36 Szenarienanalyse mit Anwendungsfalldiagrammen, Kollaborationen und Teams (Querschneidende dyn. Modellierung)

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 21-0.1, 01.07.21

1) Anwendungsfalldiagramme
2) Szenarienanalyse mit Interaktionsdiagrammen
3) Szenarienanalyse mit Kommunikationsdiagrammen
4) Szenarienanalyse mit Aktionsdiagrammen
5) Wozu braucht man querschneidende Verfeinerung?

Obligatorische Literatur

- ST für Einsteiner, Kap. Anwendungsfalldiagramme, Sequenzdiagramme, Aktivitätsdiagramme, Zustandsdiagramme
- Zuser, Kap. 7-9, insbes. 7.3+7.5
- Störrle Kap 9, Kap 12, Störrle 5.3, 5.4

Weitere Literatur

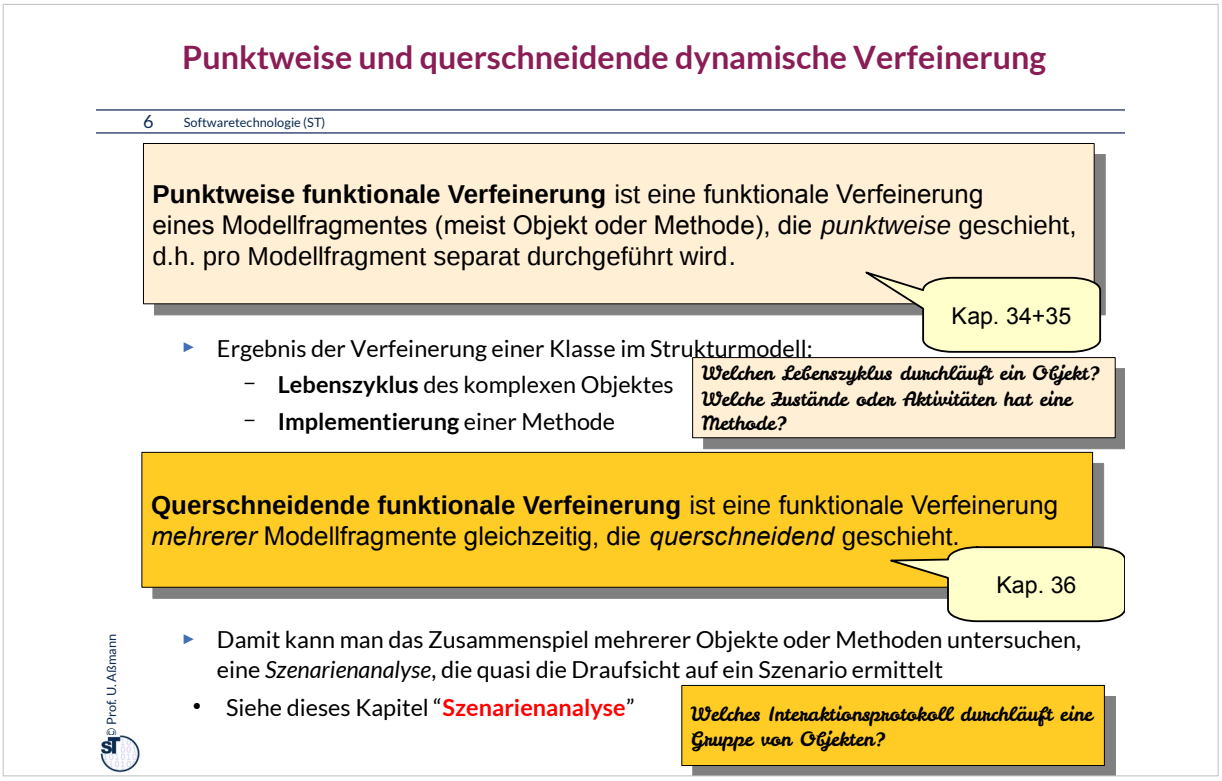
- Die Beispiele zur Servicestation finden sich in
  - S. Pfleeger, Software Engineering. Theory and Practice. Prentice-Hall.
- L. Maciaszek. Requirements Analysis and System Design – Developing Information Systems with UML. Addison-Wesley.
- Giancarlo W. Guizzardi. Ontological foundations for structure 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. Strukturelle Modellierung mit CRC-Karten
2. Strukturelle metamodellgetriebene Modellierung mit UML für das Domänenmodell
   1. Strukturelle metamodellgetriebene Modellierung
      1. Modellierung von komplexen Objekten
   2. Strukturelle Modellierung für Kontextmodell und Top-Level-Architektur
3. Analyse von funktionalen Anforderungen (Verhaltensmodell)
   1. Funktionale Verfeinerung: Dynamische Modellierung von Lebenszyklen mit Aktionsdiagrammen
   2. Funktionale querschneidende Verfeinerung: Szenarienanalyse mit Anwendungsfällen, Kollaborationen und Interaktionsdiagrammen (Kap 34, **36**)
4. Beispiel Fallstudie EU-Rent

Warum braucht man Modellierung?

- Kleinere Projekte kommen oft ohne Modelle aus. Aber...
- Große Produkte und Produktlinien
  - https://hbr.org/1994/11/the-logic-of-product-line-extensions
- Wiederverwendung im Multiprojektmanagement
  - http://www.biglever.com/images/solution/3D.jpg
- Tests und Softwarequalität zu niedrigen Kosten
- Modelle erleichtern das Verständnis des Kunden und die Kommunikation mit ihm



Ein Beispiel:

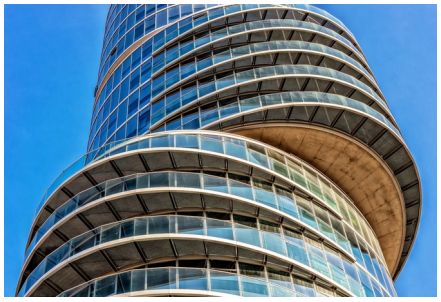
Schreibt man an einem Dokument, bedeutet punktweise Verfeinerung, dass man an einer Stelle neuen Text hinzufügt.

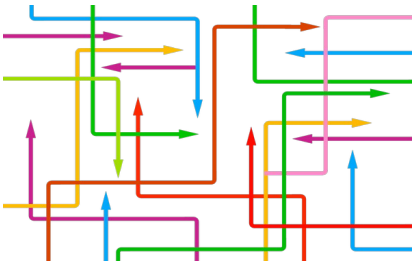
Querschneidende Verfeinerung dagegen bedeutet, dass man an vielen Stellen eine ähnliche Ersetzung oder Erweiterung vornimmt, z.B. die Ersetzung eines Namens oder einer Definition.

Wozu braucht man Szenarioanalyse?

- Objekte stehen in vielen Beziehungen in ihren Netzen
- Jede Szenarioanalyse analysiert *einen Pfad* durch das Objektnetz, bzw. *ein Subnetz* des Gesamtnetzes
- Szenarien werden vom Kunden vorgegeben und sind seine “Scheiben” durch die Anwendung

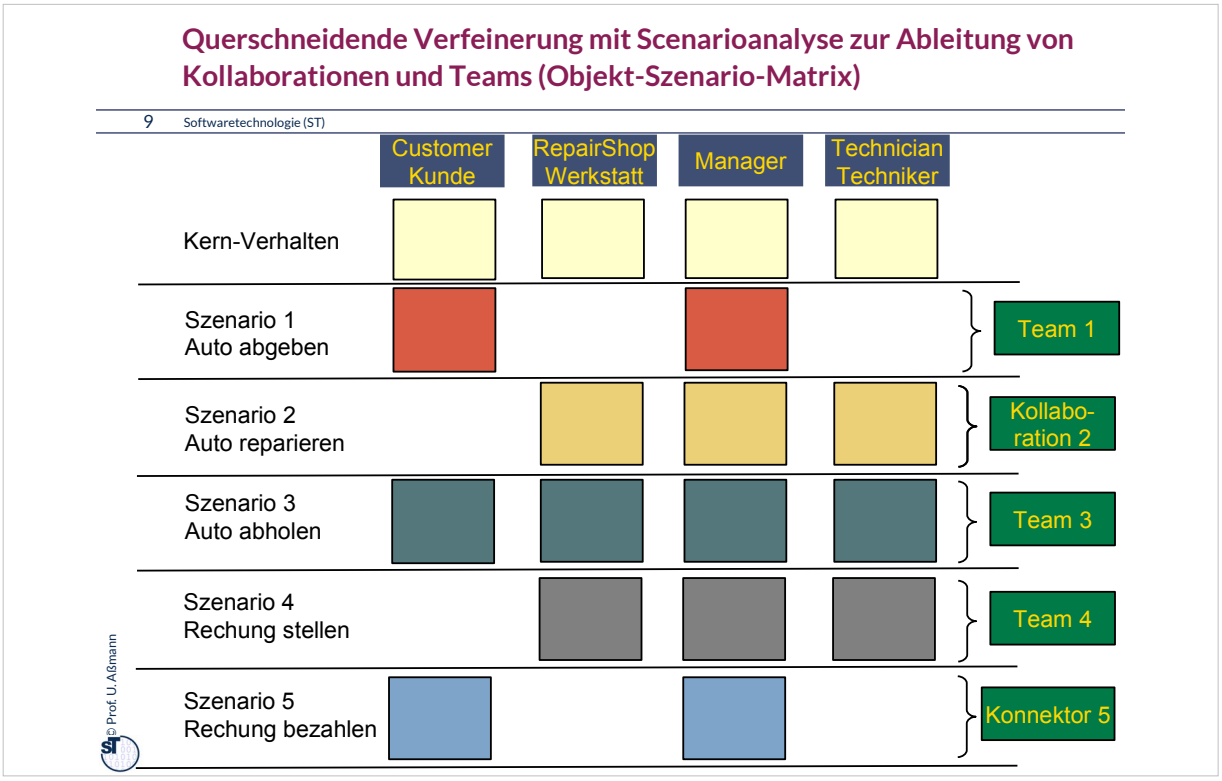
- Software ohne Szenarienanalse muss schief entwickelt werden!





www.pixabay.com

Kollaborationen können in UML, aber nicht in Java beschrieben werden; Teams und Konnektoren schon, denn sie bilden Objekte bzw. Klassen.



Eine Objekt-Szenario-Matrix listet alle Szenarien auf, die die die Objekte eines Programms involviert sind.

Man stellt die OSM zusammen aus allen Anwendungfalldiagrammen

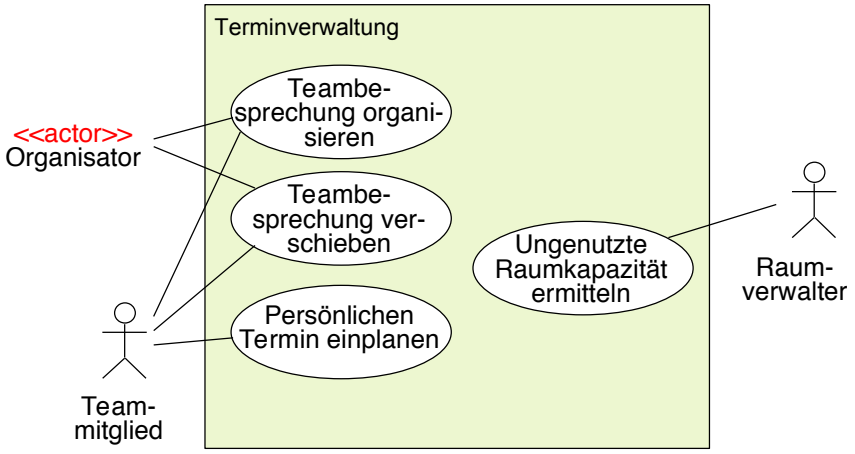
Jede Zeile wird von einer Szenarioanalyse entwickelt und endet in einer Kollaboration der beteiligten Objekte, in einem Team oder einem Konnektor.

Die Kollaborationen, Teams und Konnektoren verfeinern die Objekte querschneidend: Jede neue Szenarioanalse fügt also querschneidende Verfeinerungen zum Entwurf, zur Anwendungslogik, hinzu.

36.1 Nutzfalldiagramme (Anwendungsfalldiagramme, use case diagrams)

UML-Anwendungsfall-Diagramm (Nutzfall-, Use-Case-Diagramm)

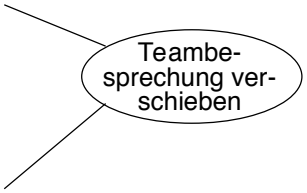
- Ein ***Anwendungsfall*** beschreibt die Interaktion (Kollaboration) der Akteure mit dem System (*querschneidend* durch das System)
- Aus einem Anwendungsfall wird, durch ***Szenarienanalyse,*** eine Kollaboration der beteiligten Objekte abgeleitet, oder ein Team (Kollaboration mit Teamobjekt), oder ein Konnektor (technisches Team)



- ***Nutzer (Stakeholder, Beteiligte):*** 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) .
- Eine ***Kollaboration*** fasst mehrere Interaktionen zu einem Szenario zusammen.

Anwendungsfälle (Nutzfälle)

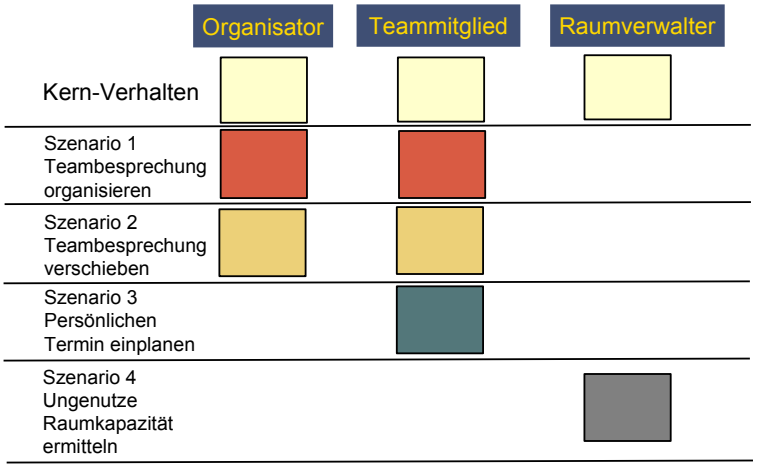
- **Anwendungsfälle** beschreiben funktionale Anforderungen an ein System *im Kontext* von Anwendern (Aktoren)
- Aus den Anwendungsfällen setzt man mittels ***Szenarienanalyse*** dann zusammen:
  - das **Kontextmodell** mit der Schnittstelle des Systems, also die sichtbaren Funktionen des Systems
  - die **Top-Level-Architektur**, die die Realisierung der sichtbaren Funktionen des Systems auf oberster Ebene zeigt
  - **Kollaborationen** zwischen Objekten in der **Anwendungslogik,**
    - der **Architektur**



Szenarioanalysen für Anwendungsfälle verfeinern alle Schichten des Systems querschneidend

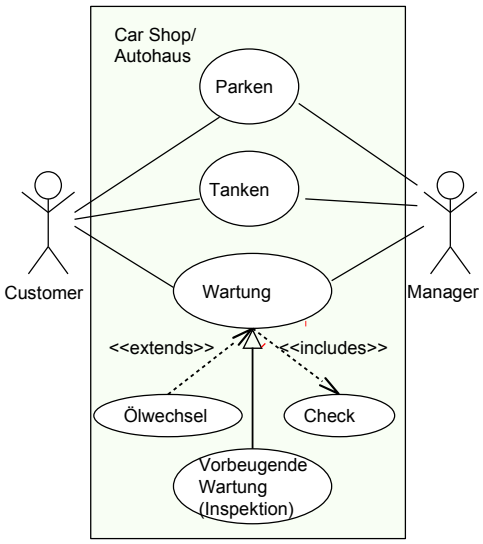
Übung

- Erstellen Sie von allen Anwendungsfalldiagrammen dieses Kapitels eine Objekt-Szenario-Matrix, die die Beteiligung der Objekte an den Szenarien festhält.



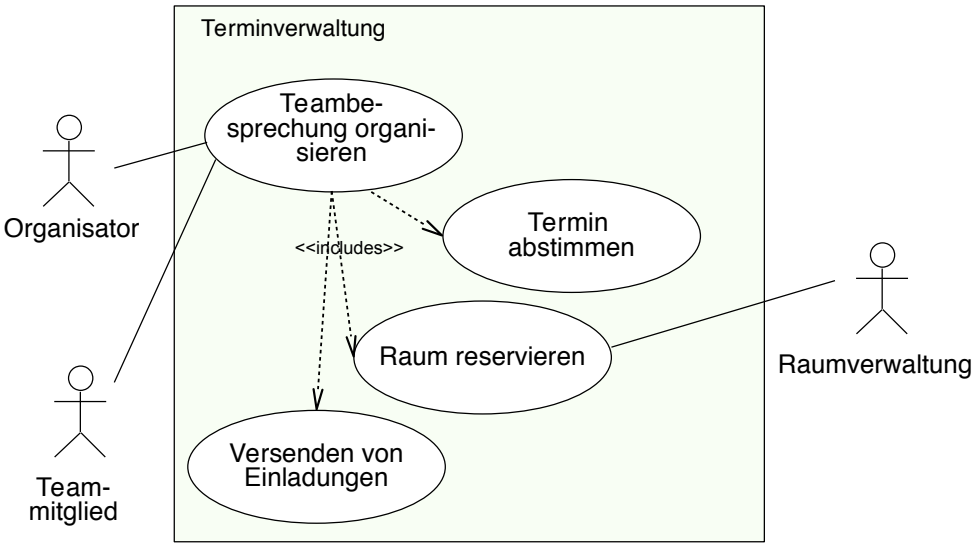
Verallgemeinerung, Erweiterung und Aufruf von Anwendungsfällen

- Die Vererbungsrelation zwischen Anwendungsfällen beschreibt Generalisierung bzw. Spezialisierung
  - Hier: *Wartung* ist allgemeiner als *Vorbeugende Wartung*
- Die ***Includes***-Relation beschreibt Bestandteile der Aktionen (Aufrufbeziehung zwischen Aktionen)
  - Hier: *Wartung* beinhaltet *Check*
- Die ***Extends***-Relation beschreibt *optionale* Erweiterungen
  - Hier: *Ölwechsel* kann Teil von *Wartung* sein



[nach Pfleeger]

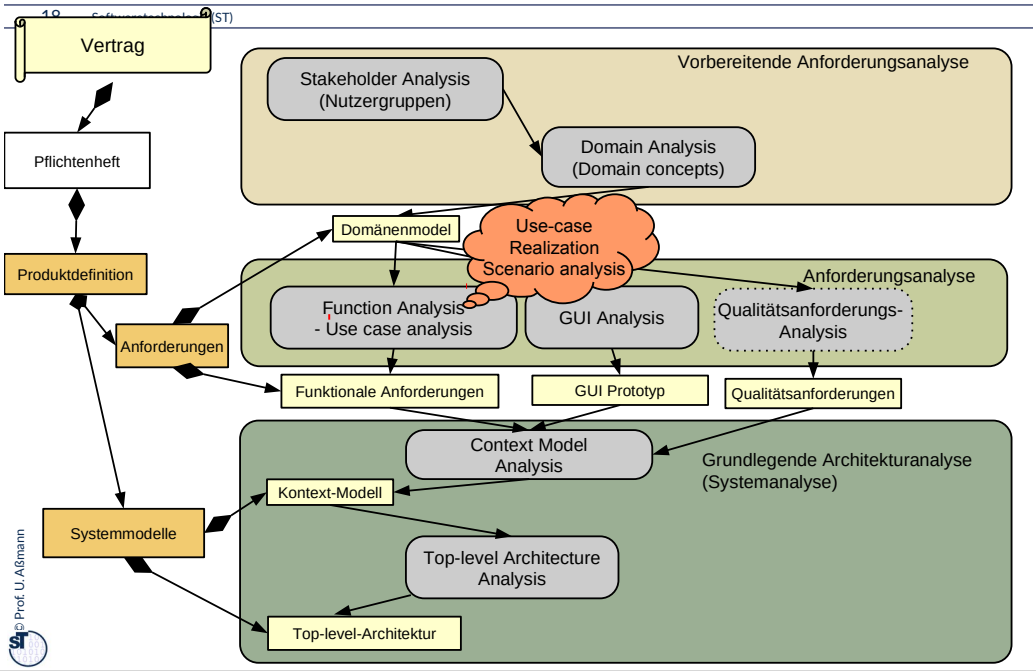
Verfeinerung des Anwendungsfalls „Teambesprechung organisieren”



36.2 Szenarienanalyse - Ableitung von Kollaborationen und Teamklassen aus Anwendungsfällen

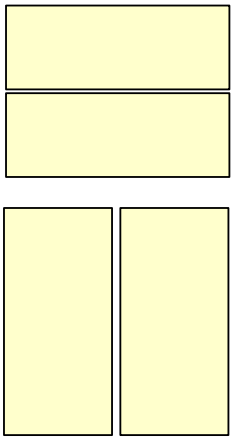
Anwendungsfallrealisierung, use case realization

Q10: Drei-Schritt der Analyse (Anforderungen und fachliches Modell)



Wege der Szenarienanalyse (use case realization analysis)

- Die Methode der ***Anwendungsfallrealisierung*** *(use case realization, Szenarienanalyse, scenario analysis)* wird verwendet, um:
  - **Systemanalyse** des Kontextmodells und der Top-Level-Architektur
  - **Querschneidende Verfeinerung** durch mehrere Klassen/Objekte in der **Anwendungslogik** durchzuführen, in dem Kollaborationen für die Objektverfettung abgeleitet werden
- Anwendungsfallrealisierung nutzt zur Verfeinerung verschiedene Interaktionsdiagramme **mit Schwimmbahnen (Lebenslinien)**:
  - mit Interaktionsdiagrammen
    - mit Lebenslinien im Sequenzdiagramm (sequence diagram, sequence chart)
    - mit Kommunikationsdiagramm (communication diagram)
  - mit Aktionsdiagrammen
    - mit Schwimmbahnen im Aktivitätsdiagramm
    - mit einem Netz von kommunizierenden Verhaltens-(Zustands-)maschinen



- Wie arbeitet man mit dem Kunden in der Szenarienanalyse?
  - Verfeinerung geschieht zusammen in Abstimmung
  - Nutze eine CRC-Analyse für die Szenarienanalyse
  - Nutze Anwendungsfälle
  - Nutze Objekt-Szenarien-Matrix

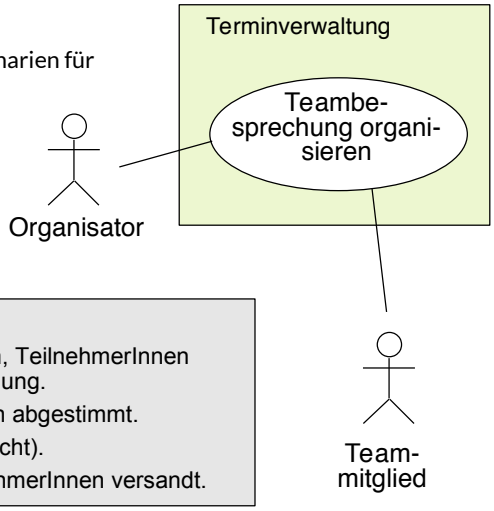
Szenarien

- **Def.:** 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'-Szenario), Ausnahmefälle ('exception case') und Fehlerfälle ('negativ'-Szenario).
- 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
  - Szenarien können durch CRC-Rollenspiel unterstützt werden
- Wähle als Szenariobeschreibung durch Interaktionsdiagramme oder Aktionsdiagramme
  - Leite daraus eine Kollaboration ab
  - Und daraus eine Teamklasse

Szenarienanalyse

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

  Durchspielen eines der Normalfall-Szenarien für 'Teambesprechung organisieren'

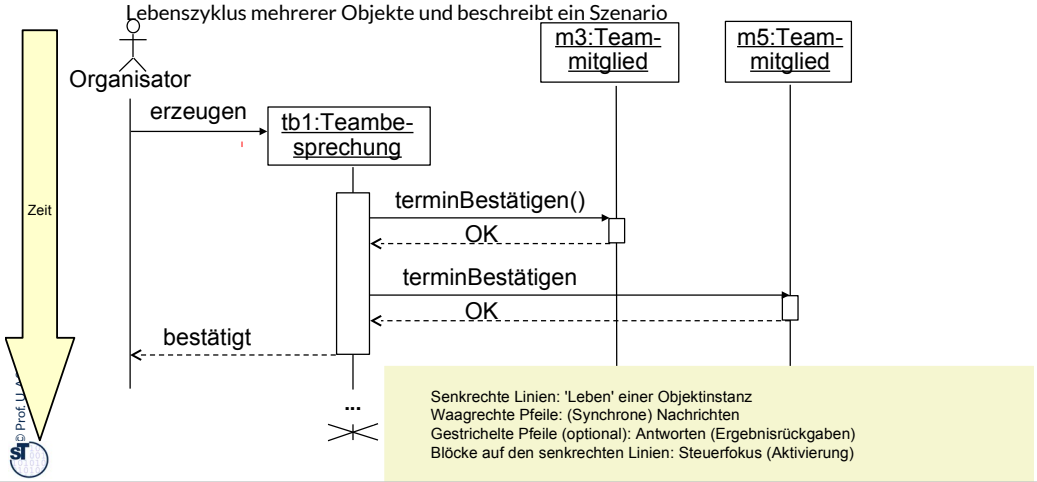


Durchspielen:

- 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.

36.2.1 Szenarienanalyse mit UML-Sequenzdiagrammen

- **Def.:** Ein **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*** oder "Schwimmbahnen")
  - Sequenzen von Nachrichten, geordnet durch die Zeit
- Achtung: das Sequenzdiagramm schneidet mit seinen Schwimmbahnen 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)

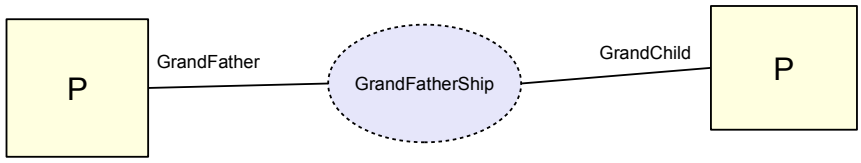
vH

Es gibt vereinfachende Varianten von Sequenzdiagrammen, z.B. kann die Darstellung der Operationsinkarnationen weggelassen werden.

Es ist in manchen Fällen sinnvoll, den Verlauf der Zeit während einer Nachrichtenübermittlung darzustellen, z.B. wenn sich zwei Vorgänge zeitlich kreuzen (Bsp. Mahnung - Zahlung). In diesem Fall ist es empfehlenswert, die Pfeile nicht waagrecht, sondern schräg

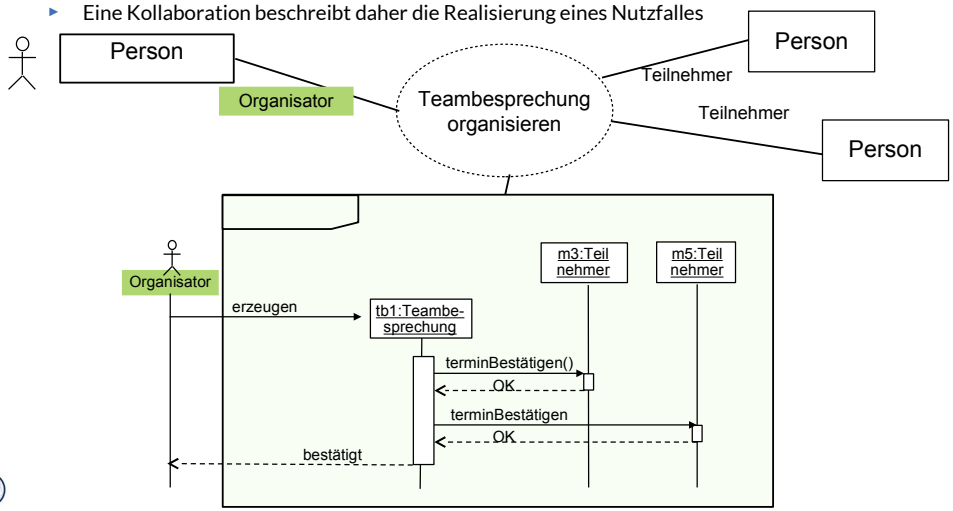
Kollaborationen (collaborations, Teamklassen) in UML

- Wdh.: Eine **Kollaboration** *(team class, collaboration, Rollenmodell) ist ein* Schema für die Zusammenarbeit von Objekten. Sie definiert mehrere Rollen von Spielern (player) im Zusammenspiel
- In UML stellt sich eine Kollaboration dar als
  - generisches Sprachkonstrukt mit Klassen-Parameter P und Rollenname als Bezeichner für Tentakel
  - konkret instantiiert mit Klassen



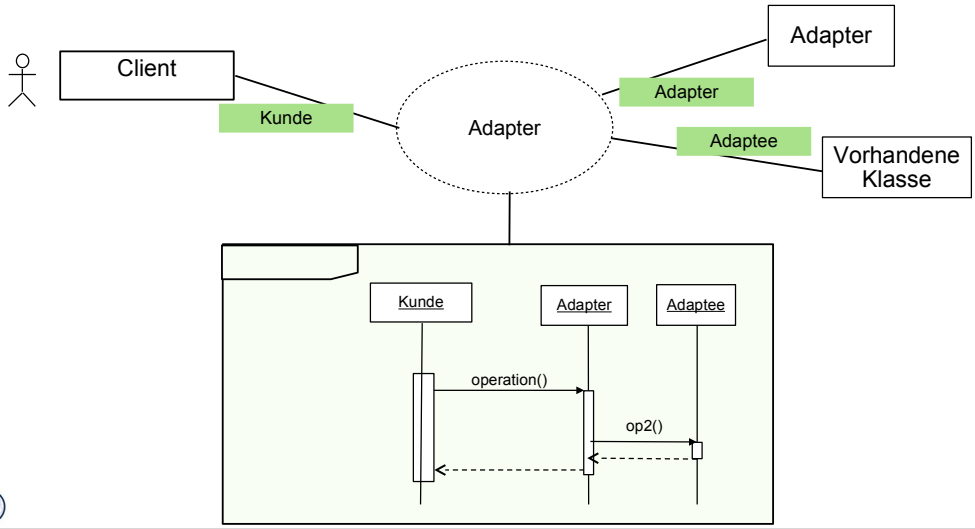
Kapseln eines Szenarios in einer Kollaboration

- Eine **Kollaboration** kann mit einem Sequenzdiagramm als Verhalten unterlegt werden
  - Die einzelnen Lebenslinien geben das Verhalten eines Objekts in der Kollaboration an
- Die Kollaboration beschreibt also ein *Szenario querschneidend durch die Lebenszyklen mehrerer Objekte*
- Eine Kollaboration beschreibt daher die Realisierung eines Nutzfalles

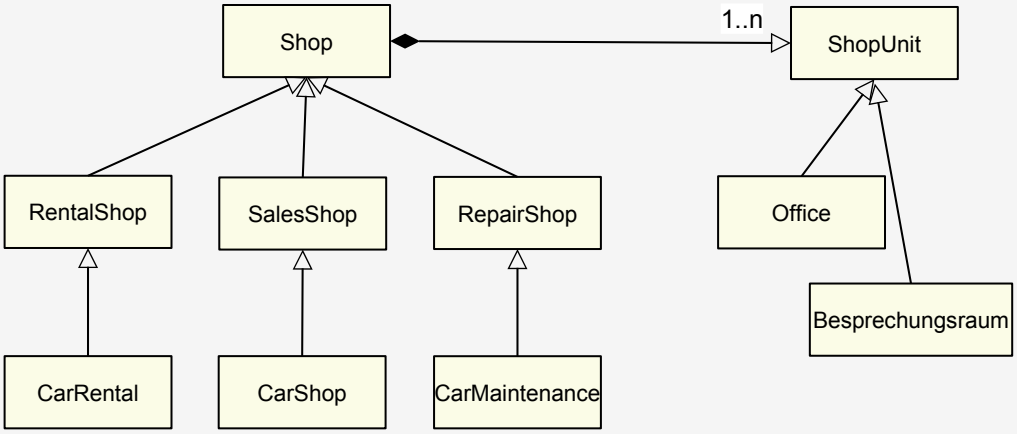


Wdh: Auch Entwurfsmuster werden in UML mit Kollaborationen spezifiziert

- Auch ein Entwurfsmuster wird in UML mit einer **Kollaboration** spezifiziert. Das Gamma-Buch ordnet jedem Entwurfsmuster ein Sequenzdiagramm zu
- **Merke**: *Bewährte Kollaborationen werden als Entwurfsmuster festgehalten!*

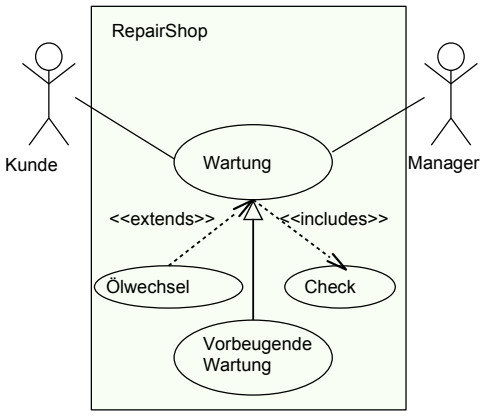


36.2.2 Beispiel Szenarienanalyse RepairShop



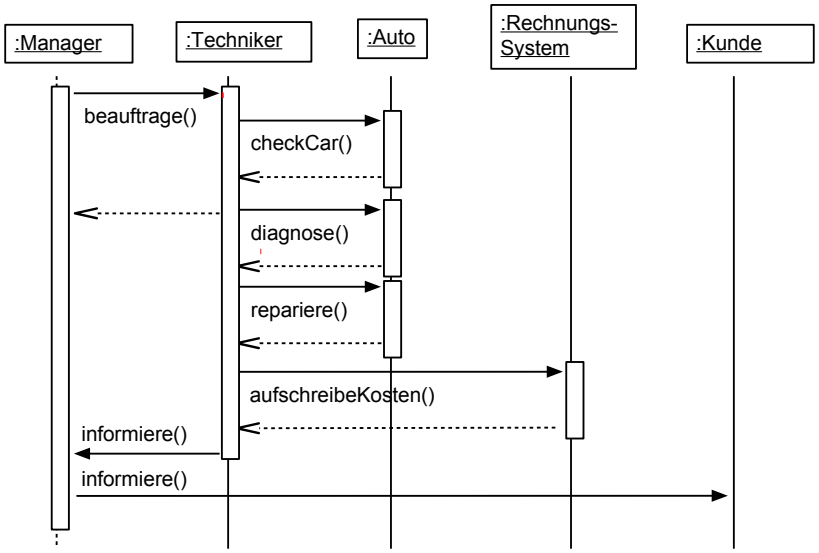
Szenarienanalyse mit Sequenzdiagrammen

- Ausgangspunkt: Anwendungsfall Auto-Wartung (Wartung) im RepairShop[Pfleeger]



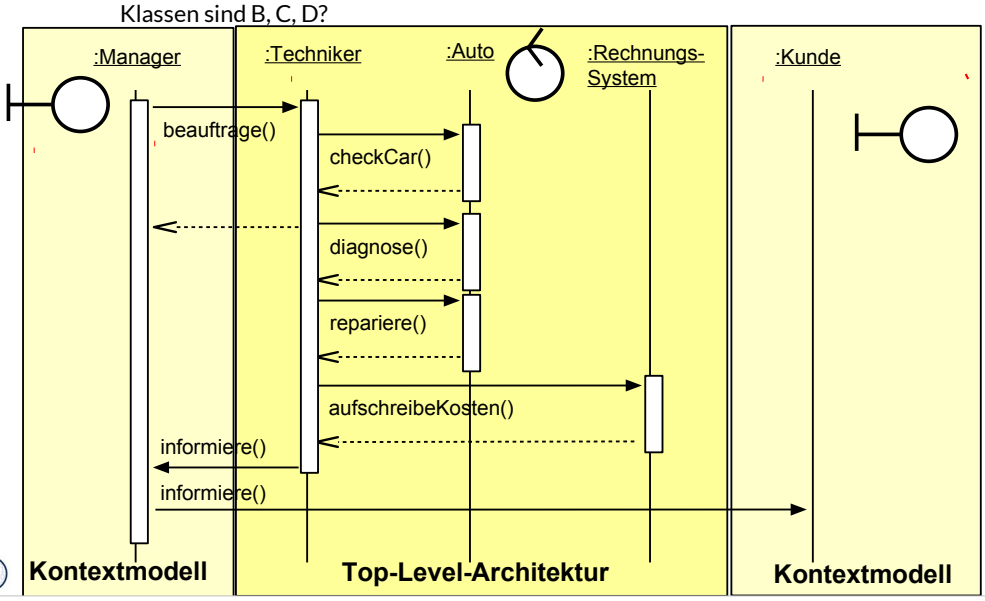
Szenarienanalyse Sequenzdiagram Service-Station

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



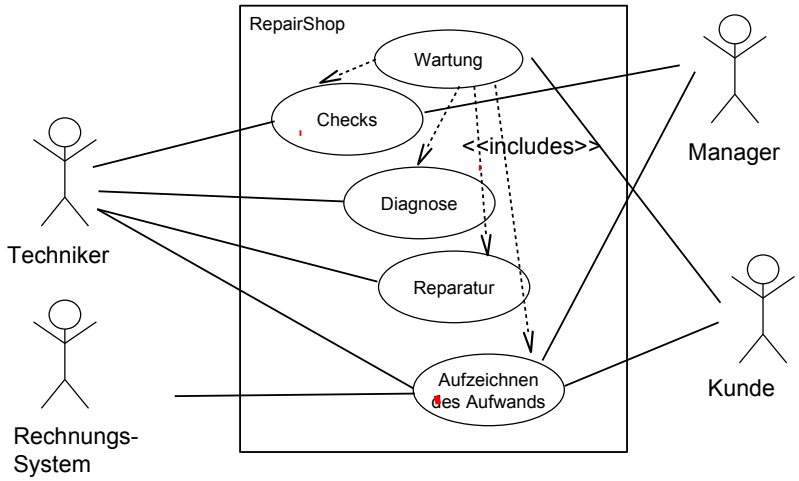
Beziehung zum Kontextmodell und Top-Level-Architektur

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



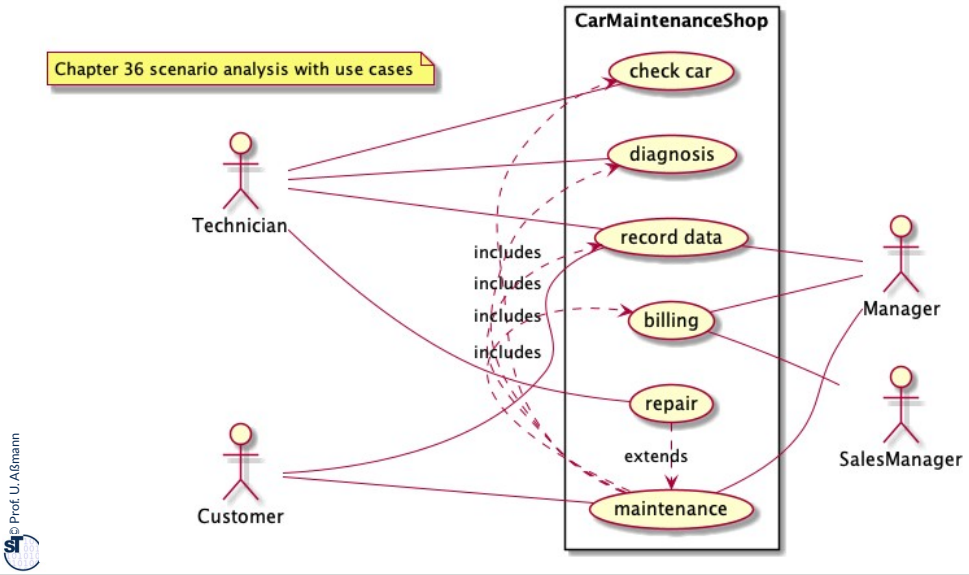
Verfeinertes Anwendungsfall-Diagramm Service-Station

- Aus dem Sequenzdiagramm kann nun ein verfeinertes Anwendungsfalldiagramm erstellt werden



Verfeinertes Anwendungsfall-Diagramm Service-Station

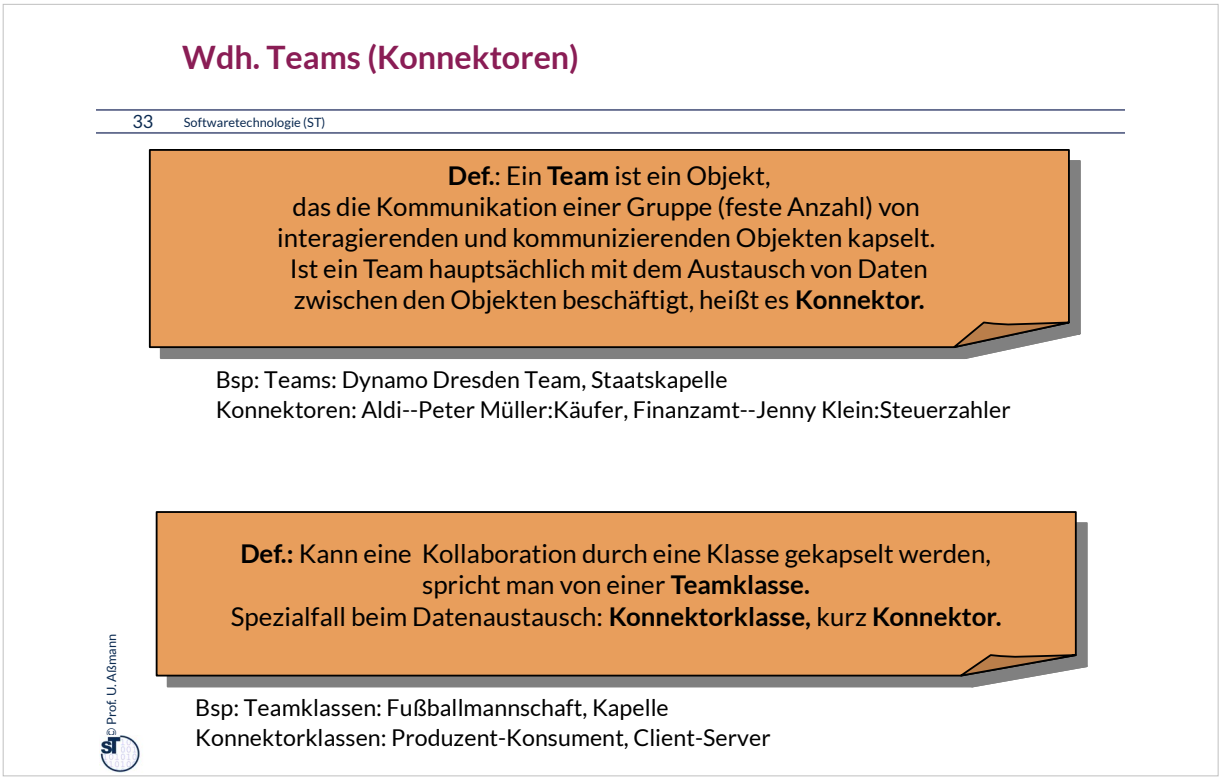
- Mit plantUML erstelltes Diagramm:




```
@startuml
' @author Uwe Assmann, version 0.1,  @date 2020-05-30
note "Chapter 36 scenario analysis with use cases" as N1
left to right direction
skinparam packageStyle rectangle
actor Technician
actor Manager
actor Customer
actor SalesManager
rectangle CarMaintenanceShop {
  (repair) .> (maintenance) : extends
  (maintenance) .> (check car) : includes
  (maintenance) .> (diagnosis) : includes
  (maintenance) .> (record data) : includes
  (maintenance) .> (billing) : includes
  Technician -- (check car)
  Technician -- (diagnosis)
  Technician -- (repair)
  (record data) -- Technician
  (maintenance) -- Manager
  (record data) -- Manager
  (billing) -- Manager
  (billing) -- SalesManager
  Customer -- (record data)
  Customer -- (maintenance)
}
@enduml
```

36.2.3 Erstellung von Teamklassen aus Szenarien

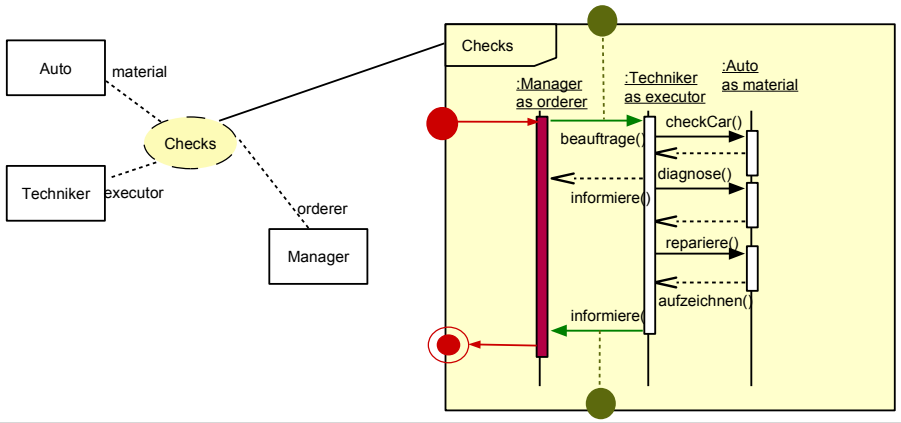
Siehe Kapitel 23 “connectors-iterators-channels”



Kollaborationen können in UML, aber nicht in Java beschrieben werden; Teams und Konnektoren schon, denn sie bilden Objekte bzw. Klassen.

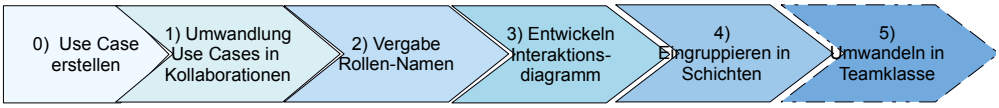
Ableitung von Kollaborationen aus der Szenarienanalyse

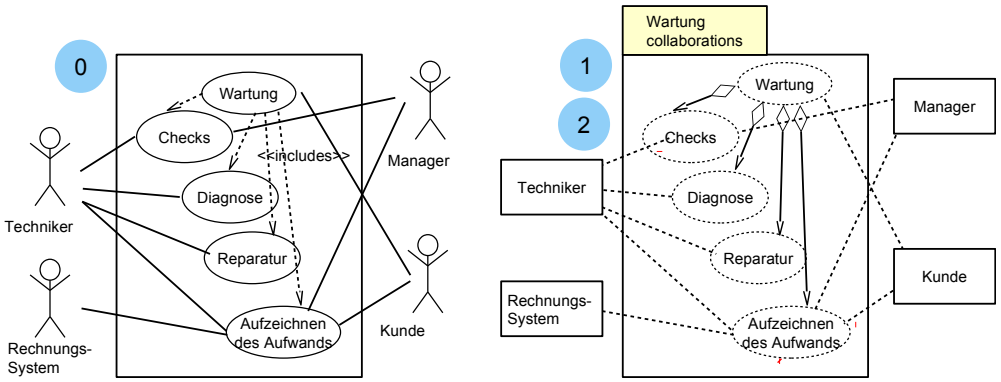
- Ein Sequenzdiagramm einer Kollaboration definiert:
  - Lebenslinien beschreiben das Verhalten der Rollen
  - Die Lebenslinie mit dem Anfangszustand kennzeichnet den **Initiator** mit **Initialzustand**
  - Die Lebenslinie mit dem Endzustand kennzeichnet den **Terminator** mit **Endzustand**;
  - **Initialbotschaft**: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



Umwandlung Anwendungsfall-Diagramm in Kollaborationen

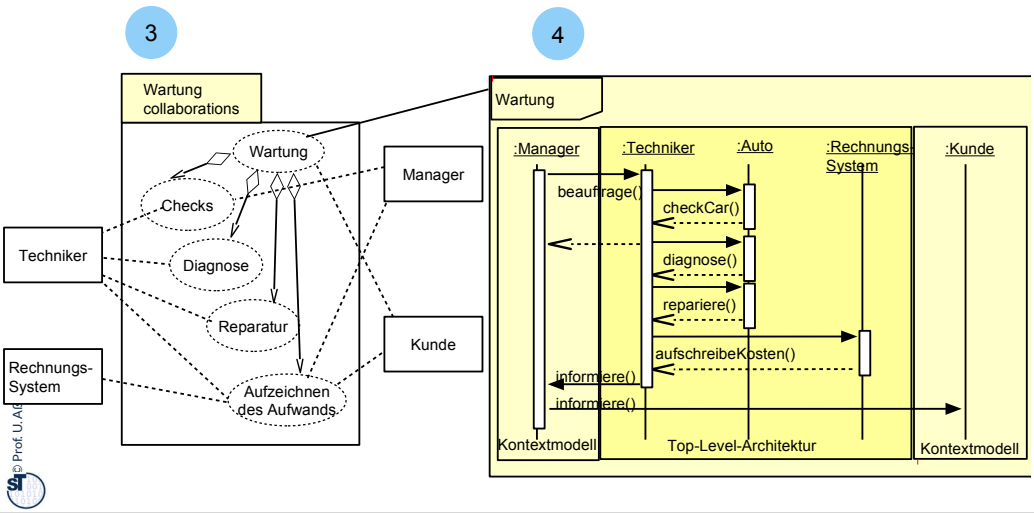
- Aus dem Anwendungsfalldiagramm kann schrittweise eine Menge von Kollaborationen erstellt werden





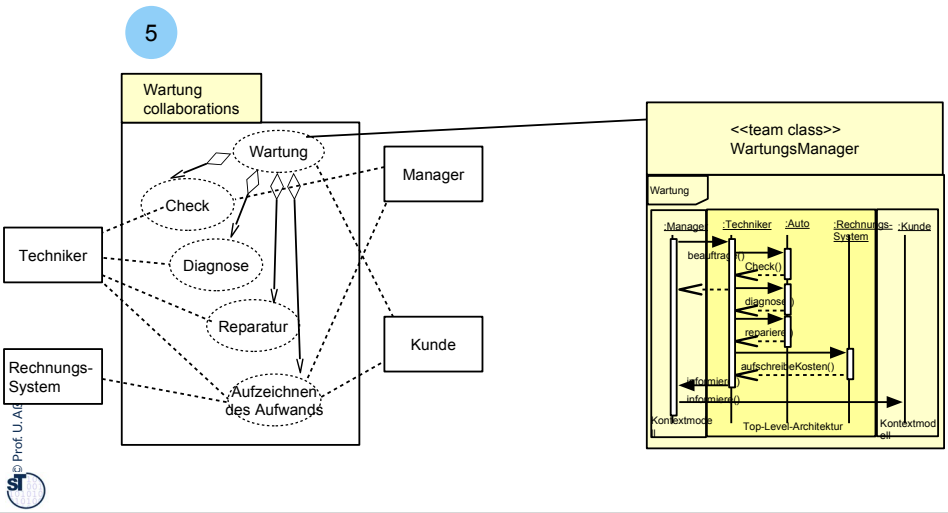
Umwandlung Anwendungsfall-Diagramm in Kollaborationen

- 3) Anhängen des Interaktionsdiagramms (hier Sequenzdiagramm) an einen Anwendungsfall
- 4) Eingruppierung Objekte in Schichten



Umwandlung Anwendungsfall-Diagramm in Teamklasse

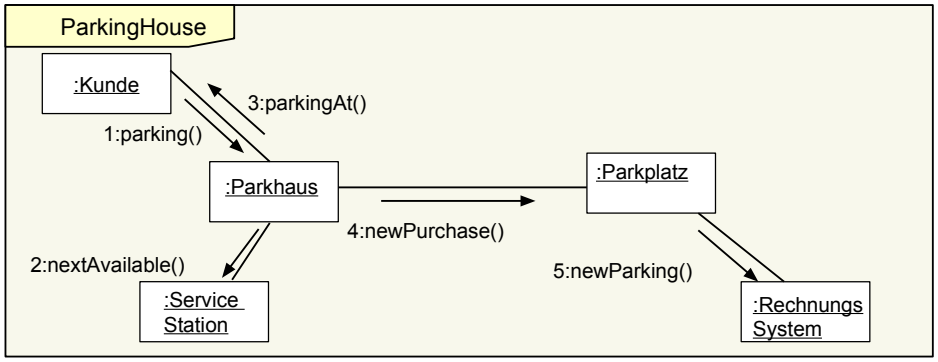
- 5) Umwandlung in Teamklasse, die die querschneidende Kollaboration steuert ("Reifikation")



36.3 Szenarienanalyse mit Kommunikationsdiagrammen

Kommunikationsdiagramm (Communication Diagram)

- Ein **Kommunikationsdiagramm** ist ein Interaktionsdiagramm, das den Fluss der Aufrufe zwischen Objekten 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 Objektnetze mit komplexem Verbundverhalten



36.4. Szenarienanalyse mit Schwimmbahnen in Aktionsdiagrammen

Szenarienanalyse funktioniert auch mit Aktionsdiagrammen: Aktivitätendiagramme (UML-AD), Statecharts (UML-SC)

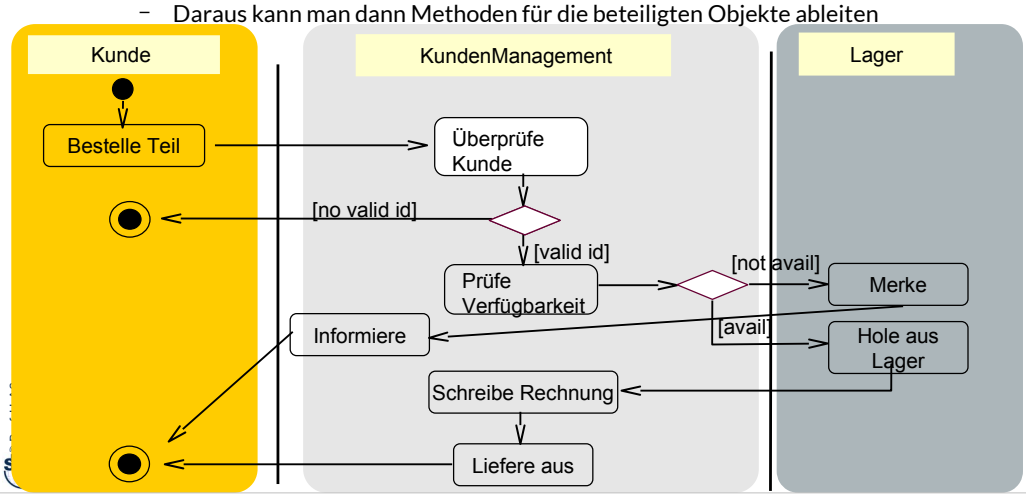
Querscheidende dynamische Modellierung mit Szenarienanalyse

- 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..*

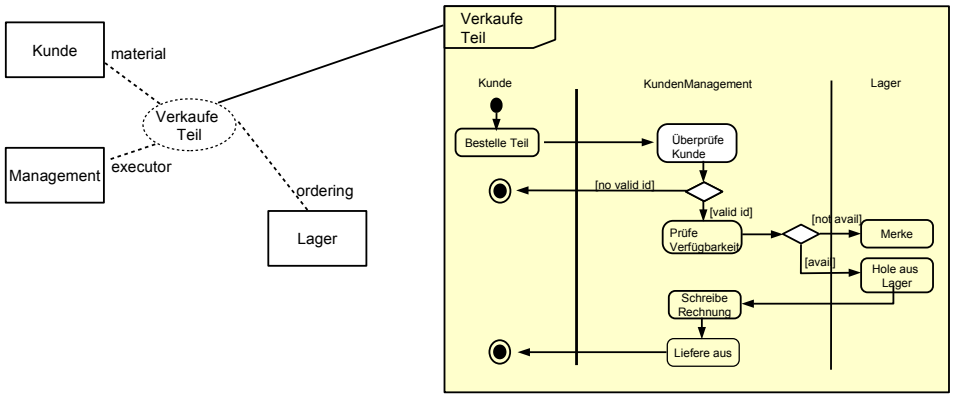
Szenarienanalyse mit UML-AD: Bearbeiten einer telefonischen Bestellung

- Aktivitäten können durch **Schwimmbahnen (swimlanes)** gegliedert werden, die Objekten zugeordnet sind
  - Jede Schwimmbahn spezifiziert eine **Rolle** eines Objekts im Kontext des Szenarios
  - Daraus kann man dann Methoden für die beteiligten Objekte ableiten



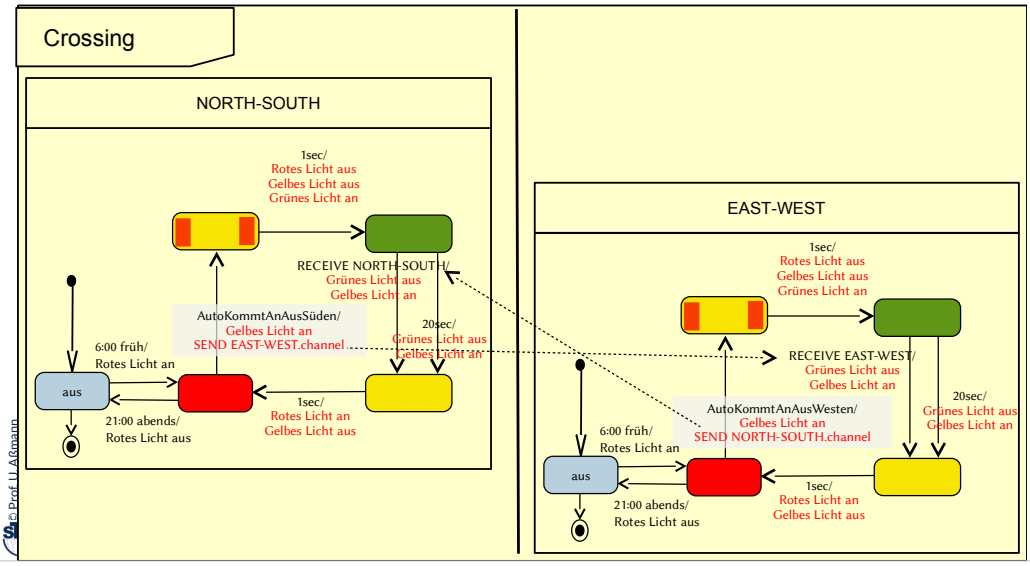
Aktivitätendiagramme als Verhalten von Kollaborationen

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



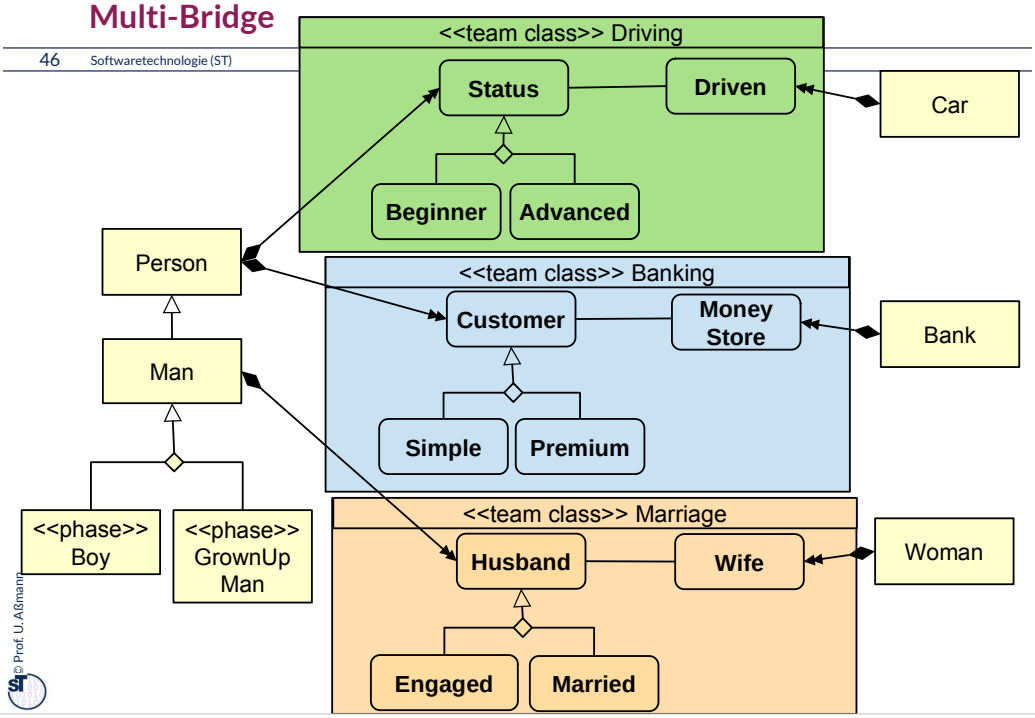
36.4.2. Kopplung zweier Ampeln an einer Kreuzung durch kooperierende Automaten

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



36.5 Wozu braucht man querschneidende Verfeinerung mit Szenarienanalyse und Kollaborationen?

Beispiel: Querschneidende Erweiterung von Objekten mit Multi-Bridge

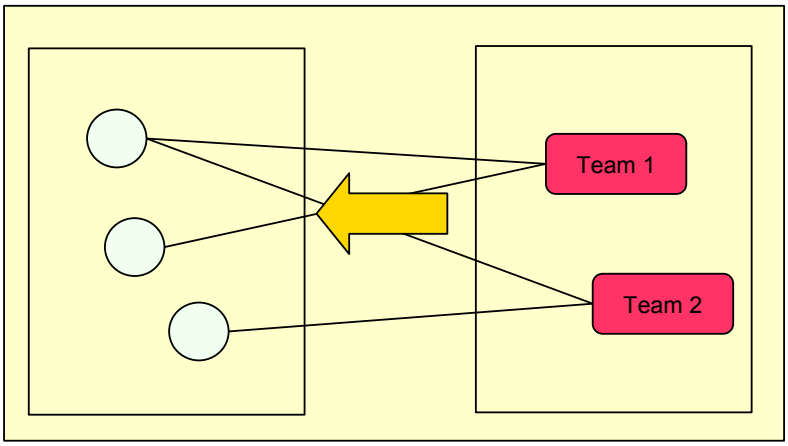


Wozu braucht man querschneidende Verfeinerung mit Szenarienanalyse, Kollaborationen?

- **Szenarienanalyse** ermittelt *querschneidendes Verhalten durch eine Menge von Klassen bzw. Objekten hindurch*
  - und entwickelt dazu aus Anwendungsfällen Kollaborationen und Teamklassen
  - Genau wie Klassen bilden Kollaborationen und Teamklassen *modulare Wiederverwendungeinheiten,* also spezielle Komponenten.
- Aber: Kollaborationen und Teams bilden *relationale, querscheidende* Module, die die Ergebnisse von Szenarienanalysen kapseln können
- Einsatz:
  - zur Verfeinerung
  - zur Erweiterung
  - zur Ersetzung (Variabilität)
  - zum Umwickeln anderer Kollaborationen (Wrapping)
- Implementierung
  - Teamklassen können als erweiterbares Multi-Bridge Entwurfsmuster realisiert werden

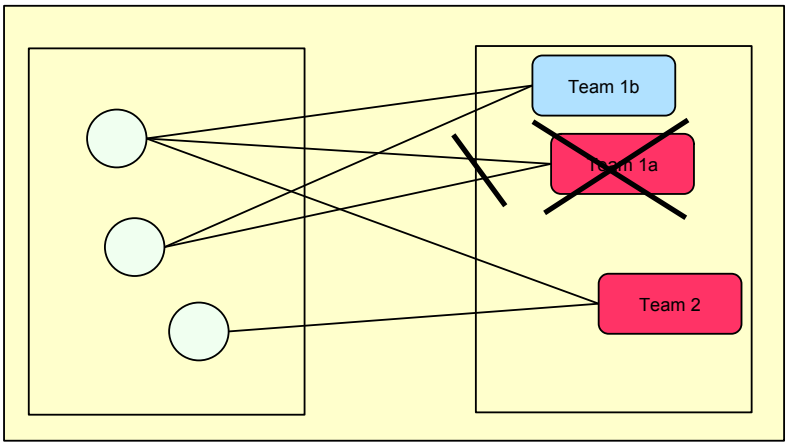
Querschneidende Erweiterung

- **Querscheidende Erweiterung** erweitert *mehrere Punkte* eines bestehenden Systems
- Implementierung: Erweiterung mehrerer komplexer Objekte mit Multi-Bridge Muster



Querschneidende Ersetzbarkeit

- **Querschneidende Ersetzbarkeit** ersetzt *Schnitte (Scheiben)* durch ein System
- Implementierung: Variation mehrerer komplexer Objekte mit Multi-Bridge Muster



Wozu braucht man querschneidende Analyse?

- Seit den 60ger Jahren exisitiert die Methodik, mit *punktweiser* Verfeinerung aus Anforderungsspezifikationen Entwürfe abzuleiten (“stepwise refinement”)
  - Erst seit den 90ger Jahren ist *querschneidende Verfeinerung* hinzugekommen, weil punktweise Verfeinerung nicht reicht → “Aspektorientierte Entwicklung” (“aspect-oriented software development”)
  - In einer querschneidenden Verfeinerung kommt eine *Kollaboration (Scheibe)* oder eine *Teamklasse* zur Anwendung hinzu
  - Szenarioanalse entwickelt mehrere querschneidende Verfeinerungen, die in der Objekt-Szenario-Matrix zusammengefasst werden
- Software braucht Kollaborationen zwischen Objekten
  - Wer querschneidende Analyse und das Ableiten von Kollaborationen und Teamklassen beherrscht, kann viel schneller als andere gute Software entwickeln

Was haben wir gelernt?

- Ein Anwendungsfall (use case im use case diagram) 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 (Sequenzdiagramm oder Kommunikationsdiagramm)
  - Oder ein Aktionsdiagramm, das ebenfalls das Protokoll zwischen den Rollen der Kollaboration beschreibt (Aktivitätendiagramm oder Statechart mit Schwimmbahnen)
- Szenarienanalyse verfeinert querschneidend, i.G. zu punktweiser Verfeinerung
- Kollaborationen sind querscheidende Module, die für Erweiterung, Ersetzung, Variabilität und Umhüllung eingesetzt werden können
- Man lagert eine Kollaboration in eine Anwendung ein, in dem man alle Spieler mit den Rollen der Kollaboration überlagert.

The End

- Warum ergibt die Szenarienanalyse querschneidende Verfeinerungen der Analysemodelle?
- Wieso kann eine Kollaboration mehrere Objekte erweitern?
- Wie kann das Entwurfsmuster Bridge genutzt werden, auf ein Kernobjekt einen Rollensatelliten aufzuprägen (zu superimponieren)?
- Wie superimponiert eine Kollaboration mehrere Spieler durch neue Rollen?
- Warum ergibt sich aus der Superimposition mehrere Kollaborationen für jedes beteiligte Objekt eine Multi-Bridge?
- Beschreiben Sie die Schritte vom Use Case Diagramm über die Szenarioanalyse zur querschneidenden Verfeinerung.
  - Wie viele Ausprägungen des Musters Bridge erhält man?
  - Wie viele Ausprägungen von Multi-Bridge?
- Erklären Sie den Unterschied zwischen einer Kollaboration und einer Teamklasse.

36.A.1 Teams als spezielle Kollaborationen

- Im Entwurf werden Kollaborationen zu *Teams*, d.h. speziellen Kommunikationsobjekten, die querschneidend Kommunikation kontrollieren

36.4.3 Teams

- Ein **Team(-objekt)** ist ein Assoziationsobjekt, das aus bisher nicht kooperierenden Objekten ein Netz aufbaut, ihre Interaktion und Kommunikation leitet und dann wieder auflöst.
- Eine **Teamklasse** ist also eine Kollaborationsklasse, die definiert:
  - eine Hauptklasse
  - die Spieler als innere Klassen (Rollenklassen) mit dem Multi-Bridge-Muster
  - Netzaufbau-Methoden, die das Hauptobjekt mit den Spielern verbinden
  - Netzabbau-Methoden
  - Kommunikationsmethoden, die auf die inneren Objekte delegieren
  - Kanäle, die Daten zwischen den Objekten hin- und herschieben
- Die Verhalten des **Teams** wird durch eine Kollaboration beschrieben
  - Sie kann **teamgetrieben** erfolgen, so dass auf ein Ereignis hin alle Objekte angestoßen werden (passive Objekte werden exogen vom Hauptobjekt angesteuert)
  - Die Bearbeitung kann **spielergetrieben** erfolgen, sodass ein oder mehrere Objekte aktiv über dieKopllaboration kooperieren
- In Java implementiert man eine Kollaboration immer als **Hauptklasse mit inneren Klassen**

Schematische Realisierung von Teams mit inneren Klassen in Java

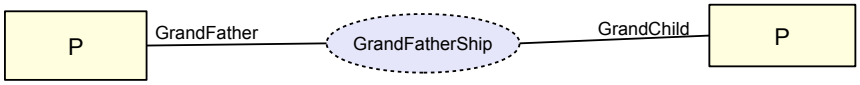

```
class Team {
  /* role as inner class */
  class RoleA {
     do() {..}
  }
  /* role as inner class */
  class RoleB {
     do() {..}
  }

// 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(); }
```


Bsp.: Teams mit inneren Klassen in Java

- In Java implementiert man eine Kollaboration immer als **Teamklasse** mit **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(); }
```

Teamklassen 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();
  }
}
```

Querschneidende Umwicklung

- **Bridge** ist nicht das einzige Muster für Kollaborationen
- **Querschneidende Umwicklung** wickelt Großobjekte mit Umwicklern (wrapper, decorator)
  - Reihenfolge der Umwicklung wichtig
- Implementierung: Umwicklung mehrerer komplexer Objekte mit Multi-Bridge und Decorator Muster