35 Szenarienanalyse mit Anwendungsfalldiagrammen (Querschneidende dyn. Modellierung)

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-1.0, 29.06.13

1) Anwendungsfalldiagramme
2) Szenarienanalyse mit Interaktionsdiagrammen
3) Szenarienanalyse mit Aktionsdiagrammen
4) Konnektoren
5) Querschneidende Verfeinerung

Obligatorische Literatur

- Zuser, Kap. 7-9, insbes. 7.3+7.5
- Störrle Kap 9, Kap 12, Störrle 5.3, 5.4
- Imprint, S. 58 – 77

Übung

- U12
- Übungsskript, Anhang

Weitere Literatur

- L. Maciaszek. Requirements Analysis and System Design – Developing Information Systems with UML. Addison-Wesley.
- Giancarlo W. Guizzardi. Ontological foundations for structure conceptual models. PhD thesis, Twente University, Enschede, Netherlands, 2005.
- Nicola Guarino, Chris Welty. Supporting ontological analysis of taxonomic relationships. Data and Knowledge Engineering, 39:51-74, 2001.

Überblick Teil III: Objektorientierte Analyse (OOA)

1. Überblick Objektorientierte Analyse
   1. (schon gehabt:) Strukturelle Modellierung mit CRC-Karten
2. Strukturelle metamodellgetriebene Modellierung mit UML für das Domänenmodell
   1. Strukturelle metamodellgetriebene Modellierung
   2. Modellierung von komplexen Objekten
      1. Modellierung von Hierarchien
      2. (Modellierung von komplexen Objekten und ihren Unterobjekten)
      3. Modellierung von Komponenten (Groß-Objekte)
   3. Strukturelle Modellierung für Kontextmodell und Top-Level-Architektur
3. Analyse von funktionalen Anforderungen
   1. Funktionale Verfeinerung: Dynamische Modellierung und Szenarienanalyse mit Aktionsdiagrammen
   2. Funktionale querschneidende Verfeinerung: Szenarienanalyse mit Anwendungsfällen, Kollaborationen und Interaktionsdiagrammen (35)
4. Beispiel Fallstudie EU-Rent

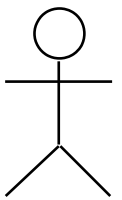
Motivation

- Bisher haben wir in der Analyse objektzentriert vorgegangen, d.h., wir haben Objekte verfeinert (*punktweise Verfeinerung*)
- Wir können aber auch das Objektnetz im Fokus haben, d.h. man erweitert mehrere Objekte auf einmal (*querscheidende Verfeinerung*)
  - Assoziationen
  - Kollaborationen
  - Interaktionsdiagrammen
- Dazu nutzen wir sog. Szenarien, in denen mehrere Objekte kooperieren

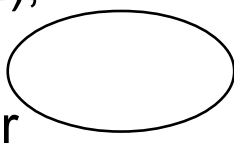
35.1 Anwendungsfalldiagramme

Nutzeranalyse (Stakeholder Analysis)

- ***Nutzer (Stakeholder)***: Nutznießer des Systems
  - **Akteur, Aktor** (Benutzer des Systems oder Interakteur)
  - Eigner von involvierten Betrieben
  - Die, die mit dem System Geld verdienen oder verlieren
  - Menschen, die unter Seiteneffekten des Systems leiden
- Die einfachste Form von Stakeholderanalyse kümmert sich nur um *Akteure*
  - und liefert eine *Liste von Akteuren*
  - Diese Akteure werden dann weiter in Anwendungsfalldiagrammen eingesetzt



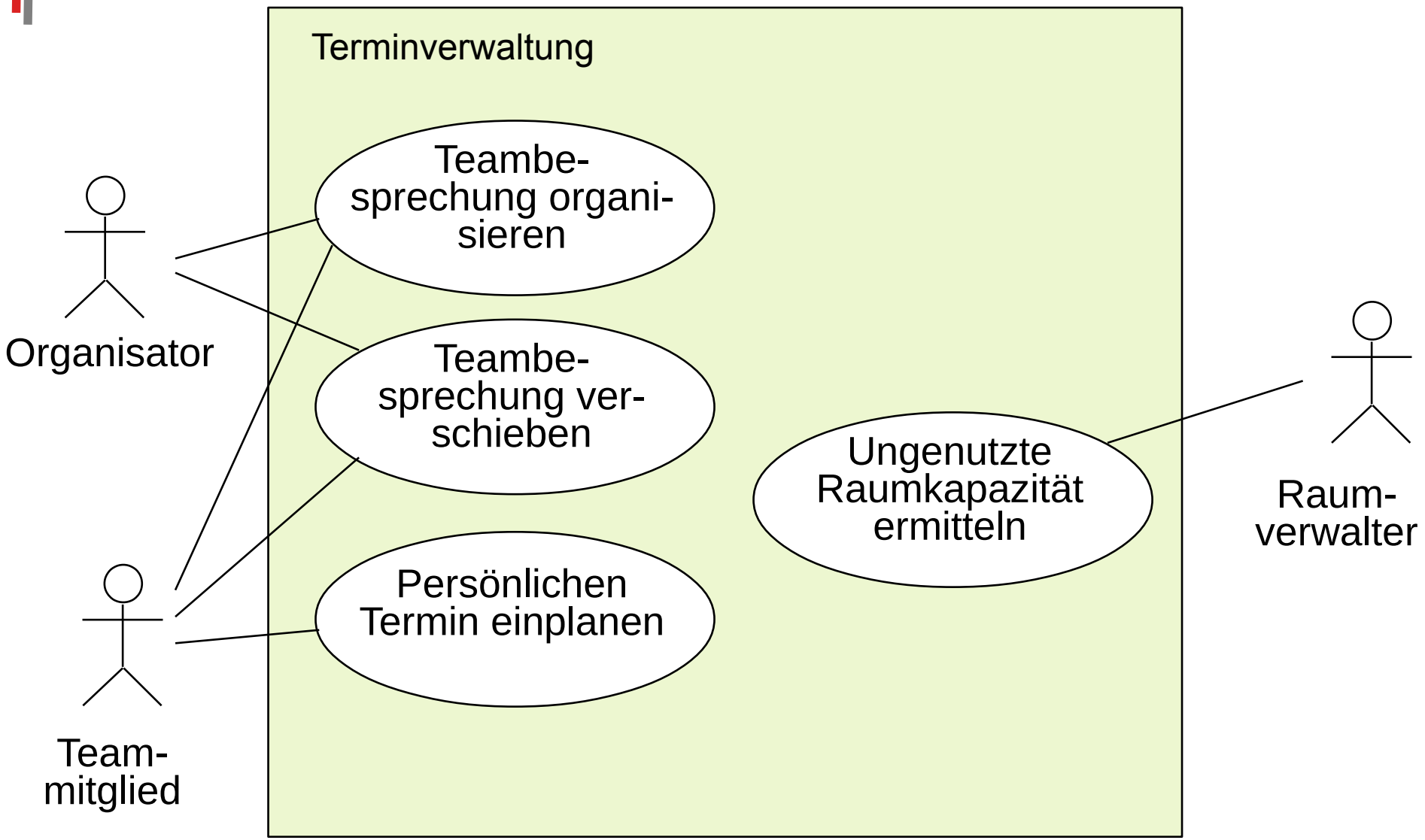
- Ein ***Akteur*** beschreibt eine Rolle, die ein Benutzer (oder ein anderes System) spielt, wenn er/es mit dem System interagiert.
- Ein ***Anwendungsfall*** (Nutzfall, Use-Case, engl. *use case*) ist die Beschreibung einer Klasse von Aktionsfolgen (einschließlich Varianten), die ein System ausführen kann, wenn es mit Akteuren interagiert.



- Eine ***Interaktion*** ist der Austausch von Nachrichten unter Objekten zur Erreichung eines bestimmten Ziels (Akteur-Anwendungsfall-Kommunikation) .

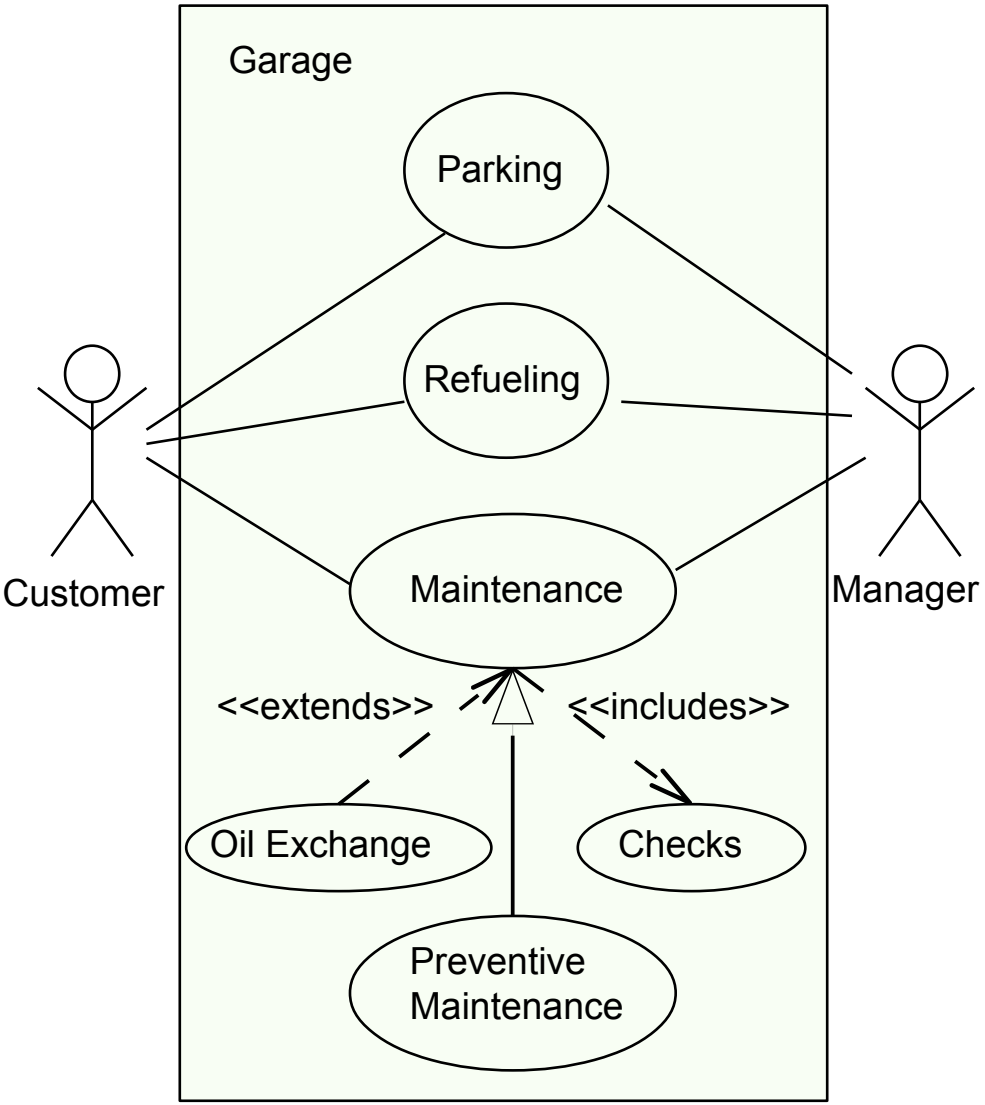
UML-Anwendungsfall-Diagramm (Use-Case-Diagramm) mit Akteuren

- Ein Anwendungsfall beschreibt die Interaktion (Kollaboration) der Akteure



Verallgemeinerung, Erweiterung und Aufruf von Anwendungsfällen

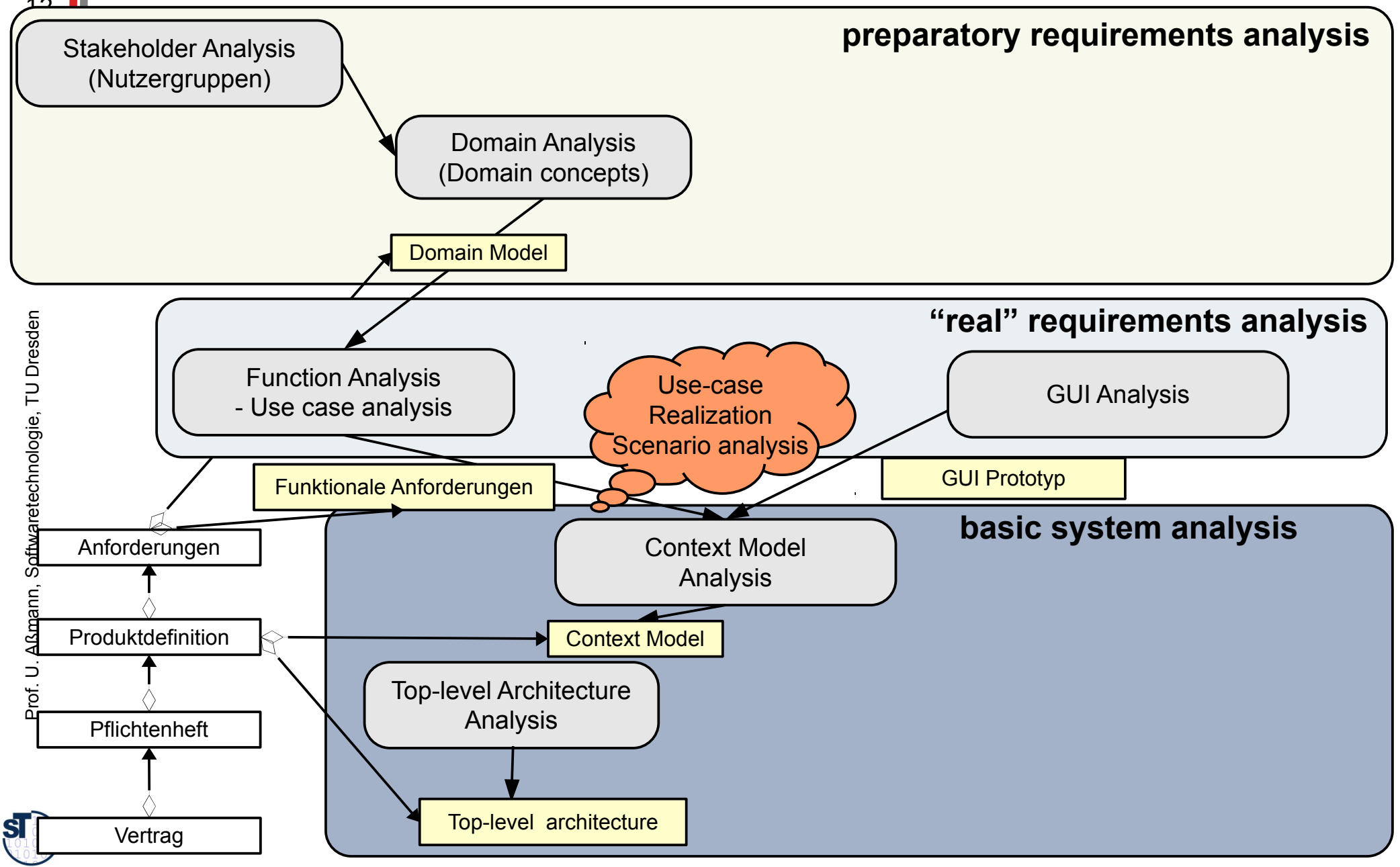
- Die Vererbungsrelation beschreibt Generalisierung bzw. Spezialisierung
  - Hier: Maintenance ist allgemeiner als Preventive Maintenance
- Die *Includes*-Relation beschreibt Bestandteile der Aktionen (Aufrufbeziehung zwischen Aktionen)
  - Hier: Maintenance beinhaltet Checks
- Die *Extends*-Relation beschreibt *optionale* Erweiterungen
  - Hier: Oil Exchange *kann* Teil von Maintenance sein



35.2 Szenarienanalyse - Ableitung von Kollaborationen aus Anwendungsfällen

Anwendungsfallrealisierung, use case realization

Erinnerung: Schematischer Ablauf der Analyse



Wege der Szenarienanalyse (use case realization analysis)

- Die Methode der ***Anwendungsfallrealisierung*** *(use case realization, Szenarienanalyse, scenario analysis)* wird verwendet, um:
  - Kontextmodell und Top-Level-Architektur abzuleiten
  - Querschneidende Verfeinerung durch mehrere Klassen/Objekte durchzuführen
  - Kollaborationen (Teams) und Konnektoren für die Objektverfettung abzuleiten
- Anwendungsfallrealisierung nutzt verschiedene Szenariendiagramme:
  - Verfeinere Anwendungsfalldiagramm mit *Interaktionsdiagrammen*
    - mit Sequenzdiagramm (sequence diagram, sequence chart)
    - mit Kommunikationsdiagramm (communication diagram)
  - Verfeinere Anwendungsfalldiagramm mit *Aktionsdiagrammen*
    - mit Schwimmbahnen im Aktivitätsdiagramm
    - mit einem Netz von kommunizierenden Verhaltens-(Zustands-)maschinen

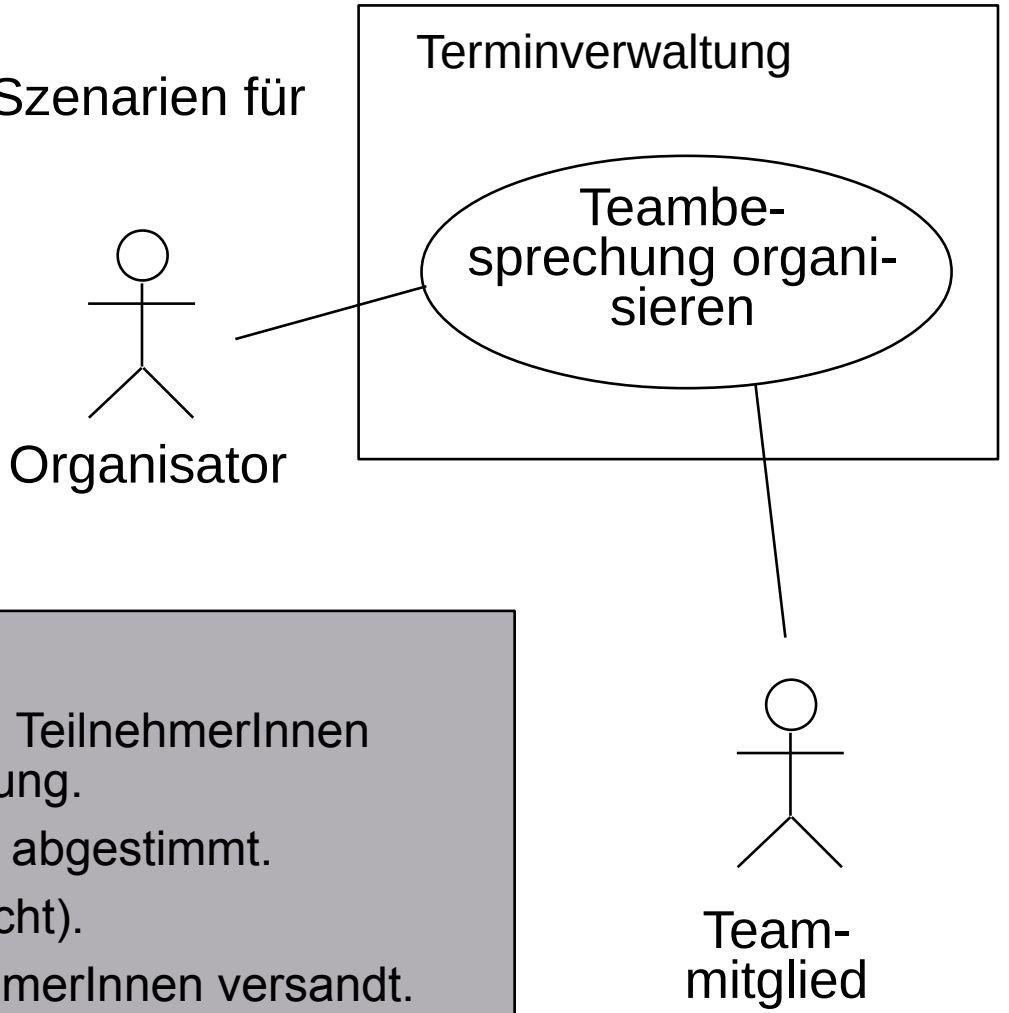
Szenarien

- **Definition:** Ein ***Szenario*** ist eine Beschreibung einer beispielhaften Folge von *Interaktionen* von Akteuren mit dem System zur Beschreibung eines Anwendungsfalls (use case realization).
  - Es gibt Szenarien für Normalfälle ('gut-Fälle'), Ausnahmefälle ('exception case') und Fehlerfälle ('negativ'-Fall).
- Szenarien spielen Anwendungsfälle durch
  - ermittle zeitliches Zusammenspiel, verfeinere über der Zeit
  - ermittle feinere Aktionen und binde sie mit Vererbung ein
  - ermittle Unteraktionen und binde sie mit <<includes>> ein
  - ermittle optionale Erweiterungen von Aktionen und binde sie mit <<extends>> ein
- Wähle als Szenariobeschreibung durch Interaktionsdiagramme oder Aktionsdiagramme
  - Leite daraus eine Kollaboration ab (Konnektor, Team)

Szenarienanalyse

- beginnt mit Anwendungsfällen und analysiert das Zusammenspiel der Akteure
- **Beispiel:**

  Durchspielen eines der Normalfall-Szenarien für 'Teambesprechung organisieren'

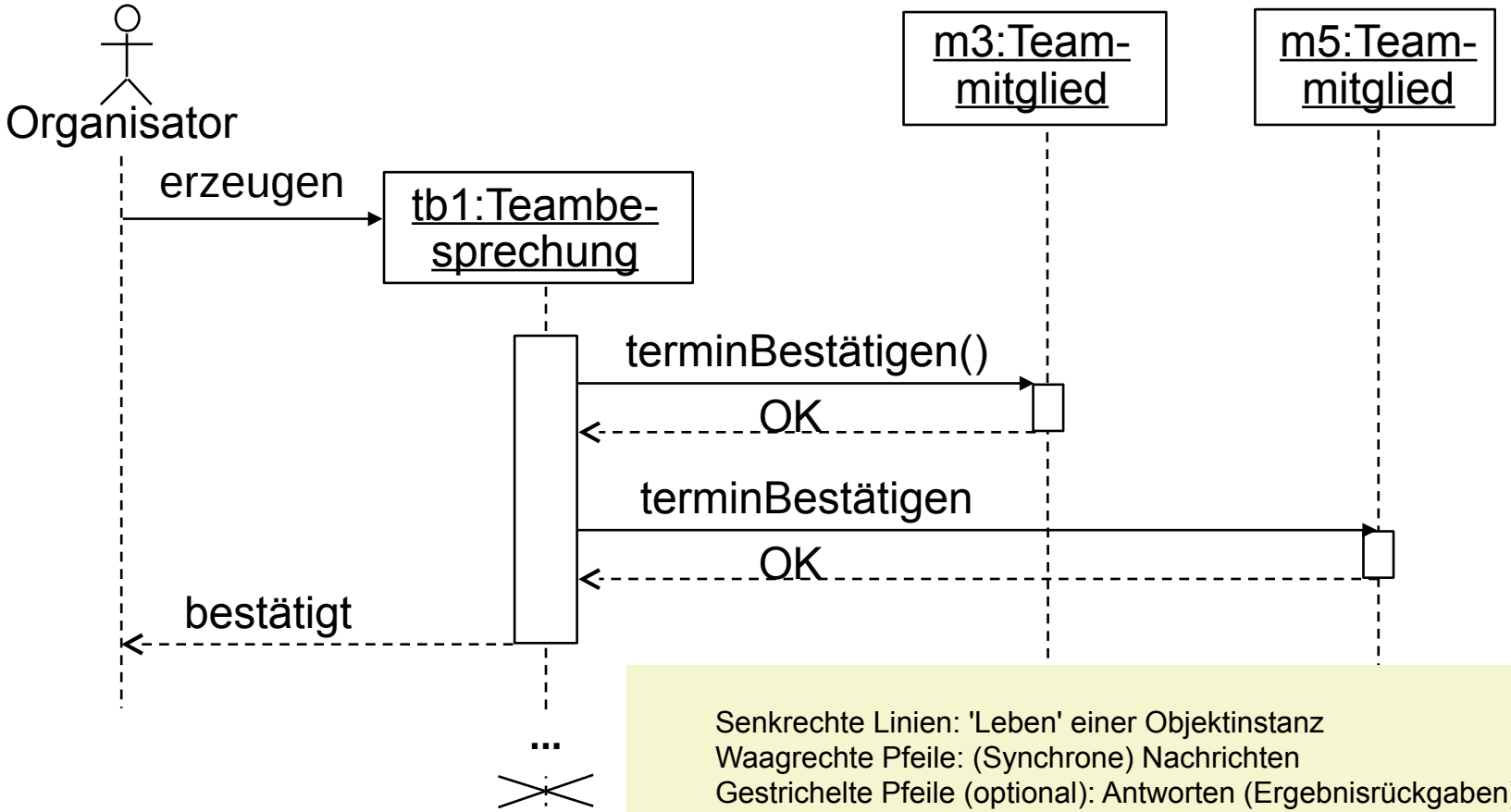


Durchspielen:

- Organisator erfährt Thema, Termin, TeilnehmerInnen einer neu geplanten Teambesprechung.
- Zeitpunkt wird mit TeilnehmerInnen abgestimmt.
- Raum wird reserviert (falls gewünscht).
- Einladungen werden an die TeilnehmerInnen versandt.

35.2.1 Szenarienanalyse mit UML-Sequenzdiagrammen

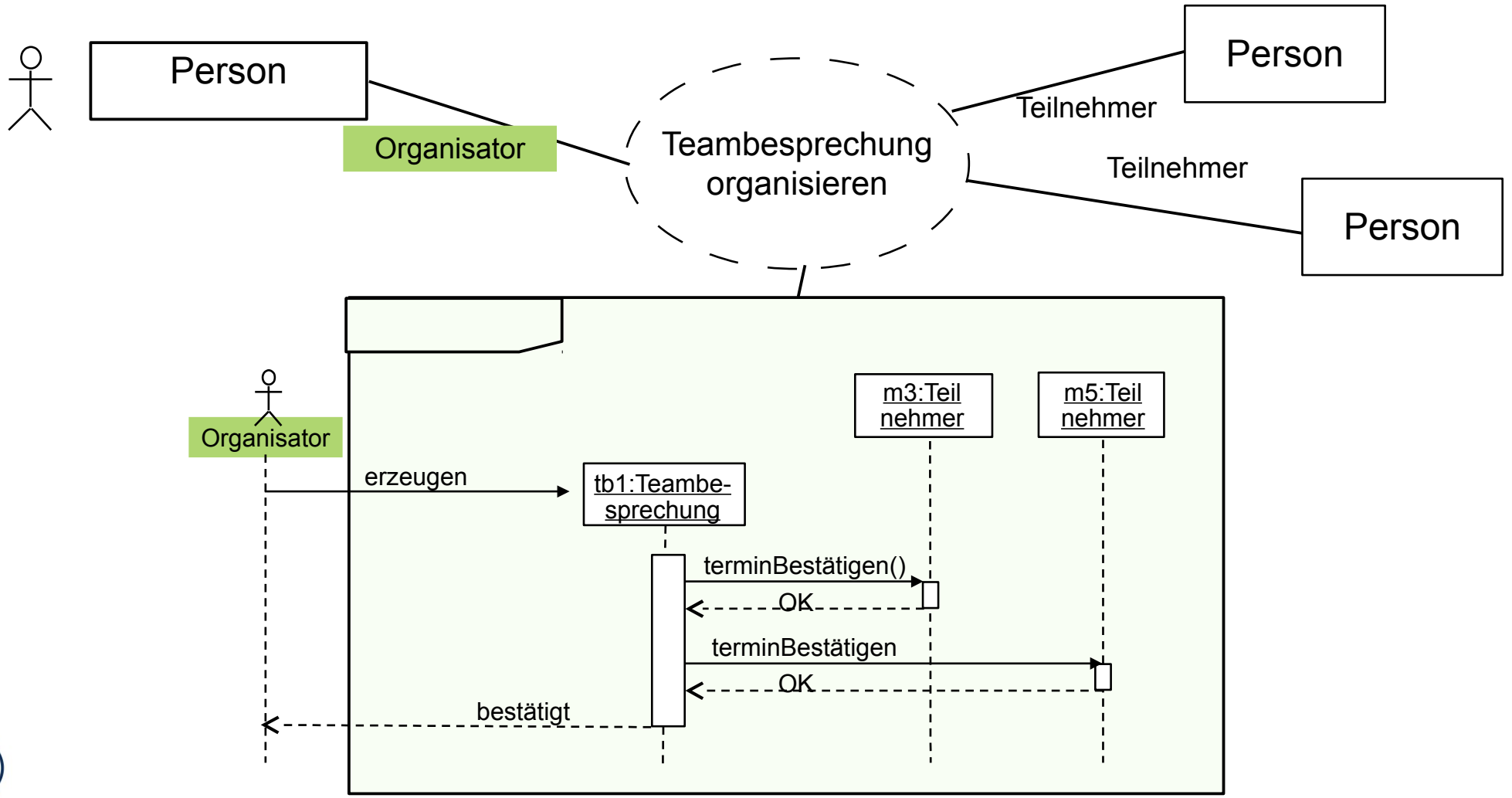
- **Sequenzdiagramm** ist eine **Objekt-Lebenszeit-Matrix**, in der die Objekte von links nach rechts aufgereiht sind und die Zeit von oben nach unten läuft (Objekt-Lebenslinien)
  - Sequenzen von Nachrichten, geordnet durch die Zeit
- Achtung: das Sequenzdiagramm schneidet quer durch das Lebenszyklus mehrerer Objekte und beschreibt ein Szenario



Senkrechte Linien: 'Leben' einer Objektinstanz
Waagrechte Pfeile: (Synchrone) Nachrichten
Gestrichelte Pfeile (optional): Antworten (Ergebnisrückgaben)
Blöcke auf den senkrechten Linien: Steuerfokus (Aktivierung)

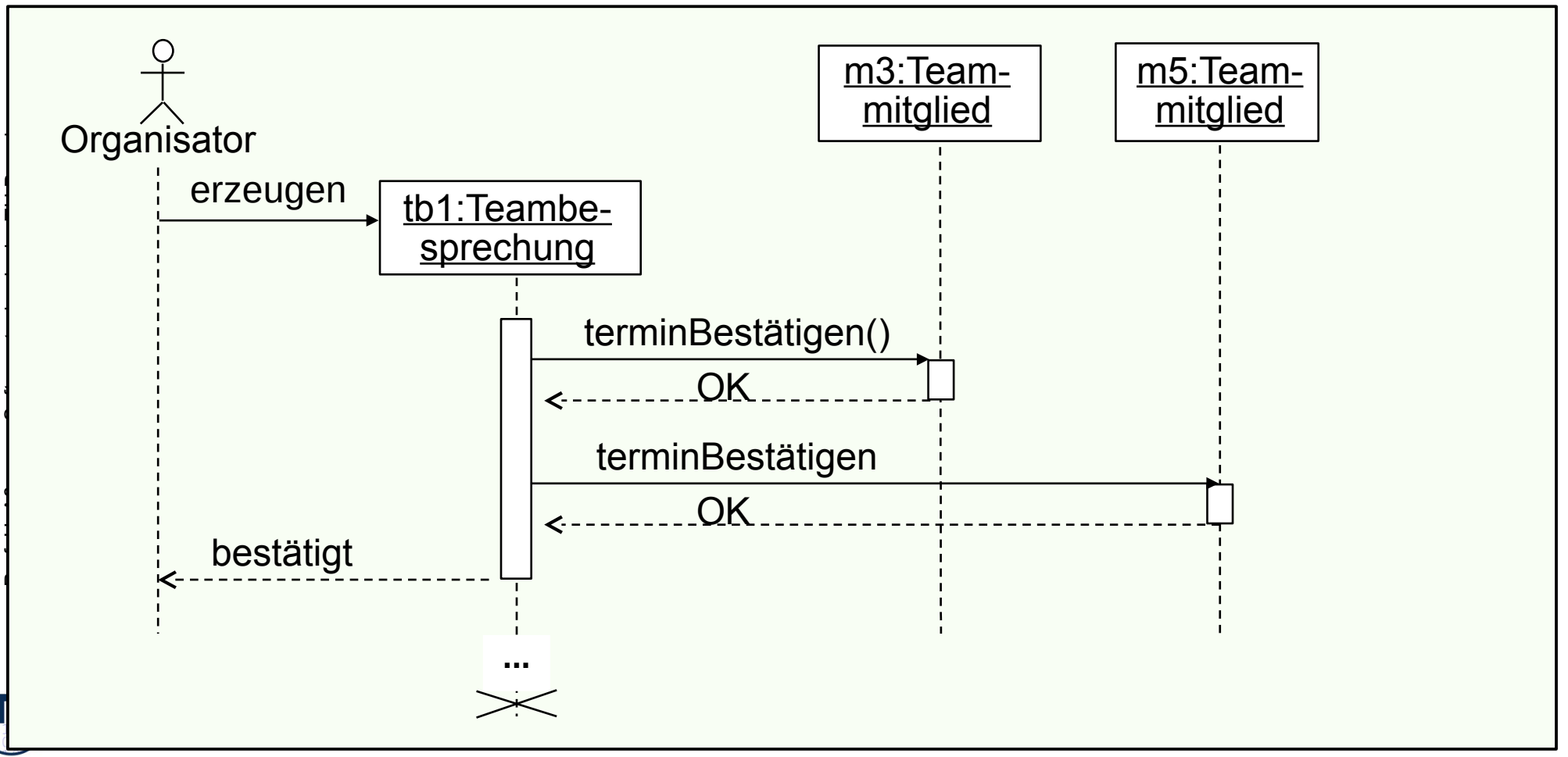
Kapseln eines Szenarios in einer Kollaboration

- Eine **Kollaboration** kann mit einem Sequenzdiagramm als Verhalten unterlegt werden
  - Die einzelnen Lebenslinien geben das Verhalten einer Rolle der Kollaboration an
- Die Kollaboration beschreibt also ein Szenario querschneidend durch die Lebenszyklen mehrerer Objekte

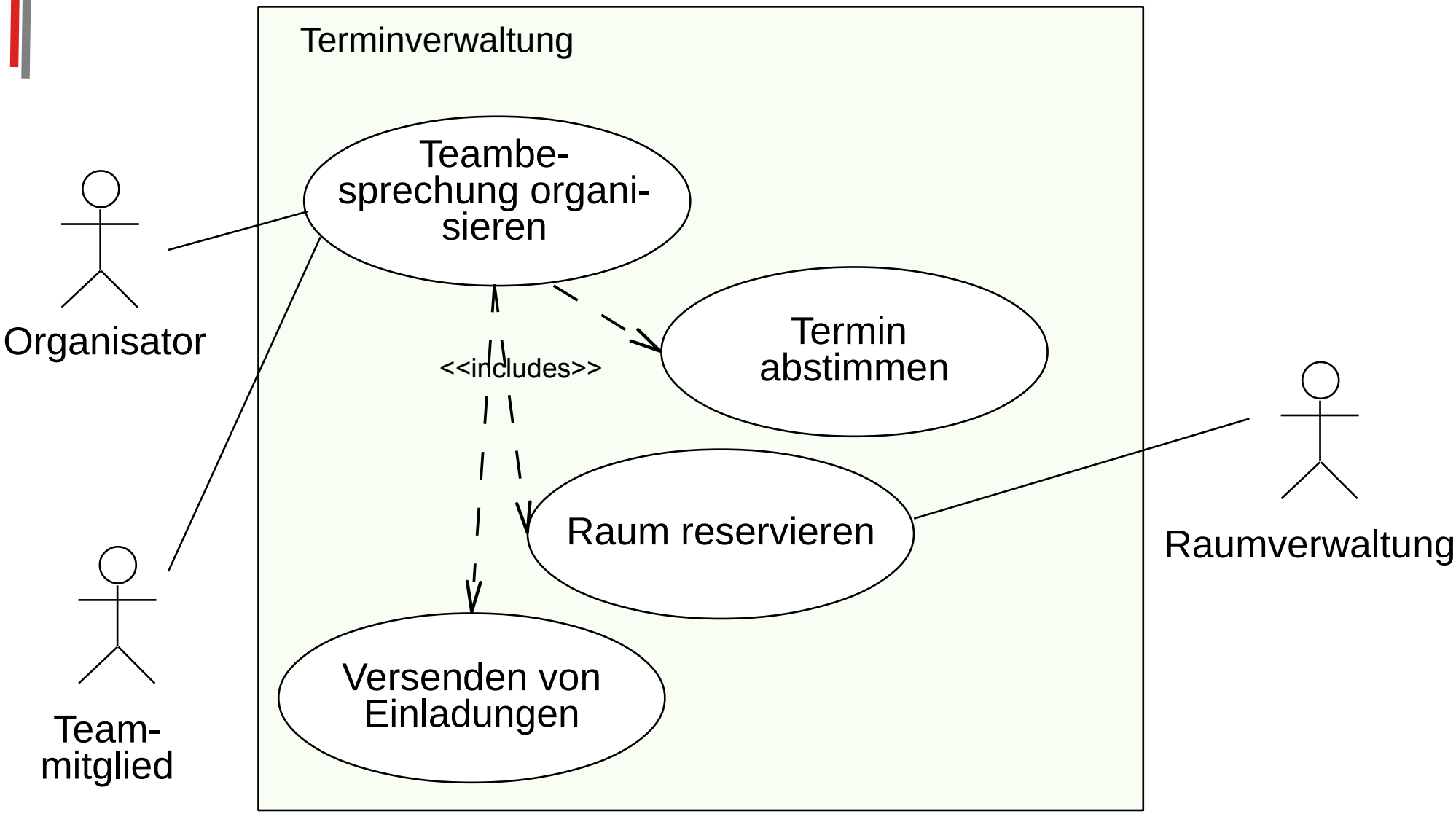


Einordnung in Kontextmodell und Top-Level-Architektur

- Nach der Szenarienanalyse muss unterschieden werden, welche Klassen zum Kontextmodell und welche zur Top-Level-Architektur gehören
- Hier: noch alles im Kontextmodell

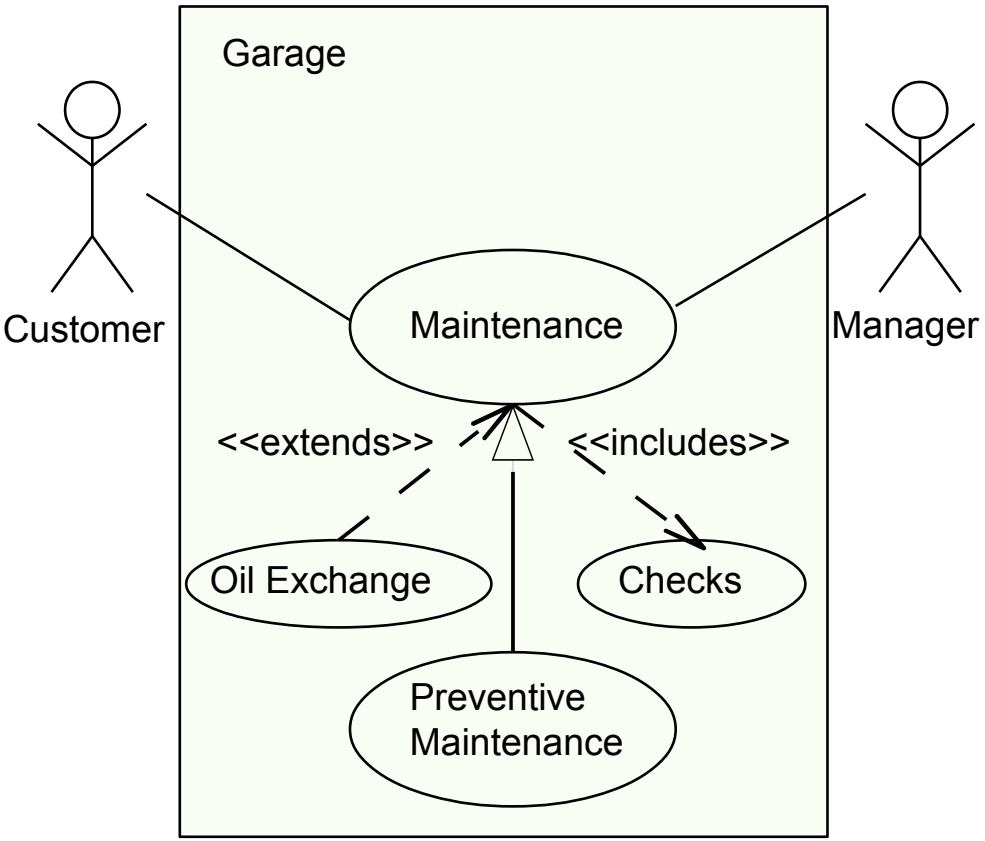


Verfeinerung des Anwendungsfalls mit Ergebnissen der Szenarienanalyse



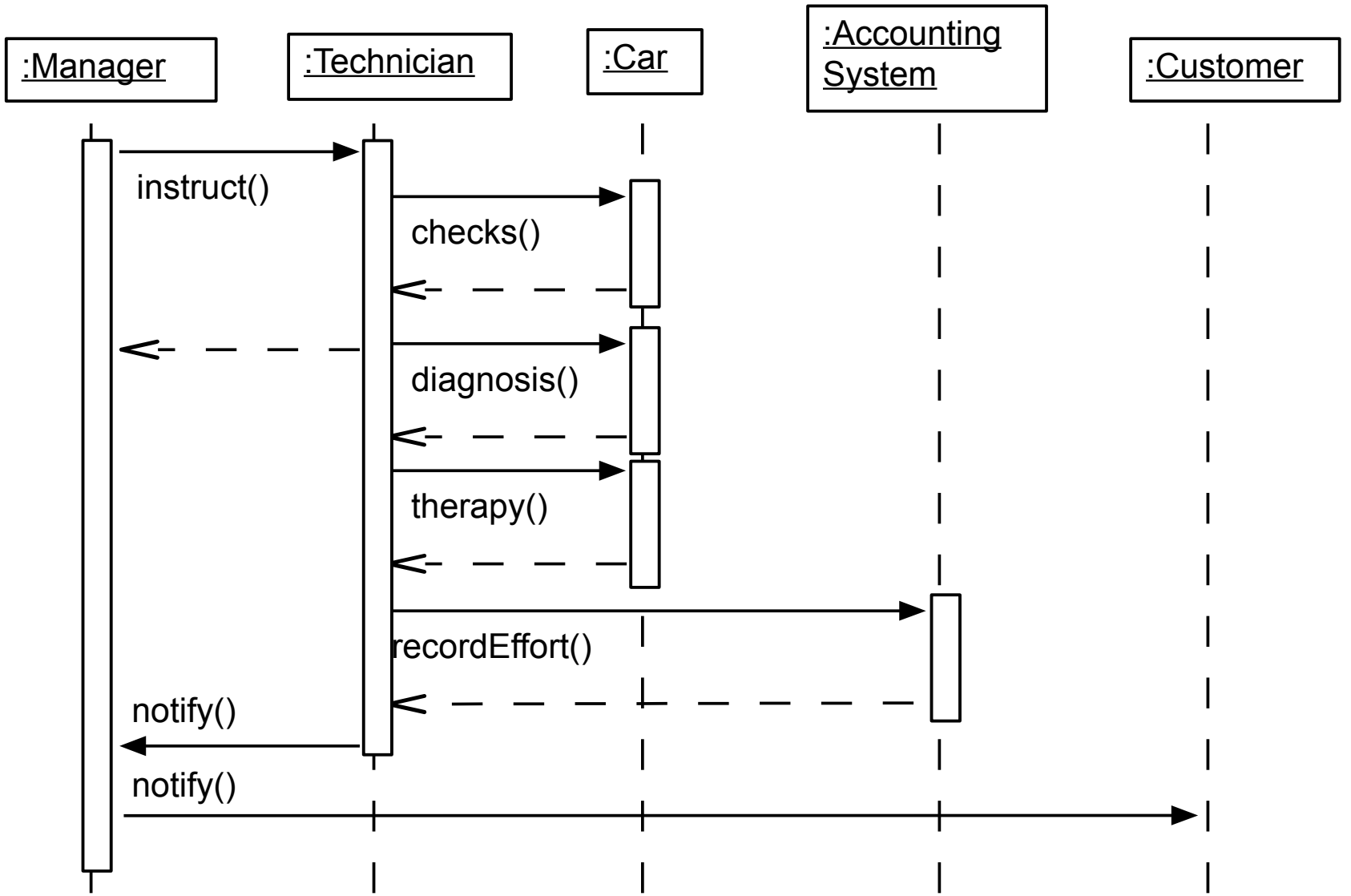
Szenarienanalyse mit Sequenzdiagrammen

- Ausgangspunkt: Anwendungsfall Auto-Wartung (Maintenance) [Pfleeger]



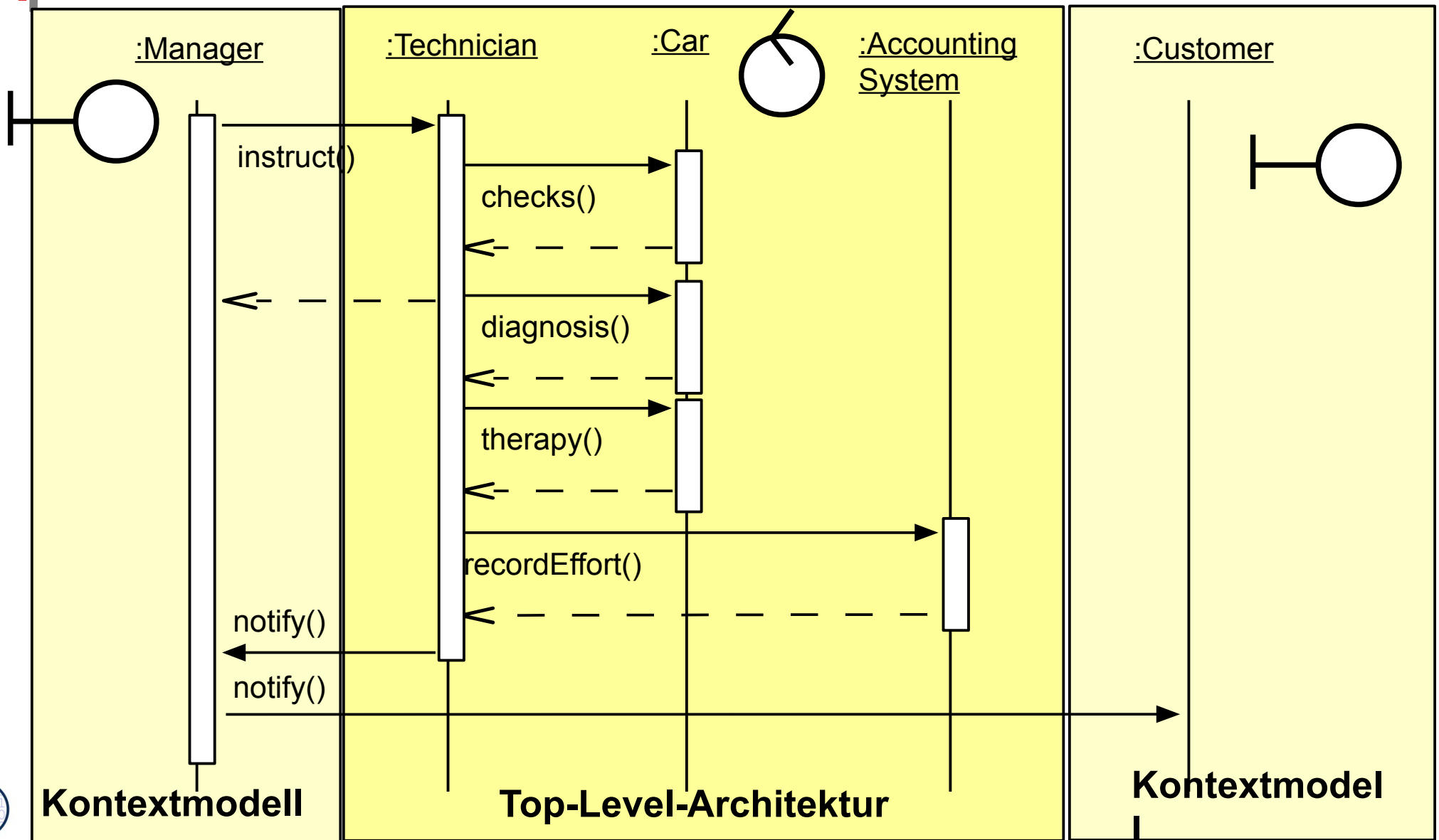
Szenarienanalyse Sequenzdiagram Service-Station

- Sequenzdiagramme werden benutzt zur Analyse von Szenarien mit wenigen Objekten, die viel kommunizieren



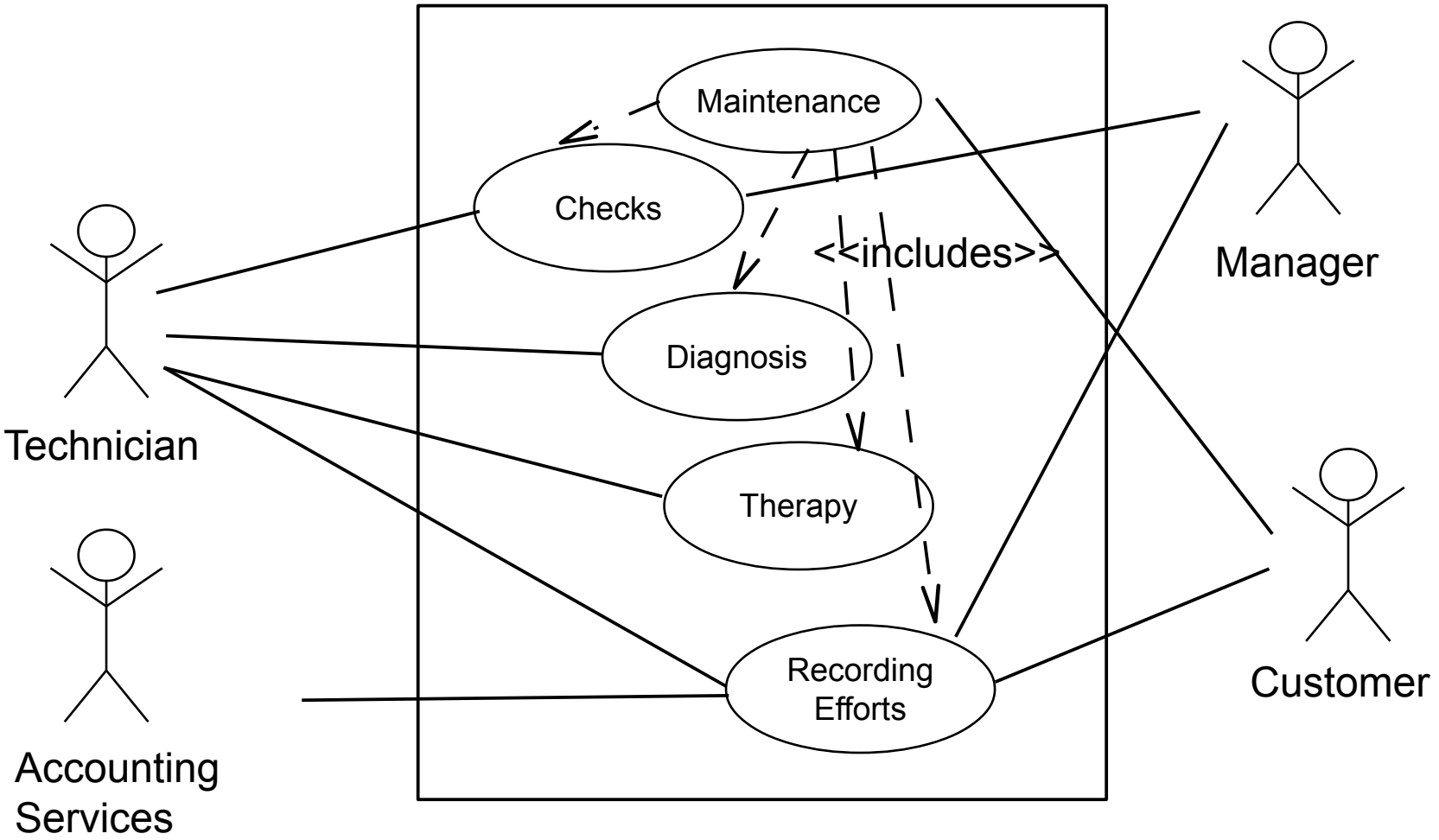
Beziehung zum Kontextmodell und Top-Level-Architektur

- Ein Sequenzdiagramm eines Szenarios muss in die TLA eingeordnet werden: Welche Klassen sind B, C, D?



Verfeinertes Anwendungsfall-Diagramm Service-Station

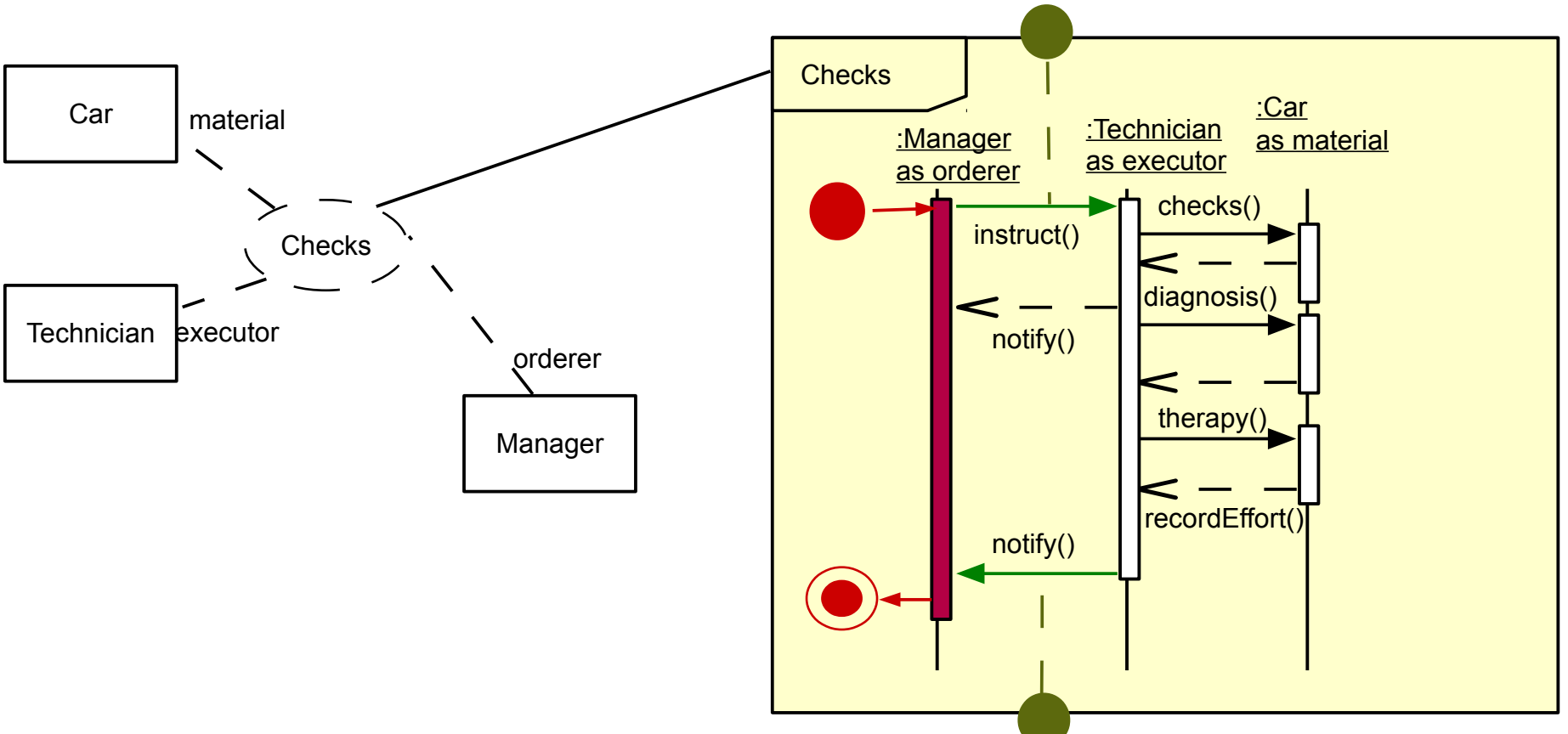
- Aus dem Sequenzdiagramm kann nun ein verfeinertes Anwendungsfalldiagramm erstellt werden



35.2.2 Erstellung von Kollaborationen aus Szenarien

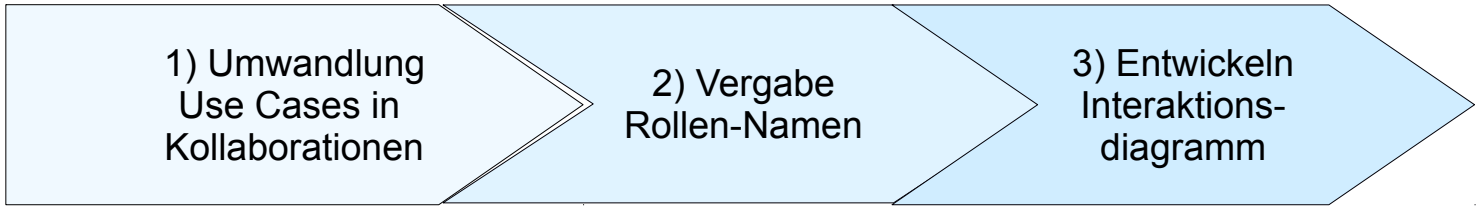
Ableitung von Kollaborationen aus der Szenarienanalyse

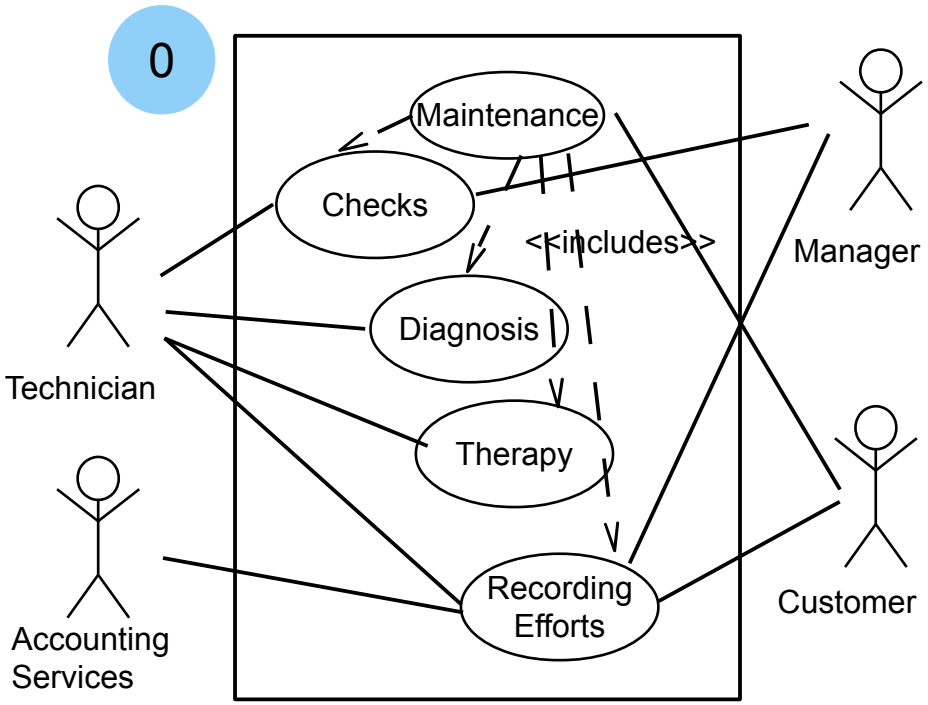
- Ein Sequenzdiagramm einer Kollaboration definiert:
  - Lebenslinien beschreiben das Verhalten der Rollen
  - Die Lebenslinie mit dem Anfangszustand kennzeichnet den **Initiator** mit **Initialzustand**
  - Die Lebenslinie mit dem Endzustand kennzeichnet den **Terminator** mit **Endzustand**;
  - **Initialbotschaft**: erste Botschaft, anliegend am Initialzustand
  - **Terminalbotschaft**: letzte Botschaft, anliegend am Endzustand

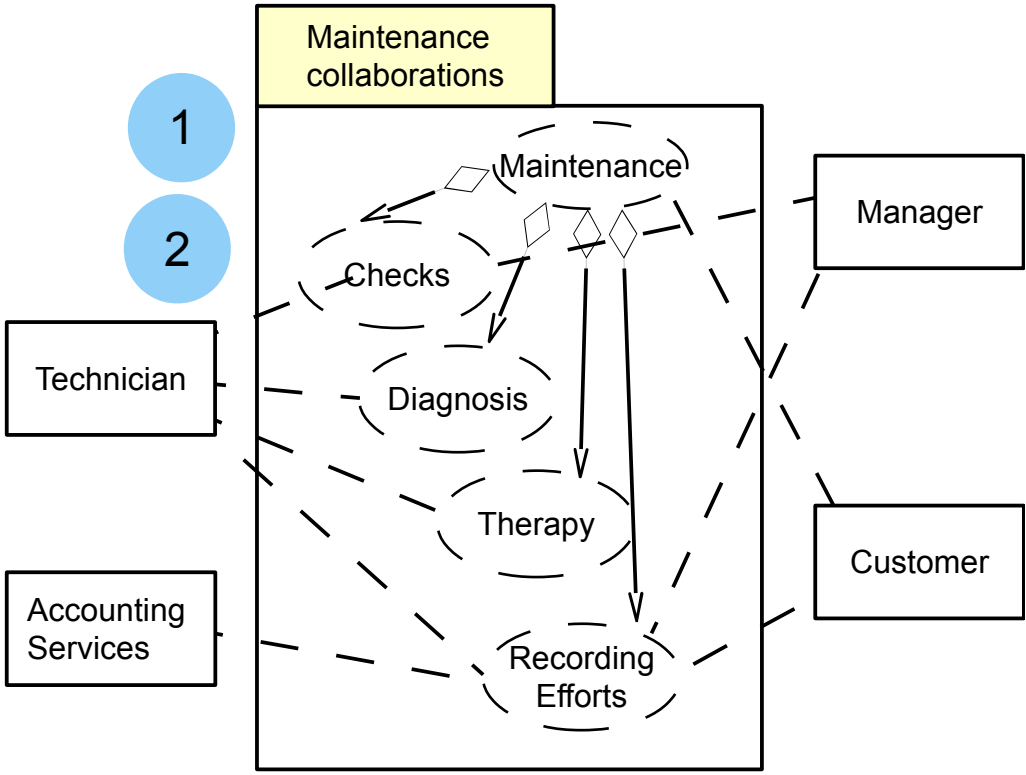


Umwandlung Anwendungsfall-Diagramm in Kollaborationen

- Aus dem Anwendungsfalldiagramm kann schrittweise eine Menge von Kollaborationen erstellt werden

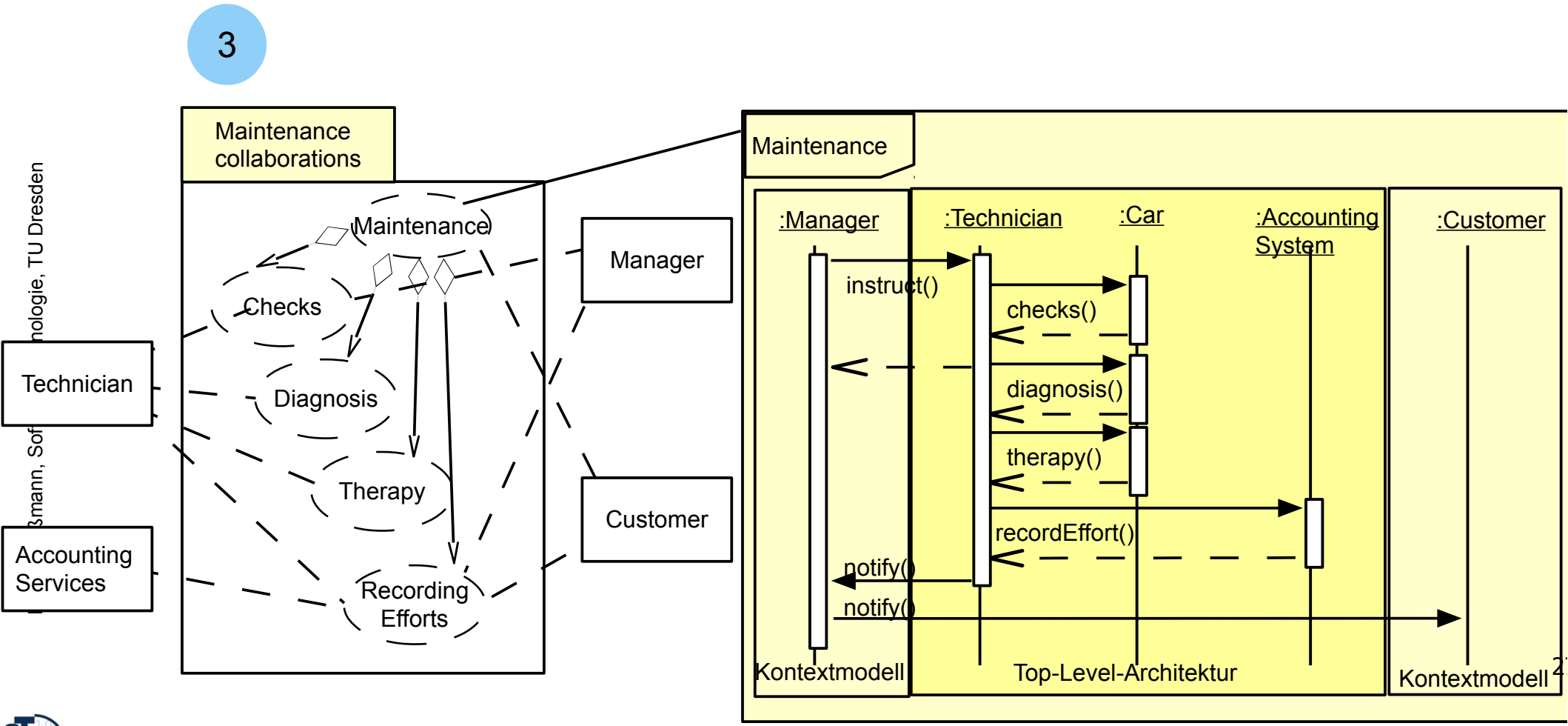






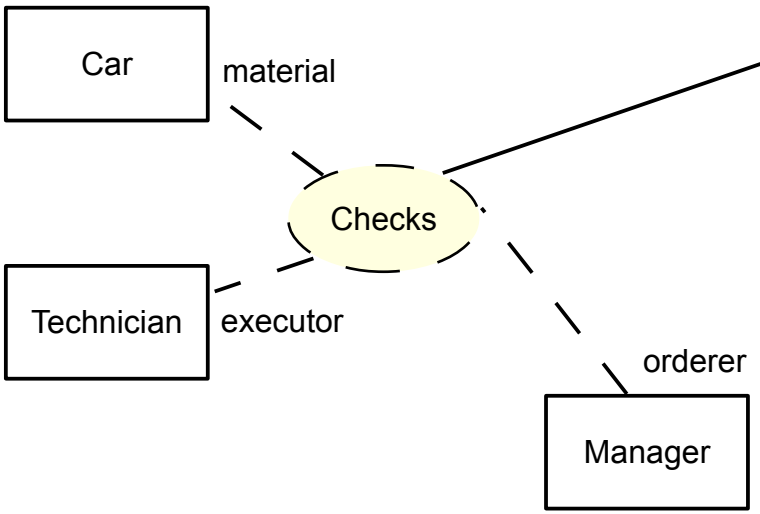
Umwandlung Anwendungsfall-Diagramm in Kollaborationen

- 3) Anhängen des Interaktionsdiagramms (hier Sequenzdiagramm)

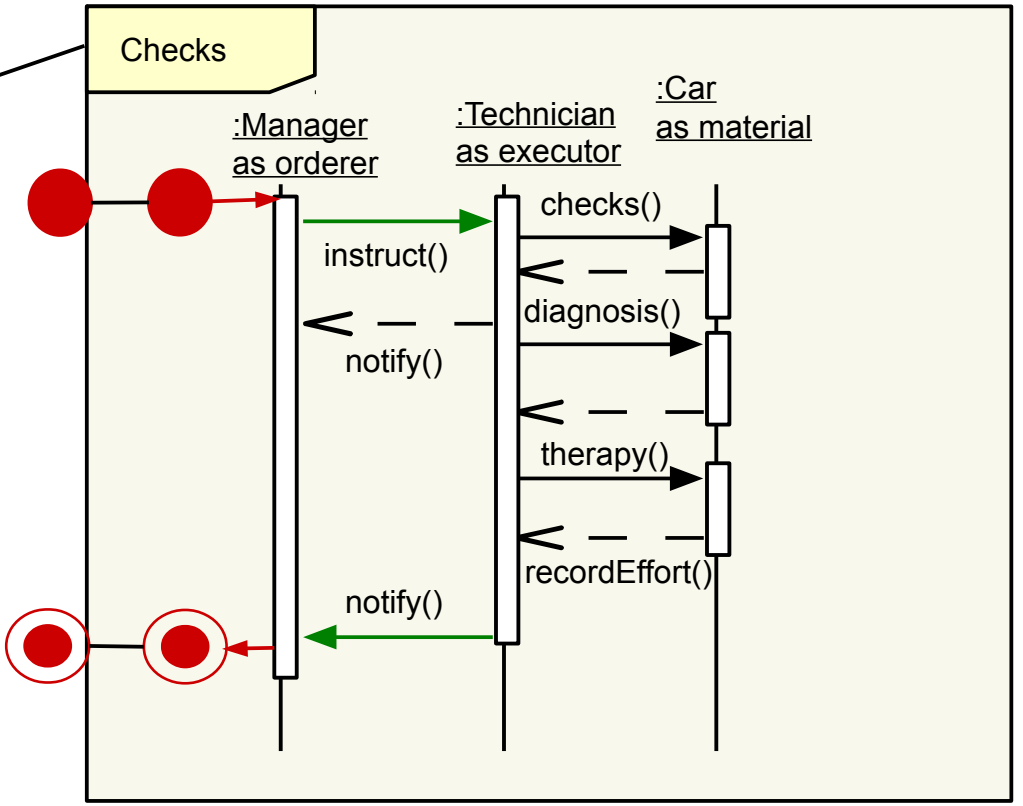


Superimpositionen von Kollaborationen auf Kernobjekte

- **Kollaborationsanknüpfung** (Komposition, Superimposition) an Kernobjekte:
  - Man bettet die Initial- und Terminalzustände bzw. -Botschaften der Lebenslinie einer Initiator- und Terminator-Rolle in die Lebenslinie des Kernobjekts ein
  - Damit werden auch die Initial- und Terminalbotschaften eingebettet
- Die **Superimposition** einer Kollaboration auf ein Klassendiagramm erfolgt:
  - **statisch** durch Codetransformation von Hand oder mit Webewerkzeug
  - **dynamisch** durch Entwurfsmuster



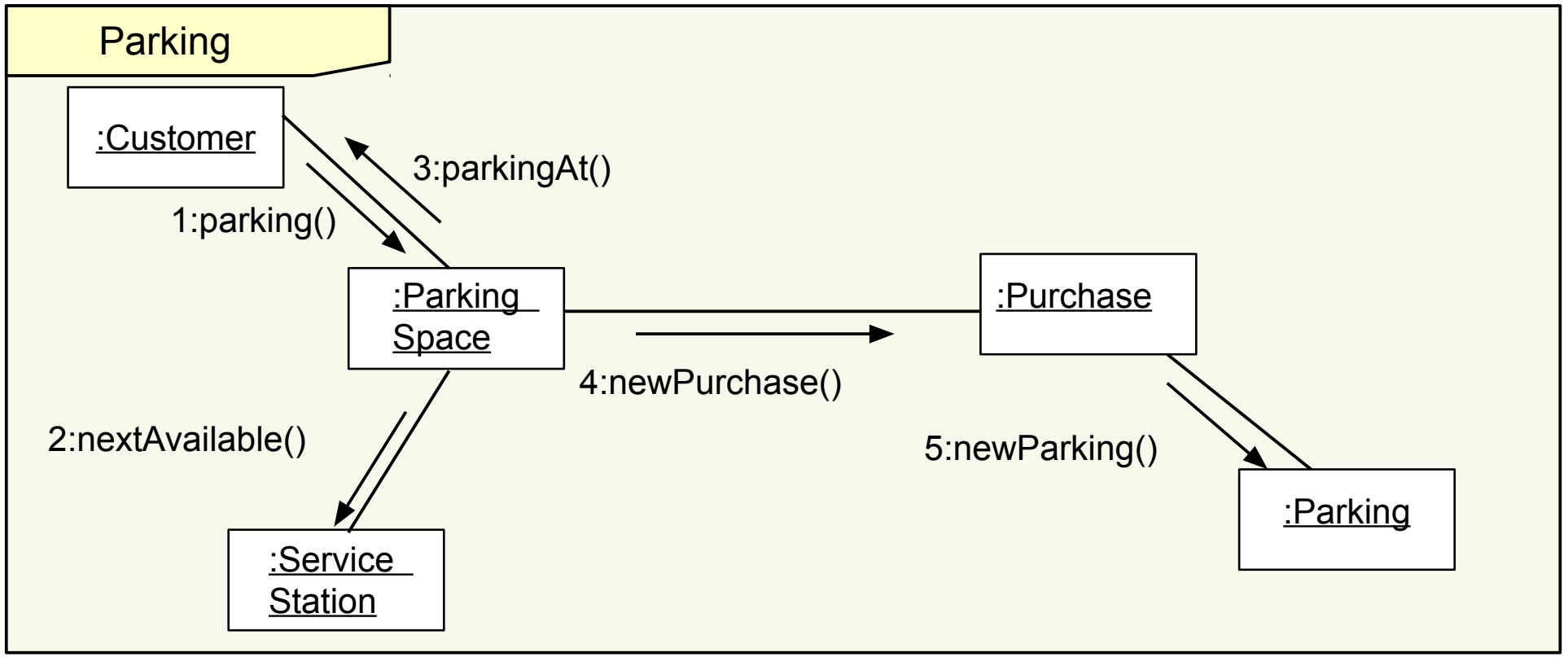
orderer: Initiator, Terminator
orderer.start->copy message instruct()
orderer.stop->union result message notify()



35.2.3 Szenarienanalyse mit Kommunikationsdiagrammen

Kommunikationsdiagramm (Communication Diagram)

- Ein **Kommunikationsdiagramm** ist ein Interaktionsdiagramm, das den Fluss der Aufrufe über der Zeit aufzeichnet
  - Sequenzdiagramm „von oben gesehen“
  - Ohne Objektlebenslinien, flexibles Layout
  - Hierarchische Nummerung drückt die Zeit aus (zeitliche Abfolge der Nachrichten und Aufrufe)
  - Geeignet für viele Objekte, die komplex miteinander verbunden sind



35.3. Szenarienanalyse mit Schwimmbahnen in Aktionsdiagrammen

Szenarienanalyse

durch Aktionsdiagramme: Aktivitätendiagramme, Statecharts

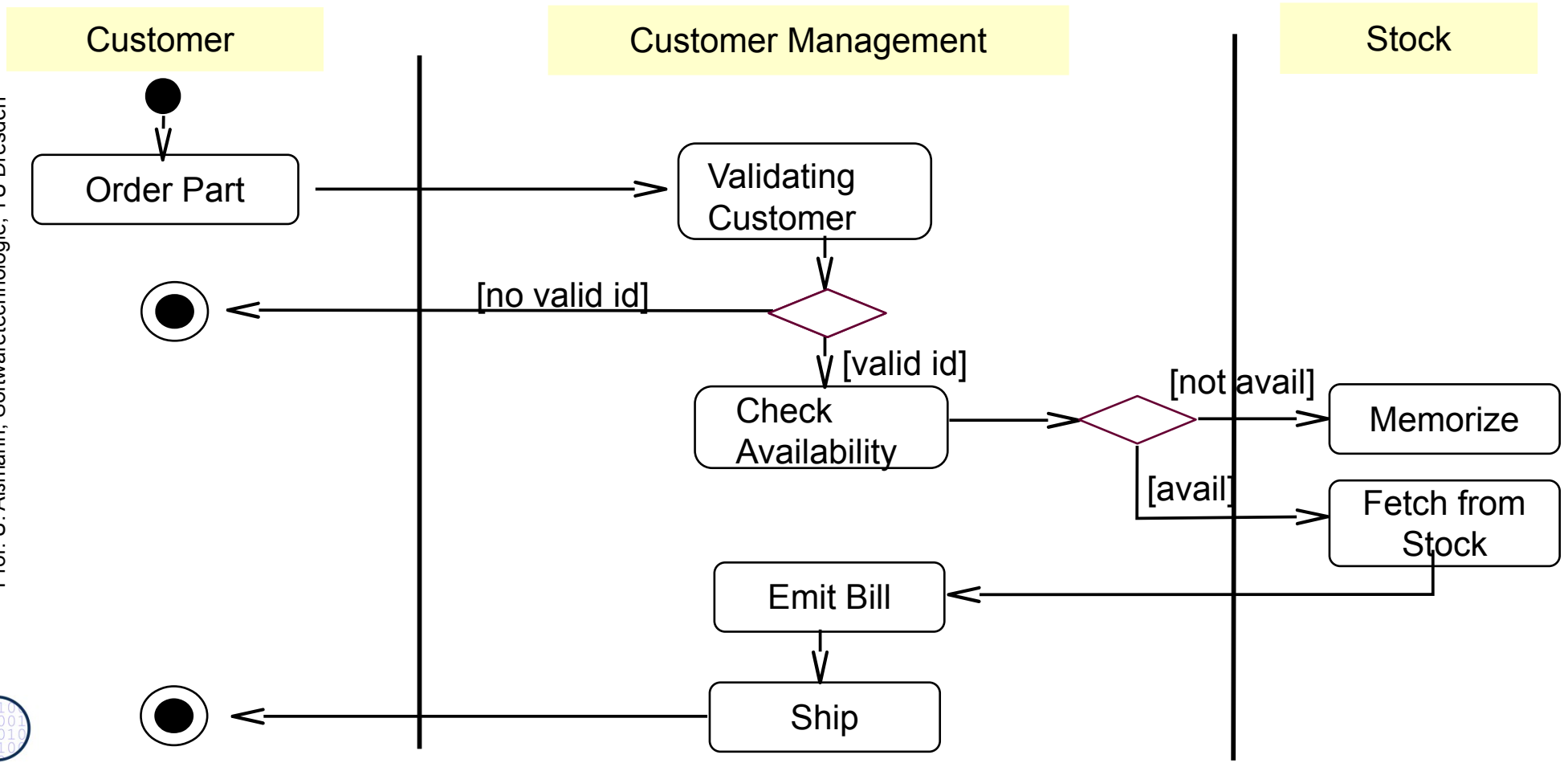
Querscheidende dynamische Modellierung mit Szenarienanalyse

- Mit Aktionsdiagrammen kann man **Lebenszyklen** von Objekten spezifizieren (punktweise Verfeinerung)
- Benutzt man *Schwimmbahnen,* kann man das Zusammenspiel mehrerer Objekte oder Methoden untersuchen (*querschneidende dynamische Modellierung, querschneidende funktionale Verfeinerung*).
- Dazu führt man eine *Szenarienanalyse* durch, die quasi die Draufsicht auf ein Szenario ermittelt

- *Achtung: in UML wird eine Aktivität genau wie ein Zustand mit einem abgerundeten Rechteck dargestellt..*

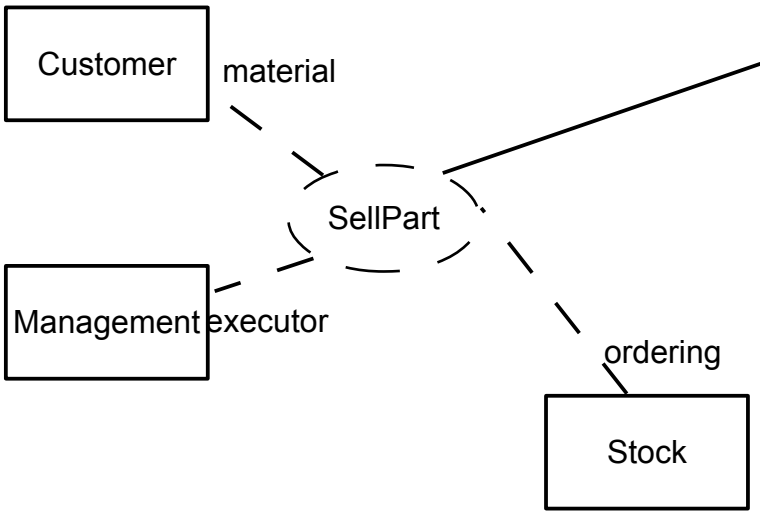
Szenarienanalyse: Bearbeiten einer telefonischen Bestellung

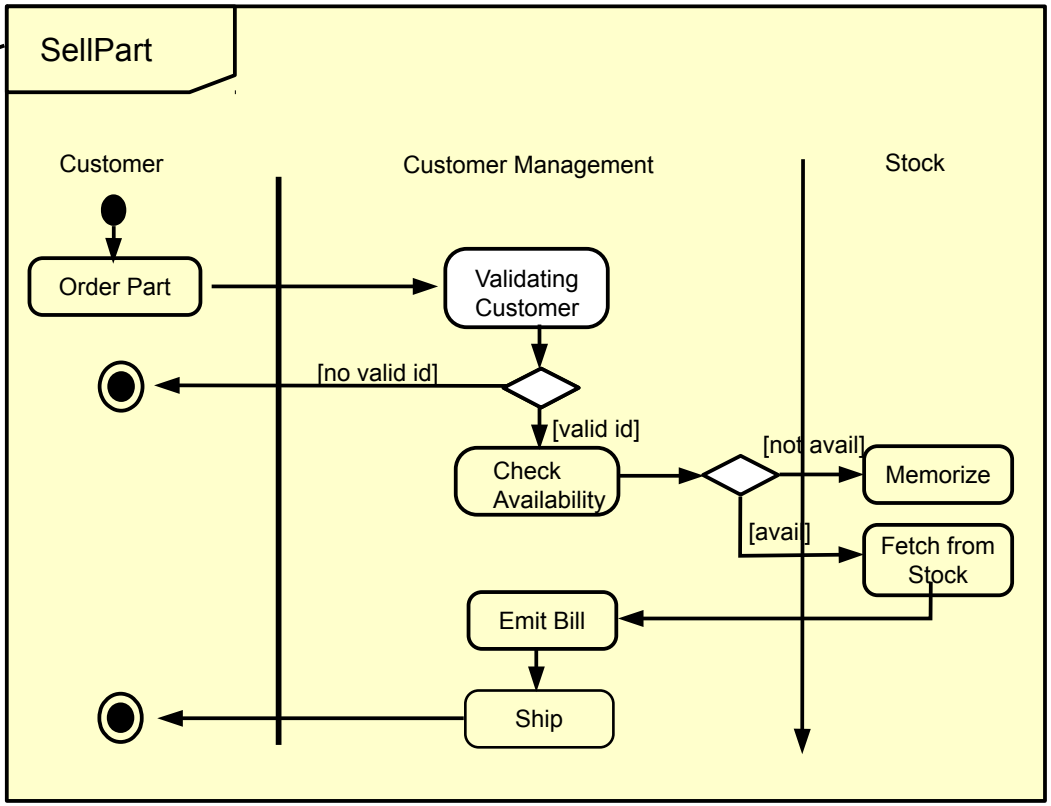
- Mit Aktivitätendiagrammen kann man Szenarienanalyse betreiben
- Aktivitäten können durch **Schwimmbahnen (swimlanes)** gegliedert werden, die Objekten zugeordnet sind
  - Daraus kann man dann Methoden für die beteiligten Objekte ableiten



Aktivitätendiagramme als Verhalten von Kollaborationen

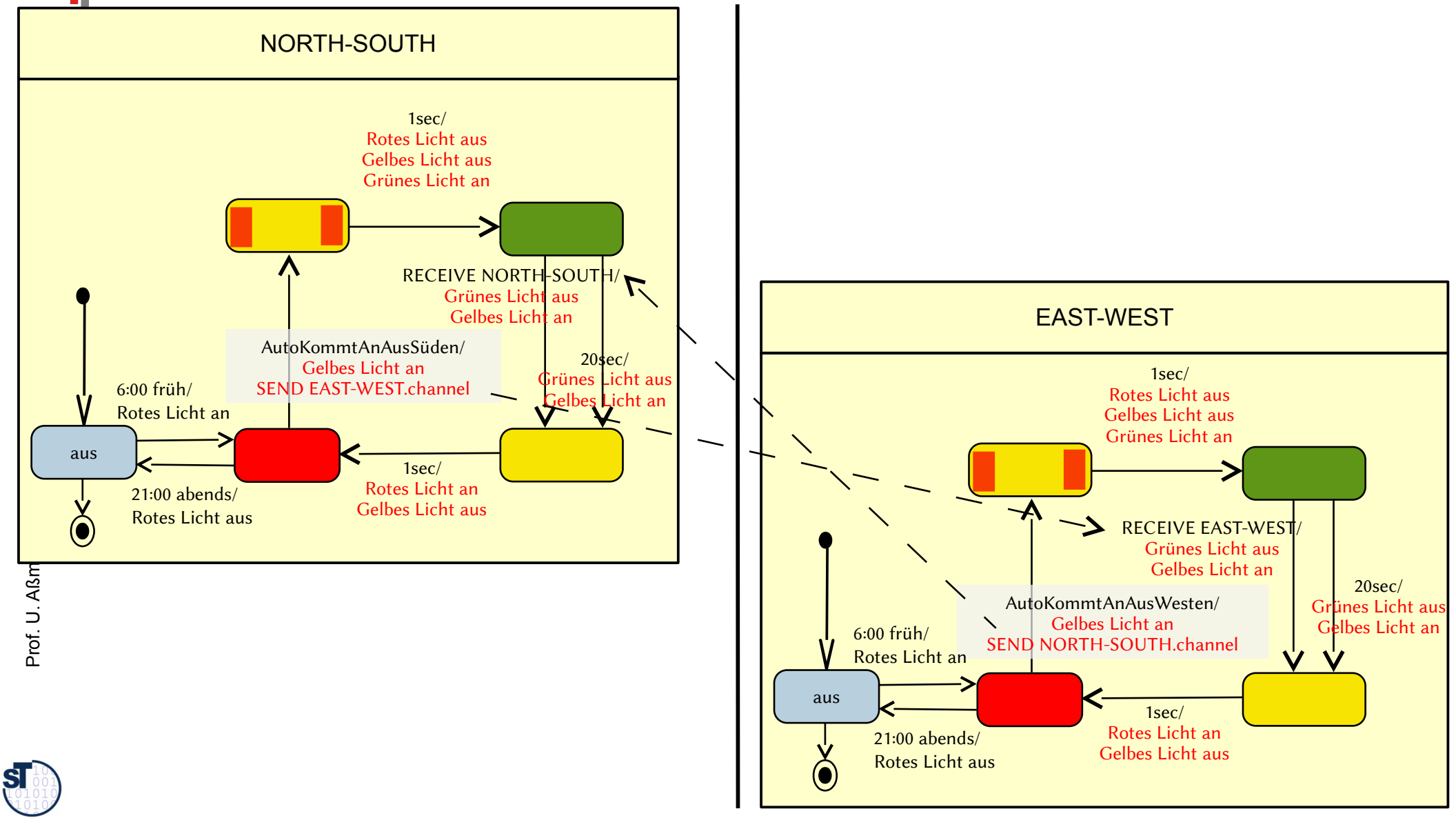
- Aktivitätendiagramme mit swimlanes (Lebensbereichen) können, ähnlich wie Sequenzdiagramme, zu Kollaborationen als Implementierung hinzugefügt werden
  - Die einzelnen Lebensbereiche (swimlanes) geben das Verhalten einer Rolle der Kollaboration an
  - Wieder gibt es Initiator und Terminator-Lebensbereich mit Initial- und Finalzustand





35.3.2. Kopplung zweier Ampeln an einer Kreuzung durch kooperierende Automaten

- Szenarienanalyse mit Statecharts funktioniert ähnlich; es entstehen Netze von kommunizierende Verhaltensmaschinen



36.4 Konnektoren als spezielle Kollaborationen

Konnektoren

- Ein **Konnektorobjekt** ist eine Assoziationsobjekt, das aus bisher nicht kooperierenden Objekten ein Netz aufbaut, bearbeitet und wieder auflöst.
- Eine **Konnektorklasse** ist eine Kollaborationsklasse, die definiert
  - ein Konnektorobjekt
  - die Rollenklassen als innere Klassen
  - Netzaufbau-Methoden, die das Konnektorobjekt mit den Spielern verbinden
  - Netzabbau-Methoden
  - Delegationsmethoden, die auf die inneren Objekte delegieren
- Die Verhalten des Konnektorobjekts wird durch eine Kollaboration beschrieben
  - Sie kann **konnektorgetrieben** erfolgen, so dass auf ein Ereignis hin alle Objekte angestoßen werden (passive Objekte werden exogen vom Konnektor angesteuert)
  - Die Bearbeitung kann **spielergetrieben** erfolgen, sodass ein oder mehrere Objekte aktiv über den Konnektor kooperieren
- **Kanäle** sind objektgetriebene Konnektoren
- In Java implementiert man eine Kollaboration immer als **Konnektor**

Schematische Realsierung von Kollaborationen mit inneren Klassen in Java


```
class Connector {
  /* role as inner class */
  class RoleA {
     do() {..}
  }
  /* role as inner class */
  class RoleB {
     do() {..}
  }

// Definition of inner objects
  PlayerA player_A;
  PlayerB player_B;
  RoleA role_A;
  RoleB role_B;
}
```

```
// Net construction
void link(PlayerA a, PlayerB b) {
  player_A = a;
  player_B = b;
}
// Net destruction
void unlink() {
  player_A = null;
  player_B = null;
}
// Delegation methods
void doA() { role_A.do(); }
void doB() { role_B.do(); }
```

35.5 Querschneidende Verfeinerung von komplexen Objekten mit Kollaborationen im Entwurf

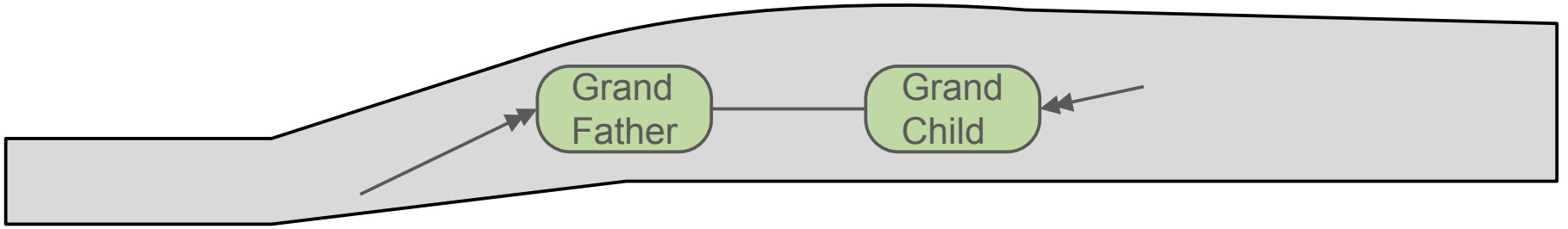
Szenarienanalyse wird nicht nur für Kontextmodell und TLA eingesetzt, sondern auch im Entwurf

Querschneidende Verfeinerung besteht aus zwei Schritten:

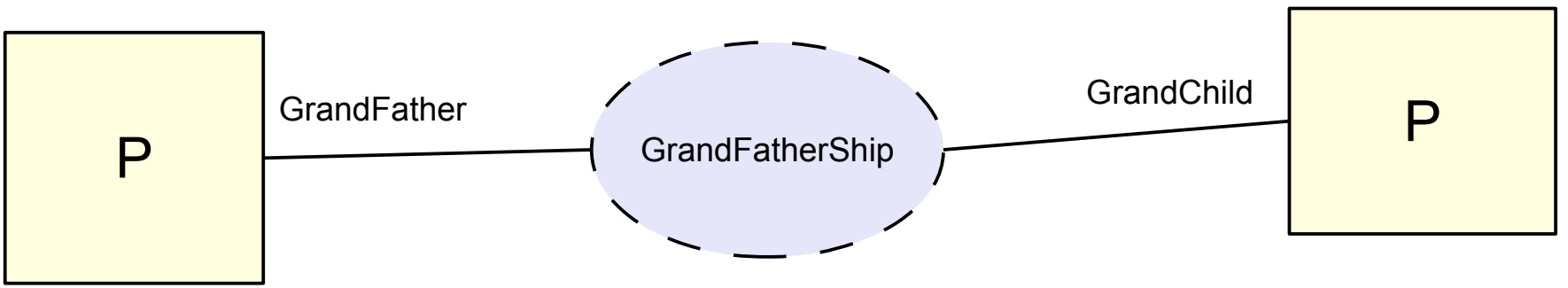
- Szenarienanalyse zur Erstellung von Kollaborationen
- Superimposition der Kollaborationen auf das bisherige Entwurfsmodell

Kollaborationen (collaborations, Teams) in UML

- Eine **Kollaboration** *(team, collaboration, Rollenmodell) ist ein* Schema für die Zusammenarbeit von Objekten. Sie definiert mehrere Rollen von Spielern (player) im Zusammenspiel
- Eine Kollaboration ist ein Netz mit offenen “plays-a”-Beziehungen
  - Die von natürlichen Typen gespielt werden müssen



- In UML stellt sich eine Kollaboration dar als generisches Sprachkonstrukt mit *Rollentyp-Parameter* P und Rollenname als Bezeichner für Tentakel:



Kollaborationen als Teams in ObjectTeams

- In fortgeschrittenen Programmiersprachen bilden Kollaborationen und ihre Rollen Sprachkonzepte.
- So auch in der Sprache ObjectTeams der TU Berlin (www.objectteams.org).
  - Hier heißt eine Kollaboration *Team* (Notation als Block, ähnlich zur Klasse)
  - *Rollenklassen bilden innere Klassen* des Teams

```
team Grandfathership {
  /* role class */
  class GrandFather {
    void caressing ();
  }
  /* role class */
  class GrandChild {
    void visiting ();
  }
}
```

```
team NewspaperReading {
  Readable buy();
  /* role class */
  class Reader {
    void breakfast () {
      Readable rd = buy();
      rd.read();
    }
  }
  /* role class */
  class Readable {
    void read();
  }
}
```

Kollaborationen mit inneren Klassen in Java

- In Java implementiert man eine Kollaboration immer als **Konnektorklasse** mit **Konnektorobjekt** und ggf. **inneren Rollenobjekten**
- **Vorteil:** alle Spieler und Rollen sind gekapselt; Code kann zusammenhängend wiederverwendet werden

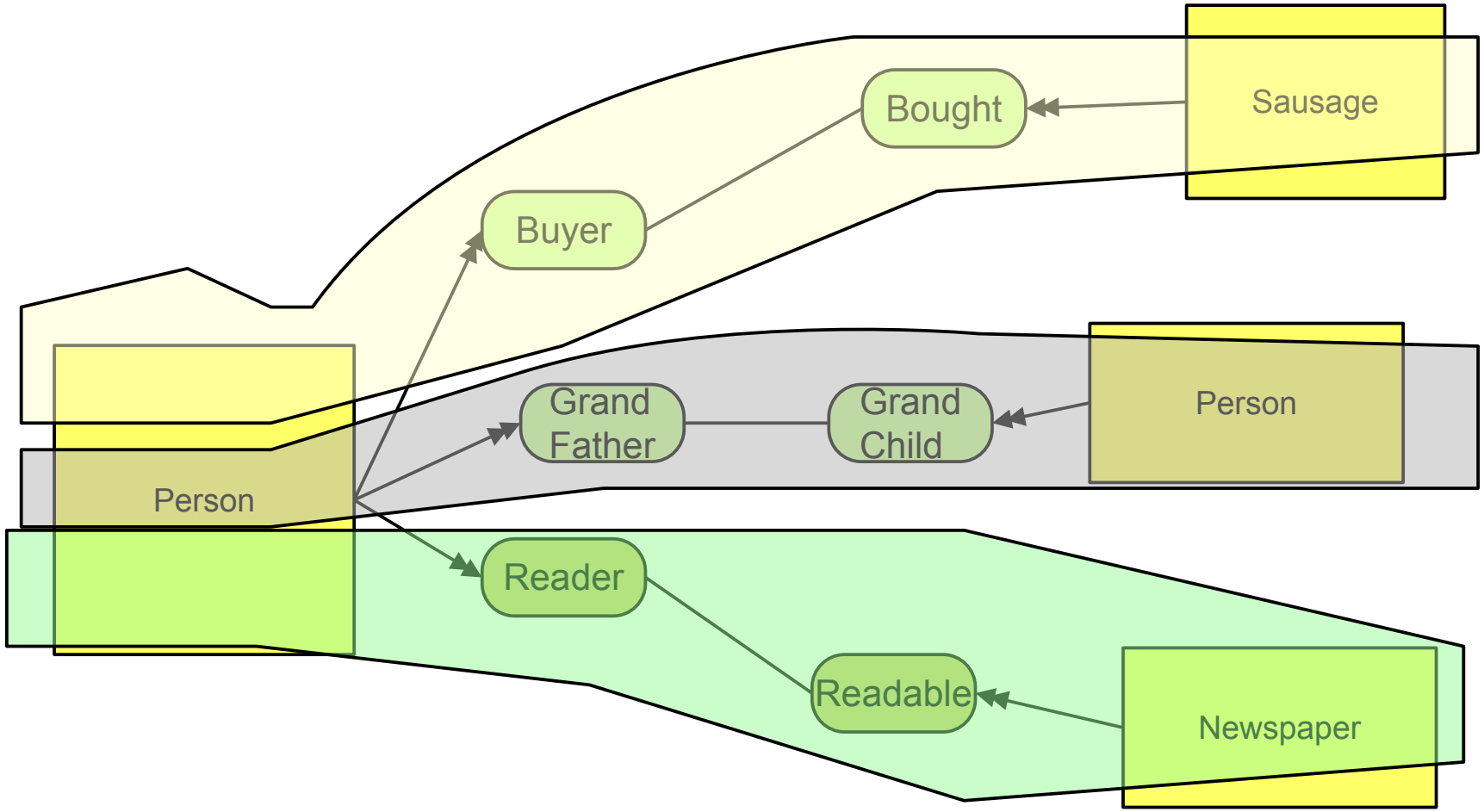
```
class Grandfathership {
  /* role as inner class */
  class GrandFather {
    void caressing ();   }
  /* role as inner class */
  class GrandChild {
    void visiting ();   }
  Person player_gf;
  GrandFather role_gf = new
GrandFather();

  Person player_gc;
  GrandChild role_gc = new
GrandChild();
}
```

```
void linkGrandfatherAndGrandChild
 (Person gf, gc) {
  player_gf = gf;
  player_gc = gc;
}
void unlinkGrandfatherAndGrandChild
(Person gf, gc) {
  player_gf = null;
  player_gc = null;
}
// delegation method
void caressing()
{ role_gf.caressing(); }
void visiting()
{ role_gc.visiting(); }
```

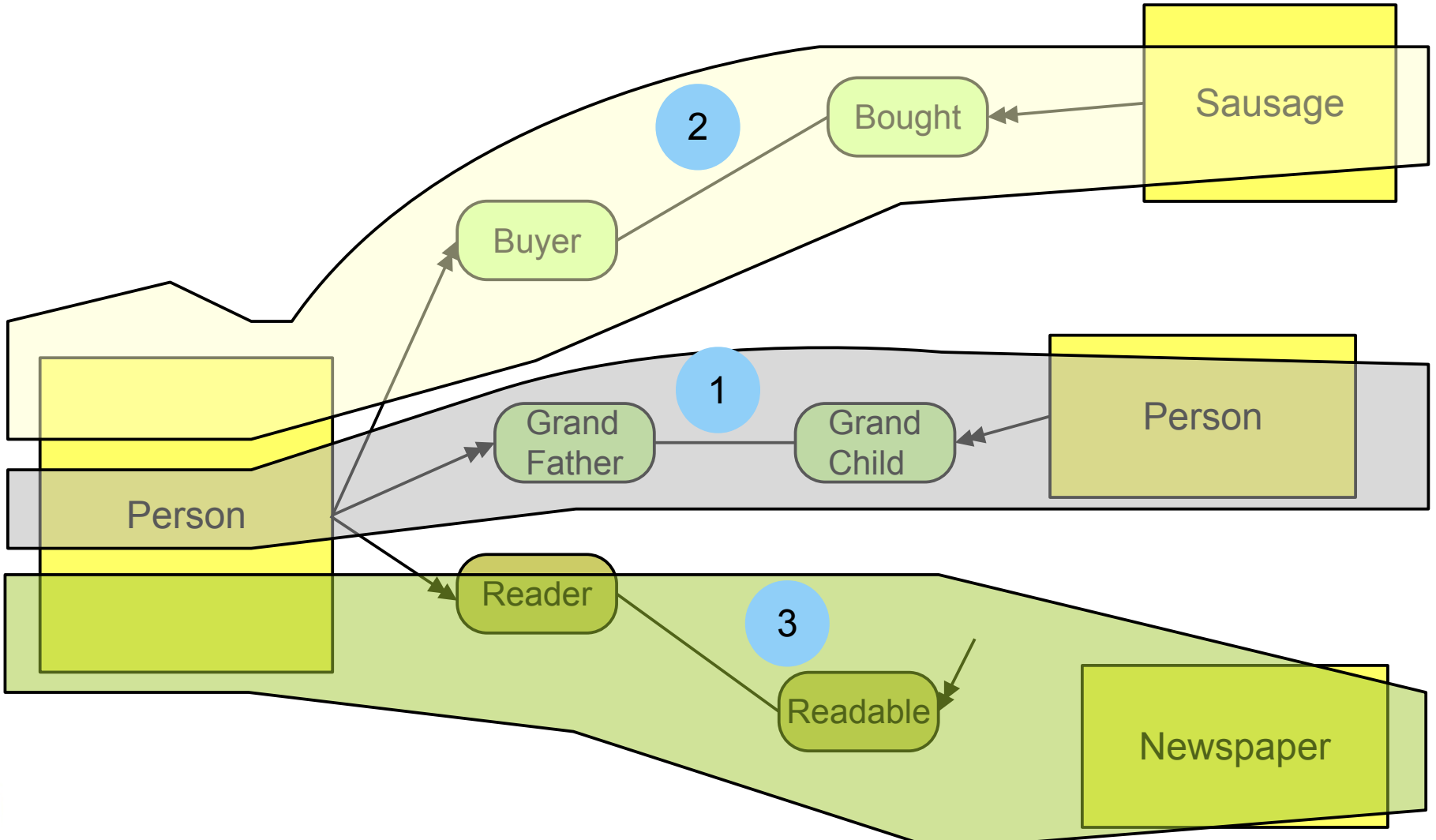
Kollaborationen als Schnitte durch die Anwendung

- Kollaborationen bilden *Schnitte (slices)* durch die Anwendung
- Mehrere Kollaborationen können auf die Anwendung superimponiert werden



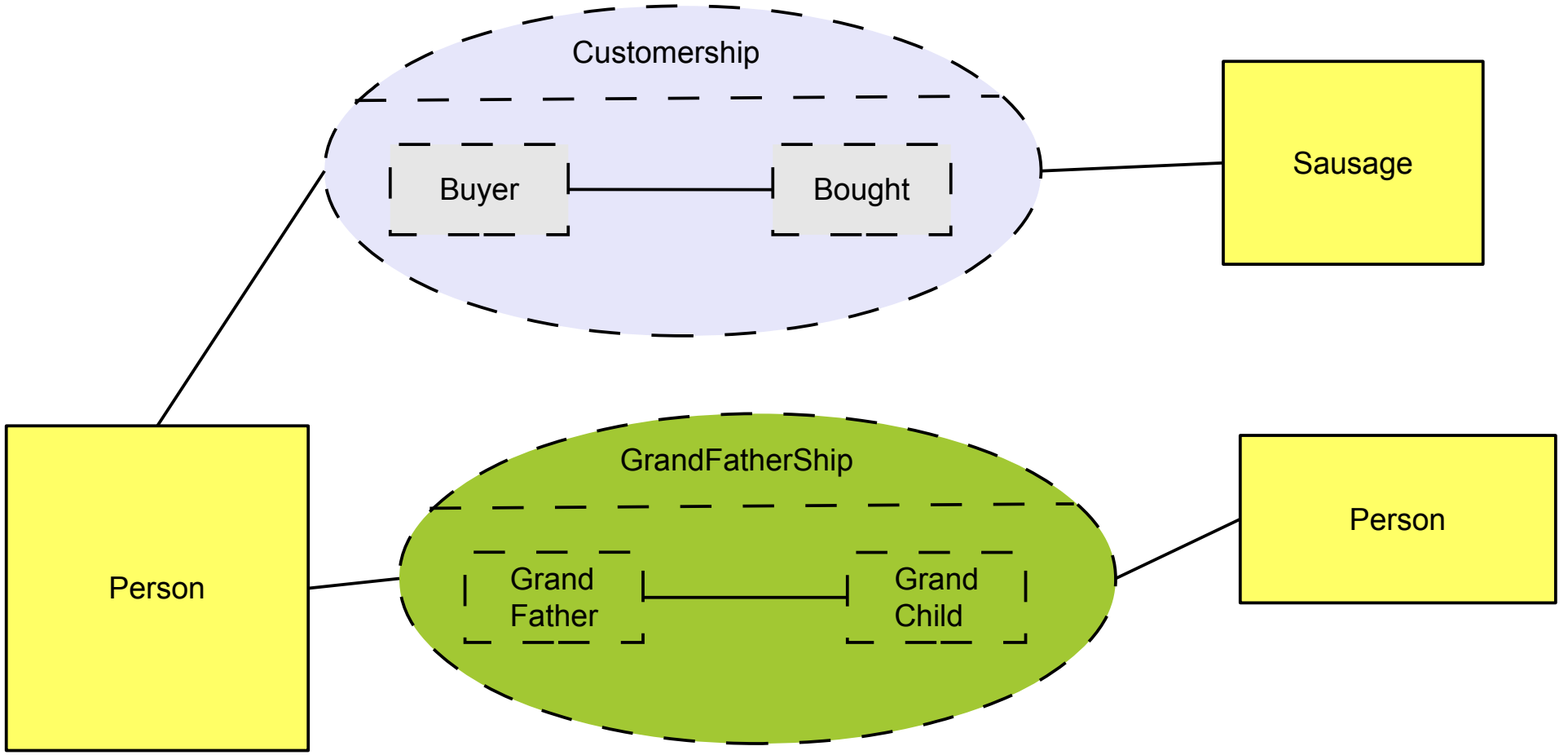
Querschneidende Erweiterung von Anwendungen mit Kollaborationen

- Analyse- und Entwurfsmodelle können sukzessive durch Kollaborationen **querschneidend** erweitert werden

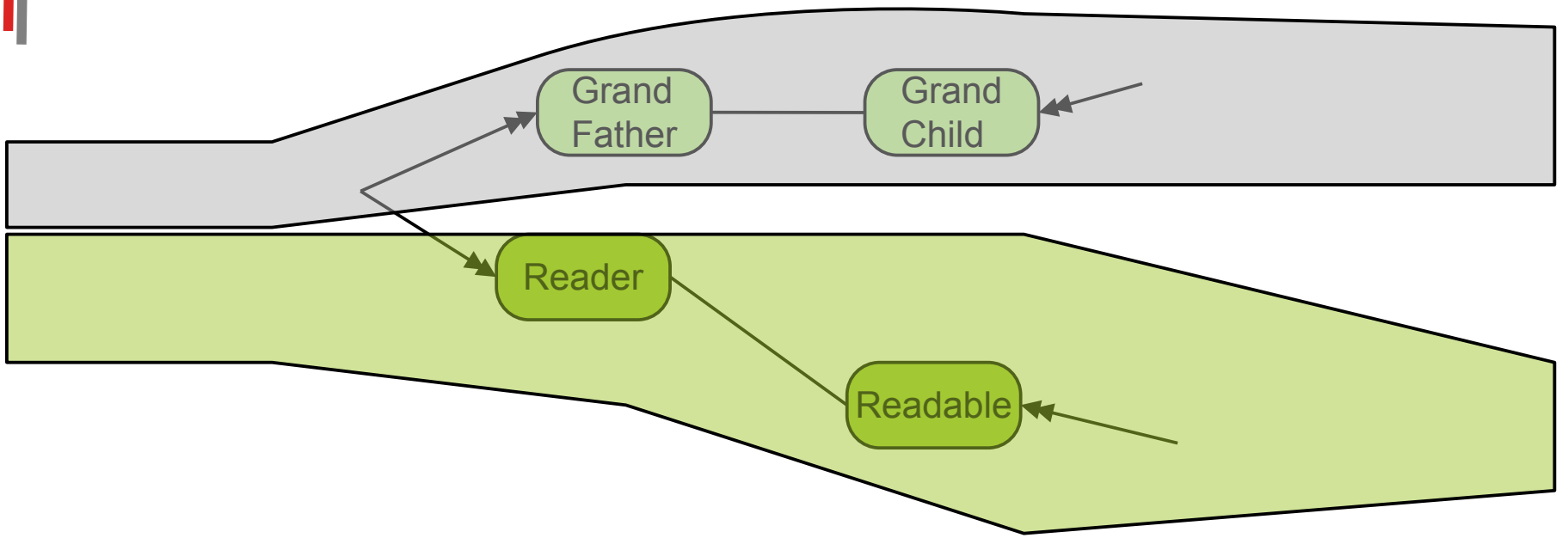


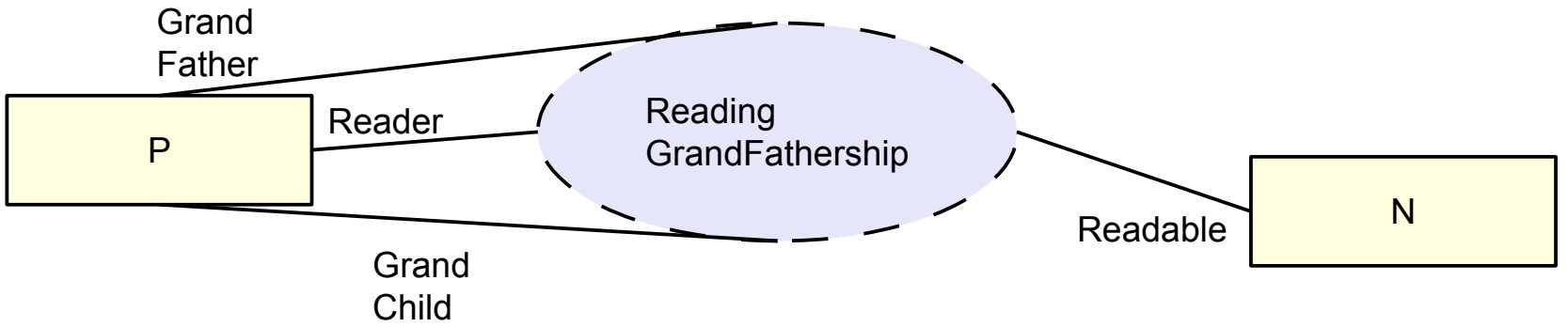
Verfeinerung mit Kollaboration-Superimposition in UML

- Das Überlagern von Kollaborationen und Konnektoren nennt man Superimposition **(Collaboration, connector superimposition)**
- Alternative Notation in UML: Kollaborationen *mit Abteilen*

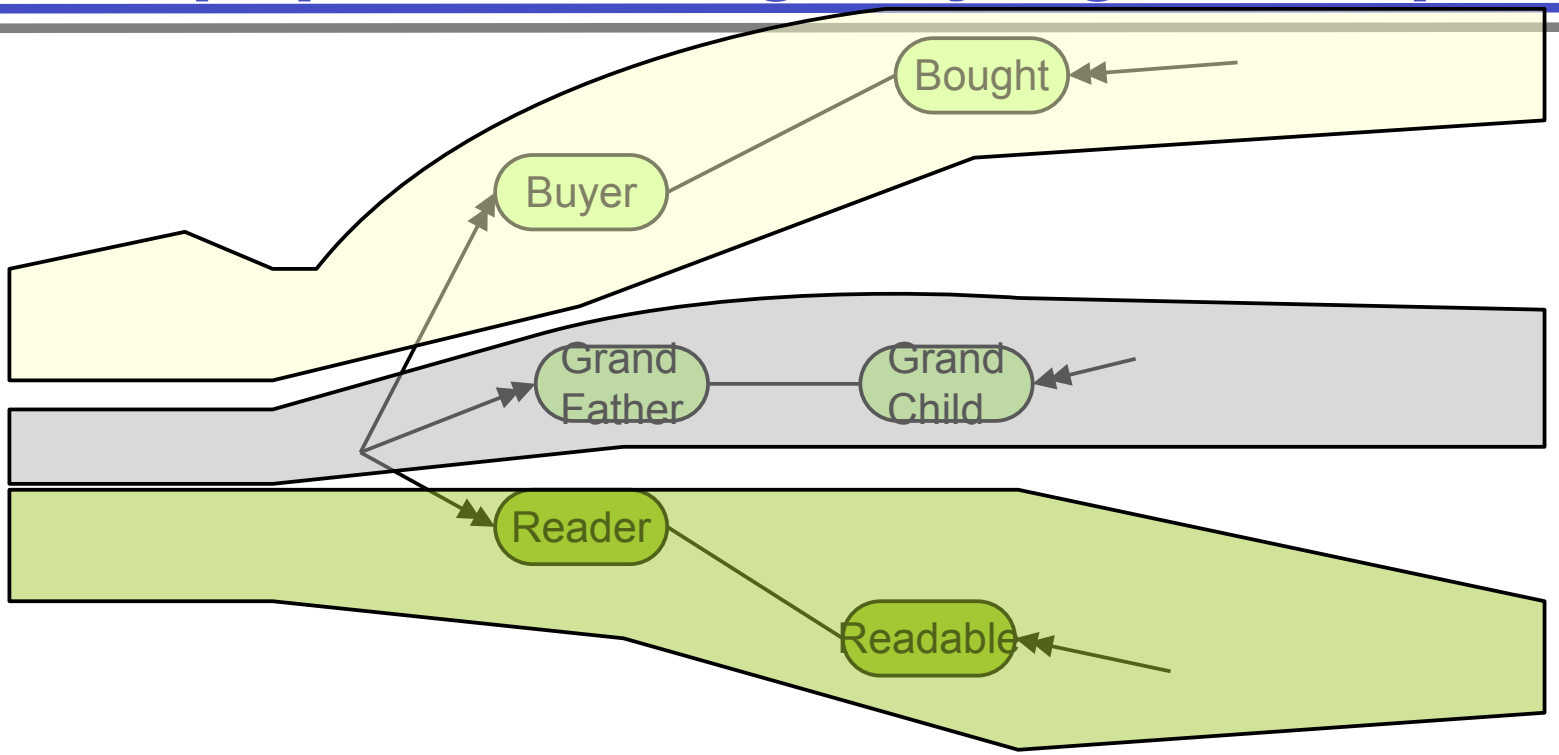


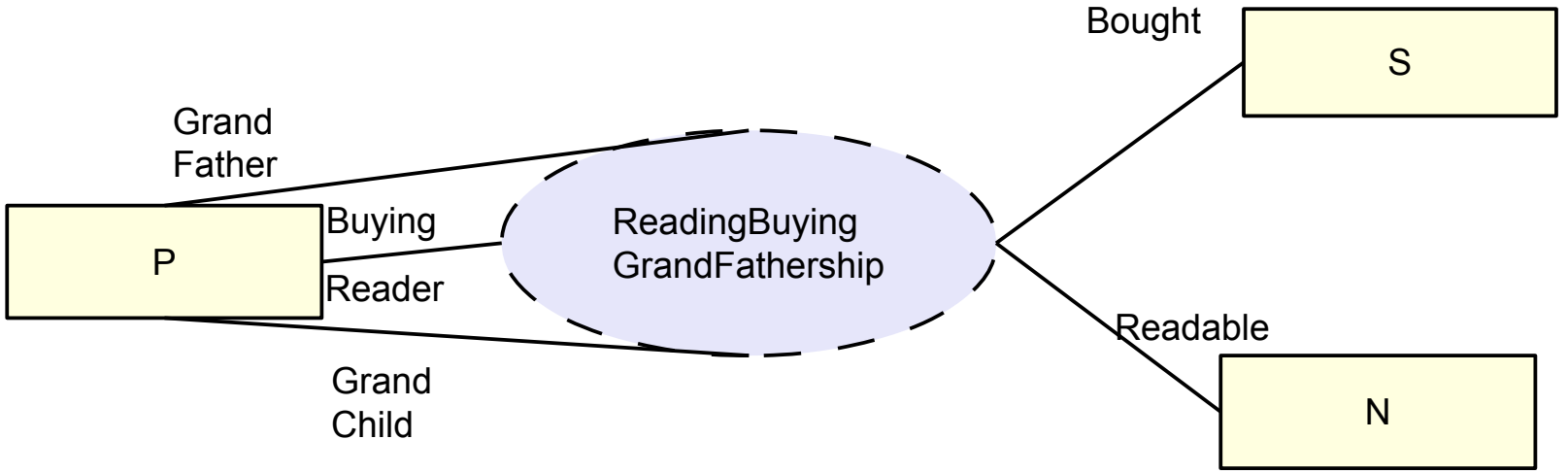
Verschmelzen von Kollaborationen: Newspaper-Reading GrandpaShip





Verschmelzen von Kollaborationen: Newspaper-Reading Buying GrandpaShip





Kollaborationsbasierte Verfeinerung

- Kollaborationsbasierte (querschneidende) Verfeinerung bedeutet, Schritt für Schritt neue Kollaborationen in das Analyse- und das Entwurfsmodell zu integrieren,
  - d.h. neue Kollaborationen zu superimponieren
- In einer Programmiersprache wie ObjectTeams kann man das direkt umsetzen, in dem man zu einem Kern-Programm neue Teams hinzufügt
  - In Java ist es schwieriger

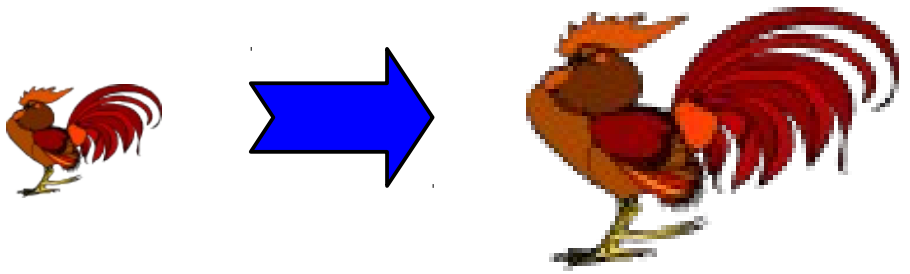
36.4.2 Verfeinerungsbeispiel für Objektanreicherung

.. Verfeinerung durch Integration von Unterobjekten..

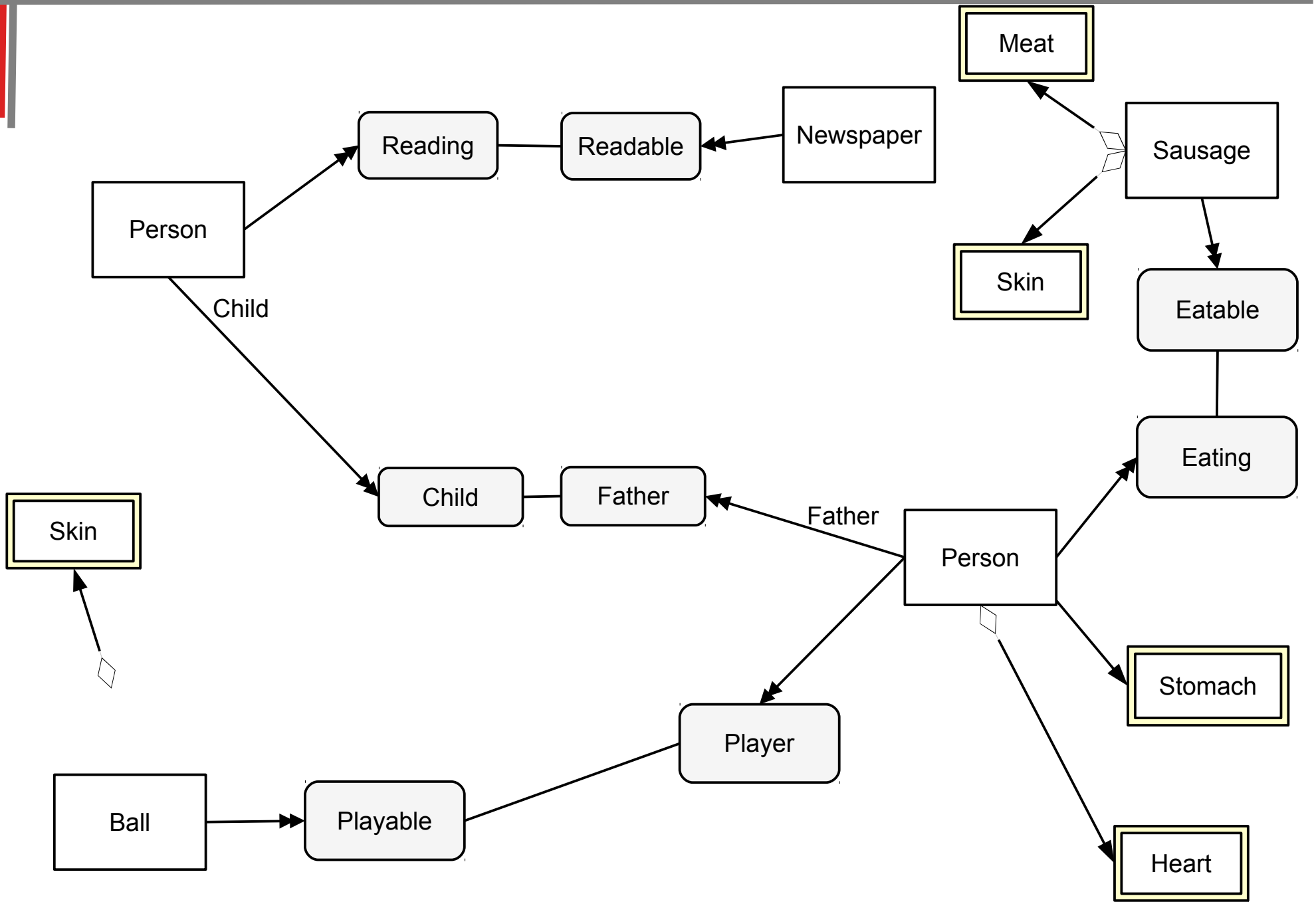
Teile und Rollen

Objektanreicherung (Wdh.)

- **Objektanreicherung (object fattening)** ist ein Verfeinerungsprozess zur *Entwurfszeit*, der an ein Kernobjekt aus dem Domänenmodell Unterobjekte anlagert (Domänenobjekt-Verfeinerung durch Integration), die
  - Teile ergänzen (Teile-Verfeinerung)
  - Rollen und Kollaborationen ergänzen (Kollaboration-Verfeinerung),
  - Rollen und Kollaborationen ergänzen (Kollaboration-Verfeinerung), die Beziehungen klären zu
    - Plattformen (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
- Ziel: Entwurfsobjekte, Implementierungsobjekte

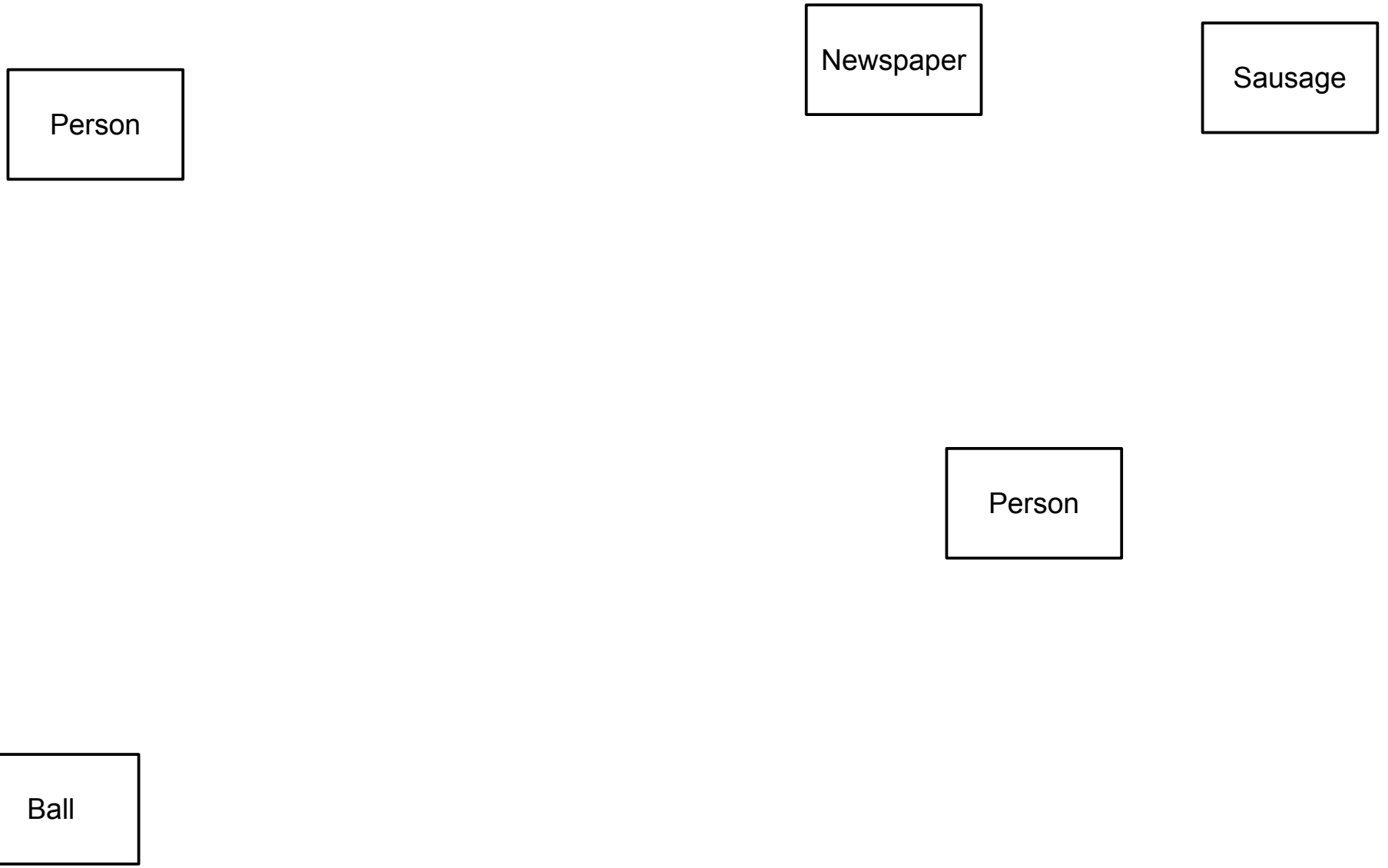


Personen-Analysemodell mit Rollenobjekten und Teilen – Wie komme ich bloß dahin?



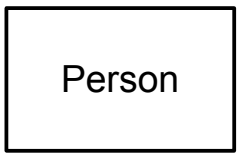
Mit Verfeinerung durch Integration von Unterobjekten (Objektanreicherung, Object Fattening)

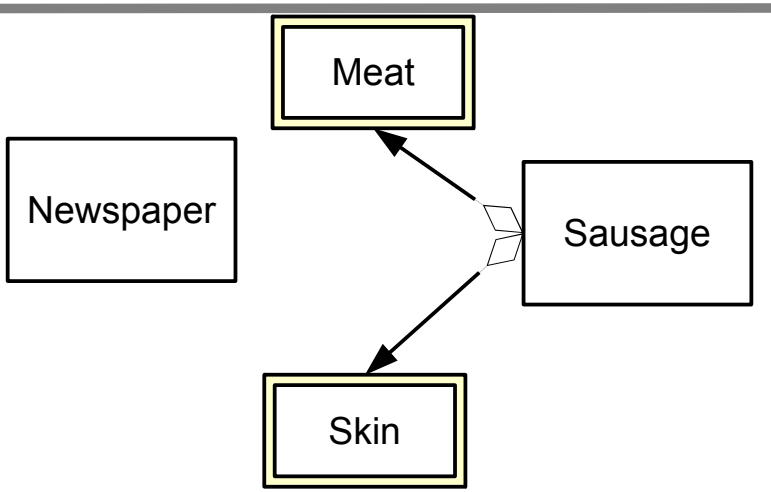
- Rohzustand: Identifikation der natürlichen Typen (in dem Domänenmodell)

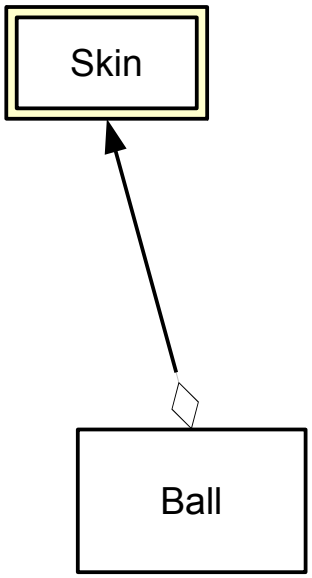


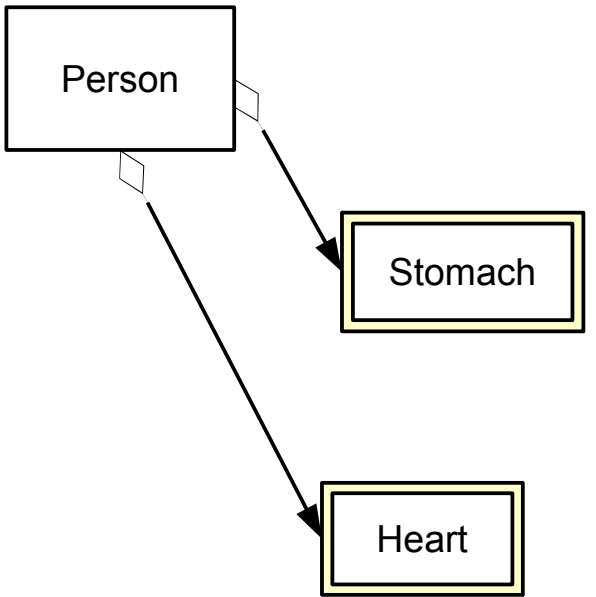
Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 1: Teile-Verfeinerung



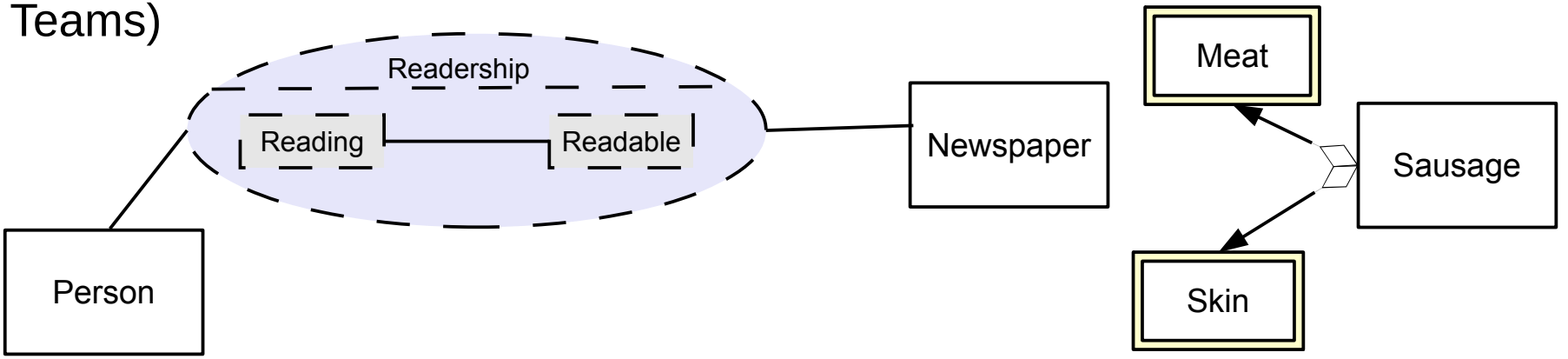


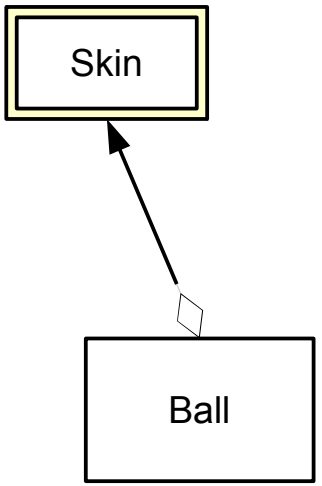


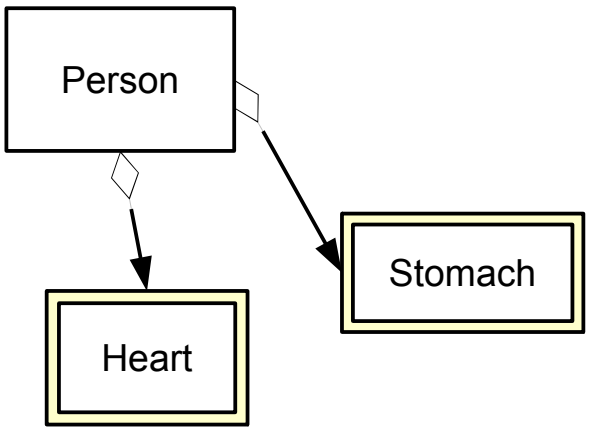


Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 2: Schrittweise Erweiterung durch Kollaborationen (Konnektoren, Teams)

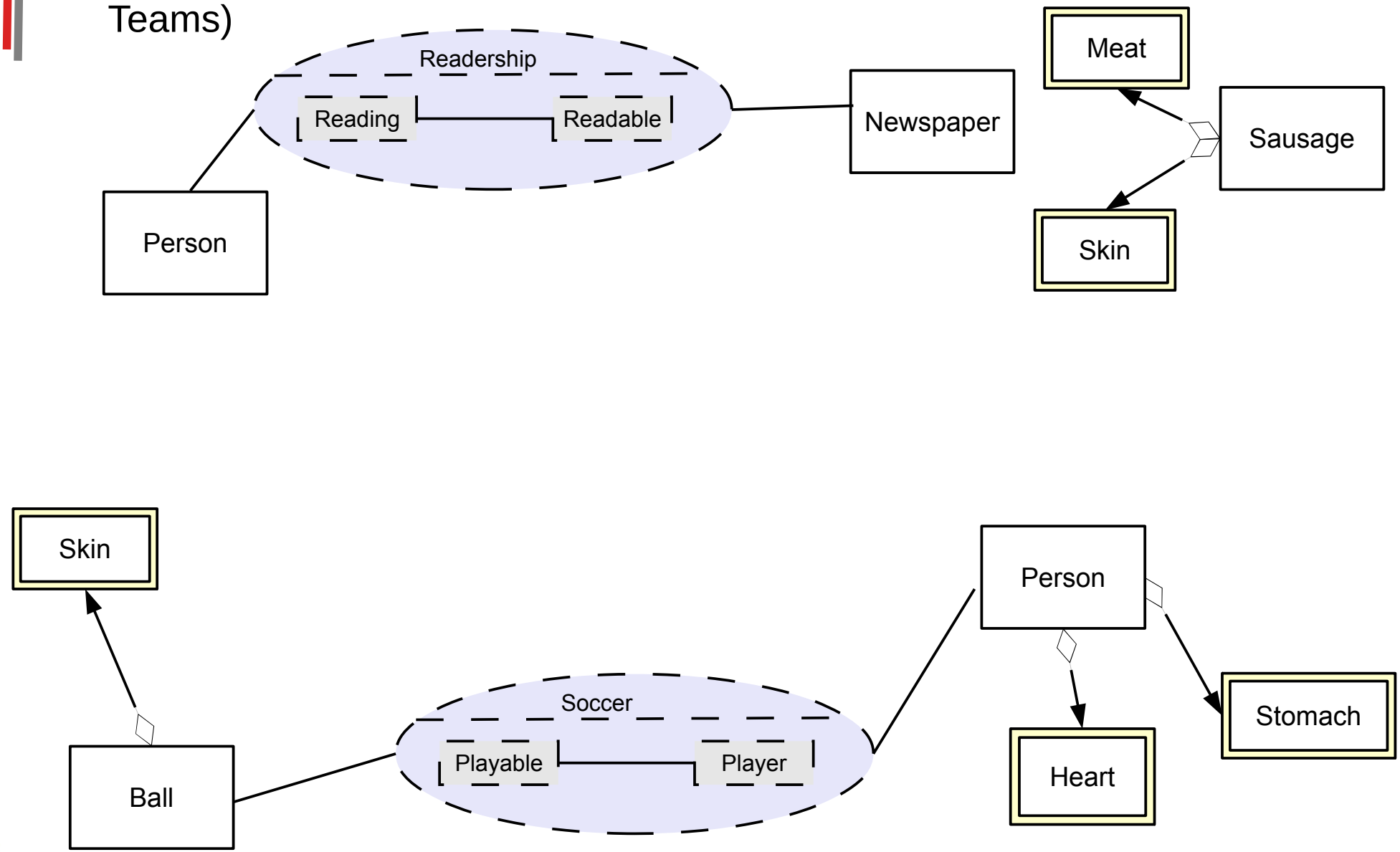






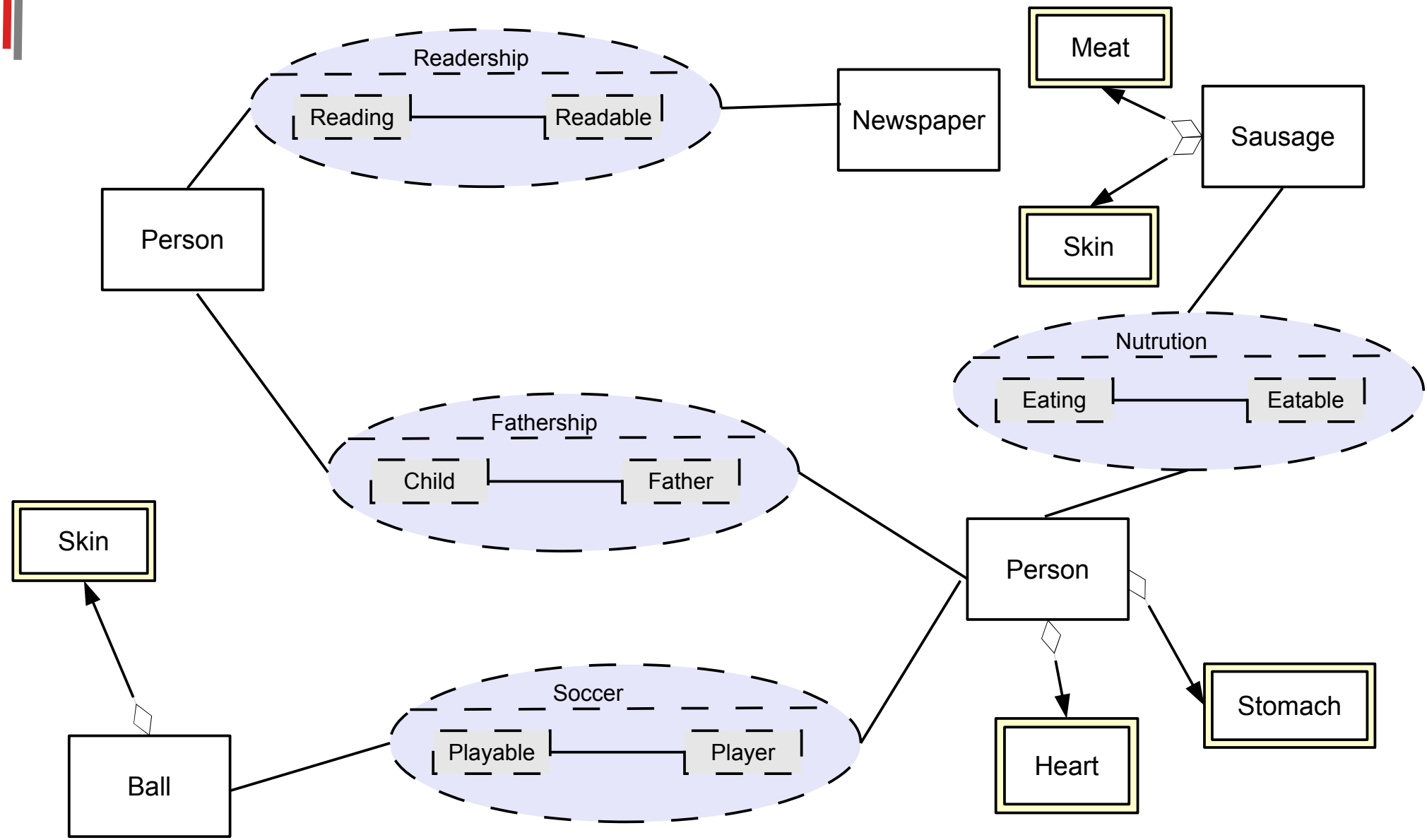
Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 2: Schrittweise Erweiterung durch Kollaborationen (Konnektoren, Teams)



Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 2: final: alle Kollaborationen



Objektanreicherung – Weitere Schritte im Entwurf

- Teile- und Rollenverfeinerung laufen noch im Analysemodell ab
  - Kollaboration-Verfeinerung wird durch Szenarienanalyse angeregt
  - Facetten- und Phasen-Verfeinerung kommt optional hinzu
- In ObjectTeams entspricht dies dem Schreiben neuer “Teams”
  - Teile können in Rollen von Teams eingelagert werden
  - In ObjectTeams kann man querschneidende Objektanreicherung ganz einfach realisieren; Superimposition geht einfach
  - In Java können Kollaborationsklassen mit inneren Rollenklassen verwendet werden, aber die Superimposition schwieriger
- Bei Entwurfsobjekten kommt hinzu:
  - Finden von **Plattform-Kollaborationen,** fundierte Unterobjekte, die das spezifische Verhalten bezüglich eines Plattformobjektes kapseln
- Beim Implementierungsmodell kommt hinzu:
  - Realisierung der Kollaborationen und der Integrationsrelation

Was haben wir gelernt?

- Ein Anwendungsfall kann durch Szenarienanalyse verfeinert werden
  - Aus dem Anwendungsfall kann eine Kollaboration abgeleitet werden
  - Sowie ein Interaktionsdiagramm, das das Protokoll zwischen den Rollen der Kollaboration beschreibt
  - Oder ein Aktionsdiagramm, das ebenfalls das Protokoll zwischen den Rollen der Kollaboration beschreibt
- Szenarienanalyse verfeinert querschneidend, i.G. zu punktweiser Verfeinerung

The End

Anhang 36.A: Nebenbemerkung

- Integration von Unterobjekten in Kernobjekte kann *zu verschiedenen Zeiten* erfolgen
  - Zur Entwurfszeit
  - Zur Bindezeit
  - Zur Allokationszeit eines Objekts
  - Zur Laufzeit
  - Zur Zeit der Software-Pflege und -Migration

Anhang: Rollen in der Literatur

- Rollenorientiertes Datenmodell (Bachmann 77)
- Entity-Relationship-Modell (ER model, Chen 76): Hier bilden die Enden einer Assoziation eine Rolle. Vorbild für UML-Klassendiagramme
  - Kurs “Softwarewerkzeuge (SEW)” im WS
- Entwurfsmuster (Riehle 98)
  - Kurs “Design patterns and frameworks (DPF)” im WS
- Produktlinien-Engineering (Smaragdakis,Batory 02)
- Kollaborationen in Architektursprachen (Garlan, Shaw 95)
  - Kurs “Component-based Software Engineering (CBSE)” im SS
- Objektorientierte Modelierung mit der OORAM Methode (Reenskaug 95)

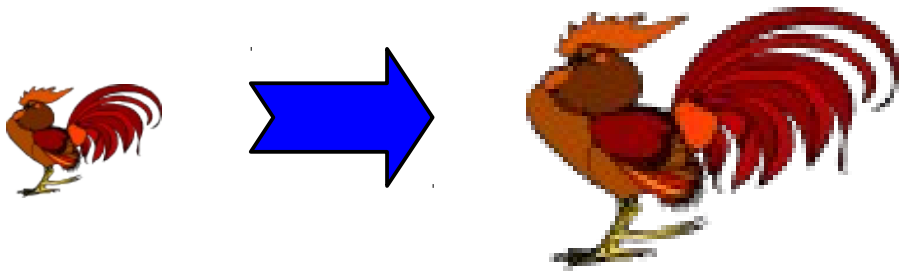
Anhang 36.B Volles Verfeinerungsbeispiel für Objektanreicherung mit allen Arten von Unterobjekten

.. Verfeinerung durch Integration von Unterobjekten..

(optional)

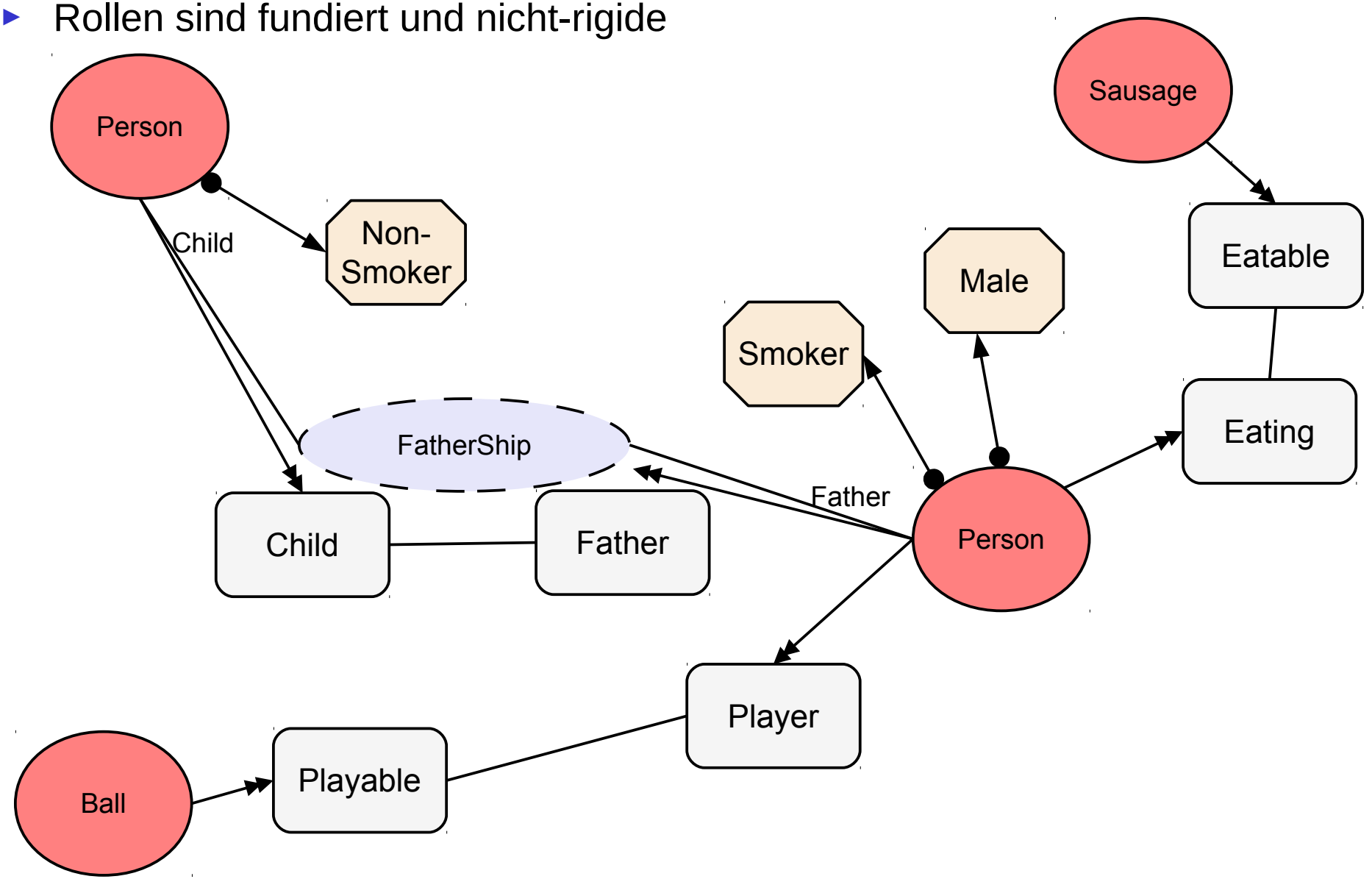
Erweitertes Konzept der Objektanreicherung

- Weitere Unterobjekte können integriert werden
  - *Phasen ergänzen (Phasen-Verfeinerung)*
  - *Facetten ergänzen (Facetten-Verfeinerung)*
  - Teile ergänzen (Teile-Verfeinerung)
  - Rollen ergänzen (Kollaboration-Verfeinerung),
  - Rollen ergänzen (Kollaboration-Verfeinerung), die Beziehungen klären zu
    - Plattformen (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
- Ziel: Entwurfsobjekte, Implementierungsobjekte

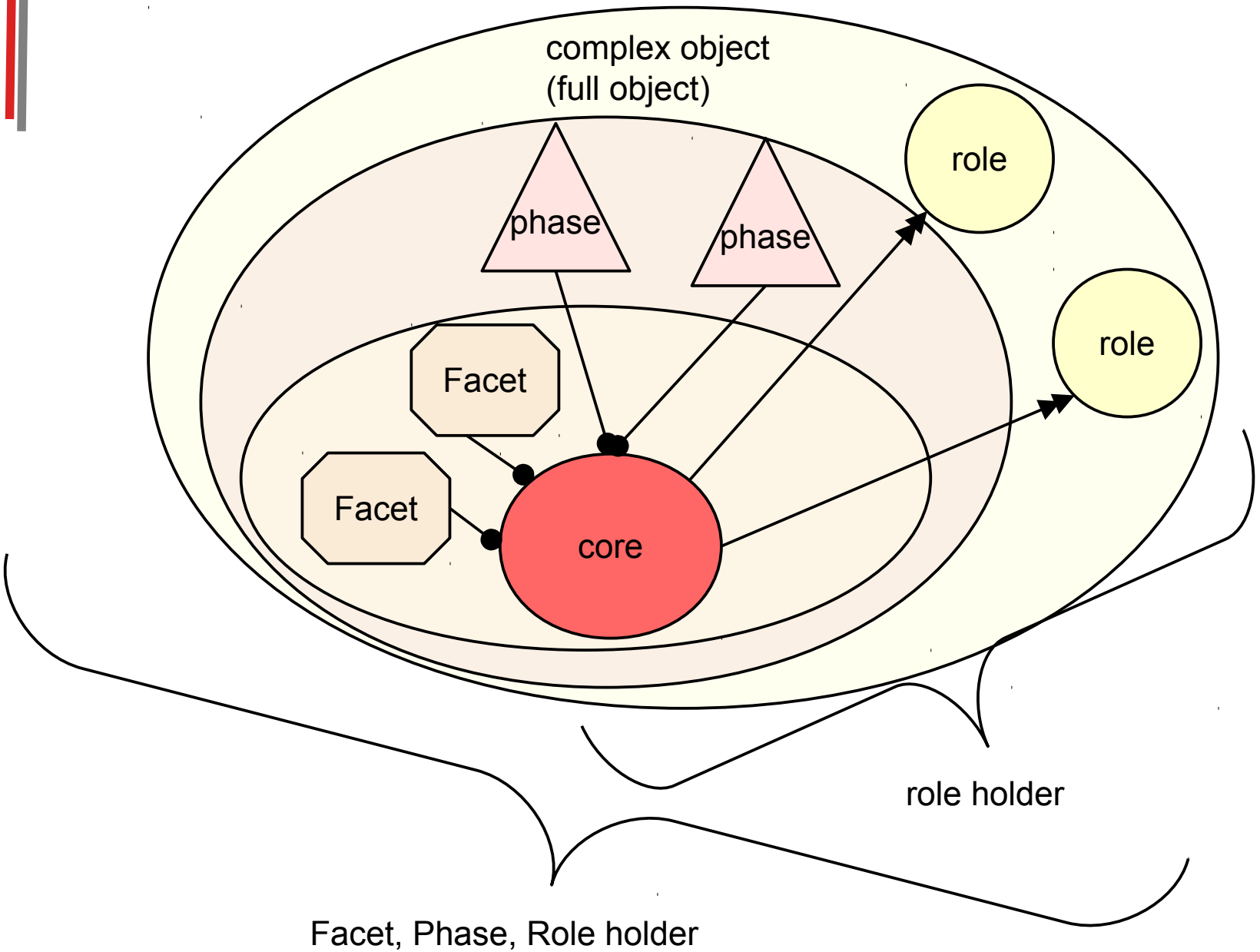


Facetten im Vergleich zu Rollen (Wdh.)

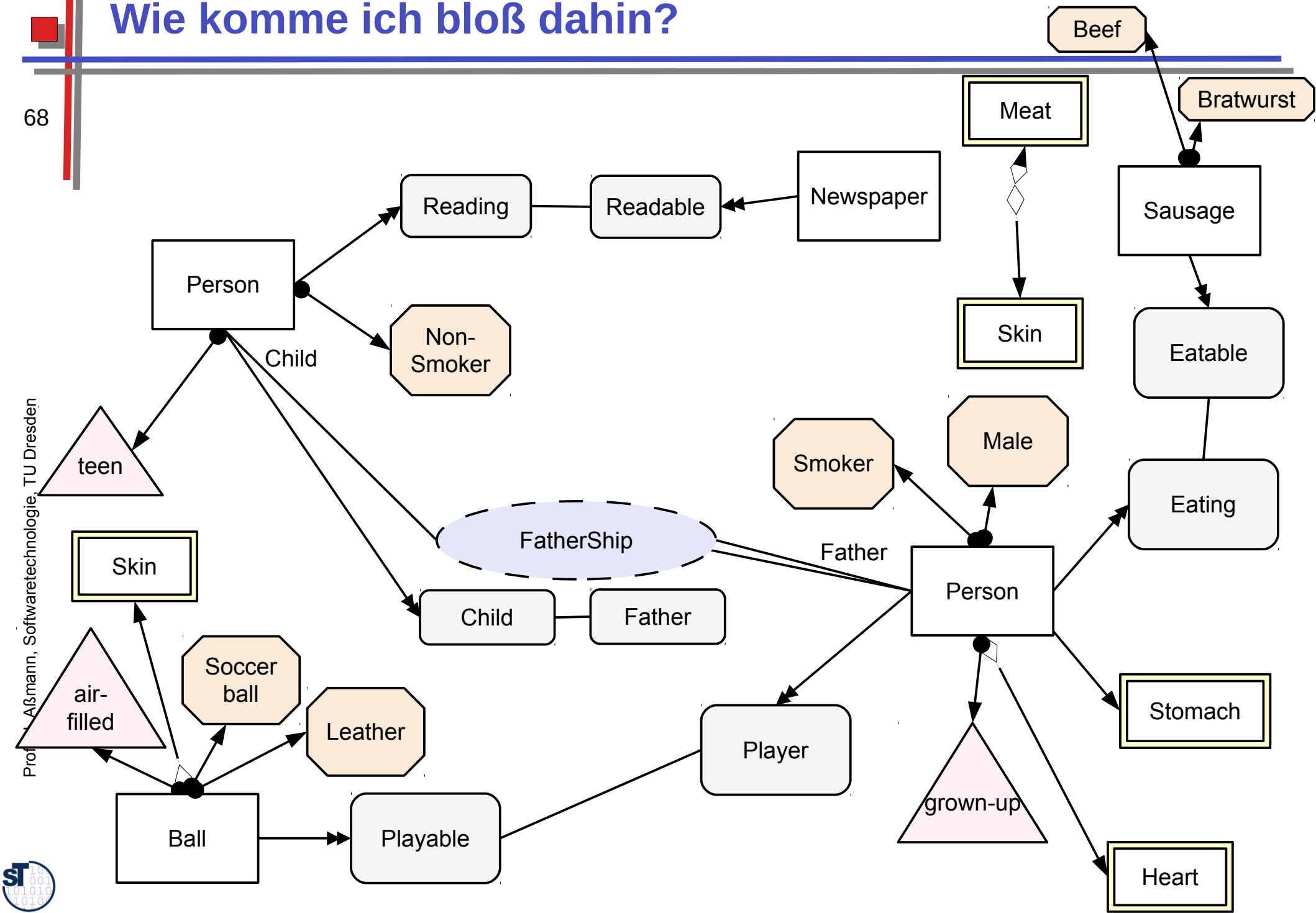
- Facetten sind nicht-fundiert und rigide (natürlich)
- Rollen sind fundiert und nicht-rigide



Komplexe Objekte

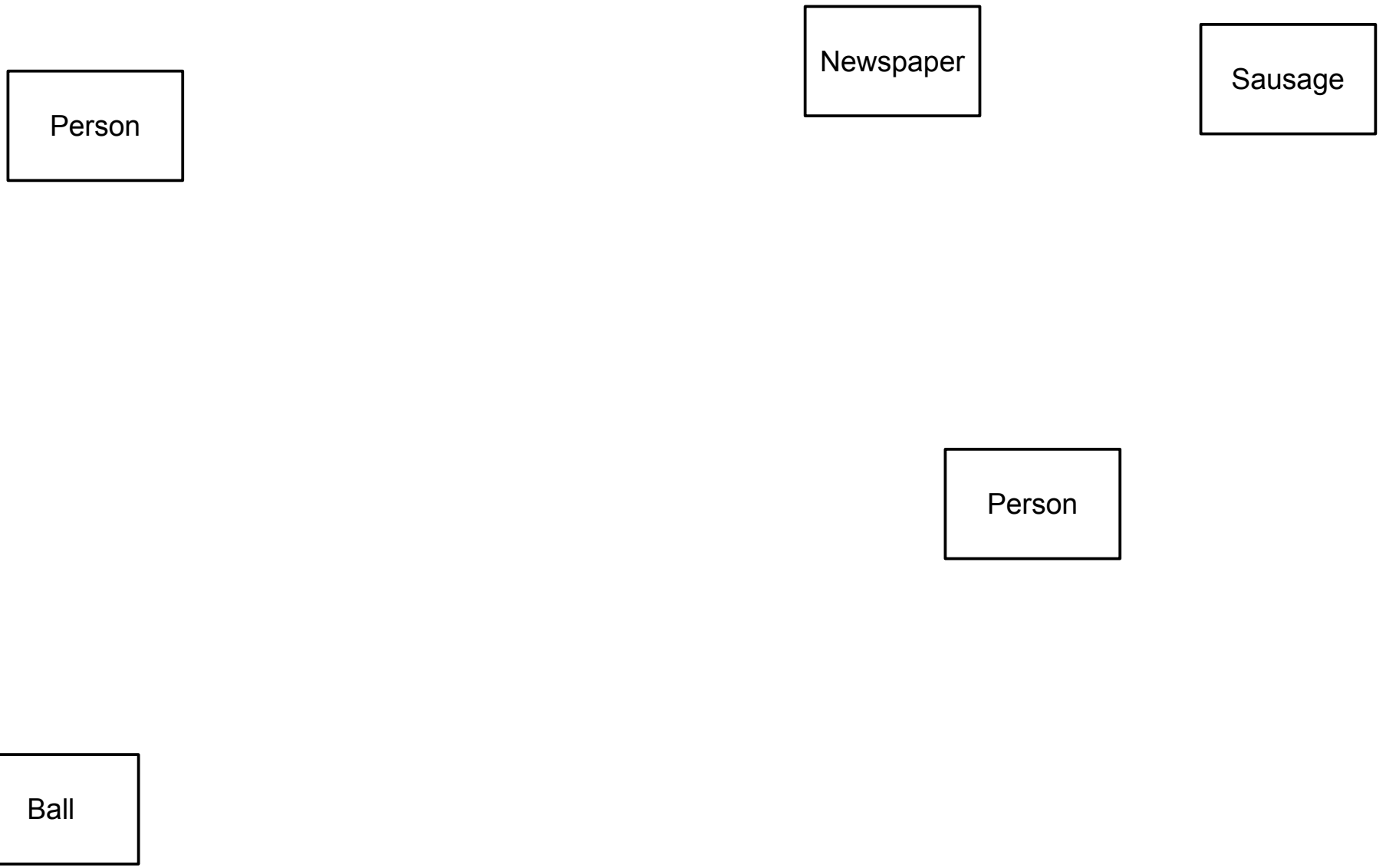


Personen-Analysemodell – Wie komme ich bloß dahin?



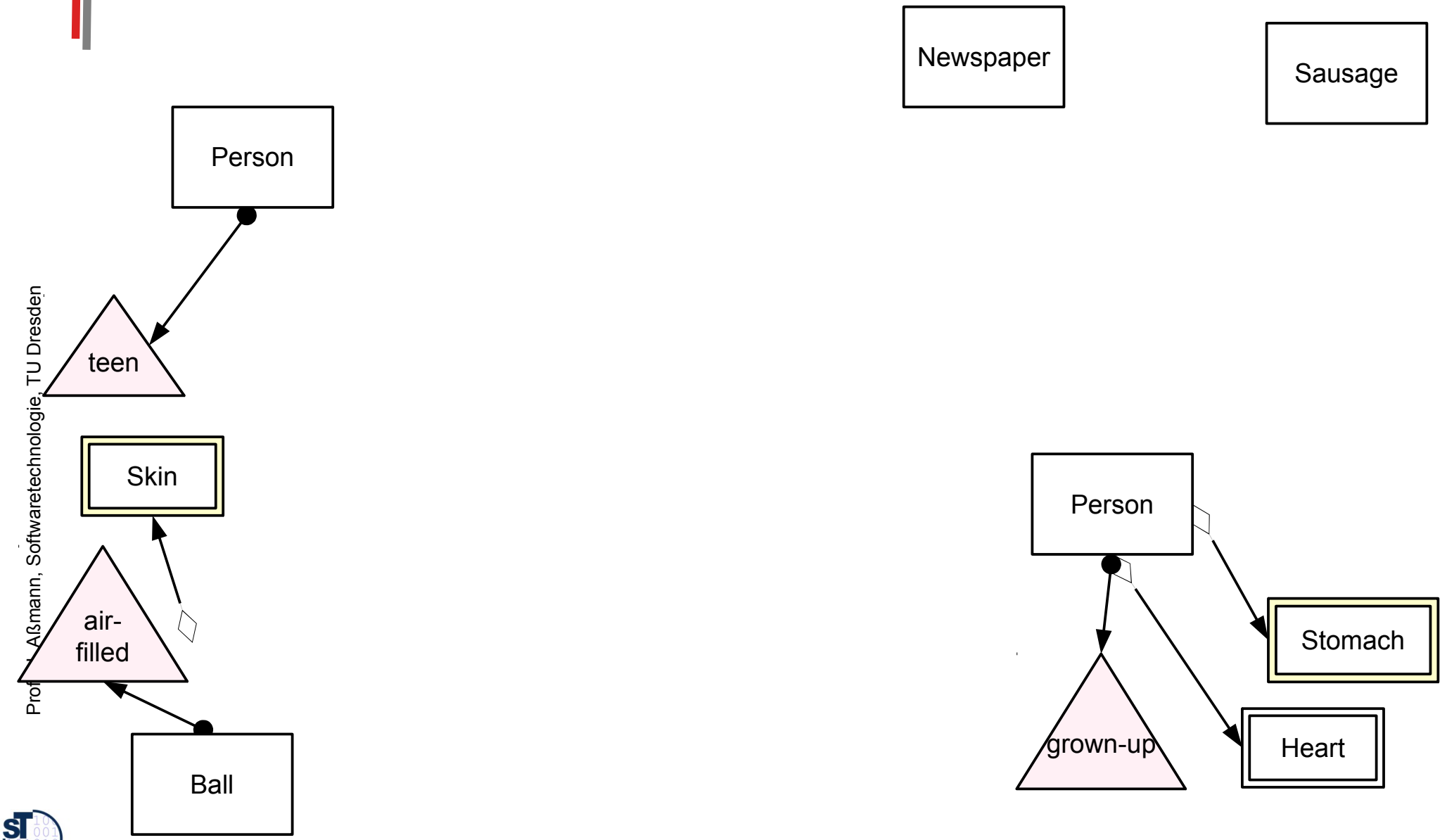
Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Rohzustand: Identifikation der natürlichen Typen



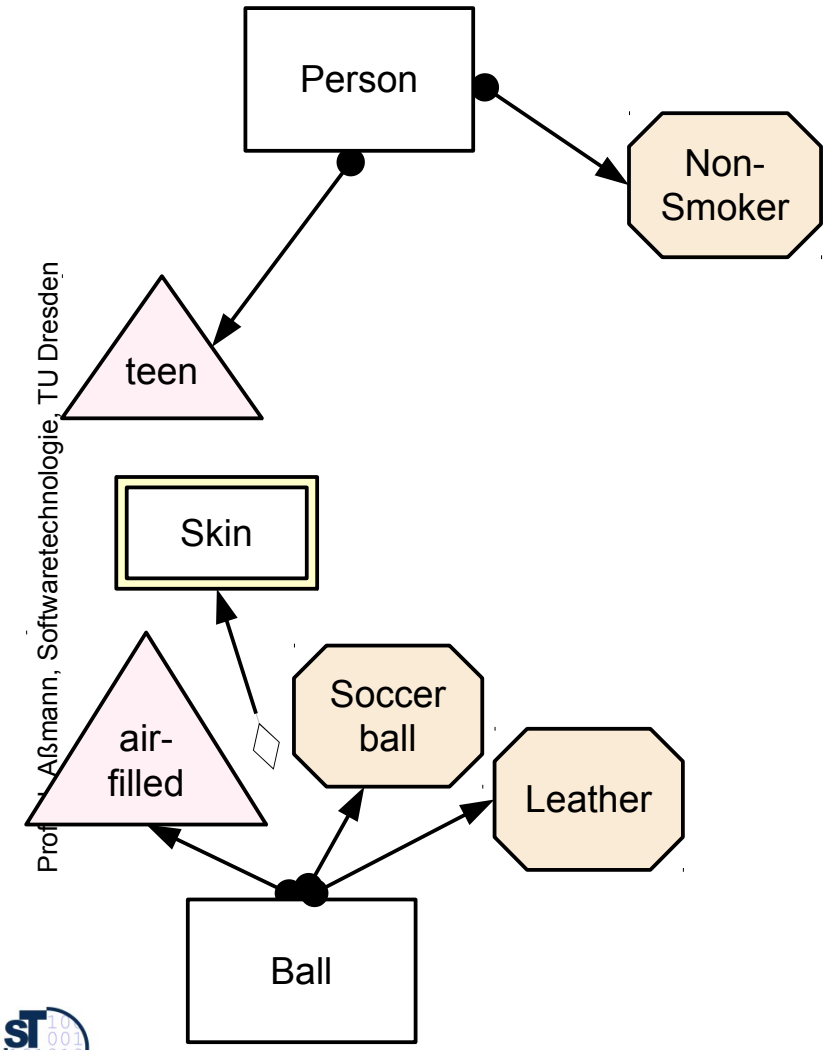
Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

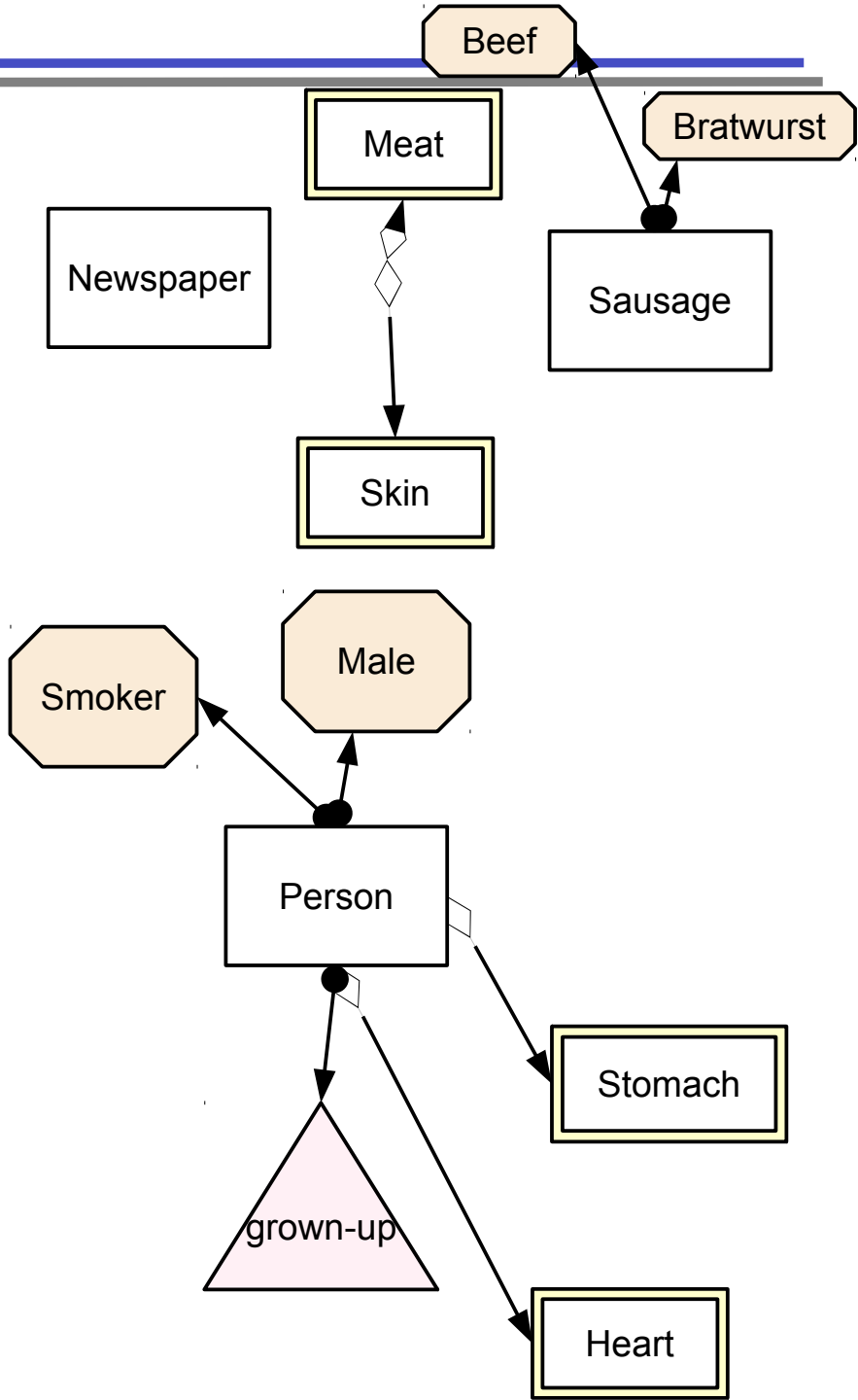
- Schritt 1: Teile-Verfeinerung, Phasen-Verfeinerung



Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 2: Facetten-Verfeinerung





Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 3: Erweiterung durch Kollaborationen