33. Strukturelle Analyse für Kontextmodell und Top-Level-Architektur (Systemanalyse)

Wie man ein System auf grobkörniger Ebene strukturiert und was der Kunde davon sehen muss

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 20-0.5, 15.06.20

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Literatur

- ST für Einsteiger, Kap. Klassendiagramme
- Zuser, Kap. 7-9
- Störrle Kap. 6 (!!)
- Balzert Kap. 6-7, 9-10

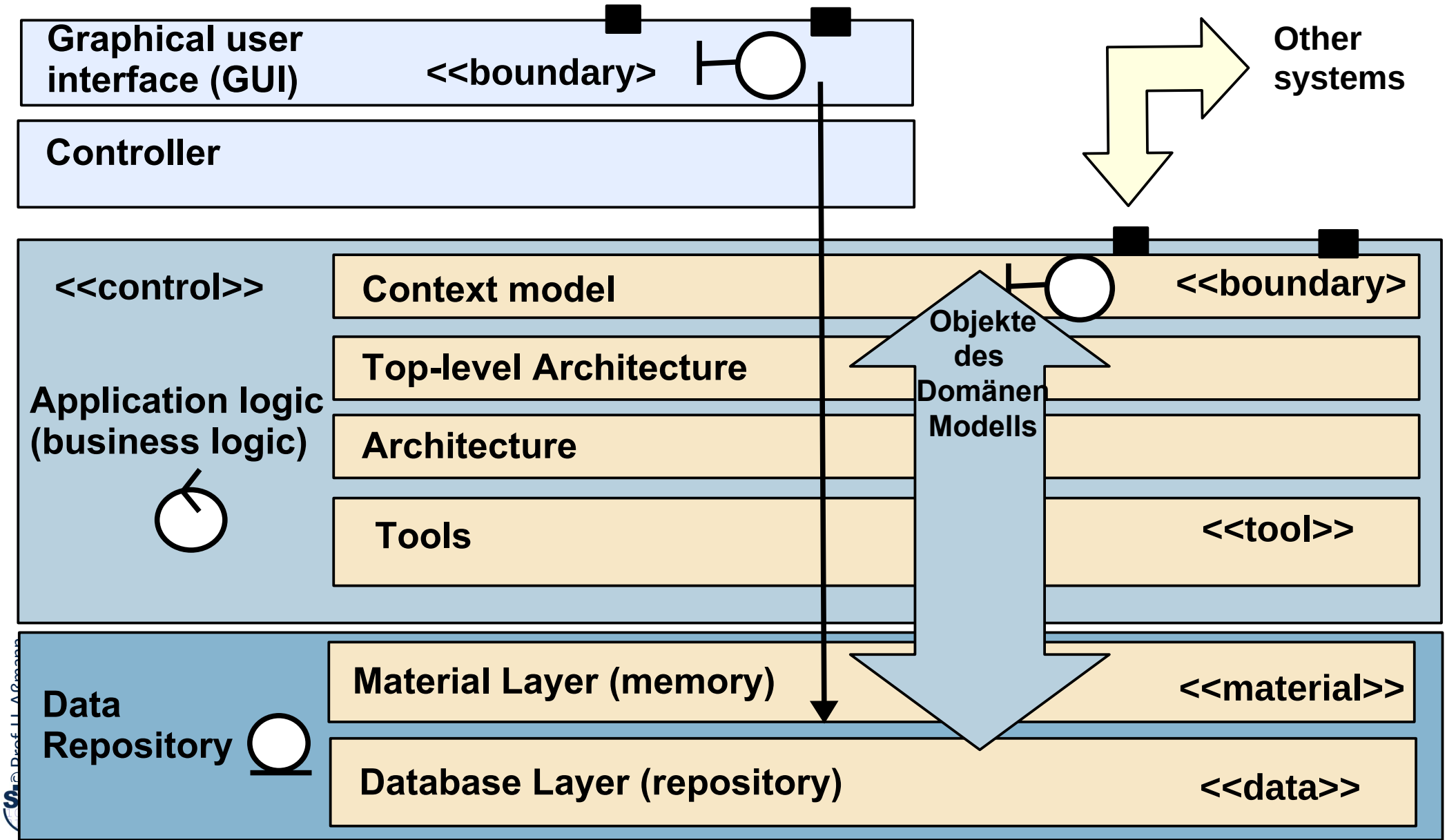
Überblick Teil III: Objektorientierte Analyse (OOA)

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

Warum ist dieses Kapitel wichtig?

- Mit Verbundobjekten und Komponenten können wir jetzt spezifizieren:
  - Kontextmodell des Systems (Schnittstellenmodell, muss der Kunde sehen)
  - Top-Level-Architektur, soweit sie der Kunde sehen muss
  - Schichten einer BCED-Architektur
- Kontextmodelle und Top-Level-Architektur gehören ins Pflichtenheft und bilden einen wesentlichen Teil des Vertrags zwischen Softwarefirma und Kunde.
- Ihre präzise Spezifikation ist essentiell für das Projekt.

Warum komplexe Objekte des Domänenmodells alle Schichten kreuzen, auch CM und TLA



Q5: Schritte der Modellierung in Bezug auf Schichten des Systems

	Schichten	Analyse	Entwurf	Feinentwurf	Implementierung
		Verfeinerung			
Benutzungsschnittstelle (Boundary)	GUI				
	Controller				
Anwendungslogik (Control)	**Kontextmodell**	Aufstellung aus Domänenmodell; Erarbeitung System-schnittstellen	stabil	Umsetzen auf jUML	stabil
	Top-Level-Architektur	Verfeinerung des Kontextmodells	stabil	Umsetzen auf jUML	stabil
	Architektur		Ausarbeitung Architektur (PSM)	Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML; Serialisierung	Details ausfüllen, Methoden ausprogrammieren
	Tools		Ausarbeitung Tools	Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML; Serialisierung	Details ausfüllen, Methoden ausprogrammieren
Datenhaltung (Database)	Material	Aufstellung aus Domänenmodell		Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML;	Details ausfüllen, Methoden ausprogrammieren

Toll Collect

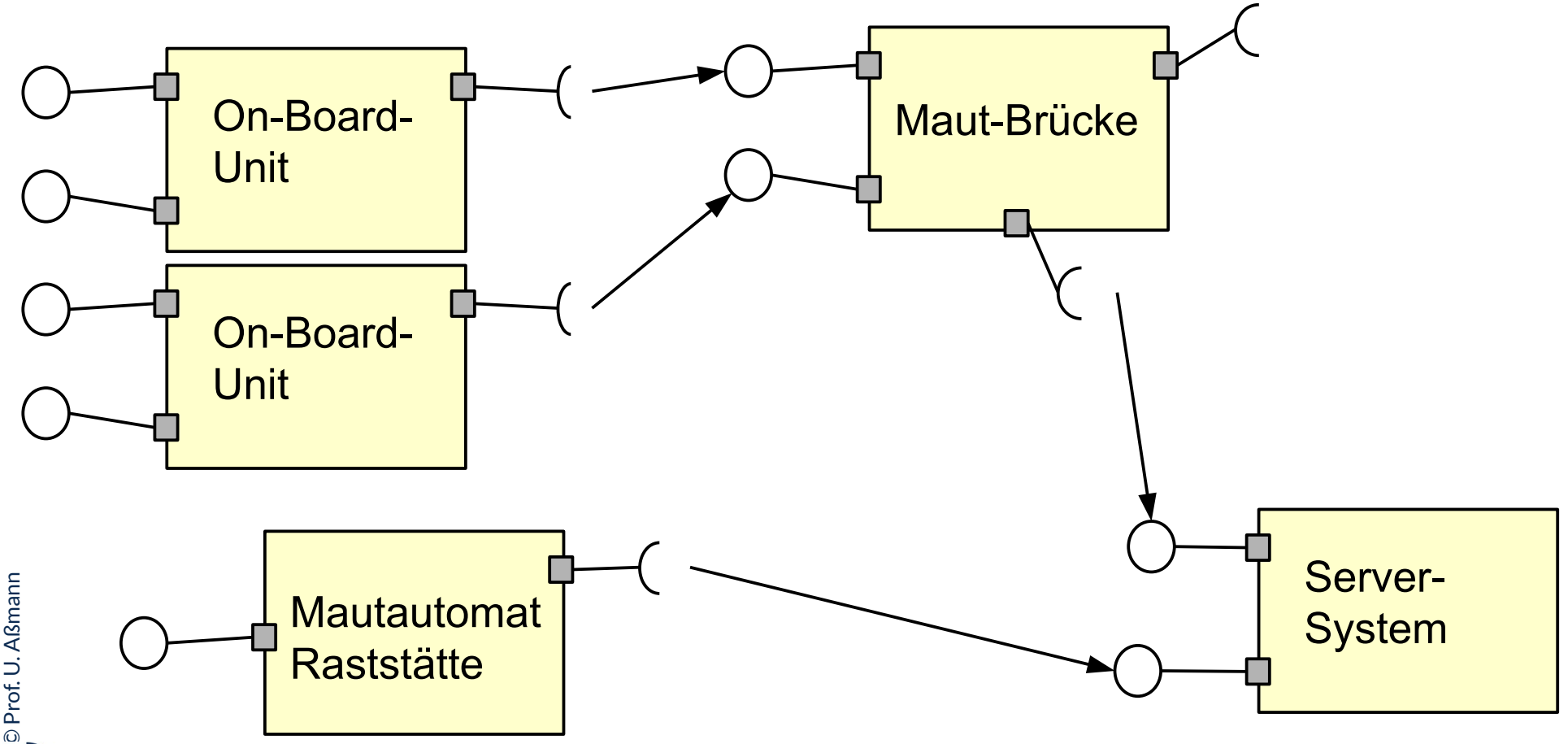
http://de.wikipedia.org/wiki/Lkw-Maut_in_Deutschland

- Auftragnehmer Telekom und Daimler Financial Services
- Schätzung der Einnahmen auf 3 Mrd jährlich
- Inbetriebnahme geplant 31.8.2003, dann 2.11. 2003
  - Tägliche Vertragsstrafe 250k€
- Realisiert 1.1. 2005, vollständig 1.1.2006

- Vertrag mit allen Anlagen und Nebenvereinbarungen aus 17.000 Seiten.
  - Der Kernvertrag, in dem Fragen der Haftung, Vertragsstrafen und Kündigungsfristen geregelt sind, umfasst 190 Seiten.

Warum war TollCollect verspätet? (Beobachtungen)

- Die Integrationsarbeiten auf oberster Ebene wurden unterschätzt
- Die Interaktionen zwischen den Teilsystemen (Komponenten und Schnittstellen zbw. ihren Kontextmodellen) wurden zu spät auf Skalierbarkeit getestet



Notation: UML Komponenten

Die Fragen der Systemanalyse im Kundengespräch

- Def.: Die **Systemanalyse**, ein Teil der objektorientierten Analyse, fragt nach den Schnittstellen und der Struktur der obersten Schichten eines Systems.

Die obersten Schichten eines Systems sind meist hierarchisch gestaltet. Typische Fragen der Systemanalyse:

- *Welche **Komponenten** hat das System?*
- *Welche Dienste soll das System liefern? (**Schnittstellen**)*
- *Welche Daten sollen in das System fließen? (Streams, Senken)*
  - *Datentypen, die in und aus dem System fließen, gehören ins Domänenmodell*
- *Wie kommuniziert das System mit anderen Systemen? (**Kanäle, Konnektoren**)*
- *Welche asynchronen Ereignisse treten außerhalb des Systems auf und wie fließen sie in das System? (**Ereigniskanäle**)*
- *Welche Ereignisse treten innerhalb des Systems auf und wie reagiert das System? (**Ausnahmen**, exceptions)*

33.1 Das Kontextmodell

- Das Kontextmodell enthält
  - alle Schnittstellen eines Systems (angeboten, benötigt)
  - die GUI (graphischen Oberfläche, interaktive Schnittstelle) mit ihren Kommandos (Knöpfe, Menüs, etc)

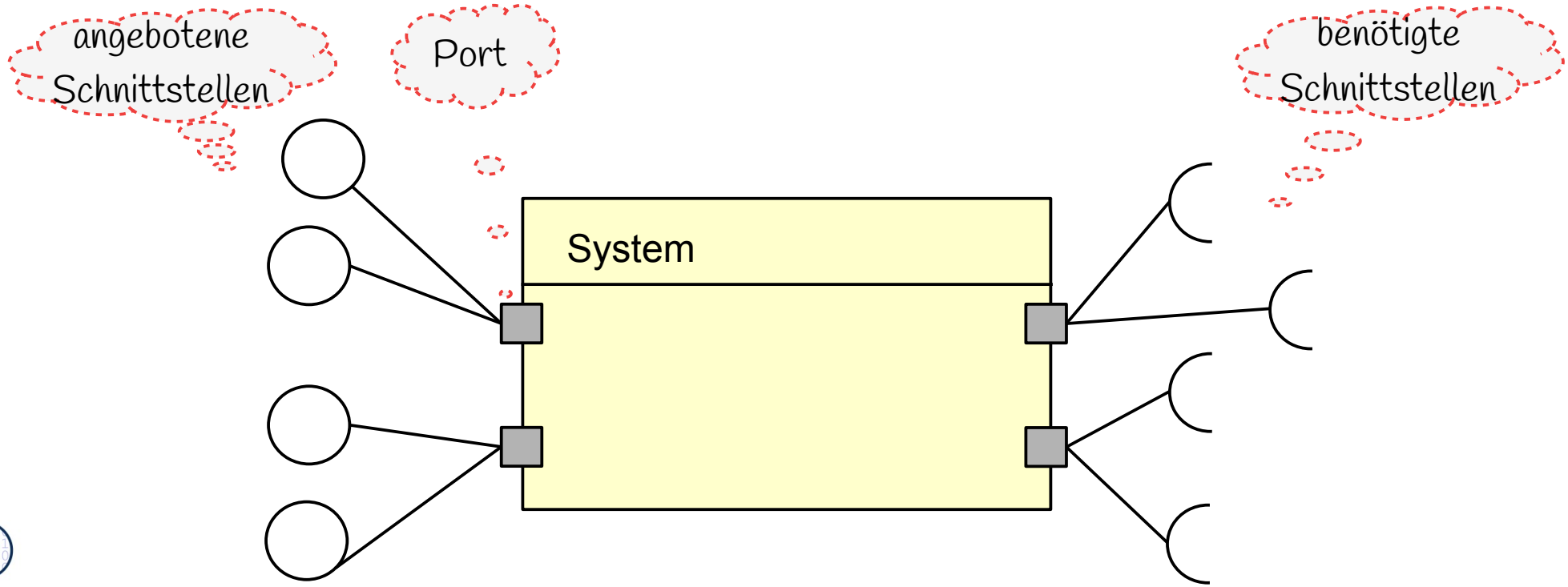
Kontextmodell (Schnittstellenmodell)

- Das System und seine obersten Schichten wird meist als *Klassenkomponente* verstanden
  - Mit angebotenen und benötigten Schnittstellen, mit ports
  - Hierarchisch durch die Ganz/Teile-Relation verfeinert
- Das ***Kontextmodell (Schnittstellenmodell) eines Systems*** enthält alle äusseren Schnittstellen des Systems (Portschnittstellen)
  - Funktionen für alle Anwendungsfälle (oberste Schicht der Anwendungslogik)
  - Ein- und Ausgabekanäle (ports, streams)
  - Benutzerschnittstelle (Eingabeformulare, GUI mit Menüs, Masken und Abfragen)
  - Daten, die in und aus dem System fliessen (Typen aus dem Domänenmodell),
    - Ereignisse, Ausnahmen

Das System wird oft als eine große komplexe Komponente gesehen, dessen Schnittstellen vom Kontextmodell beschrieben werden.

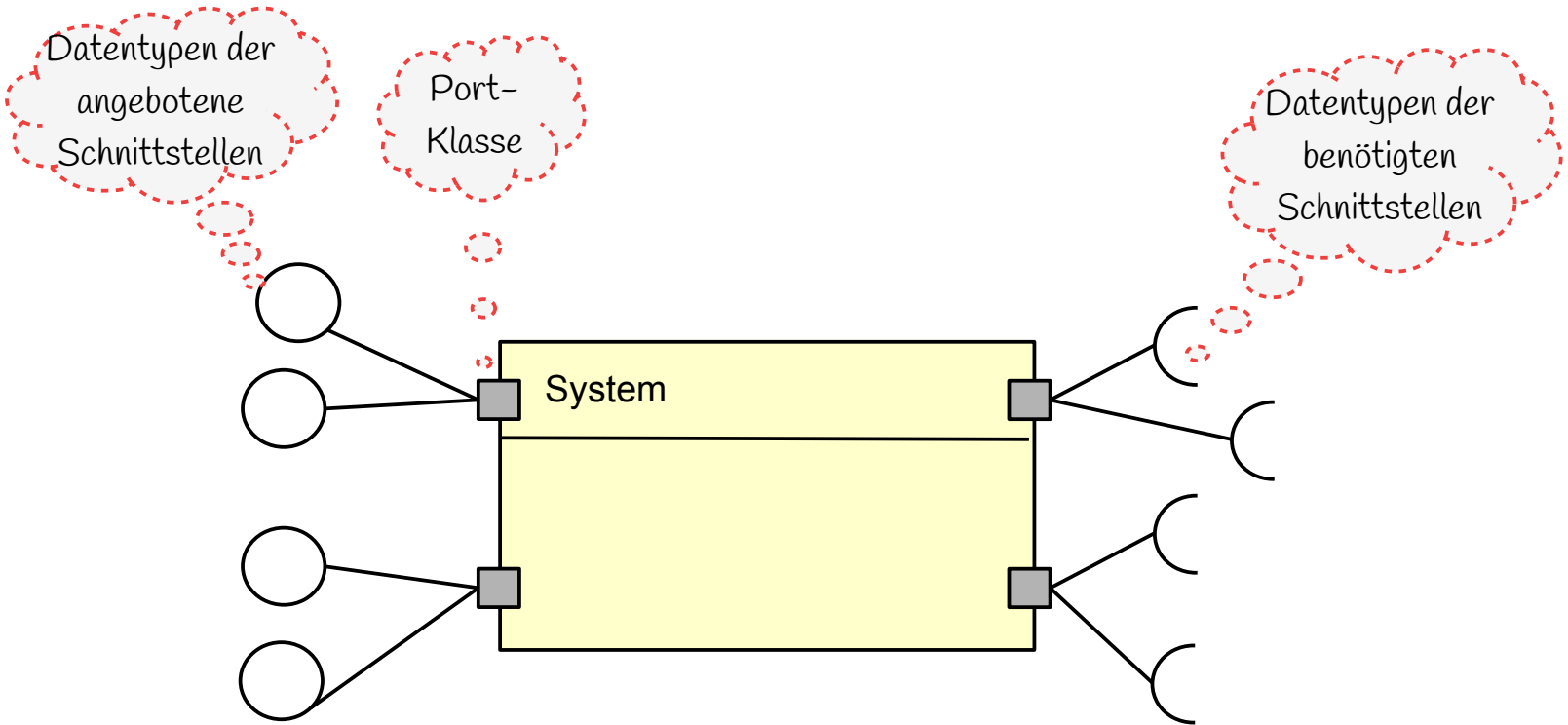
Komponenten im Kontextmodell

- Ein ***Montagediagramm*** ist ein Netz von Komponenten (Störrle)
- Das **Kontextmodell** des Softwaresystems ist die Menge seiner Schnittstellen
  - Das System ist selbst eine Komponente
  - Das Kontextmodells enthält *angebotene* und *benötigte* Schnittstellen:
    - Funktionale (call ports) und Strom-Schnittstellen (stream ports), gerade zu anderen Systemen
    - GUI-Bildschirme, Masken, Formulare
- Reine Klassen genügen nicht immer, denn sie spezifizieren keine benötigten Schnittstellen



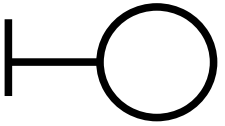
Datentypen des Kontextmodells stammen aus dem Domänenmodell

- Das Domänenmodell stellt die Typen für die technischen Objekte, Prozesse und Funktionalitäten des Kontextmodells zur Verfügung
  - Anschlüsse: Funktionale und strombasierte Schnittstellen (call und stream ports)
  - GUI-Bildschirme, Masken, Formulare
  - Port-Klassen und Schnittstellen sind Grenzklassen (boundary)

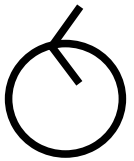


Unterscheidung von Systemschnittstellenklassen von anderen Klassen mit dem BCD-Profil

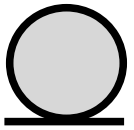
- **Schnittstellen-Klassen** (Portschnittstellen, Portklassen, GUI-Klassen, Grenzklassen, boundary classes)
  - Teil einer Schnittstelle, Benutzerschnittstelle, oder Port-Mixin der System-Komponente
- **Steuerungsklassen** (control classes)
  - Aktive Klasse der Anwendungslogik, die die Ausführung eines Prozesses steuert
  - Mit oder ohne Zustand
- **Material**: Passive Klasse, beschreibt Daten
  - Entitätsklassen (Entity): Beschreibt komposite, persistente Datenobjekte der Domäne
  - Datenklasse (Database): Adapterklasse für eine Entity in der Datenbank
    - Oft sind Entity and Database vereinigt
  - MaterialContainer: Container für Material
- BCD-Architektur: 3-Schichten-Architektur (3-tier architecture)
- BCED-Architektur: 4-Schichten-Architektur (4-tier architecture)



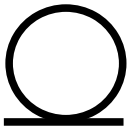
<<boundary>>



<<control>>
<<workflow>>
<<tool>>
<<process>>



<<material>>
<<entity>>

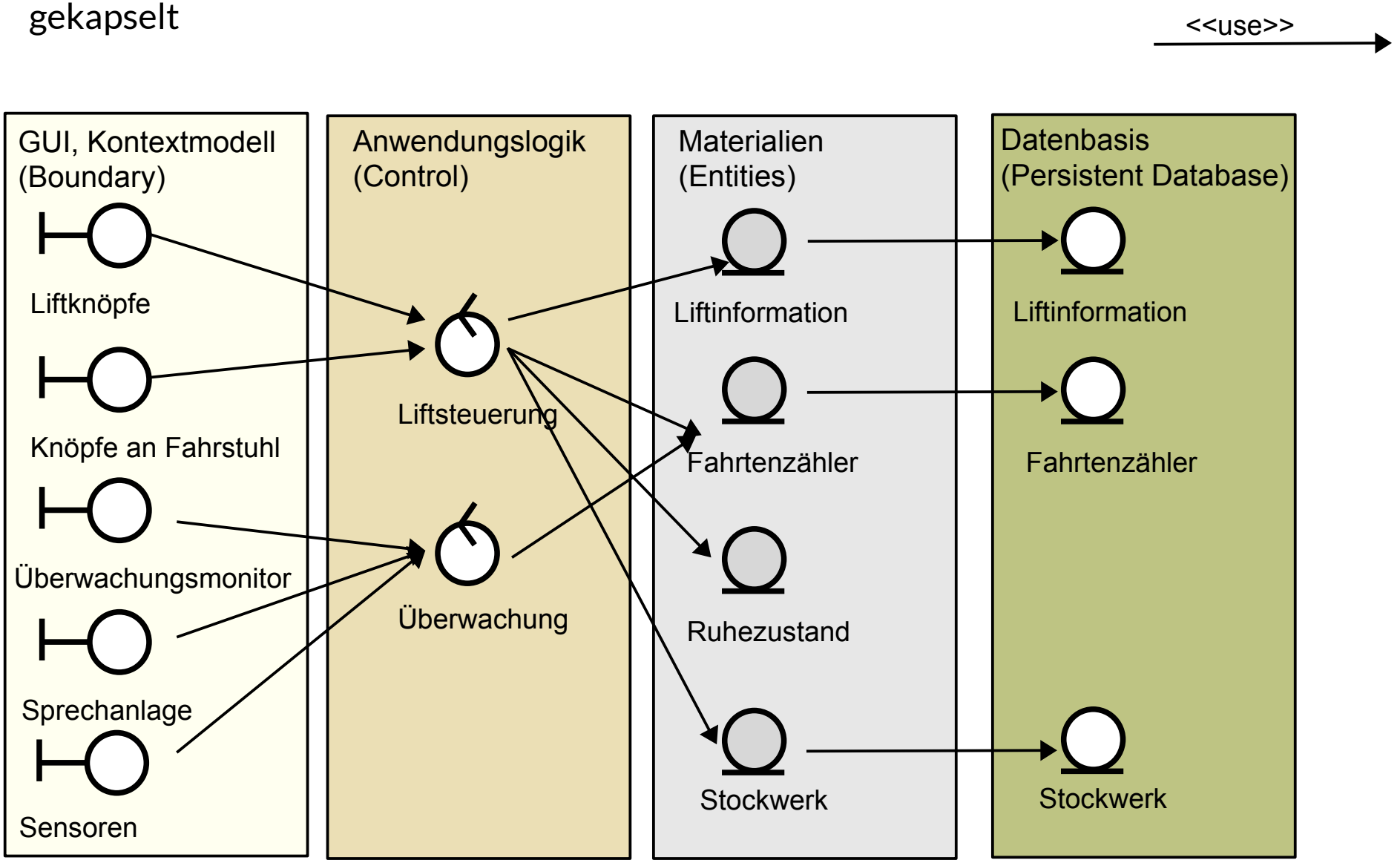


<<database>>

<<container>>

Bsp: Analyse-Klassenmodell Liftsteuerung im BCED-Stil

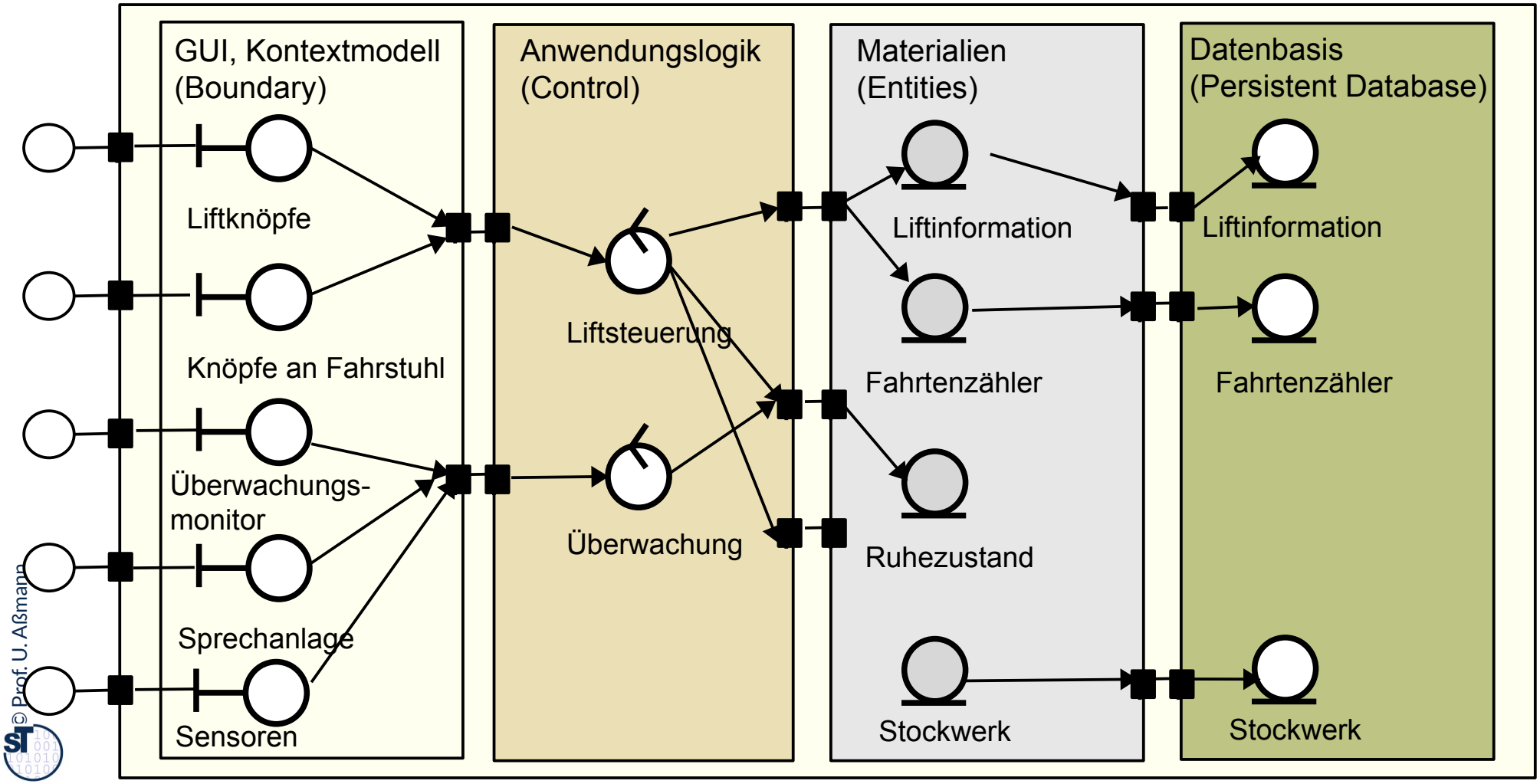
- als 4-Schichten-Architektur (4-tier architecture) von **Klassen:** Schichten sind nicht voll gekapselt



Bsp: Analyse-Montagediagramm: Liftsteuerung im BCED-Stil

- Als 4-Schichtenarchitektur von **Komponenten** → Schichten sind voll gekapselt und besser vorbereitet auf Austausch von Komponenten und Verteilung
- Viele Klassen haben *einen* Ein- und Ausgabe-Port; andere teilen sich einen Port

<<use>>

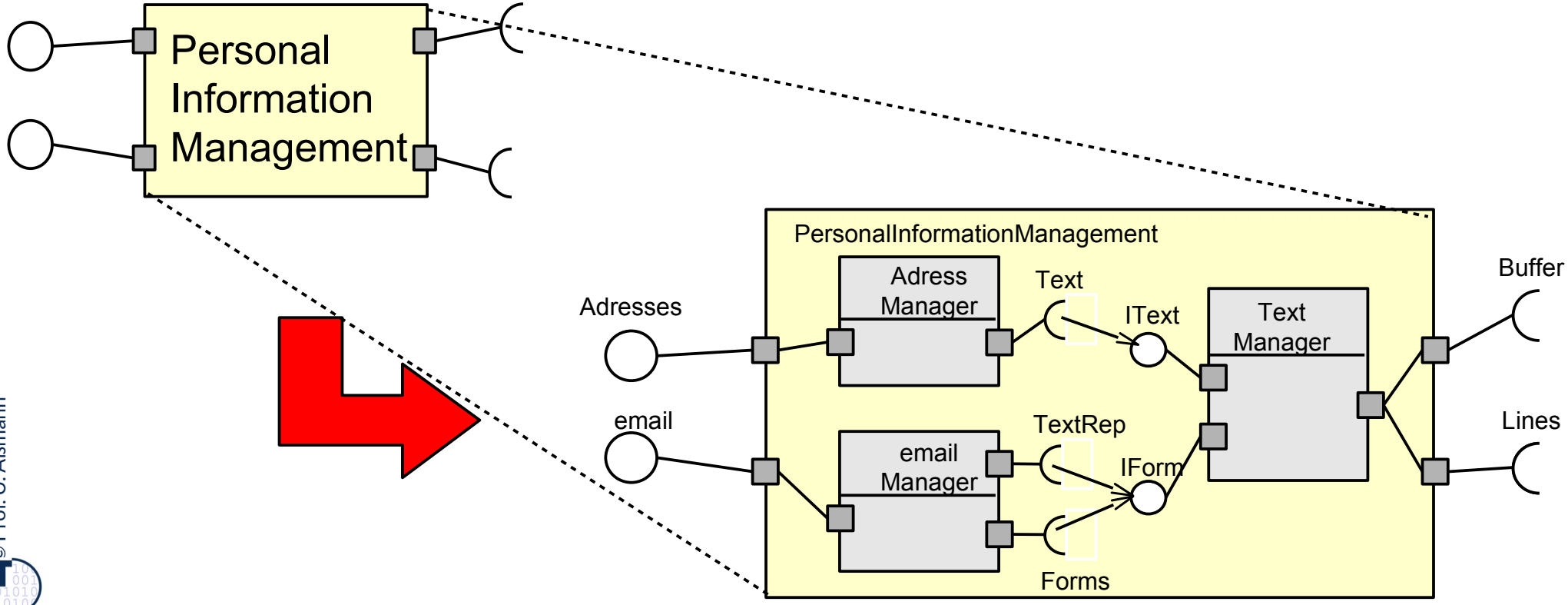


33.2 Top-Level-Architektur

Das System ist eine große Komponente, ein Verbundobjekt mit Ports,
dessen Schnittstellen vom Kontextmodell beschrieben werden
und das intern hierarchisch organisiert ist (Top-Level-Architektur).

Top-Level-Architektur als punktweise Verfeinerung der System-Komponente

- Wird das System als Komponente mit angebotenen und benötigten Schnittstellen betrachtet, kann man es hierarchisch mit der part-of-Relation verfeinern
  - Das Kontextmodell wird als Komponente, d.h., hierarchisch, spezifiziert
  - Die **Top-Level-Architektur** entsteht durch die erste Verfeinerung des Kontextmodells
  - Die inneren Komponentenklassen werden sichtbar



Warum wird die Top-Level-Architektur in der Anforderungsanalyse ermittelt?

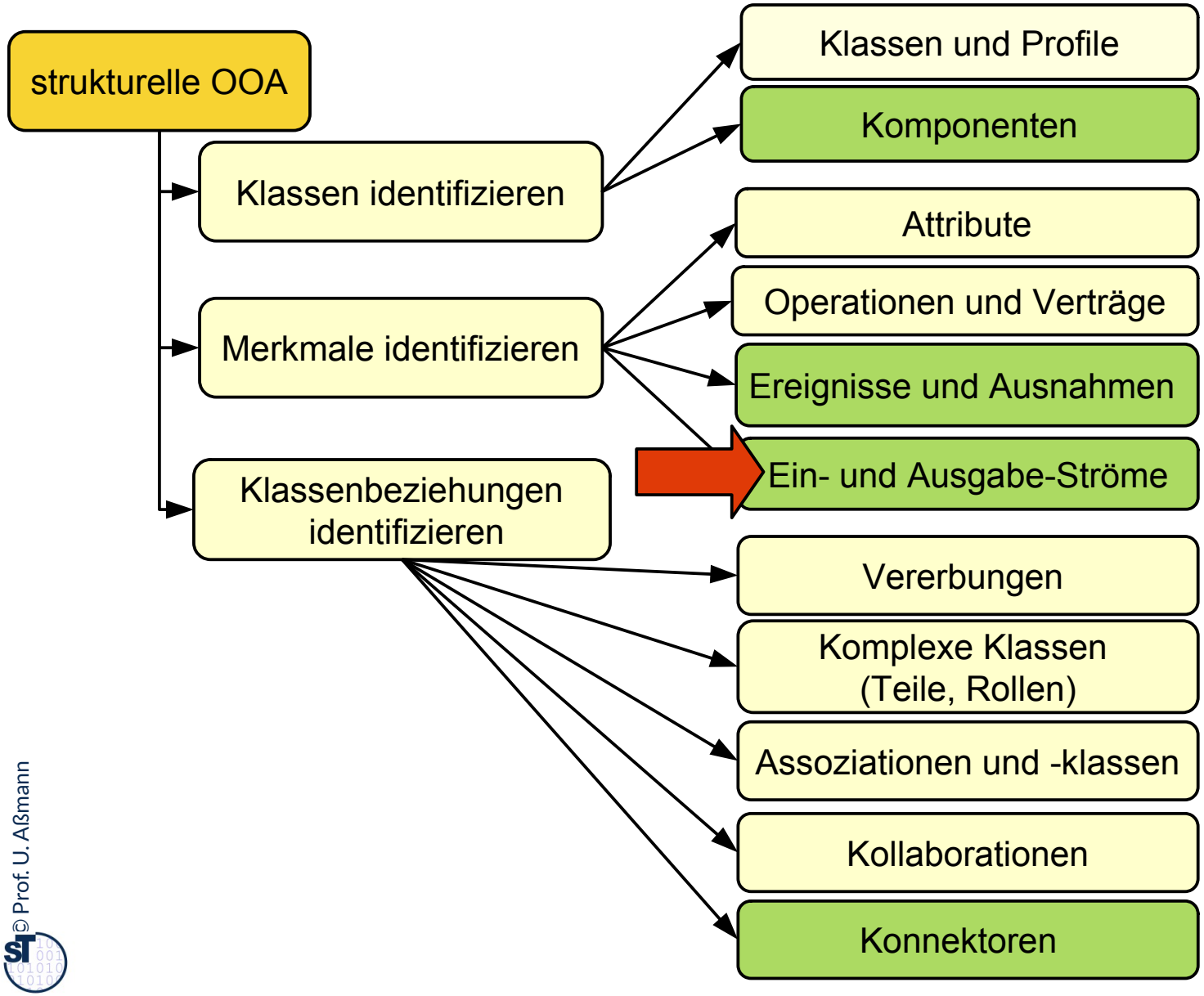
- Man sieht die Antwort an TollCollect: *Die Top-Level-Architektur der Anwendung gehört mit zur Anforderungsspezifikation*, weil
  - die Komponenten der ersten Schicht dem Benutzer sichtbar sind (was sind die Top-Level-Komponenten der TollCollect-Software?)
  - sie oft Aufgaben direkt zugeordnet werden können, die der Benutzer kennt (was sind die Aufgaben und Top-Level-Komponenten einer Groupware?)
  - sie zur Kostenplanung und Abrechnung für das Projekt verwendet werden (Produktstrukturplan, siehe Vorlesung Softwaremanagement)
  - sie dem Manager eine Einteilung von Projektmitarbeitern vereinfachen
- Wenn sie nicht zur Anforderungsspezifikation hinzukommt, ist sie als erstes Dokument im Entwurf auszuarbeiten
  - um die Planung zu vereinfachen

33.3 Asynchrone Systemmerkmale im Kontextmodell

Ereignisse und Ausnahmen (Exceptions)

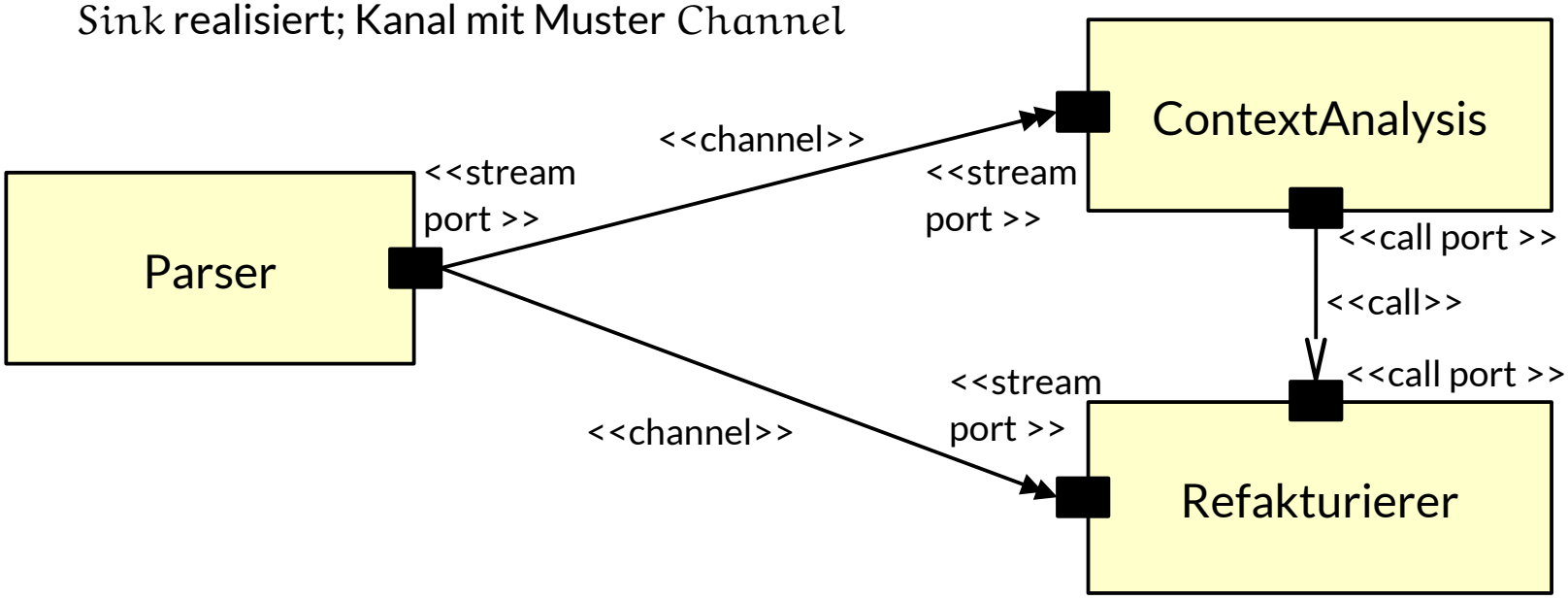
Q6: Schritte der strukturellen, metamodellgetriebenen Analyse

- gelb: Domänenmodell; grün: Kontextmodell, TopLevel-Architektur



Strom-Anschlüsse von Komponentenklassen und Kanal-Konnektoren

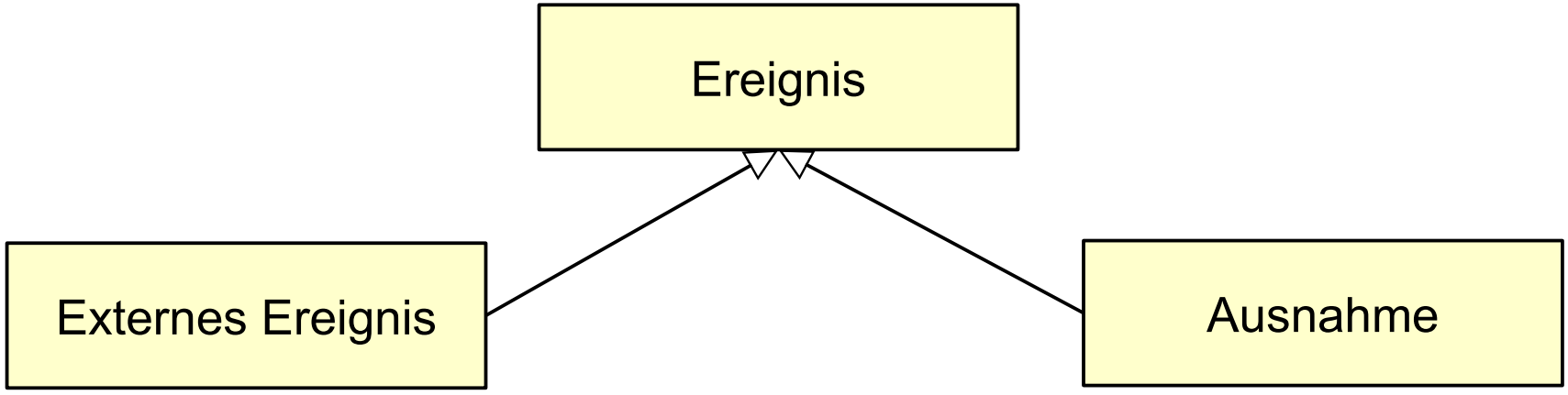
- Ein ***Aufrufsanschluß (call port)*** ist eine Portschnittstelle, die angebotene oder benötigte Operationen enthält, über die es synchron aufgerufen wird oder synchron aufruft
- Ein ***Stromanschluss (stream port)*** ist eine Portschnittstelle, die einen Ein- oder Ausgabestrom enthält, über den kontinuierlich Daten ein (Senke-Muster) und aus fließen (Iterator-Muster)
  - Ein Stromanschluß läßt auf ein aktives Objekt (Prozess) schließen, das den Strom bedient.
  - Daten können einfach oder strukturiert sein (große Objekte, Werte, Formulare, Webseiten)
  - Datentypen, die über den Stromanschluss fließen, stammen aus dem Domänenmodell
- Strom-Ports werden durch ***Konnektoren*** (***Strom-Kanäle, channels, pipes)*** zu Strömen verbunden
  - Es entstehen Pipe-und-Filter-Architekturen (Datenflussarchitekturen)
- Entwurfs- und Implementierungsmodell: Portschnittstellenklasse wird mit Muster Iterator oder Sink realisiert; Kanal mit Muster Channel



Asynchrones Verhalten des Systems

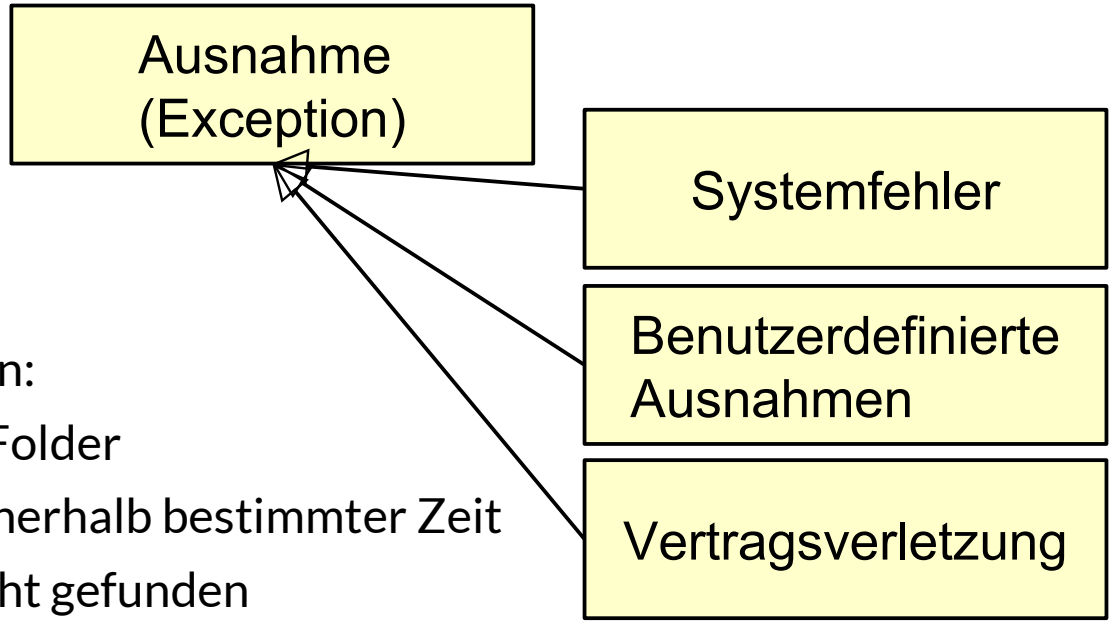
- Im Kontextmodell muß zusätzlich das *asynchrone Verhalten des Systems* spezifiziert werden

- **Externe Ereignisse** – und wie soll das System darauf reagieren?
  - Benutzererzeugte Ereignisse (ButtonPressed, MenuItemSelected,..)
  - Plattform-erzeugte Ereignisse (Platte voll, Power out, ...)
  - Sensorwerte (Temperatur, GPS)
  - Externe Ereignisse können durch Ströme (stream ports) in das System fließen



Ausnahmen im Kontextmodell

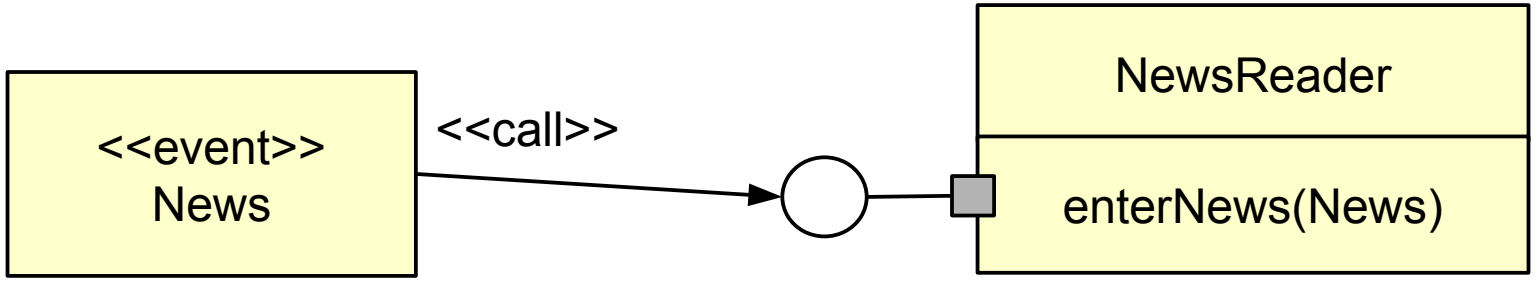
- Eine **Ausnahme (exception)** ist ein internes Ereignis, das vom System selbst ausgelöst wird
  - Systemfehler:
    - Division durch 0
    - Zugriff durch null-Zeiger
    - Platte voll
  - Benutzerdefinierte Ausnahmen:
    - Zuviele Dateien in einem Folder
    - Benutzer reagiert nicht innerhalb bestimmter Zeit
    - Material in Datenbank nicht gefunden
  - Vertragsverletzungen von Methoden:
    - Zeiger null
    - integer-Wert zu gross

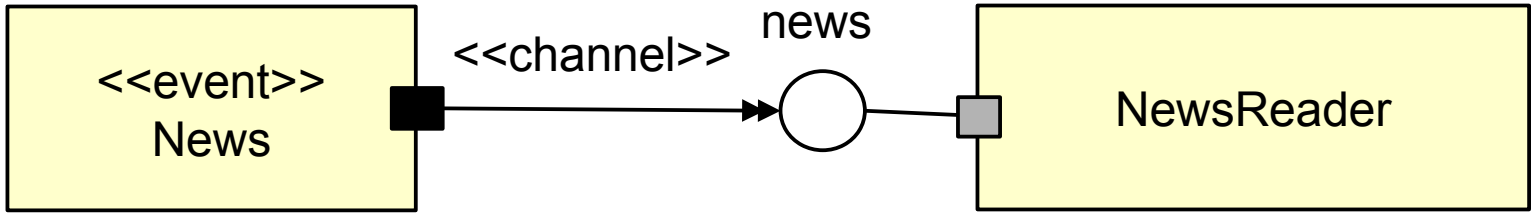


- Ausnahmen des Systems und die Reaktion des Kontextes auf diese gehören ins Kontextmodell!
- Implementierung: Ausnahmen sind als Konzept in Java vorhanden (exception)

Externe Ereignisse gelangen ins System über das Kontextmodell

- Ein Ereignis ist also ein asynchron auftretendes Geschehen, das als Ereignisobjekt repräsentiert und schliesslich mit einem Konnektor in eine Klasse, Komponente, oder das System hineingereicht wird
  - als Parameter einer Methode (call port)
  - als Objekt in einem *Stromanschluss* über einen *Konnektor*





Unterschied Domänen- vs. Kontextmodell

Domänen-Modell .- ***Das Abbild der realen Welt***	**Kontext-Modell -** ***Teile der Technik, die der Kunde sehen muss***
Notation: UML Objekte: Fachgegenstände Klassen: Fachbegriffe Attribute ohne Typen Operationen: ohne Typen, Parameter und Rückgabewerte Assoziationen: partiell, bidirektional i. Allg. ohne Datentypen Aggregationen, Kompositionen Leserichtung, partielle Multiplizitäten Vererbung: Begriffsstruktur Annahme perfekter Technologie Funktionale Essenz Grobe Strukturskizze	Notation: UML Objekte: Softwareeinheiten Klassen: Komponenten, Teile, Rollen, Ports, Interface, Stereotypen aus Profilen Attribute: Klassenattribute, Ports, Ströme Unidirektionale Assoziationen mit voller Multiplizität, Navigation Vererbung: Programmableitung Asynchrones Verhalten: Ereignisse, Ausnahmen Genauere Strukturdefinition in der Top-Level-Architektur: Adapter, Konnektoren

The End – Anhang mit optionalem Material

- Einige Folien sind eine überarbeitete Version von Vorlesungsfolien zur Vorlesung Softwaretechnologie von © Prof. H. Hussmann. Used by permission.

Fragen:

- Was unterscheidet Komponenten und Klassen?
- Warum modelliert man auf den äußeren Schichten des Systems mit Komponenten statt mit Klassen?
- Welche Schnittstellen besitzt ein System neben einer graphischen Benutzerschnittstelle noch?
- Welches “required interface” hat eine Anwendung gegenüber dem Betriebssystem?
- Warum ist ein System ein komplexes Großobjekt? Wie gliedert es sich?
- Erkläre den Zusammenhang zwischen Anschlüssen, Konnektoren und Ereignissen.

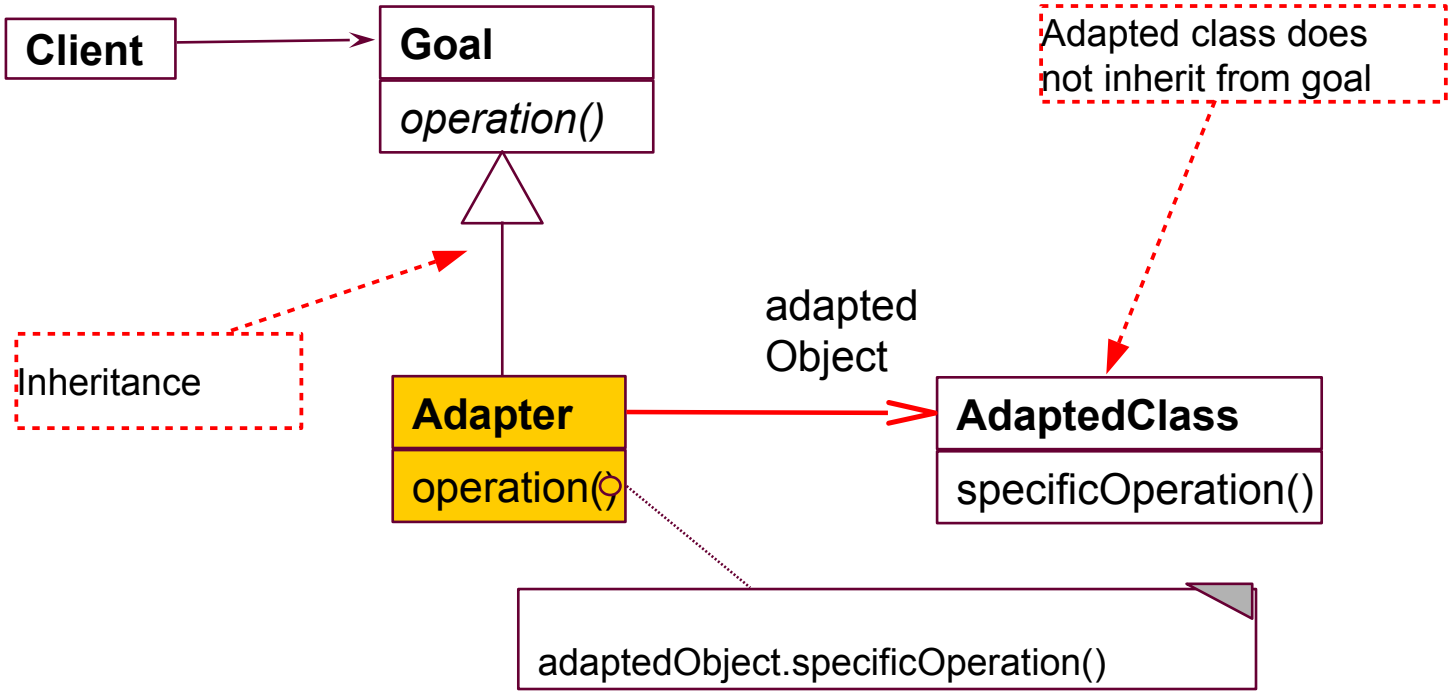
33.A.1 Einsatz von Adapter zum Brücken von Schnittstellen in der Top-Level-Architektur

Aufgabe des Kontextmodells: Verbindung nach außen

- Eine wesentliche Aufgabe des Kontextmodells ist die Verbindung des Systems mit dem Kontext, d.h. der restlichen Welt
- Die Top-Level-Architektur detailliert die Verantwortlichkeiten für die Verbindungen
- Jede neue Verbindung erzeugt eine neue Rolle des Systems (neuer Kontext)
- Da die Schnittstellen der externen Systeme mit denen der internen Komponenten oft nicht zusammenpassen *(architectural mismatch)*, werden Adapter konzipiert, die die Rollen implementieren
  - Dazu kann man das Entwurfsmuster *Adapter* verwenden

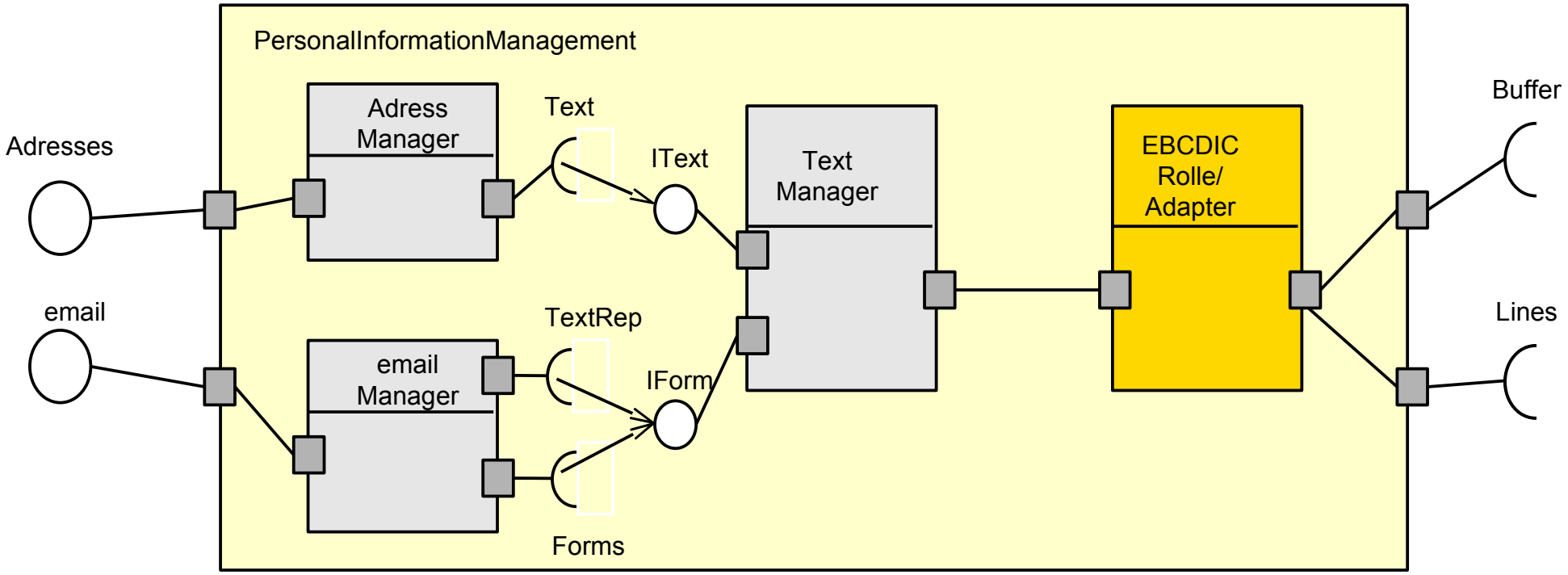
Entwurfsmuster Adapter (Objektadapter) (Wdh.)

- Ein Adapter (Objektadapter) ist ein Objekt, das
  - eine Schnittstelle auf eine andere abbildet
  - ein Protokoll auf ein anderes abbildet
  - Datenformate auf einander abbildet
  - Delegation verwendet



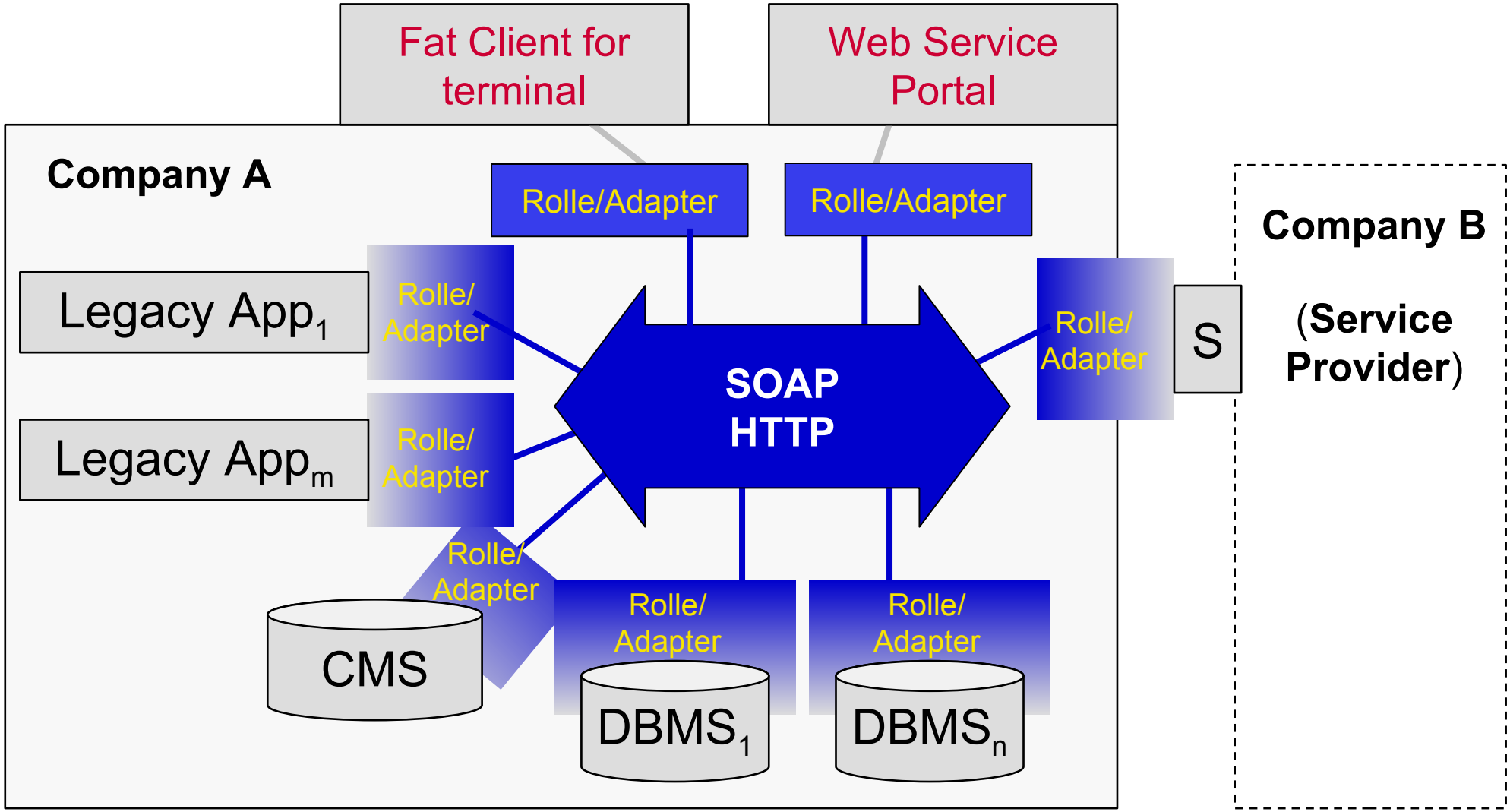
Top-Level-Architektur mit Adaptern

- Die Rollen bzw. Adapter werden nun zwischen den Top-Level-Komponenten und Komponenten der Umgebung eingesetzt
  - z.B. als Daten-Adapter für die unterschiedlichen Character-Codes der unterschiedlichen Maschinen



Beispiel: Web Services mit Entwurfsmuster Konnektor (SOAP/HTTP)

- **Enterprise Application Integration (EAI)** steckt Systeme aus Komponenten mit Hilfe von Rollen bzw. Adaptern zusammen



Beispiel: Bahrami-Profil für Domänenmodell

- Das Bahrami Profil wird hauptsächlich im Domänenmodell eingesetzt
  - Und damit als Typen für die Schnittstellen im Kontextmodell

- Konzept, Begriff (concept) <<concept>>
  - Etwas, worauf sich viele Leute eines Anwendungsbereiches geeinigt haben. Oft angeordnet in Taxonomien oder Ontologien
  - Benutzt im Domänenmodell
- Ereignisklasse (Event) <<event>>
  - Ereignis in der Zeit, extern zum Objekt
- Organisation <<organization>>
  - Verkörpert Wissen über eine Organisationseinheit
- Menschen (People) <<people>>

- Plätze (Places class) <<place>>

Beispiel: Rumbaugh-Profil

- Domänenmodell:
  - Physical class (e.g., Boat) <<physical>>
  - Business class (e.g., Bill) <<business>>
- Anwendungslogik in Kontextmodell und Toplevel-Architektur:
  - Logical class (e.g., Timetable) <<logical>>
  - Application class (e.g., BillingTransaction) <<application>>
  - Behavioral class (e.g., Cancellation) <<behavior>>
- Plattform im Implementierungsmodell:
  - Computer class (e.g., Network) <<computer>>

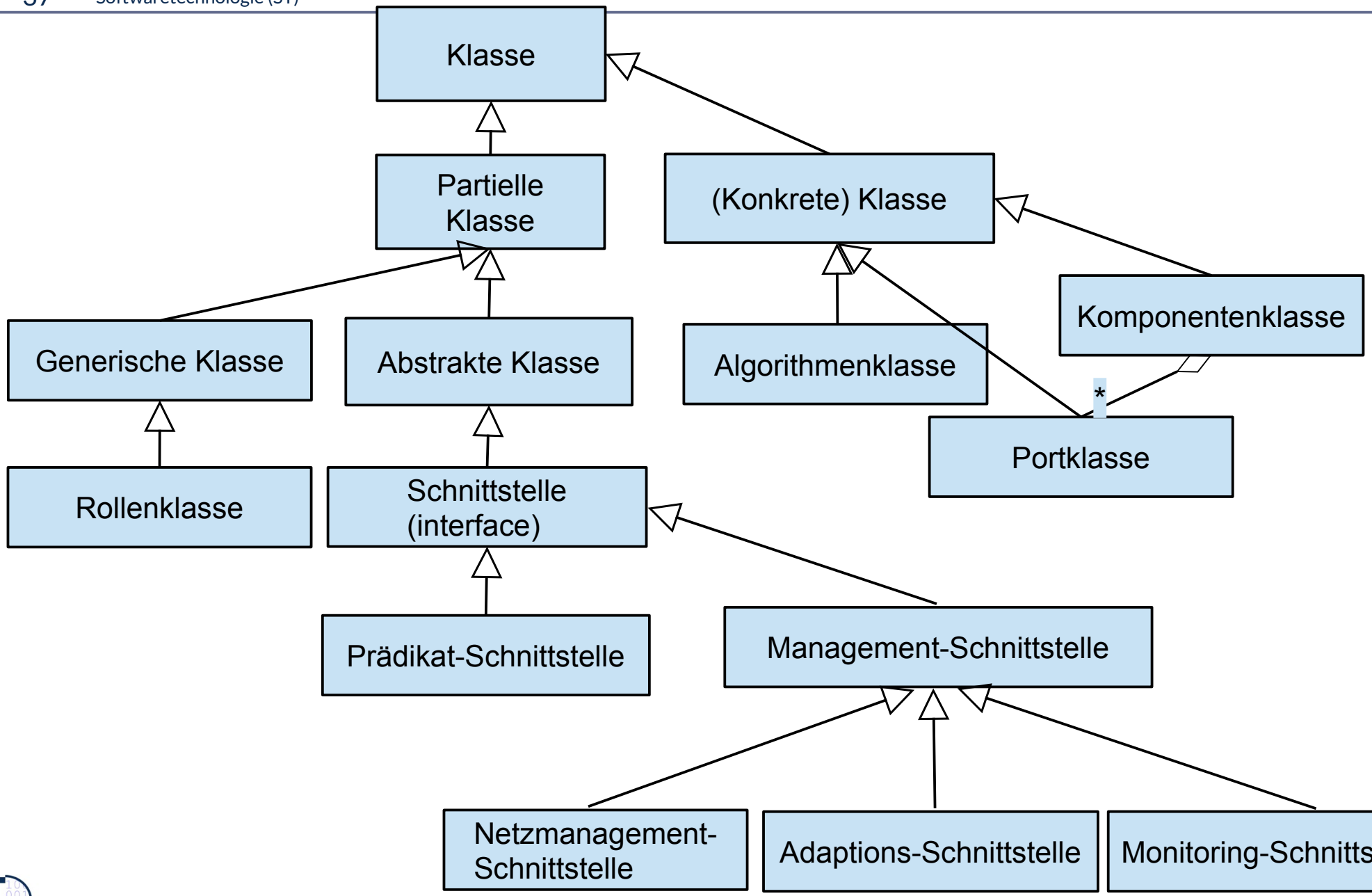
33.A.2 Weitere Arten von Schnittstellen und Klassen

(zur Information)

Weitere Arten von Schnittstellen in komplexen Objekten

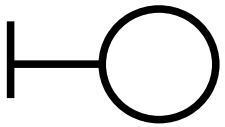
- **Funktionale Schnittstellen** enthalten Funktionen, die direkt auf den Zustand des Objekts zugreifen
  - **Prädikat-Schnittstellen**, die Prädikate auswerten
- **Managementschnittstellen** enthalten Funktionen, die das Objekt und seine Nachbarn verwalten:
  - **Netzmanagement-Funktionen** verändern das Netz
  - **Adaptions-Funktionen** verändern Parameter des Objekts, passen das Objektverhalten auf den Kontext an, optimieren das Objekt, verändern seinen Lebenszyklus
  - **Monitoring-Funktionen** messen bestimmte Parameter des Objekts

Q2: Begriffshierarchie von Klassen (Erweiterung)



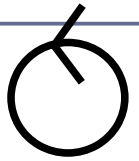
Identifikation von Systemschnittstellenklassen für das Kontextmodell

- **Schnittstellenklassen (boundary, Grenzklassen)** bestehen oft aus
  - Formularen, die dem Benutzer präsentiert werden
  - html-Seiten
  - Abfragen, Meldungen, Sichten auf Daten
  - Schnittstellenklassen zu anderen Systemen, inklusive Adaptern für andere Systeme
  - Strom-Anschlüsse für Datenströme, die ein und aus fließen



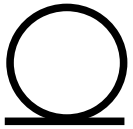
- Oft können Grenzklassen als Portklassen der System-Komponentenklasse dargestellt werden
- Bsp: Terminanwendung:
  - Tabelle der gebuchten Besprechungen
  - Formular, um eine neue Buchung eines Raumes einzutragen
  - Tabelle der Besprechungen pro Mitarbeiter
  - Report über die Auslastung der Besprechungsräume

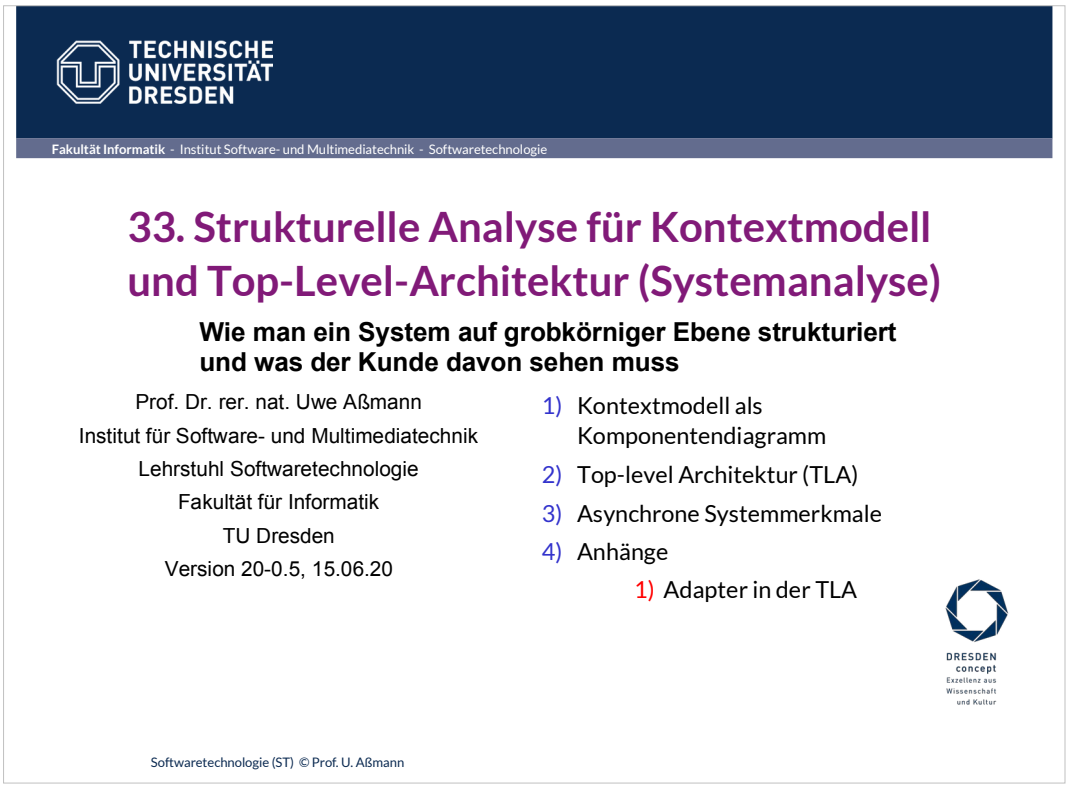
Identifikation von Steuerungs- und Entitätsklassen



- **Steuerungsklassen (control)** enthalten *Anwendungslogik*, d.h. die anwendungsspezifische Funktionalität
  - Steuerungsklassen treten oft in der Top-Level-Architektur auf, wo sie aus Schnittstellenklassen im Kontextmodell entwickelt werden
  - Im Entwurf werden die Steuerungsklassen der Top-Level-Architektur weiter verfeinert

- **Entitätsklassen (data, Datenklassen, Materialien)** werden aus den passiven Klassen eines Domänenmodells bestimmt
  - Entitätsobjekte können physikalisch aus sehr vielen einzelnen Objekten bestehen (physikalische Splitterung)
  - --> Aggregations- und Kompositionsoperation in UML





Der Kunde muss vom System die oberen Schichten sehen (Kontextmodell und Top-Level-Architektur). Das zu definieren ist Teil der *Systemanalyse*.

Literatur

- ST für Einsteiger, Kap. Klassendiagramme
- Zuser, Kap. 7-9
- Störrle Kap. 6 (!!)
- Balzert Kap. 6-7, 9-10

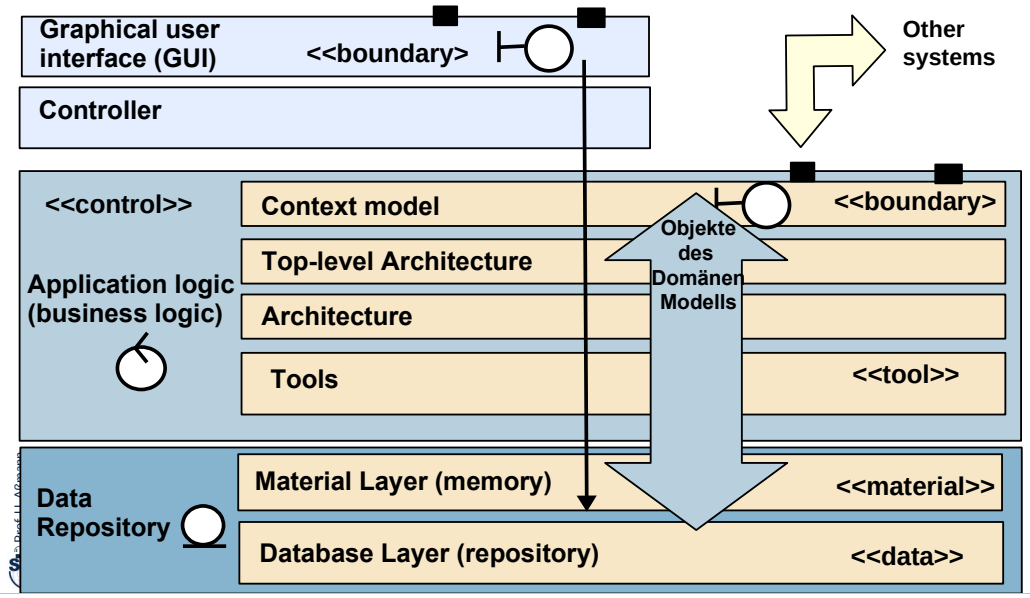
Überblick Teil III: Objektorientierte Analyse (OOA)

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

Warum ist dieses Kapitel wichtig?

- Mit Verbundobjekten und Komponenten können wir jetzt spezifizieren:
  - Kontextmodell des Systems (Schnittstellenmodell, muss der Kunde sehen)
  - Top-Level-Architektur, soweit sie der Kunde sehen muss
  - Schichten einer BCED-Architektur
- Kontextmodelle und Top-Level-Architektur gehören ins Pflichtenheft und bilden einen wesentlichen Teil des Vertrags zwischen Softwarefirma und Kunde.
- Ihre präzise Spezifikation ist essentiell für das Projekt.

Warum komplexe Objekte des Domänenmodells alle Schichten kreuzen, auch CM und TLA



Q5: Schritte der Modellierung in Bezug auf Schichten des Systems

	Schichten	Analyse	Entwurf	Feinentwurf	Implementierung
Benutzungsschnittstelle (Boundary)	GUI		Verfeinerung		
	Controller				
Anwendungslogik (Control)	**Kontextmodell**	Aufstellung aus Domänenmodell; Erarbeitung System-schnittstellen	stabil	Umsetzen auf jUML	stabil
	Top-Level-Architektur	Verfeinerung des Kontextmodells	stabil	Umsetzen auf jUML	stabil
	Architektur		Ausarbeitung Architektur (PSM)	Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML; Serialisierung	Details ausfüllen, Methoden ausprogrammieren
	Tools		Ausarbeitung Tools	Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML; Serialisierung	Details ausfüllen, Methoden ausprogrammieren
Datenhaltung (Database)	Material	Aufstellung aus Domänenmodell		Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML;	Details ausfüllen, Methoden ausprogrammieren

Toll Collect

http://de.wikipedia.org/wiki/Lkw-Maut_in_Deutschland

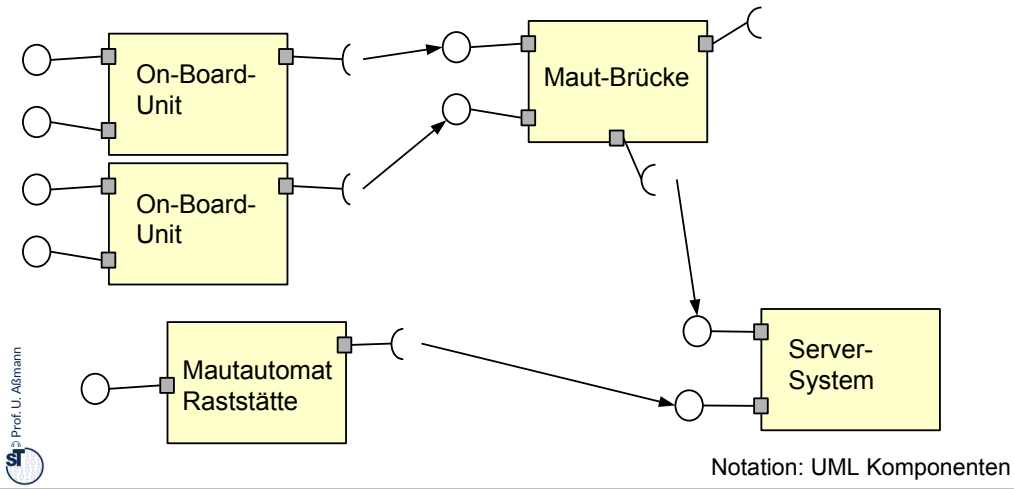
- Auftragnehmer Telekom und Daimler Financial Services
- Schätzung der Einnahmen auf 3 Mrd jährlich
- Inbetriebnahme geplant 31.8.2003, dann 2.11. 2003
  - Tägliche Vertragsstrafe 250k€
- Realisiert 1.1. 2005, vollständig 1.1.2006

- Vertrag mit allen Anlagen und Nebenvereinbarungen aus 17.000 Seiten.
  - Der Kernvertrag, in dem Fragen der Haftung, Vertragsstrafen und Kündigungsfristen geregelt sind, umfasst 190 Seiten.

Warum wurde der Aufwand für den Integrationstest des Toll-Collect-Systems unterschätzt?

Warum war TollCollect verspätet? (Beobachtungen)

- Die Integrationsarbeiten auf oberster Ebene wurden unterschätzt
- Die Interaktionen zwischen den Teilsystemen (Komponenten und Schnittstellen zbw. ihren Kontextmodellen) wurden zu spät auf Skalierbarkeit getestet



Notation: UML Komponenten

Hat man alles in Komponenten spezifiziert, kann man die Komponenten durch *Mockups* ersetzen

und so den Integrationstest in drei Teile aufteilen:

1) Test der Komponenten (unit test)

2) Test des Montagediagramms mit den Mockups (Test des Netzes)

3) Test des Gesamtsystems (eigentliche Integration)

Die Fragen der Systemanalyse im Kundengespräch

- Def.: Die **Systemanalyse**, ein Teil der objektorientierten Analyse, fragt nach den Schnittstellen und der Struktur der obersten Schichten eines Systems.

Die obersten Schichten eines Systems sind meist hierarchisch gestaltet. Typische Fragen der Systemanalyse:

- *Welche **Komponenten** hat das System?*
- *Welche Dienste soll das System liefern? (**Schnittstellen**)*
- *Welche Daten sollen in das System fließen? (Streams, Senken)*
  - *Datentypen, die in und aus dem System fließen, gehören ins Domänenmodell*
- *Wie kommuniziert das System mit anderen Systemen? (**Kanäle, Konnektoren**)*
- *Welche asynchronen Ereignisse treten außerhalb des Systems auf und wie fließen sie in das System? (**Ereigniskanäle**)*
- *Welche Ereignisse treten innerhalb des Systems auf und wie reagiert das System? (**Ausnahmen**, exceptions)*

33.1 Das Kontextmodell

- Das Kontextmodell enthält
  - alle Schnittstellen eines Systems (angeboten, benötigt)
  - die GUI (graphischen Oberfläche, interaktive Schnittstelle) mit ihren Kommandos (Knöpfe, Menüs, etc)

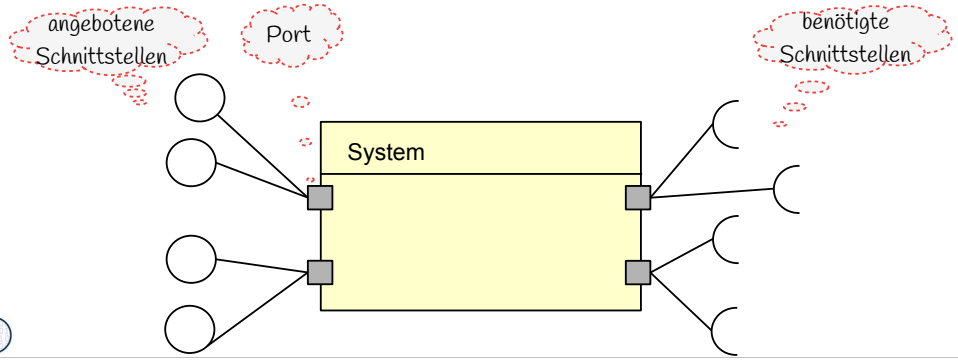
Kontextmodell (Schnittstellenmodell)

- Das System und seine obersten Schichten wird meist als *Klassenkomponente* verstanden
  - Mit angebotenen und benötigten Schnittstellen, mit ports
  - Hierarchisch durch die Ganz/Teile-Relation verfeinert
- Das ***Kontextmodell (Schnittstellenmodell) eines Systems*** enthält alle äusseren Schnittstellen des Systems (Portschnittstellen)
  - Funktionen für alle Anwendungsfälle (oberste Schicht der Anwendungslogik)
  - Ein- und Ausgabekanäle (ports, streams)
  - Benutzerschnittstelle (Eingabeformulare, GUI mit Menüs, Masken und Abfragen)
  - Daten, die in und aus dem System fliessen (Typen aus dem Domänenmodell),
    - Ereignisse, Ausnahmen

Das System wird oft als eine große komplexe Komponente gesehen, dessen Schnittstellen vom Kontextmodell beschrieben werden.

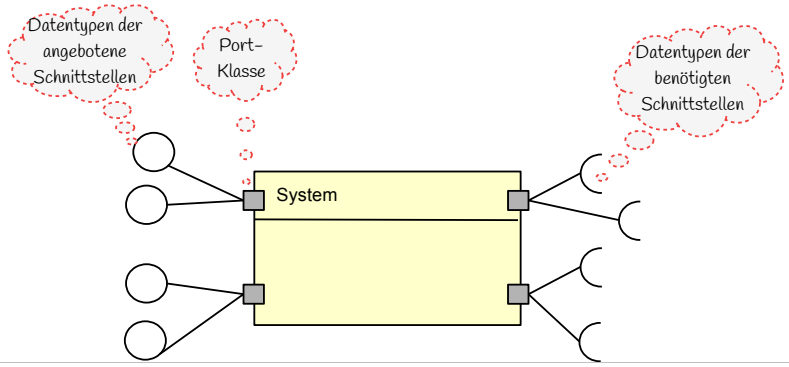
Komponenten im Kontextmodell

- Ein ***Montagediagramm*** ist ein Netz von Komponenten (Störrle)
- Das **Kontextmodell** des Softwaresystems ist die Menge seiner Schnittstellen
  - Das System ist selbst eine Komponente
  - Das Kontextmodells enthält *angebotene* und *benötigte* Schnittstellen:
    - Funktionale (call ports) und Strom-Schnittstellen (stream ports), gerade zu anderen Systemen
    - GUI-Bildschirme, Masken, Formulare
- Reine Klassen genügen nicht immer, denn sie spezifizieren keine benötigten Schnittstellen



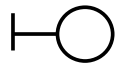
Datentypen des Kontextmodells stammen aus dem Domänenmodell

- Das Domänenmodell stellt die Typen für die technischen Objekte, Prozesse und Funktionalitäten des Kontextmodells zur Verfügung
  - Anschlüsse: Funktionale und strombasierte Schnittstellen (call und stream ports)
  - GUI-Bildschirme, Masken, Formulare
  - Port-Klassen und Schnittstellen sind Grenzklassen (boundary)



Unterscheidung von Systemschnittstellenklassen von anderen Klassen mit dem BCD-Profil

- **Schnittstellen-Klassen** (Portschnittstellen, Portklassen, GUI-Klassen, Grenzklassen, boundary classes)
  - Teil einer Schnittstelle, Benutzerschnittstelle, oder Port-Mixin der System-Komponente
- **Steuerungsklassen** (control classes)
  - Aktive Klasse der Anwendungslogik, die die Ausführung eines Prozesses steuert
  - Mit oder ohne Zustand
- **Material**: Passive Klasse, beschreibt Daten
  - Entitätsklassen (Entity): Beschreibt komposite, persistente Datenobjekte der Domäne
  - Datenklasse (Database): Adapterklasse für eine Entity in der Datenbank
    - Oft sind Entity and Database vereinigt
  - MaterialContainer: Container für Material
- BCD-Architektur: 3-Schichten-Architektur (3-tier architecture)
- BCED-Architektur: 4-Schichten-Architektur (4-tier architecture)



<<boundary>>

<<control>>
<<workflow>>
<<tool>>
<<process>>

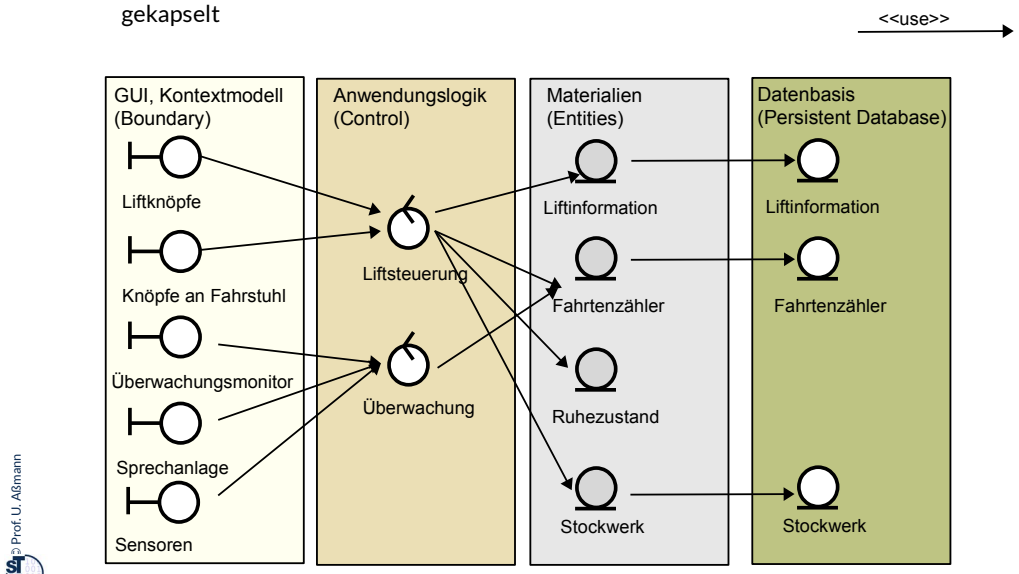
<<material>>
<<entity>>

<<database>>

<<container>>

Bsp: Analyse-Klassenmodell Liftsteuerung im BCED-Stil

- als 4-Schichten-Architektur (4-tier architecture) von **Klassen:** Schichten sind nicht voll gekapselt



Dieses Beispiel bildet ein klassisches BCED-4-Schichtensystem. Der Übersetzer

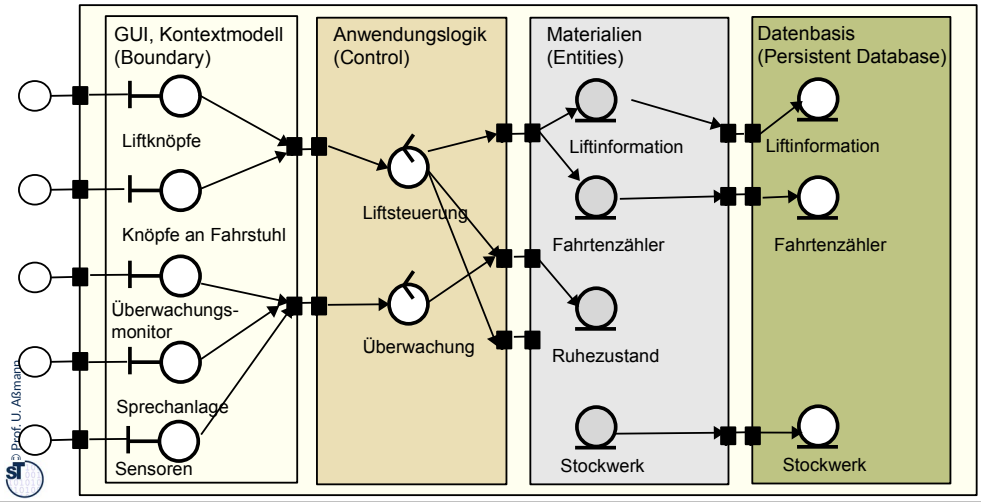
•findet heraus, welche “benötigten Schnittstellen” die von den weiter links stehenden Schichten haben

•und prüft, ob die weiter rechts stehenden Schichten sie liefern.

Will man die Anwendungslogik austauschen, muss man re-compilieren, weil der Übersetzer erneut prüfen muss.

Bsp: Analyse-Montagediagramm: Liftsteuerung im BCED-Stil

- Als 4-Schichtenarchitektur von **Komponenten** → Schichten sind voll gekapselt und besser vorbereitet auf Austausch von Komponenten und Verteilung <<use>>
- Viele Klassen haben *einen* Ein- und Ausgabe-Port; andere teilen sich einen Port



Welche Konnektoren sind Delegationskonnektoren (delegation connectors)?

Warum kann man in dem Montagediagramm die Anwendungslogik-Komponente (Schicht) gut austauschen?

Malen Sie eine Variante des Montagediagramms, bei dem alle Port-Mixins zu sehen sind.

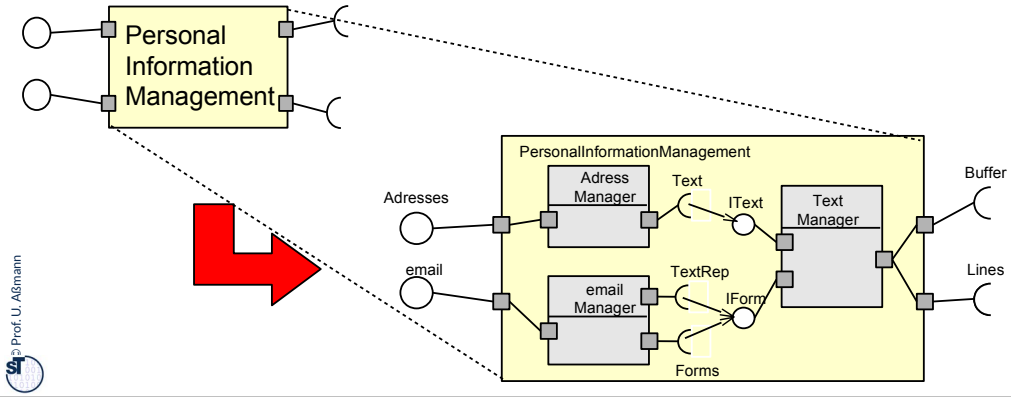
Können Sie auch eine Variante zeichnen, bei dem Lollipops und Sockets zu sehen sind?

33.2 Top-Level-Architektur

Das System ist eine große Komponente, ein Verbundobjekt mit Ports, dessen Schnittstellen vom Kontextmodell beschrieben werden und das intern hierarchisch organisiert ist (Top-Level-Architektur).

Top-Level-Architektur als punktweise Verfeinerung der System-Komponente

- Wird das System als Komponente mit angebotenen und benötigten Schnittstellen betrachtet, kann man es hierarchisch mit der part-of-Relation verfeinern
  - Das Kontextmodell wird als Komponente, d.h., hierarchisch, spezifiziert
  - Die **Top-Level-Architektur** entsteht durch die erste Verfeinerung des Kontextmodells
  - Die inneren Komponentenklassen werden sichtbar



Warum wird die Top-Level-Architektur in der Anforderungsanalyse ermittelt?

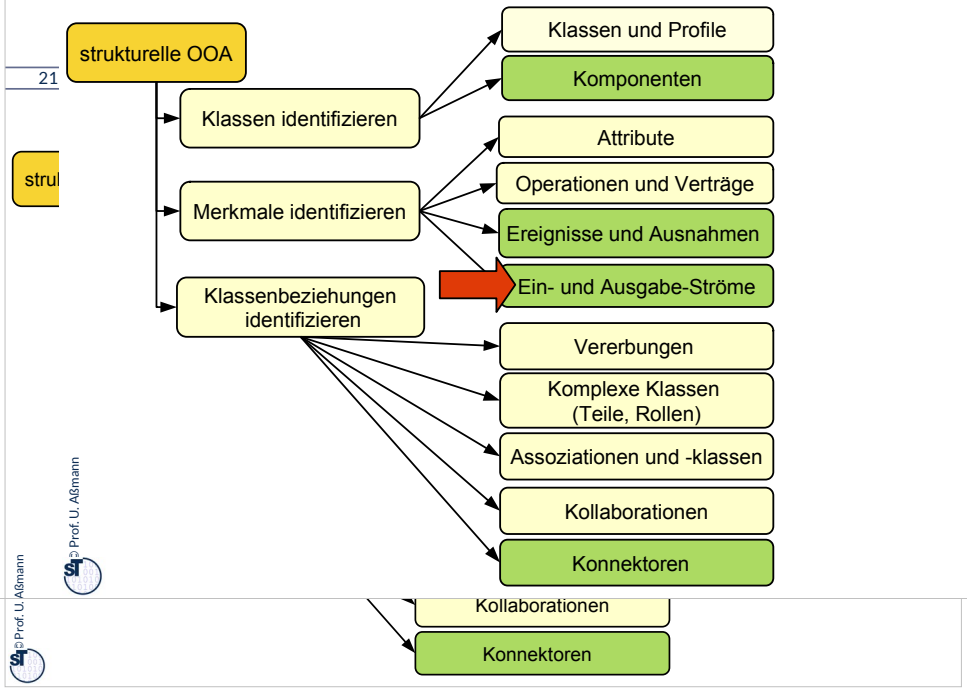
- Man sieht die Antwort an TollCollect: *Die Top-Level-Architektur der Anwendung gehört mit zur Anforderungsspezifikation*, weil
  - die Komponenten der ersten Schicht dem Benutzer sichtbar sind (was sind die Top-Level-Komponenten der TollCollect-Software?)
  - sie oft Aufgaben direkt zugeordnet werden können, die der Benutzer kennt (was sind die Aufgaben und Top-Level-Komponenten einer Groupware?)
  - sie zur Kostenplanung und Abrechnung für das Projekt verwendet werden (Produktstrukturplan, siehe Vorlesung Softwaremanagement)
  - sie dem Manager eine Einteilung von Projektmitarbeitern vereinfachen
- Wenn sie nicht zur Anforderungsspezifikation hinzukommt, ist sie als erstes Dokument im Entwurf auszuarbeiten
  - um die Planung zu vereinfachen

33.3 Asynchrone Systemmerkmale im Kontextmodell

Ereignisse und Ausnahmen (Exceptions)

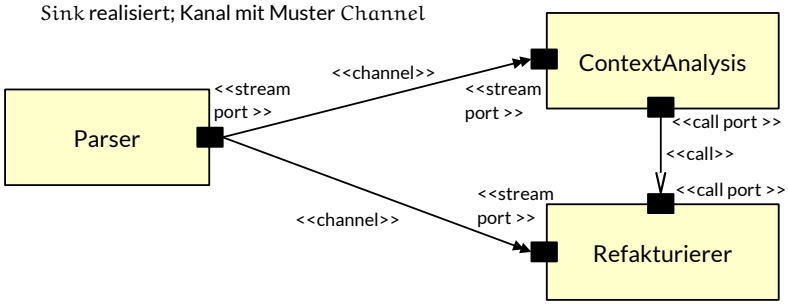
Q6: Schritte der strukturellen, metamodellgetriebenen Analyse

- gelb: Domänenmodell; grün: Kontextmodell, TopLevel-Architektur



Strom-Anschlüsse von Komponentenklassen und Kanal-Konnektoren

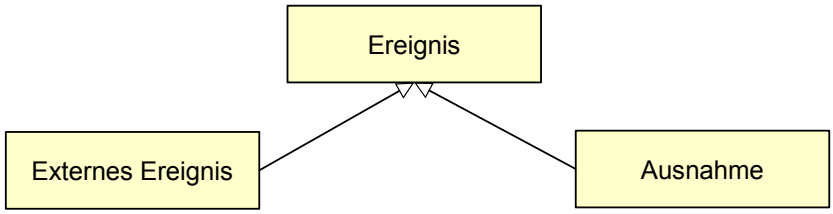
- Ein ***Aufrufsanschluß (call port)*** ist eine Portschnittstelle, die angebotene oder benötigte Operationen enthält, über die es synchron aufgerufen wird oder synchron aufruft
- Ein ***Stromanschluss (stream port)*** ist eine Portschnittstelle, die einen Ein- oder Ausgabestrom enthält, über den kontinuierlich Daten ein (Senke-Muster) und aus fließen (Iterator-Muster)
  - Ein Stromanschluß läßt auf ein aktives Objekt (Prozess) schließen, das den Strom bedient.
  - Daten können einfach oder strukturiert sein (große Objekte, Werte, Formulare, Webseiten)
  - Datentypen, die über den Stromanschluss fließen, stammen aus dem Domänenmodell
- Strom-Ports werden durch ***Konnektoren*** (***Strom-Kanäle, channels, pipes)*** zu Strömen verbunden
  - Es entstehen Pipe-und-Filter-Architekturen (Datenflussarchitekturen)
- Entwurfs- und Implementierungsmodell: Portschnittstellenklasse wird mit Muster Iterator oder Sink realisiert; Kanal mit Muster Channel



Asynchrones Verhalten des Systems

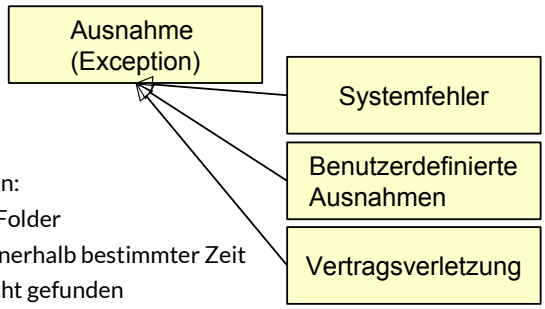
- Im Kontextmodell muß zusätzlich das *asynchrone Verhalten des Systems* spezifiziert werden

- **Externe Ereignisse** – und wie soll das System darauf reagieren?
  - Benutzererzeugte Ereignisse (ButtonPressed, MenuItemSelected,..)
  - Plattform-erzeugte Ereignisse (Platte voll, Power out, ...)
  - Sensorwerte (Temperatur, GPS)
  - Externe Ereignisse können durch Ströme (stream ports) in das System fließen



Ausnahmen im Kontextmodell

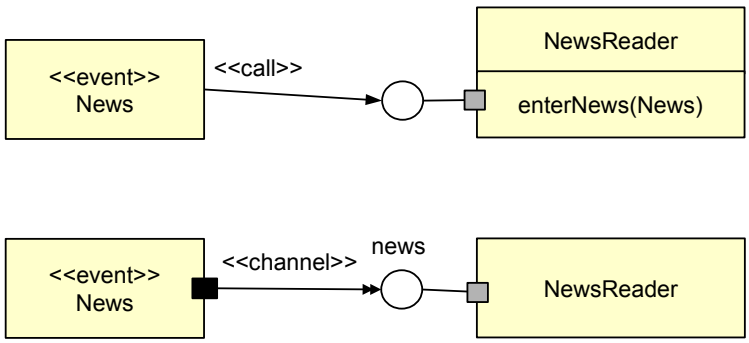
- Eine **Ausnahme (exception)** ist ein internes Ereignis, das vom System selbst ausgelöst wird
  - Systemfehler:
    - Division durch 0
    - Zugriff durch null-Zeiger
    - Platte voll
  - Benutzerdefinierte Ausnahmen:
    - Zuviele Dateien in einem Folder
    - Benutzer reagiert nicht innerhalb bestimmter Zeit
    - Material in Datenbank nicht gefunden
  - Vertragsverletzungen von Methoden:
    - Zeiger null
    - integer-Wert zu gross



- Ausnahmen des Systems und die Reaktion des Kontextes auf diese gehören ins Kontextmodell!
- Implementierung: Ausnahmen sind als Konzept in Java vorhanden (exception)

Externe Ereignisse gelangen ins System über das Kontextmodell

- Ein Ereignis ist also ein asynchron auftretendes Geschehen, das als Ereignisobjekt repräsentiert und schliesslich mit einem Konnektor in eine Klasse, Komponente, oder das System hineingereicht wird
  - als Parameter einer Methode (call port)
  - als Objekt in einem *Stromanschluss* über einen *Konnektor*



Unterschied Domänen- vs. Kontextmodell

Domänen-Modell .- ***Das Abbild der realen Welt***	**Kontext-Modell -** ***Teile der Technik, die der Kunde sehen muss***
Notation: UML Objekte: Fachgegenstände Klassen: Fachbegriffe Attribute ohne Typen Operationen: ohne Typen, Parameter und Rückgabewerte Assoziationen: partiell, bidirektional i. Allg. ohne Datentypen Aggregationen, Kompositionen Leserichtung, partielle Multiplizitäten Vererbung: Begriffsstruktur Annahme perfekter Technologie Funktionale Essenz Grobe Strukturskizze	Notation: UML Objekte: Softwareeinheiten Klassen: Komponenten, Teile, Rollen, Ports, Interface, Stereotypen aus Profilen Attribute: Klassenattribute, Ports, Ströme Unidirektionale Assoziationen mit voller Multiplizität, Navigation Vererbung: Programmableitung Asynchrones Verhalten: Ereignisse, Ausnahmen Genauere Strukturdefinition in der Top-Level-Architektur: Adapter, Konnektoren

Hussmann

The End – Anhang mit optionalem Material

- Einige Folien sind eine überarbeitete Version von Vorlesungsfolien zur Vorlesung Softwaretechnologie von © Prof. H. Hussmann. Used by permission.

Fragen:

- Was unterscheidet Komponenten und Klassen?
- Warum modelliert man auf den äußeren Schichten des Systems mit Komponenten statt mit Klassen?
- Welche Schnittstellen besitzt ein System neben einer graphischen Benutzerschnittstelle noch?
- Welches “required interface” hat eine Anwendung gegenüber dem Betriebssystem?
- Warum ist ein System ein komplexes Großobjekt? Wie gliedert es sich?
- Erkläre den Zusammenhang zwischen Anschlüssen, Konnektoren und Ereignissen.

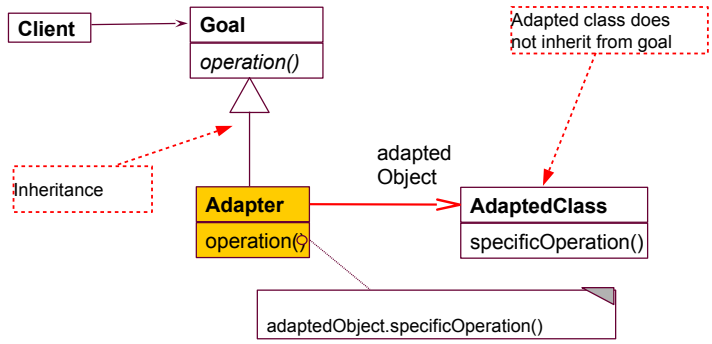
33.A.1 Einsatz von Adapter zum Brücken von Schnittstellen in der Top-Level-Architektur

Aufgabe des Kontextmodells: Verbindung nach außen

- Eine wesentliche Aufgabe des Kontextmodells ist die Verbindung des Systems mit dem Kontext, d.h. der restlichen Welt
- Die Top-Level-Architektur detailliert die Verantwortlichkeiten für die Verbindungen
- Jede neue Verbindung erzeugt eine neue Rolle des Systems (neuer Kontext)
- Da die Schnittstellen der externen Systeme mit denen der internen Komponenten oft nicht zusammenpassen *(architectural mismatch)*, werden Adapter konzipiert, die die Rollen implementieren
  - Dazu kann man das Entwurfsmuster *Adapter* verwenden

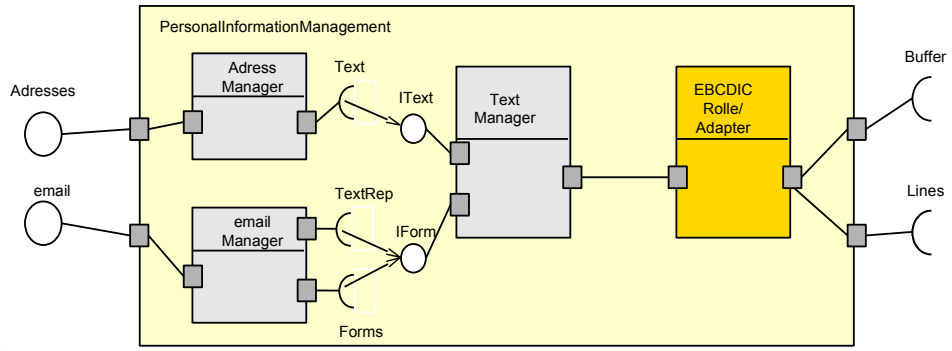
Entwurfsmuster Adapter (Objektadapter) (Wdh.)

- Ein Adapter (Objektadapter) ist ein Objekt, das
  - eine Schnittstelle auf eine andere abbildet
  - ein Protokoll auf ein anderes abbildet
  - Datenformate auf einander abbildet
  - Delegation verwendet



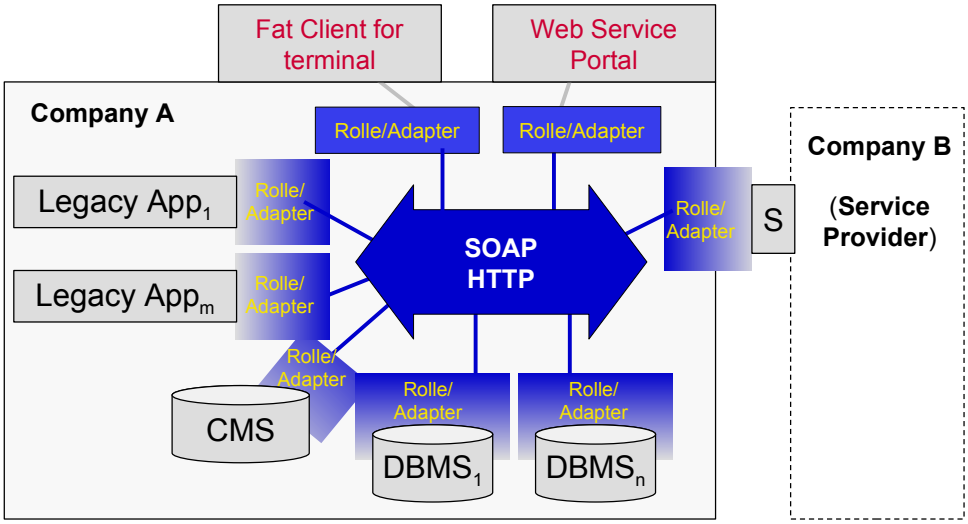
Top-Level-Architektur mit Adaptern

- Die Rollen bzw. Adapter werden nun zwischen den Top-Level-Komponenten und Komponenten der Umgebung eingesetzt
  - z.B. als Daten-Adapter für die unterschiedlichen Character-Codes der unterschiedlichen Maschinen



Beispiel: Web Services mit Entwurfsmuster Konnektor (SOAP/HTTP)

- **Enterprise Application Integration (EAI)** steckt Systeme aus Komponenten mit Hilfe von Rollen bzw. Adaptern zusammen



Beispiel: Bahrami-Profil für Domänenmodell

- Das Bahrami Profil wird hauptsächlich im Domänenmodell eingesetzt
  - Und damit als Typen für die Schnittstellen im Kontextmodell

- Konzept, Begriff (concept)
  - Etwas, worauf sich viele Leute eines Anwendungsbereiches geeinigt haben. Oft angeordnet in Taxonomien oder Ontologien
  - Benutzt im Domänenmodell
- Ereignisklasse (Event)
  - Ereignis in der Zeit, extern zum Objekt
- Organisation
  - Verkörpert Wissen über eine Organisationseinheit
- Menschen (People)

- Plätze (Places class)

<<concept>>

<<event>>

<<organization>>

<<people>>

<<place>>

Beispiel: Rumbaugh-Profil

- Domänenmodell:
  - Physical class (e.g., Boat) <<physical>>
  - Business class (e.g., Bill) <<business>>
- Anwendungslogik in Kontextmodell und Toplevel-Architektur:
  - Logical class (e.g., Timetable) <<logical>>
  - Application class (e.g., BillingTransaction) <<application>>
  - Behavioral class (e.g., Cancellation) <<behavior>>
- Plattform im Implementierungsmodell:
  - Computer class (e.g., Network) <<computer>>

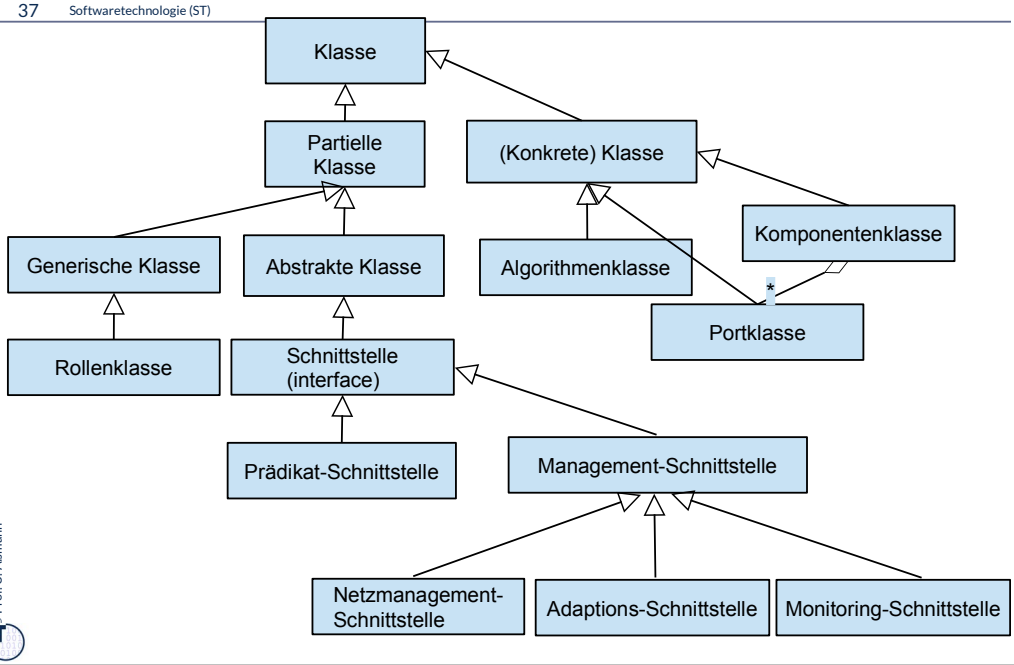
33.A.2 Weitere Arten von Schnittstellen und Klassen

(zur Information)

Weitere Arten von Schnittstellen in komplexen Objekten

- **Funktionale Schnittstellen** enthalten Funktionen, die direkt auf den Zustand des Objekts zugreifen
  - **Prädikat-Schnittstellen**, die Prädikate auswerten
- **Managementschnittstellen** enthalten Funktionen, die das Objekt und seine Nachbarn verwalten:
  - **Netzmanagement-Funktionen** verändern das Netz
  - **Adaptions-Funktionen** verändern Parameter des Objekts, passen das Objektverhalten auf den Kontext an, optimieren das Objekt, verändern seinen Lebenszyklus
  - **Monitoring-Funktionen** messen bestimmte Parameter des Objekts

Q2: Begriffshierarchie von Klassen (Erweiterung)



Identifikation von Systemschnittstellenklassen für das Kontextmodell

- **Schnittstellenklassen (boundary, Grenzklassen)** bestehen oft aus
  - Formularen, die dem Benutzer präsentiert werden
  - html-Seiten
  - Abfragen, Meldungen, Sichten auf Daten
  - Schnittstellenklassen zu anderen Systemen, inklusive Adaptern für andere Systeme
  - Strom-Anschlüsse für Datenströme, die ein und aus fließen

- Oft können Grenzklassen als Portklassen der System-Komponentenklasse dargestellt werden
- Bsp: Terminanwendung:
  - Tabelle der gebuchten Besprechungen
  - Formular, um eine neue Buchung eines Raumes einzutragen
  - Tabelle der Besprechungen pro Mitarbeiter
  - Report über die Auslastung der Besprechungsräume

Identifikation von Steuerungs- und Entitätsklassen

- **Steuerungsklassen (control)** enthalten *Anwendungslogik*, d.h. die anwendungsspezifische Funktionalität
  - Steuerungsklassen treten oft in der Top-Level-Architektur auf, wo sie aus Schnittstellenklassen im Kontextmodell entwickelt werden
  - Im Entwurf werden die Steuerungsklassen der Top-Level-Architektur weiter verfeinert

- **Entitätsklassen (data, Datenklassen, Materialien)** werden aus den passiven Klassen eines Domänenmodells bestimmt
  - Entitätsobjekte können physikalisch aus sehr vielen einzelnen Objekten bestehen (physikalische Splitterung)
  - --> Aggregations- und Kompositionsoperation in UML