite anderer Sprachen vermeiden
- Einsetzbarkeit für verschiedene Typen von Systemen: Industrielle Systeme, Informationssysteme
- Entwickelt von der Gruppe Requirements Engineering

Anforderungsspezifikationen

- Wozu Anforderungsspezifikationen? U.a.
  - Anforderungen entwickeln/erfassen
  - Mittel zur Kommunikation
- Ziele/Eigenschaften
  - Qualitätsmerkmale: Adäquatheit, Vollständigkeit, Eindeutigkeit, Widerspruchsfreiheit, Prüfbarkeit (!), Verständlichkeit (!) und Änderbarkeit
  - Weitere Qualitätsmerkmale: Rückverfolgbarkeit, Entwurfs-unabhängigkeit, Umsetzbarkeit
  - Komplexitätsbewältigung
  - Unterstützung einer inkrementellen Entwicklung

Dimensionen der Modellierung

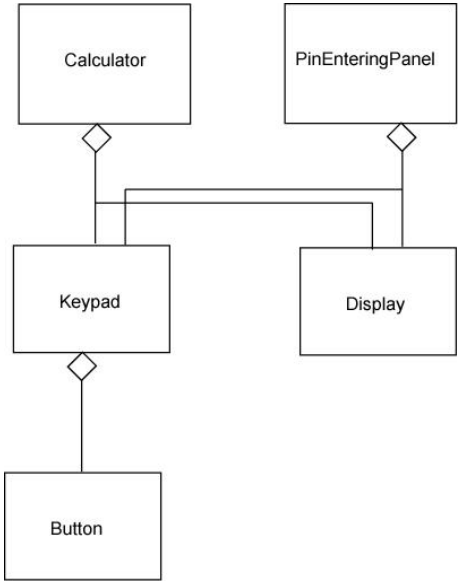
- Möglichkeiten zur Beschreibung einer Anforderungsspezifikation
  - Deskriptiv: Beschreibung von Input und Output
  - Konstruktiv: Beschreibung des Systems auf abstrakter Ebene → ADORA
- Formalität
  - Formal
  - Teilformal
  - Informal
- Projektion: Zeigen von verschiedenen Sichten oder Aspekten des Systems

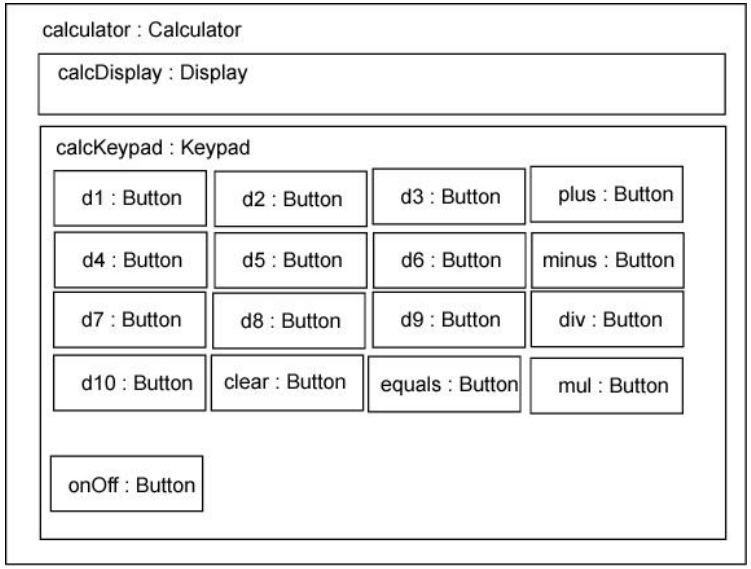
Sprachkonzepte von ADORA

- Verwendung von **abstrakte Objekten** anstelle von Klassen
- **Hierarchische Dekomposition**
  - Beherrschung von Komplexität
  - Strukturierung von Systemen
  - Inkrementelle Entwicklung von Systemen
- **Integrierte Modelle**, kein Flickwerk von vielen verschiedenen Sprachen → Komplexitätsbeherrschung
- Variabler **Grad an Formalität**
  - Unterstützung von inkrementeller Entwicklung
- **Kontextvisualisierung: Details immer mit ihrem Kontext in abstrahierter Form** darstellen
  - Unterstützt die integrierte Sichtweise auf das Systemmodell oder einen Teil davon
  - Komplexitätsbeherrschung

Abstrakte Objekte

- Anomalien bei der Darstellung von Modellen → Fehlende Ausdruckstärke in Modellen
- Mit Klassenmodellen schwierig Dekomposition zu betreiben
- Verwendung von abstrakten Objekten → Umgeht diese Probleme!
- Verwendung von Klassen als Typen



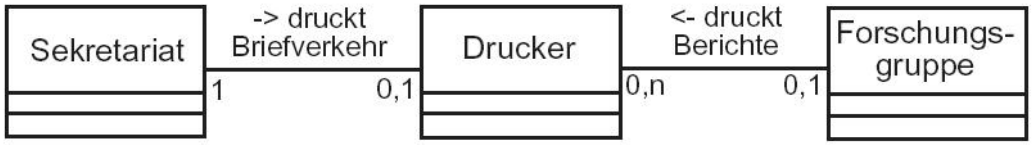


Dekomposition: Teil/Ganzes-Hierarchie

- Eigenschaften einer guten Systemgliederung
  - Logisch zusammengehörende Teile
  - Gemeinsame Information über kleine Schnittstellen
  - Kein internes Wissen über andere Teile im System
- Bestehende OO-Modellierungssprachen (Ausnahme UML 2.0) lösen dies bisher nicht hinreichend gut
  - Sammlung von verschiedenen Teilsprachen
  - Gleiche Information in Containern
- Teil/Ganzes-Hierarchie
  - Basisstruktur für die Dekomposition: Teil/Ganzes-Objekthierarchie
  - Objekte mit untergeordneten Objekten → Hierarchische Dekomposition
  - Unterstützt Prinzip der Abstraktion

Beispiel Drucker (1)

- Beispiel aus zu Modellierungsanomalie, abstrakten Objekten und Teil/Ganzes- Hierarchie

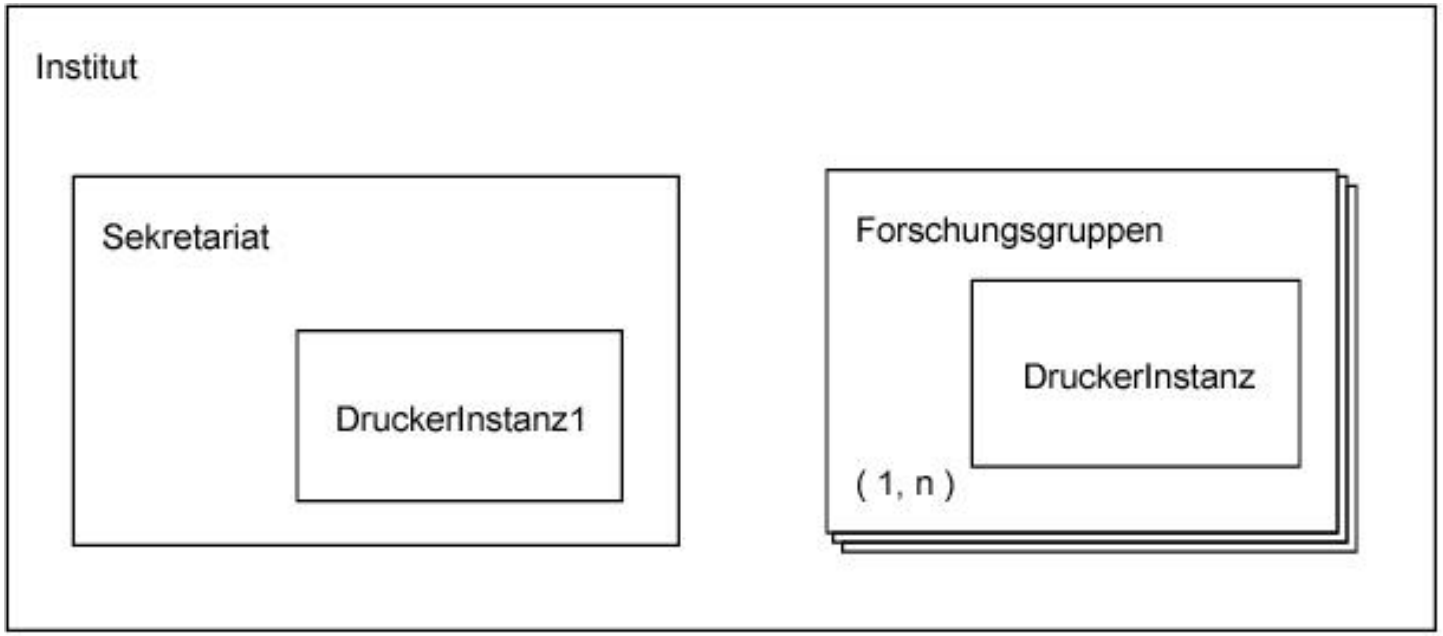


aus [Joos99]

- Modellierungsanomalie: Fehlende Kontextmodellierung!
- Dadurch fehlende Information:
  - Kann ein bestimmter Drucker (Instanz) nur vom Sekretariat oder auch von Forschungsgruppen verwendet werden oder werden alle Drucker gemeinsam von allen Institutionen gleichermassen verwendet werden?
  - Ist ein Drucker einer Forschungsgruppe zugeordnet oder können mehrere Forschungsgruppen einen Drucker gemeinsam benutzen?
  - Existieren Drucker, die weder vom Sekretariat noch von einer Forschungsgruppe verwendet werden?

Beispiel Drucker (2)

- Teil-Ganzes-Hierarchie:
  - Ausdrucksstark
  - Zerlegbar
  - Weitere Aspekte einfach integrierbar



aus [Joos99]

Integriertes Modell (1)

- Aspektmodelle pro Aspekt jeweils ein Modell
  - Zusammenhänge und Überschneidungen durch (nicht sichtbare) Integritätsbedingungen geregelt
  - In gängigen Modellierungssprachen der Fall: Beispiel UML → lose gekoppelte Sammlung von Teilsprachen
    - Verhaltens in Statecharts
    - Struktur in Packages und Klassenmodellen
    - ...
  - Nachteile:
    - Konsistenz und Vollständigkeit
    - Benutzer muss Sichten im Kopf integrieren!

Integriertes Modell (2)

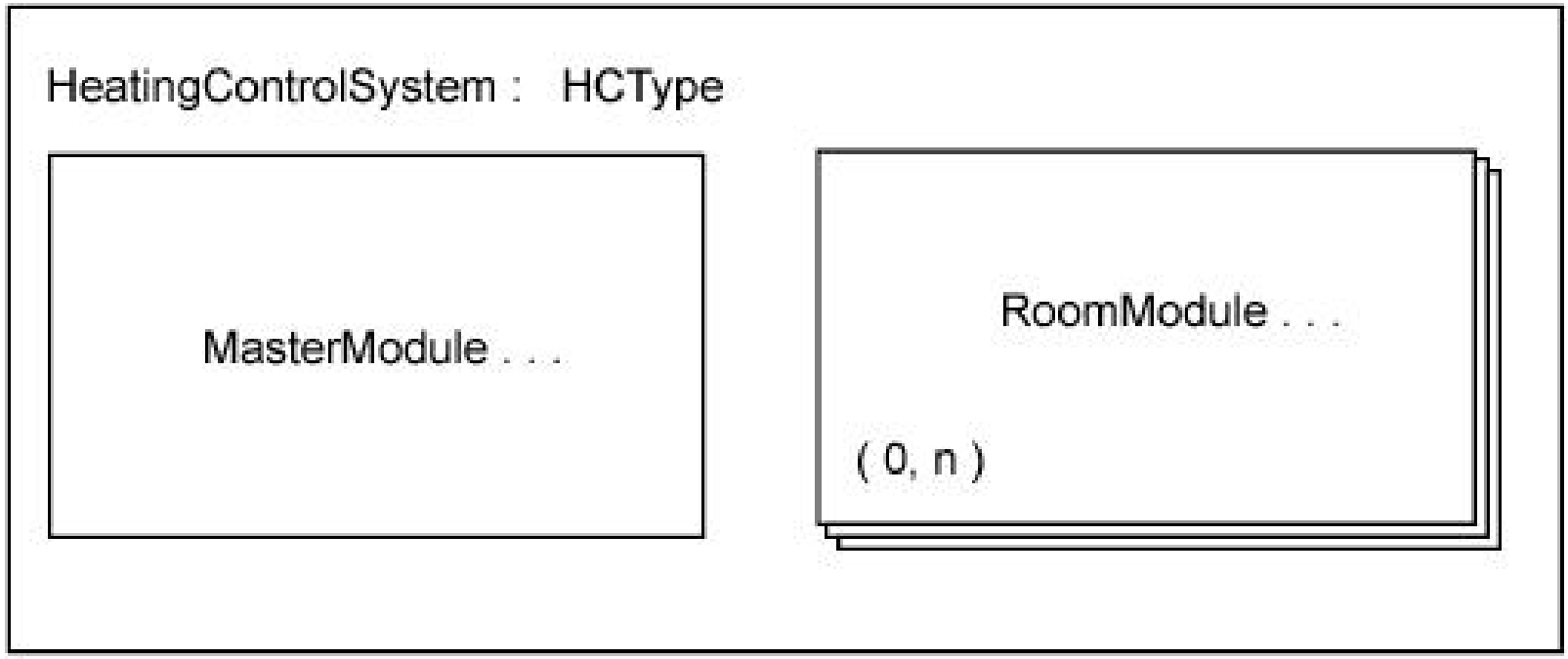
- Integrierte Modelle in ADORA
  - Zentrales Gesamtmodell enthält alle Aspekte des Systemmodells
  - Basisstruktur
    - Aspektbezogene Ein- und Ausblendung nach Bedarf
    - Hierarchische Dekomposition
- Vorteile
  - Weniger Redundanz in den Aspektmodellen
  - Bessere Strukturierung: Daher auch einfachere Modelle
  - Bessere Verständlichkeit
  - Weniger fehleranfällig
  - Komplexitätsbewältigung → Weniger intellektuelle Belastung

Sichten von ADORA

- Basis-Objektstruktur
- Sichten (aspektbezogene Einblendungen)
  - Strukturelle Sicht
  - Verhaltenssicht
  - Funktionale Sicht
  - Benutzer-Sicht
  - Systemkontext-Sicht
- Die Sichten können nach Belieben ein- und ausgeblendet werden
- Bis auf die Funktionale Sicht sind alle Sichten direkt in der Basissicht integriert

Basis-Sicht (1)

- Beispiel

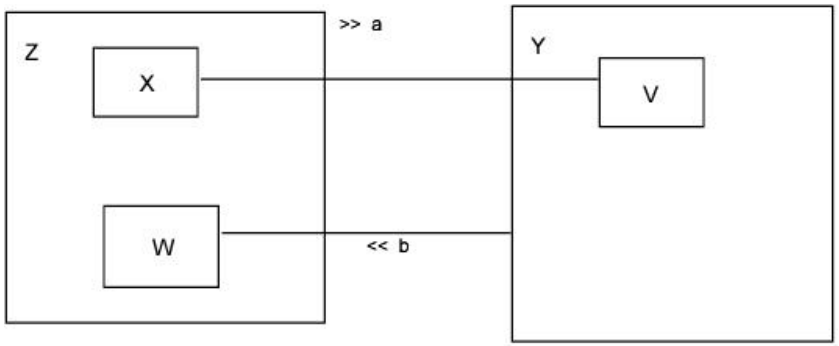


Basis-Sicht (2)

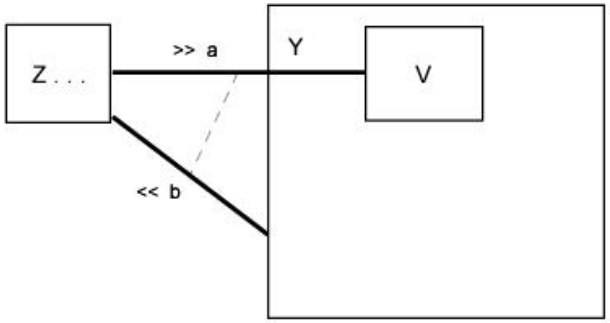
- Besteht aus abstrakten Objekten und Objektmengen
- Objektmengen können Kardinalitäten enthalten
- Ein abstraktes Objekt selber besteht aus
  - Eigenschaften (Attribute, Operationen, Beziehungen)
  - Komponenten (eingeschachtelte Objekte und Objektmengen)
- Graphisch repräsentiert durch geschachtelte Rechtecke
- Bildet Integrationsbasis für die Einblendung anderer Sichten
- Abstrahierung (drei Punkte)
  - Basisstruktur Visualisierungskonzept (Zooming)
  - Unvollständige Modelle → inkrementelle Entwicklung von Modellen
- Objekte können von bestimmtem Typ sein

Strukturelle Sicht (1)

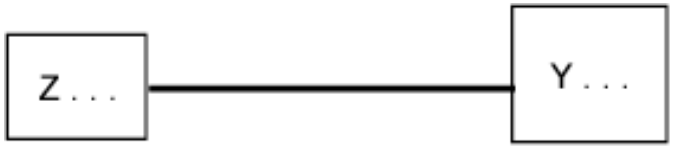
Ohne Abstrahierung



Abstrahierung von Z



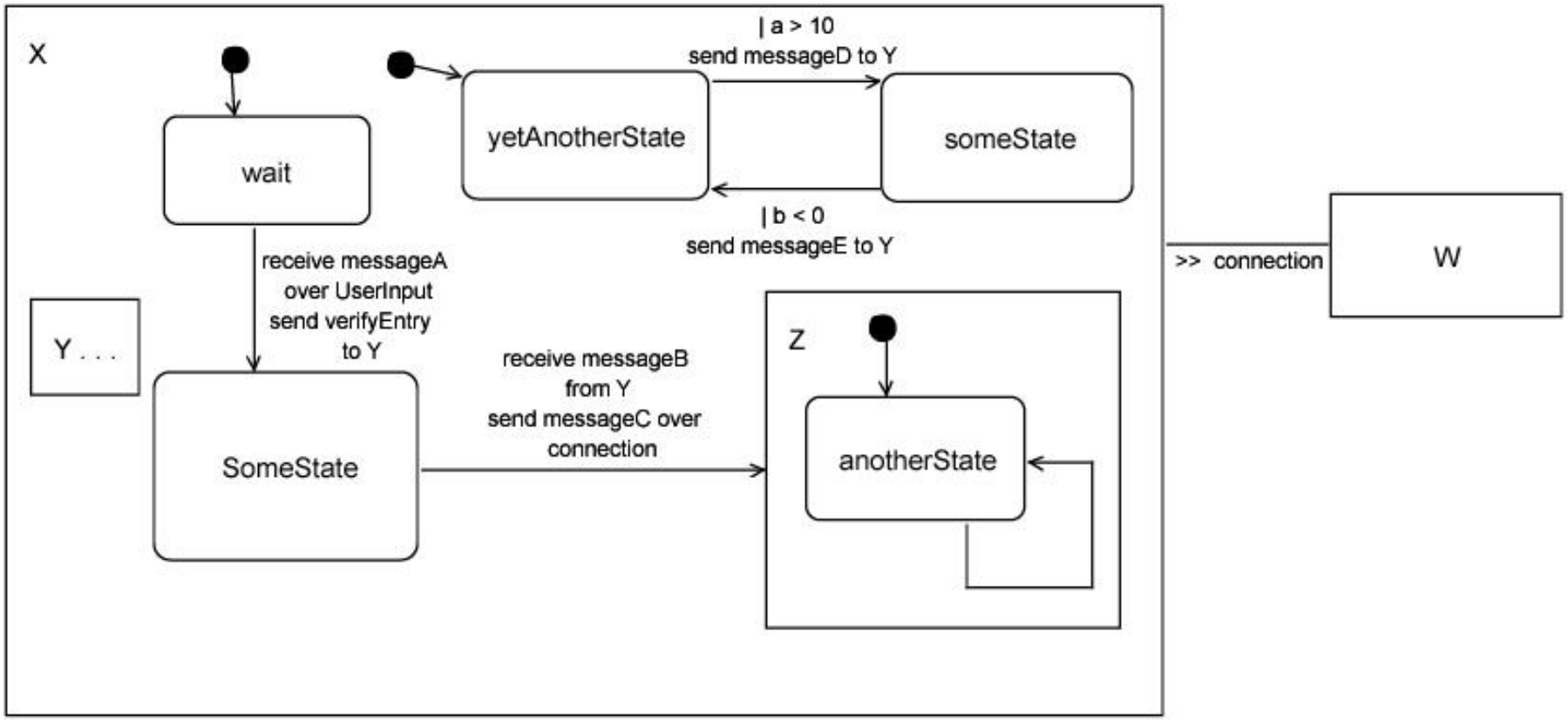
Abstrahierung von Z und Y



Strukturelle Sicht (2)

- Assoziationen
  - Zugriffe auf öffentliche Attribute und Operationen von assoziierten Objekten
  - Immer gerichtete binäre Beziehungen
    - Entkoppelt Objekte
    - Information Hiding
  - Multiplizitäten bei Verbindungen zwischen Objektmengen
- Abstrakte Beziehungen
  - Beschreibung bei Abstrahierung (Zooming)
  - Hierarchische Darstellung von Beziehungen mittels (abstrakter) Oberbeziehungen
  - Verwendung bei unvollständigen Modellen

Verhaltenssicht (1)



Verhaltenssicht (2)

- Statechart-ähnliche Notation
- Eingebettet in Objektstruktur
  - Abgerundete Rechtecke
  - Transitionen
  - Zustände können wiederum hierarchisch zerlegt werden
- Objekte sind selbst komplexe Zustände
- Nebenläufigkeit wird implizit ausgedrückt (kein explizites Konstrukt)
- Transitionen
  - Broadcast nur innerhalb des Ursprungsobjektes
  - Sonst explizites Verschicken/Empfangen zu/von einem Komponentenobjekt

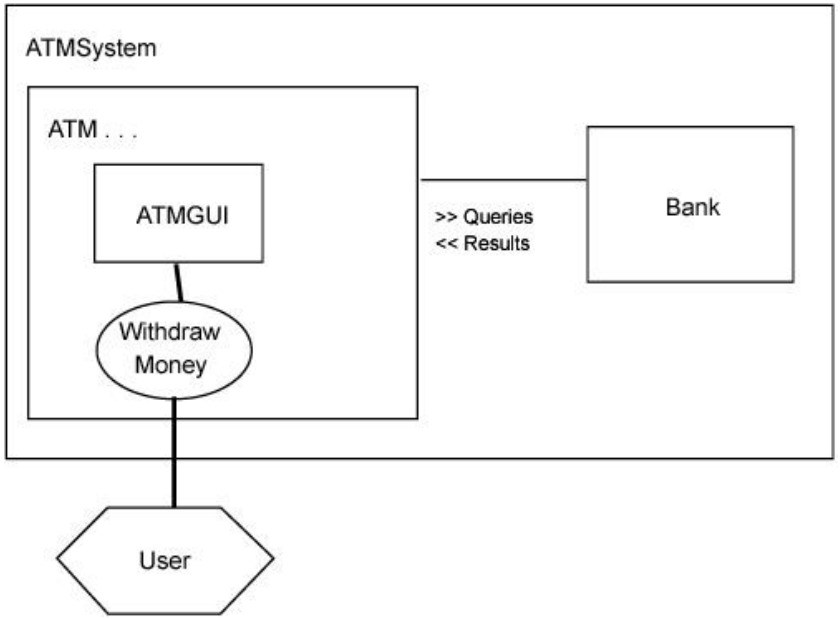
Funktionale Sicht (1)

- Definiert:
  - Eigenschaften von abstrakten Objekten (sofern nicht durch den Typ definiert)
  - Beschreibungen für Transitionen
- Beschreibt
  - Benutzerdefinierte Datentypen
  - Attribute von Objekte
  - Operationen (synchron und asynchron)
- Abgeleitet von der Sprache ASTRAL [Coen97]
- Objektbeschreibungen können teilformal sein, d.h. natürlichsprachlichen Text enthalten

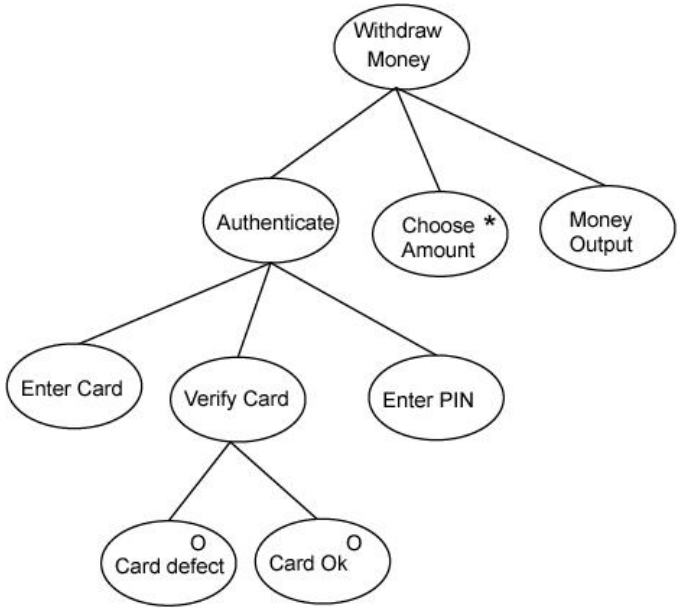
Funktionale Sicht (2)

```
object specification BankAccount is object type Account;
  types
     PinNumber : typedef code : integer(1000 < code < 100000);
  attributes
     Name : string; StoredPin : PinNumber; MoneyAmount : float; Limit  : float;
  syncoperation checkCode(in: enteredPin :PinNumber; out: returnCode : boolean)
     var checkOk : boolean = false;
     `This method checks for the pin number with the stored pin number `;
  end checkCode
  operation moneyWithdraw(amount : float)
      var acceptCode : boolean;
      pre (moneyAmount + Limit) - amount >= 0;
      post  moneyAmount@pre == moneyAmount + amount;
  end moneyWithdraw
end specification
```

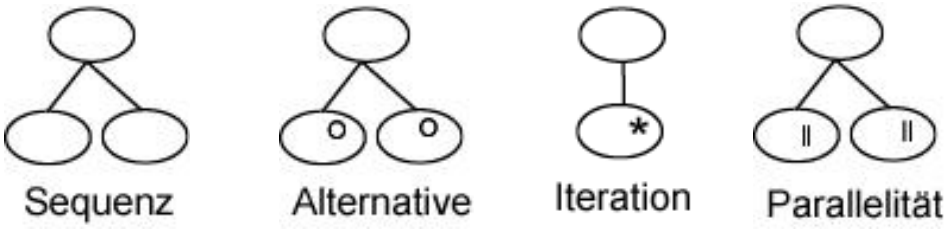
Benutzer-Sicht / Systemkontext-Sicht (1)



Withdraw Money - Szenario



Erweiterte Jackson-Syntax



Benutzer-Sicht / Systemkontext-Sicht (2)

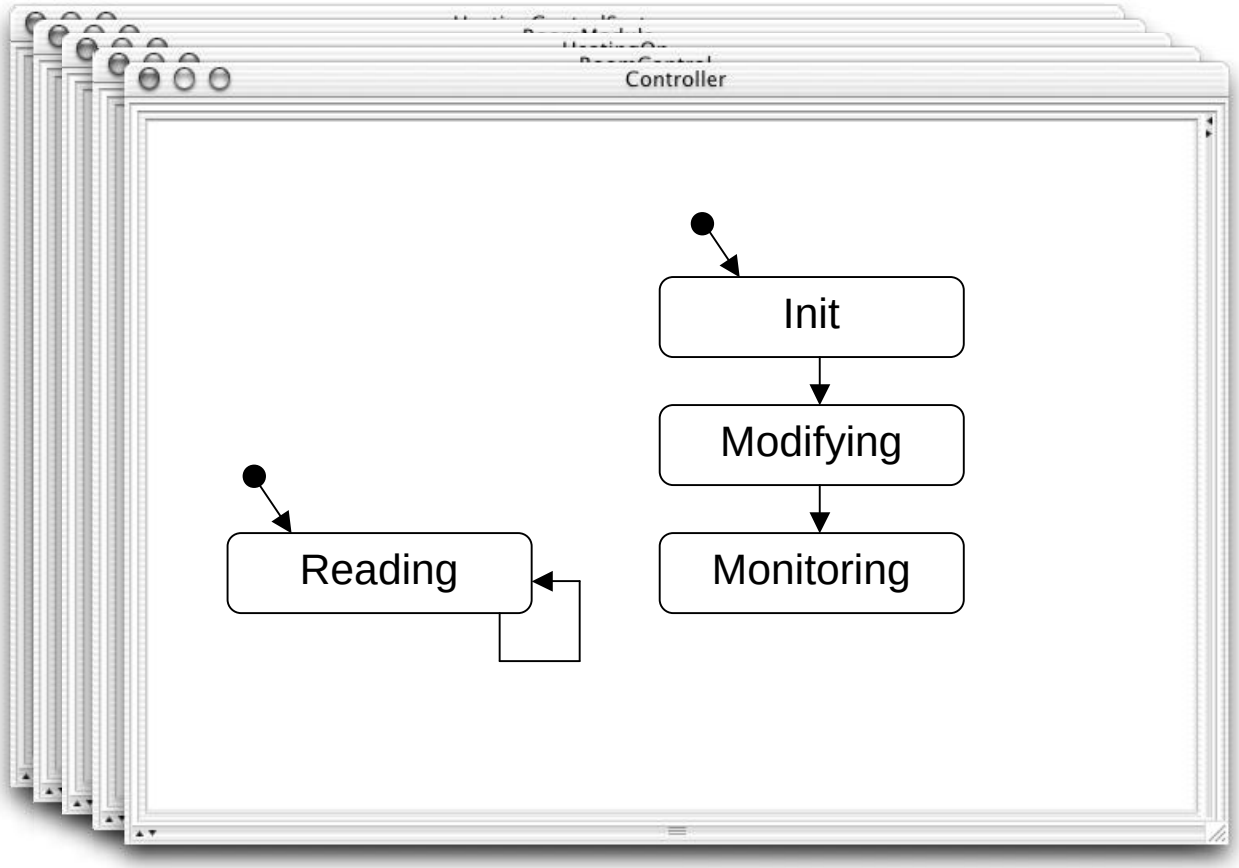
- Benutzer-Sicht
  - Modellierung von Typszenarien
  - Idee: Anwendungsfälle direkt einer bestimmten Teilkomponente zuordnen
  - Integriert in Basisstruktur, Ovale mit Namen
  - Zwei Ansätze → zu diskutieren
    - Jackson-artige Gliederung in Subszenarien
    - Statechart-artige Szenarien
- Systemkontext-Sicht
  - Externe Akteure im Kontext des Systems (6-Ecke): Benutzer, andere Systeme
  - Externe Objekte als COTS Komponenten → gehören System

Visualisierung von ADORA-Modellen

- Was wird gewünscht, um ADORA-Modelle geeignet zu visualisieren
  - Unterstützung der Dekomposition/Integrierte Modelle
  - Orientierung im Modell
  - Verständlichkeit erhalten

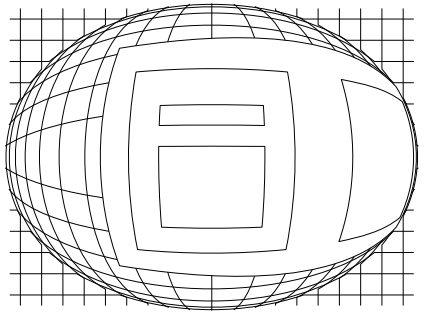
- Konventionelle Navigationstechniken in Modellen
  - Explosiv-Zoom → Fenster öffnen sich
  - Ergeben schlecht verständliche Modelle
  - Wenig Übersicht!
  - → Fehlender Kontext der Elemente

Traditioneller Explosiv-Zoom

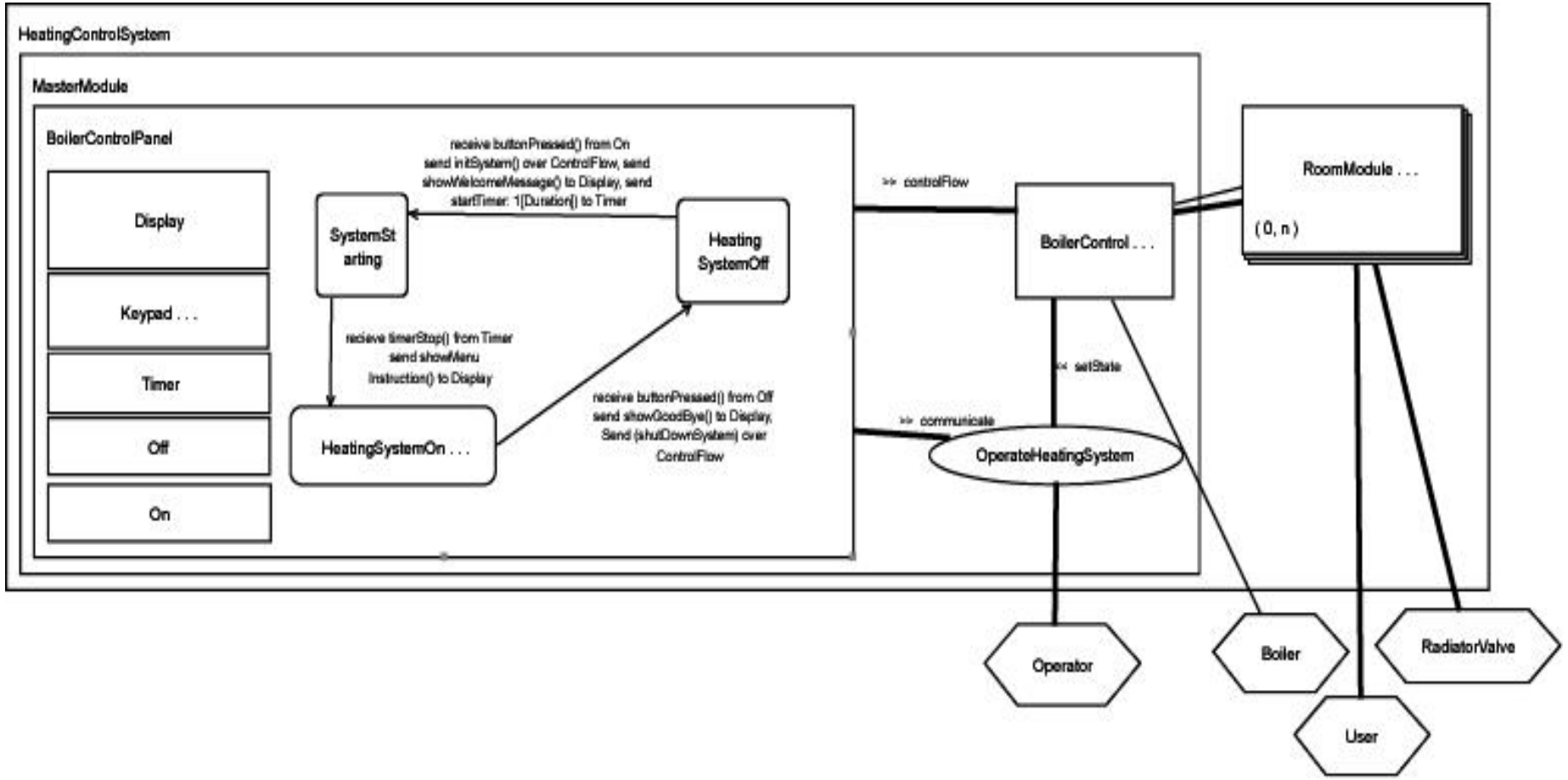


Kontext-Visualisierung

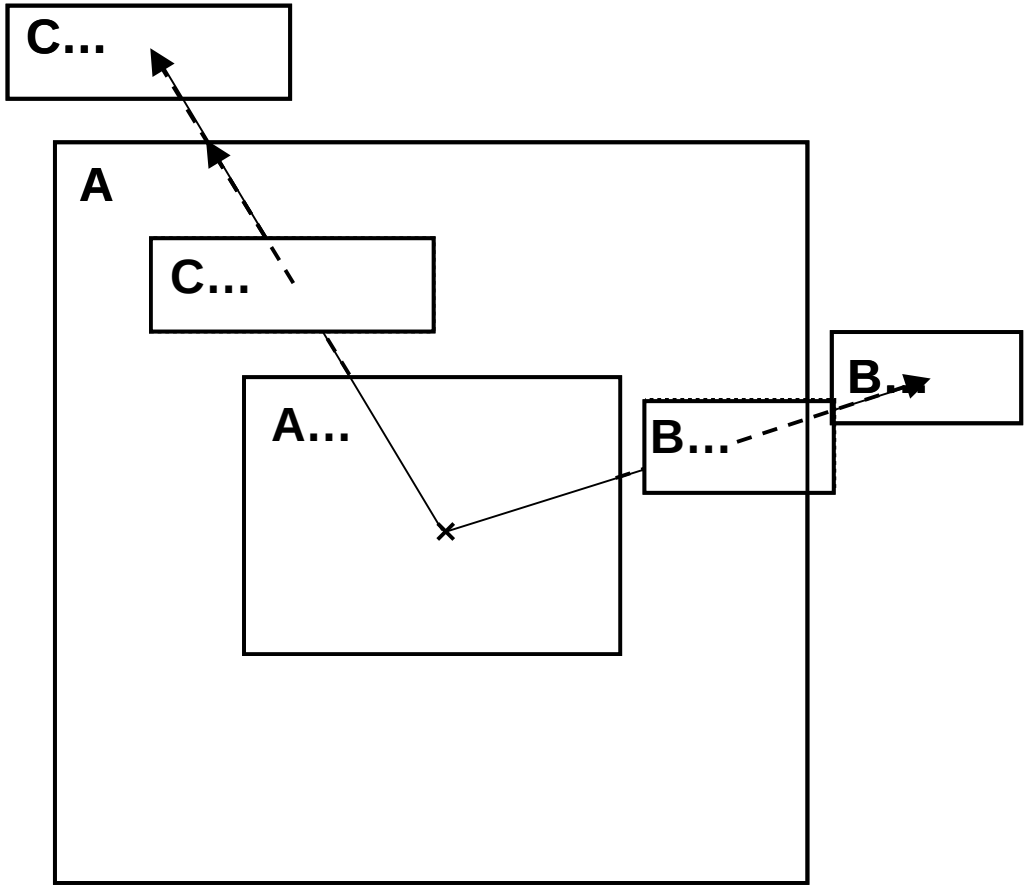
- Idee
  - Zeigen von lokal interessanten Details → Focus of Interest
  - Kontext in abstrahierter Form darstellen
  - Globaler Kontext, lokales Detail
- Mögliche Technik
  - Fischaugen-Linse [Bern02] → Weitwinkellinse
  - Physikalisches Fischauge nicht passend
  - Logisches Fischauge, d.h. auf bestimmten Ebenen in der Modellhierarchie Details darstellen, Kontext abstrahieren



Beispiel logisches Fischauge in einem ADORA-Modell



Funktionsweise Layout-Algorithmen



Layout-Algorithmus für ADORA

- Verwendungsmöglichkeiten
  - Kontext-Visualisierung (logisches Fischauge)
  - Space-Management in graphischen Modellen (siehe [Seyb03])

- Einiges zum Algorithmus
  - Radiale Verschiebung: Erhaltung Sekundärnotation durch diesen Algorithmus
  - Einige Spezialfälle sind zu beachten → Überlappungen
  - Auszoomen-Algorithmus durch umgekehrte Anwendung des Algorithmus

Ausblick

- Simulation von Modellen
  - Bisher weitestgehend erforscht: Simulation von formalen Modellen
  - Ziel: Simulation/Animation von teilformalen Modellen
- Integration von weiteren Systemaspekten in die Modelle
  - Verteilungs- und Prozessstruktur der Systeme
- Erprobung für grössere Projekte mit Feedback

Werkzeug-Demo

- ADORA implementiert in einem Werkzeug-Prototypen
  - Wichtigste Sprachelemente
  - Layout-Algorithmen
- Beispiel

Schlussbemerkungen und Fragen

- Zentrale Arbeiten zu ADORA [Glinz02], [Bern02], [Joos99], [Xia03]
- Weitere Infos unter:

  http://www.ifi.unizh.ch/req/adora

- Veröffentlichung des ADORA-Werkzeug-Prototyp in Kürze unter GPL auf der ADORA-Web-Seite.
- Fragen?

Referenzen

[Coen97] Coen-Posisini, A., Ghezzi, C., Kemmerer, R.A. (1997). Specification of Realtime System Using ASTRAL, IEEE Transactions on Software Engineeering **23**, 9 (Sept. 1997). 704 – 736

[Glinz02] Glinz, M., S. Berner, S. Joos (2002). Object-oriented modeling with ADORA. *Information Systems* **27**, 6. 425-444.

[Bern02] Berner, S. (2002). *Modellvisualisierung für die Spezifikationssprache ADORA.* [Visualization of models for the specification language ADORA (in German)] PhD Thesis, University of Zurich.

[Joos99] Joos, S. (1999). *ADORA-L – Eine Modellierungssprache zur Spezifikation von Software- Anforderungen* [ADORA-L – A modeling language for specifying software requirements (in German)]. PhD Thesis, University of Zurich.

[Seyb03] Seybold, C., M. Glinz, S. Meier, N. Merlo-Schett (2003). An Effective Layout Adaptation Technique for a Graphical Modeling Tool. (short paper in research demonstration session). *Proceedings of the 2003 International Conference on Software Engineering*, Portland.

[Xia03] Xia, Y. (2003). *A Language Definition Method for Visual Specification Languages*. PhD Thesis, University of Zurich (to appear).

