33. Strukturelle System-Modellierung (Kontextmodell und Top-Level-Architektur)

Mit komplexen Objekten und UML-Komponenten

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-1.0, 14.06.13

1) Kontextmodell
2) Top-level Architektur
3) Asynchrone Systemmerkmale

Obligatorische Literatur

- 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 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 (Verhaltensmodell)
   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

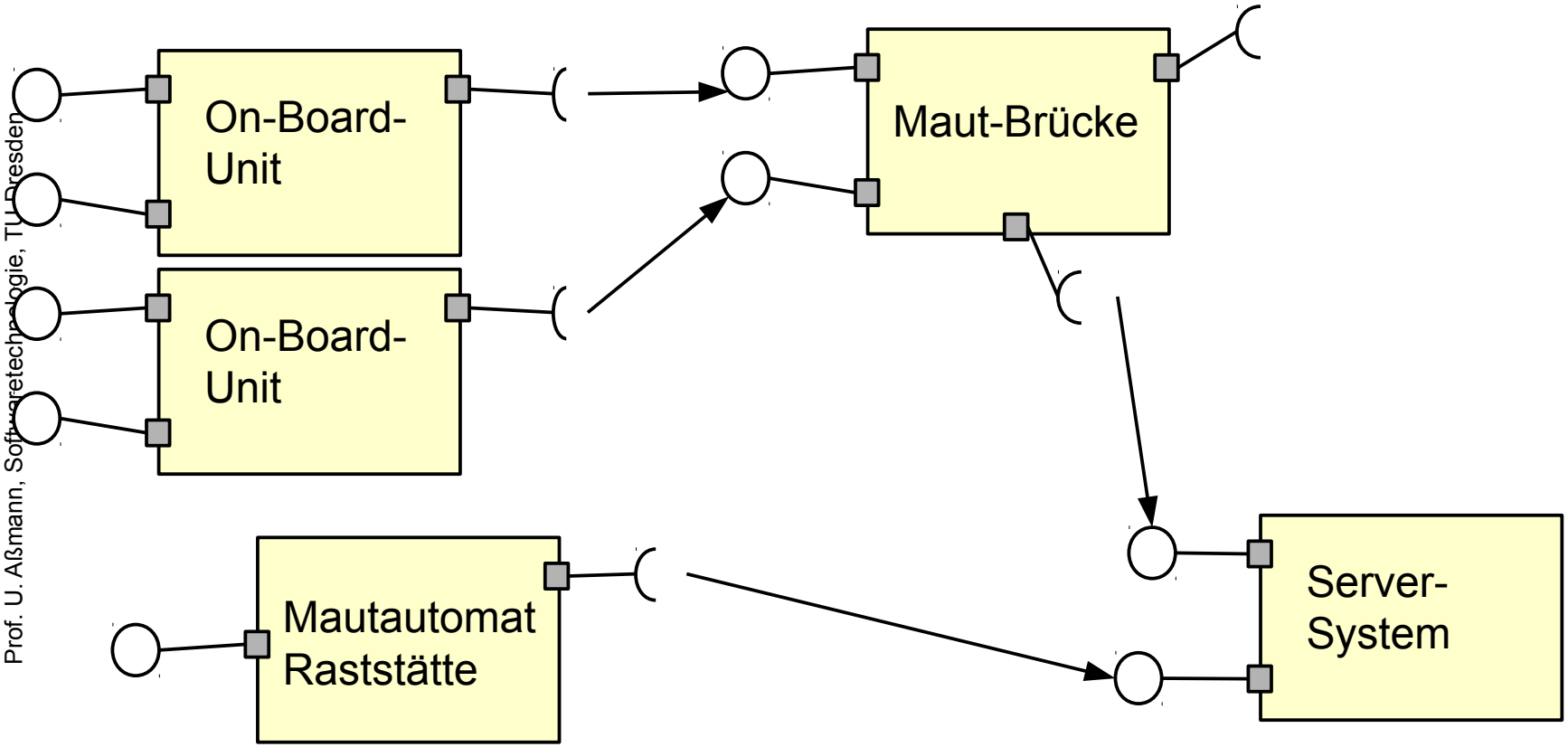
Toll Collect

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

- Auftragnehmer Telekom und Daimler Financial Services
- Schätzung der Einnahmen auf 3 Mrd jährlich
- Inbetriebnahme 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 Kündigungsfriste geregelt sind, umfasst 190 Seiten.

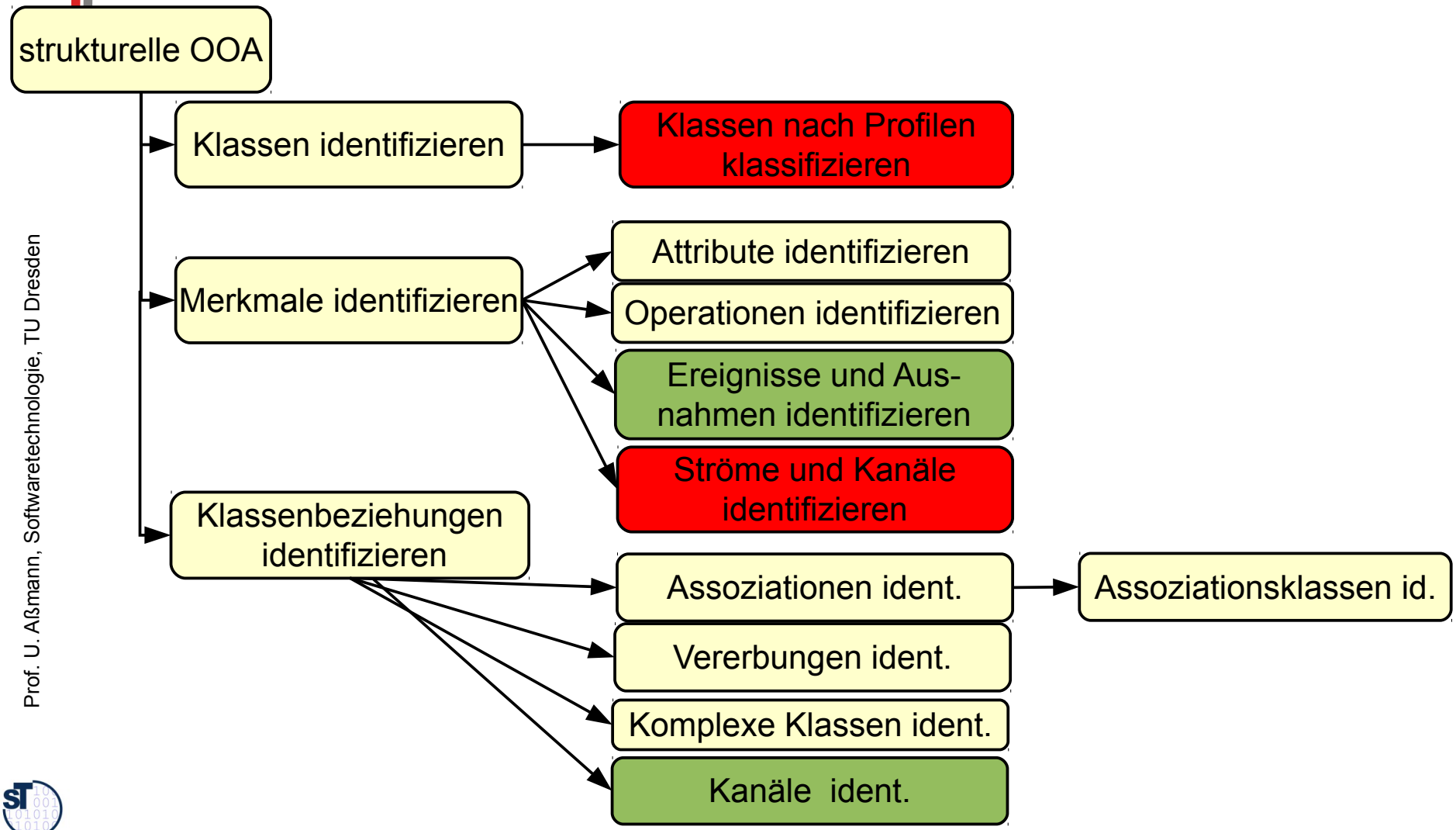
Warum verspätet? (Beobachtungen)

- Die Integrationsarbeiten auf oberster Ebene wurden unterschätzt
- Die Interaktionen zwischen den Systemen (ihren Kontextmodellen) wurden zu spät Skalierbarkeit getestet



Schritte der strukturellen, metamodellgetriebenen Analyse

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



33.1 Kontextmodell

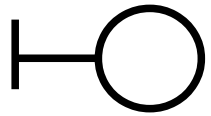
Kontextmodell

- Das *Kontextmodell* enthält die äusseren Schnittstellen des Systems
  - Funktionen für alle Anwendungsfälle
  - Ein- und Ausgabekanäle (channels mit ports, streams)
  - Eingabeformulare (GUI mit Masken und Abfragen)
- sowie
  - Daten, die in und aus dem System fliessen, mit Typen aus dem Domänenmodell typisiert
    - Ereignisse, Ausnahmen
- Es wird meist mit hierarchischen Klassen (UML-Komponenten) notiert
  - Mit angebotenen und benötigten Schnittstellen, mit ports
  - Hierarchisch durch die Ganz/Teile-Relation verfeinert

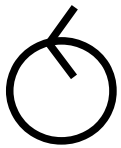
Das System ist ein großes komplexes Objekt,
dessen Schnittstellen vom Kontextmodell beschrieben werden.

Unterscheidung von Systemschnittstellenklassen mit dem BCD-Profil

- GUI-Klassen (Schnittstellen-, Grenzklassen, boundary classes)
  - Teil einer Benutzerschnittstelle
- Steuerungsklassen (control class)
  - Aktive Klasse, 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 heisst auch 3-Schichten-Architektur (3-tier architecture)



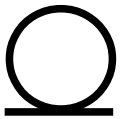
<<boundary>>



<<control>>

<<tool>>

<<process>>



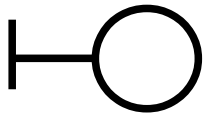
<<entity>>

<<database>>

<<container>>

Identifikation von Systemschnittstellenklassen für das Kontextmodell

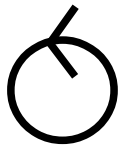
- **Schnittstellenklassen (boundary, Grenzklassen)** bestehen oft aus
  - Formularen, die dem Benutzer präsentiert werden
  - html-Seiten
  - Abfragen
  - Meldungen
  - Sichten auf Daten
  - Schnittstellenklassen zu anderen Systemen, inklusive Adaptern für andere Systeme
  - Strom-Anschlüsse für Datenströme,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



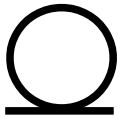
- Oft können Grenzklassen als Portklassen der Systemkomponente dargestellt werden
- Bsp: Terminanwendung:
  - Tabelle der gebuchten Besprechungen
  - Formular, um eine neue Buchung eines Raumes einzutragen
  - Tabelle der Besprechungen pro Mitarbeiter
  - Report über die Auslastung der Besprechungsräume

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

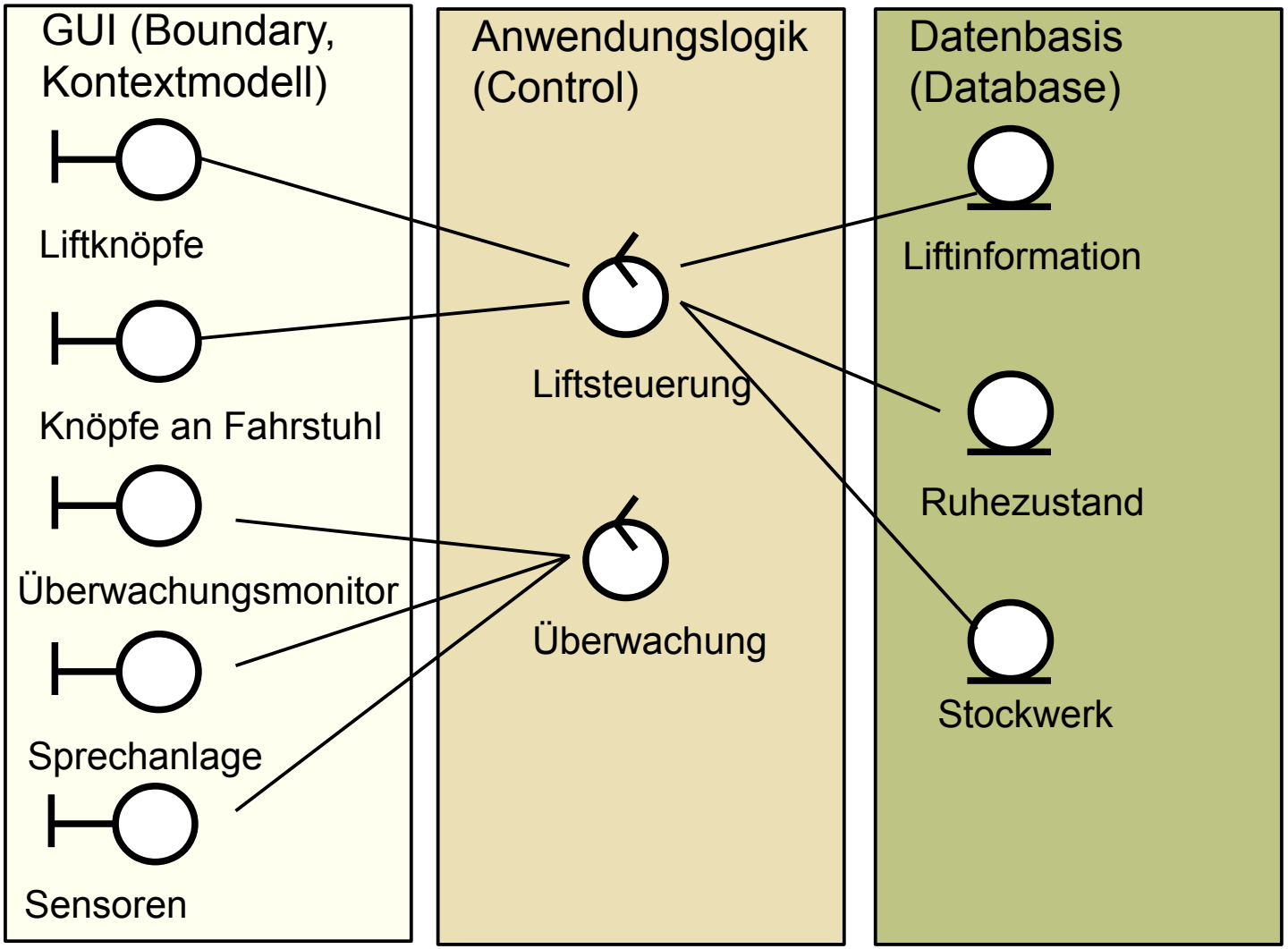


- **Entitätsklassen (data, Datenklassen, Materialien)** werden aus den passiven Klassen eines Domänenmodells bestimmt
  - Entitätsobjekte können physikalisch aus sehr vielen einzelnen Objekten bestehen (physikalische Splitterung)
  - --> Aggregations- und Kompositionsoperation in UML



Bsp: Analyse-Klassenmodell Liftsteuerung im BCD-Stil

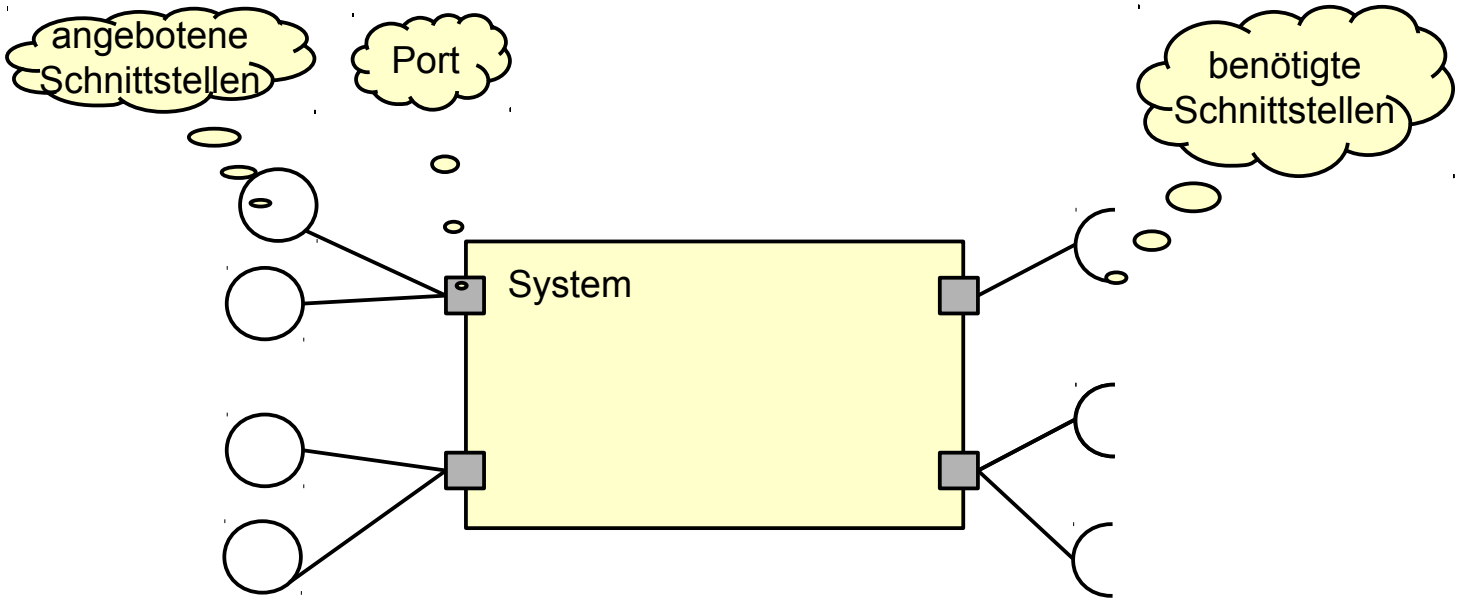
- als 3-Schichten-Architektur (3-tier architecture)



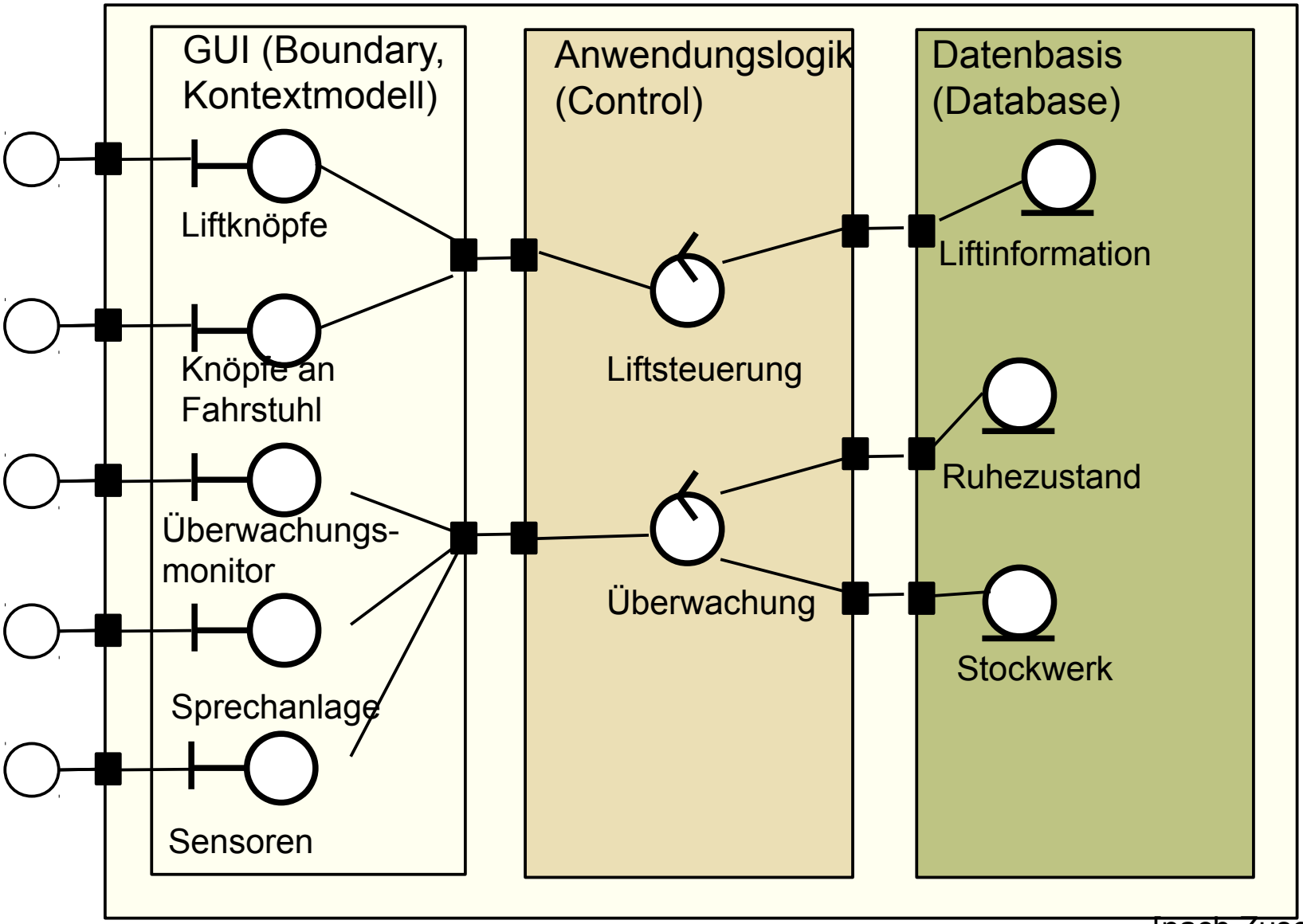
[Zuser]

Komponenten und Typen im Domänenmodell und Kontextmodell

- Das Kontextmodell wird meist als hierarchische Klasse (UML-Komponente) spezifiziert (Störrle: Montagediagramm)
- 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)



Bsp: Analyse-Modell Liftsteuerung mit Kontextmodell als UML-Komponente



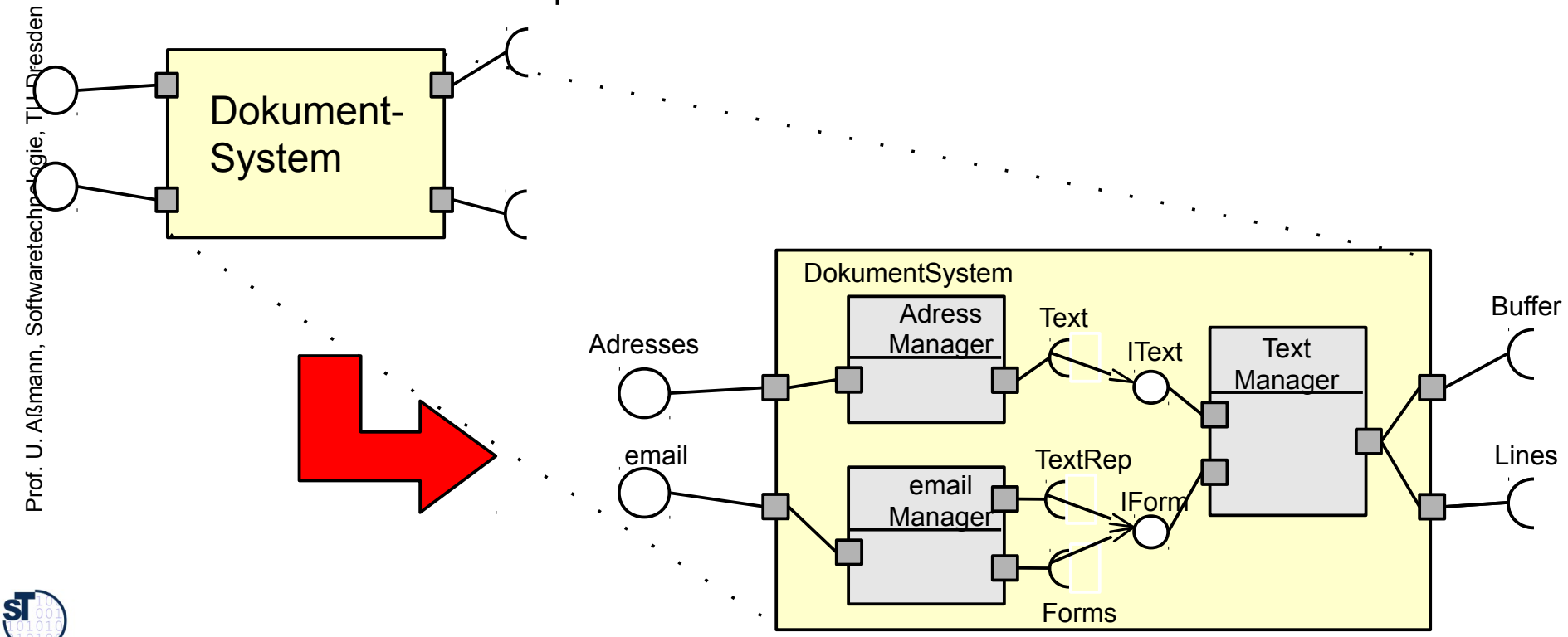
[nach Züser]

33.2 Top-Level-Architektur

Das System ist ein großes komplexes Objekt,
dessen Schnittstellen vom Kontextmodell beschrieben werden
und das hierarchisch organisiert ist (Top-Level-Architektur).

Top-Level-Architektur

- Wird das System als Großobjekt mit angebotenen und benötigten Schnittstellen betrachtet, kann man es hierarchisch mit part-of verfeinern
  - Das Kontextmodell wird als UML-Komponente, d.h., hierarchische Klasse, 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

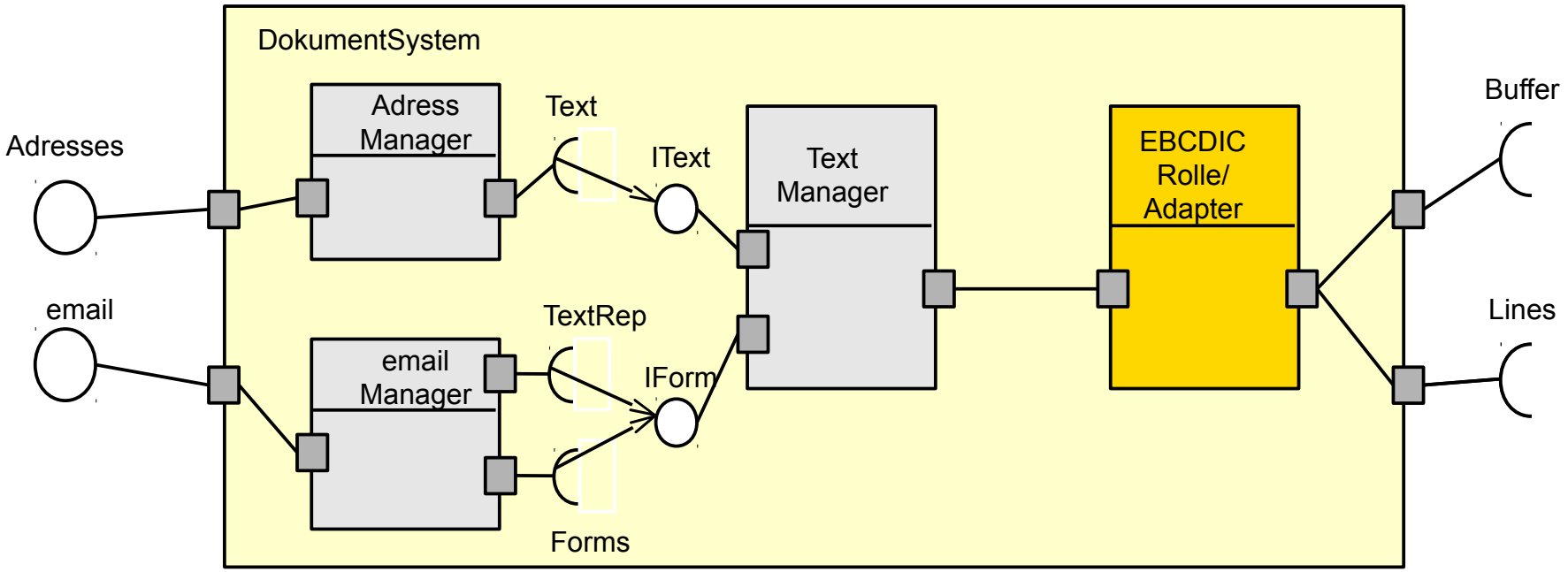
33.2.1 Adapter zum Brücken von Schnittstellen in der Top-Level-Architektur

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

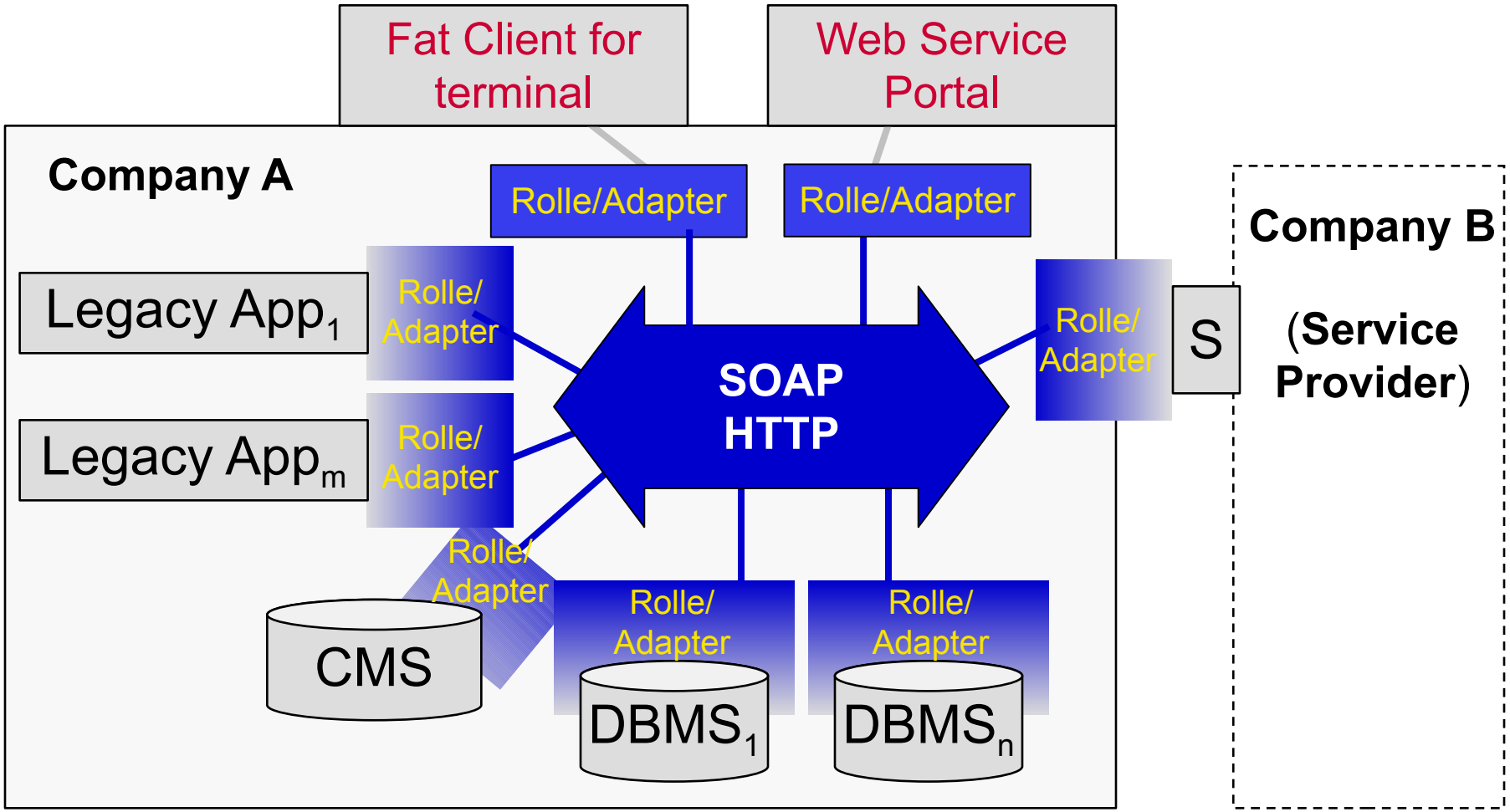
Top-Level-Architektur mit Adaptern

- 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 komplexen Konnektoren SOAP/HTTP

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



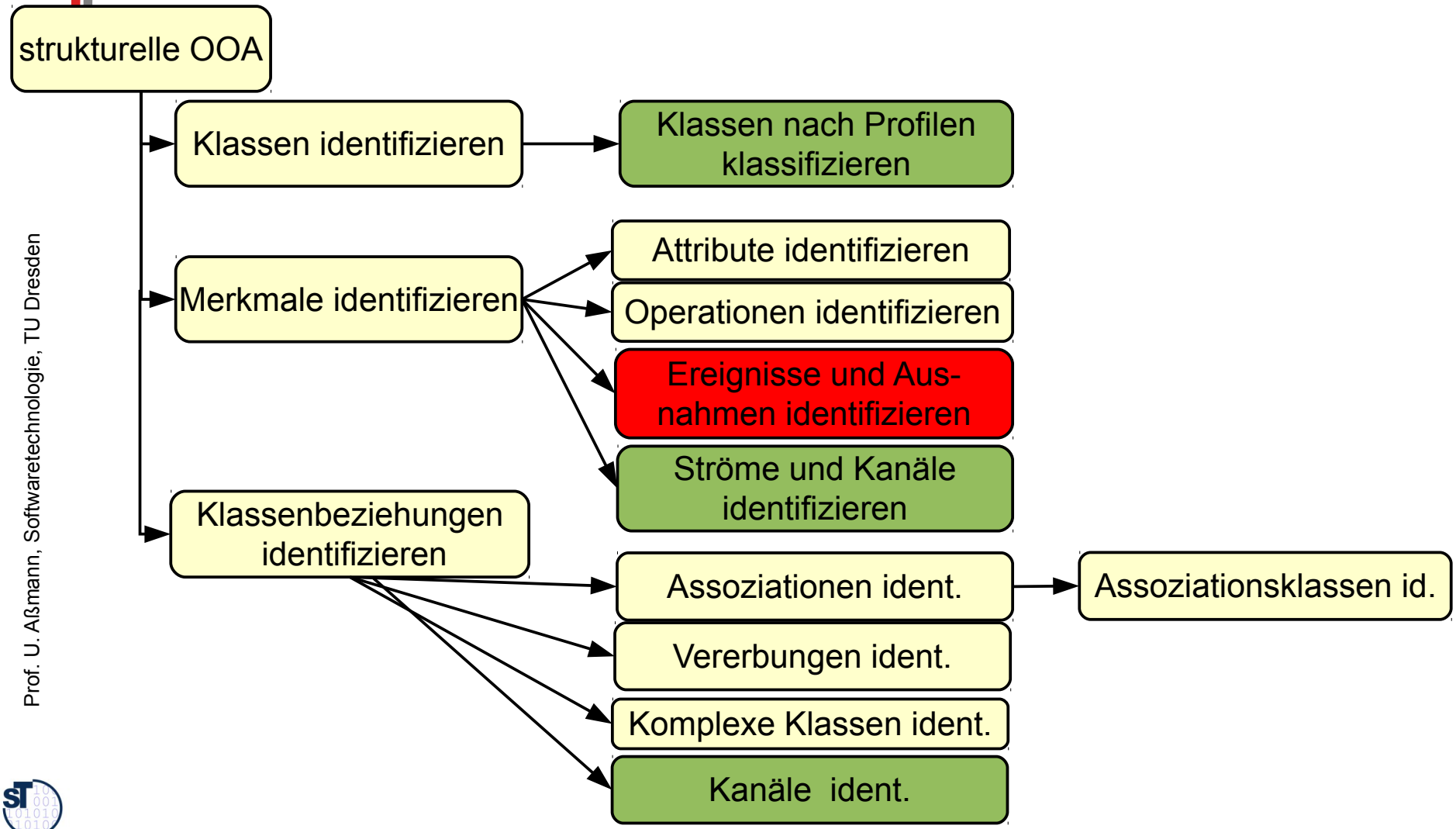
33.3 Asynchrone Systemmerkmale im Kontextmodell

(zur Information)

Ereignisse und Ausnahmen (Exceptions)

Schritte der strukturellen, datengetriebenen Analyse

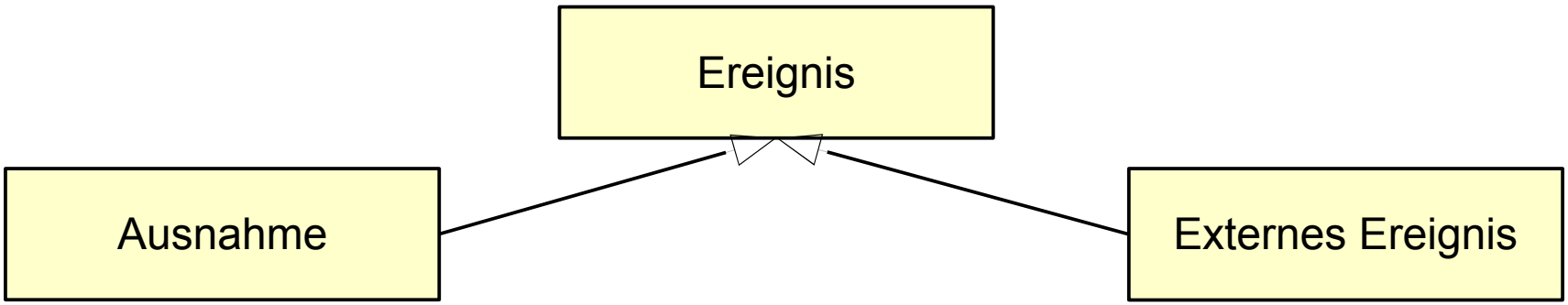
- gelb: Domänenmodell; grün: Kontextmodell, TL-Architektur



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
  - Plattform-erzeugte Ereignisse (Platte voll, Power out, ...)
  - Sensorwerte
  - Externe Ereignisse können durch Ströme (stream ports) in das System fließen

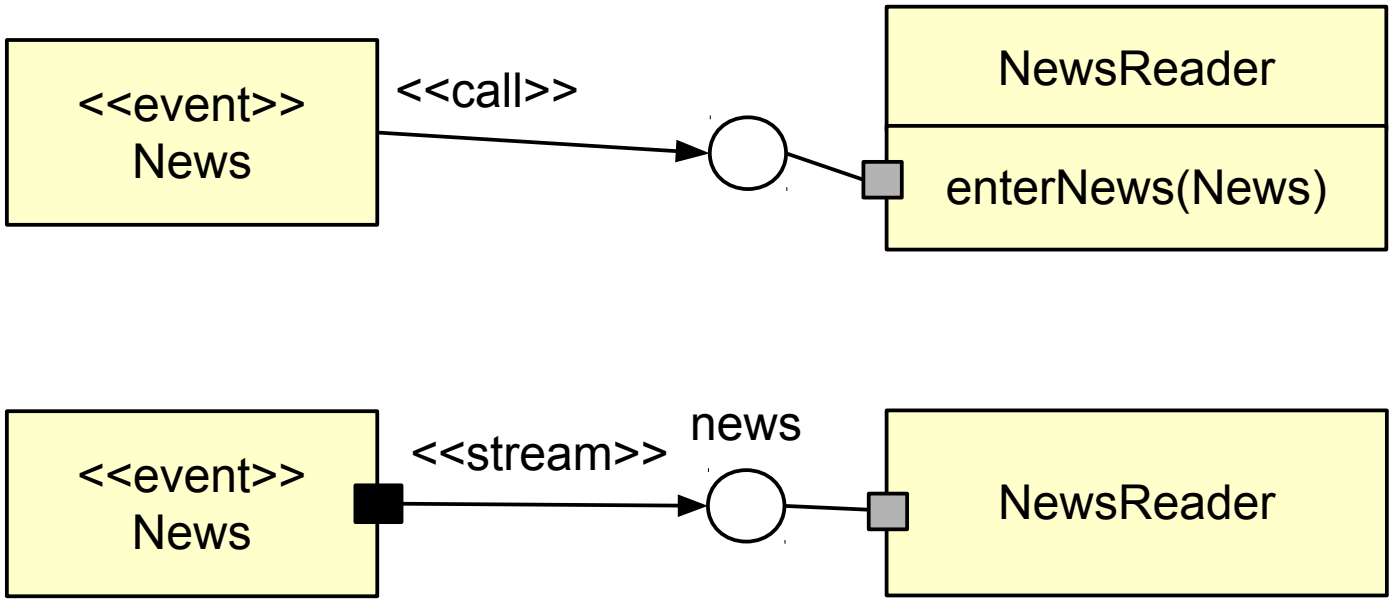


Ausnahmen im Kontextmodell

- Eine **Ausnahme (exception)** ist ein Ereignis, das vom System selbst ausgelöst wird
  - Systemfehler:
    - Division durch 0
    - Zugriff durch null-Zeiger, anschliessender Absturz
    - 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

Ereignisse gelangen ins System über das Kontextmodell

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



Unterschied Domänen- vs. Kontextmodell

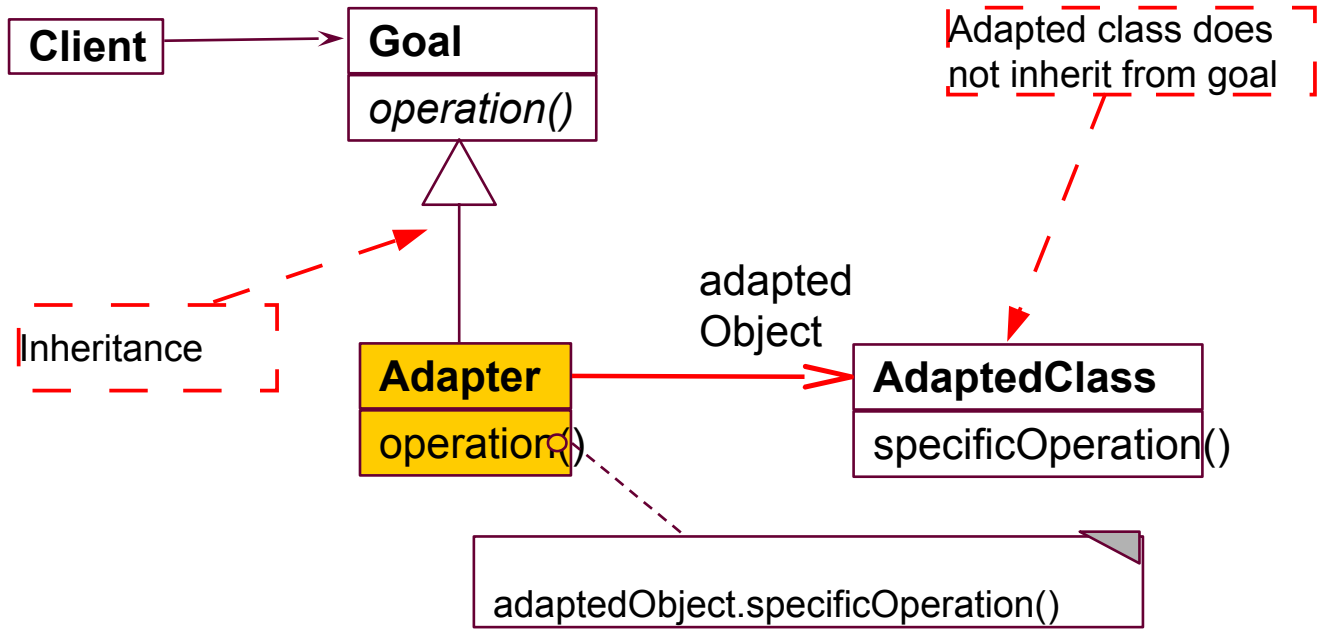
Domänen-Modell	**Kontext-Modell - *Teile der Technik, was der Kunde sehen muss***
Notation: UML Objekte: Fachgegenstände Klassen: Fachbegriffe Attribute ohne Typen Operationen: ohne Typen, Parameter und Rückgabewerte Assoziationen: partiell, bidirektional i. Allg. ohne Datentypen Aggregationen, Kompositionen Leserichtung, partielle Multiplizitäten Vererbung: Begriffsstruktur Annahme perfekter Technologie Funktionale Essenz Grobe Strukturskizze	Notation: UML Objekte: Softwareeinheiten Klassen: Komponenten, Grenzklassen, 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 der Vorlesungsfolien zur Vorlesung Softwaretechnologie von © Prof. H. Hussmann. Used by permission.

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



Profil “Aktive und Passive Klassen”

- Zum Kontextmodell gehört die Unterscheidung von *aktiven Klassen (Prozessen)* und *passiven Klassen*
  - Im einfachsten Fall existiert *eine* aktive Klasse
- Aktive Objekte
  - Aktoren
  - Prozesse
  - Workflow (interaktiver Prozess)
  - Werkzeuge (Tools)

Manager

<<actor>>

<<active class>> <<process>>

<<tool>>

- Passive Objekte (Materials, entities, data objects)
  - Aktive Objekte arbeiten auf Materialien

<<datatype>> <<material>>

<<passive class>> <<entity>>

- Kanäle (channels, pipes): Beschreiben, wie aktive Objekte miteinander kommunizieren
  - Über Kanäle fliessen Daten, sie werden mit Senken geschrieben und mit Streams gelesen

<<channel>>

<<call>>

<<stream>>

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>>