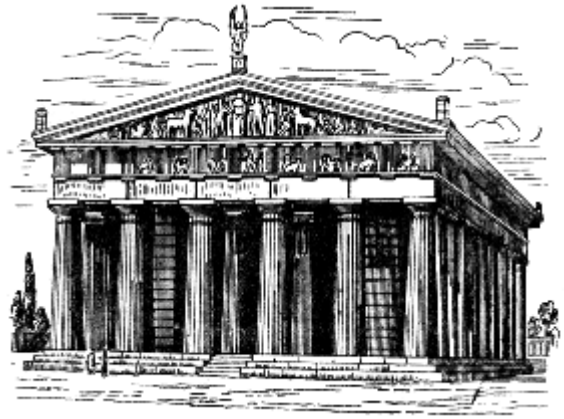
Softwarearchitektur

(Architektur: *αρχή* = Anfang, Ursprung + *tectum* = Haus, Dach)

11. Betriebliche Informationssysteme – Teil 3



Vorlesung Wintersemester 2005 / 06

Technische Universität München
Institut für Informatik
Lehrstuhl von Prof. Dr. Manfred Broy

Dr. Klaus Bergner, Prof. Dr. Manfred Broy, Dr. Marc Sihling

Inhalt

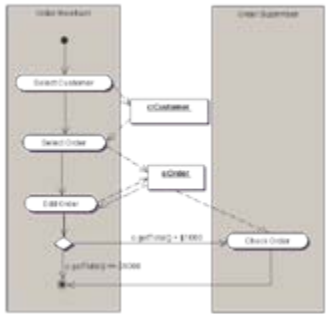
- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

Inhalt

- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

Logische Schichtenarchitektur

Die Aspekte der fachlichen und technischen Architektur werden vier Schichten zugeordnet. Die Unterstützungsschicht bietet querschnittliche Funktionalität.

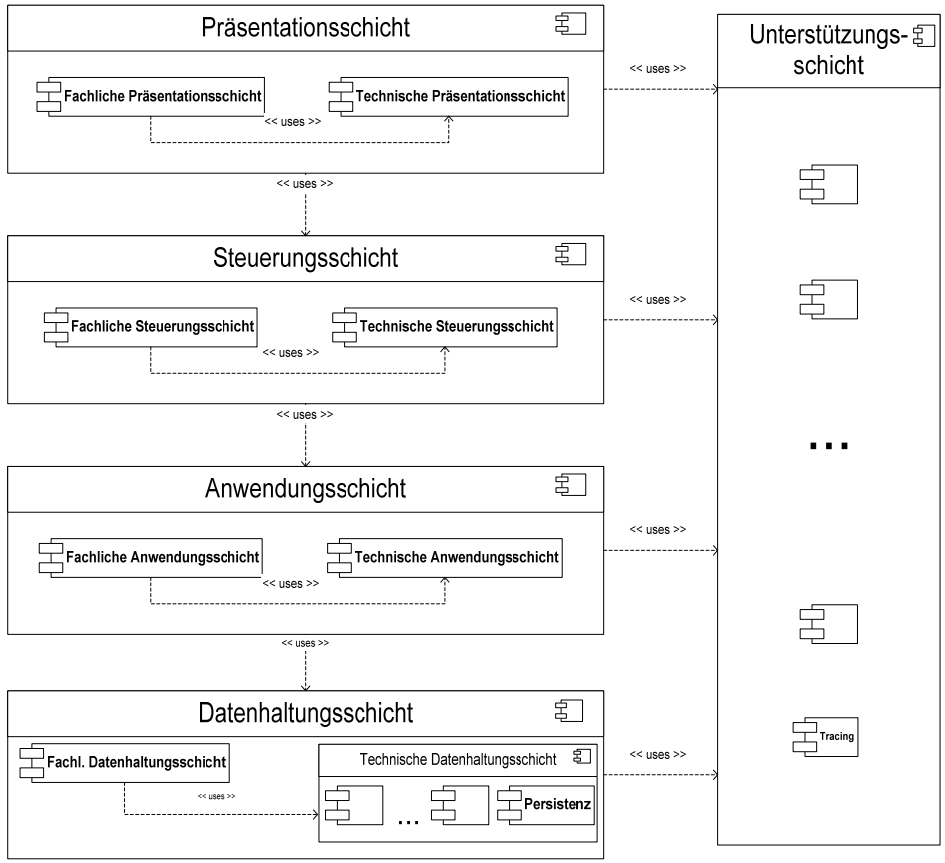


GUIs zu Aktivitäten

Workflows zu Prozessen

Fachliche Komponenten

Datenlayout zu Objekten



Beispiele für weitere CORBA Services

- **Life Cycle Service** stellt Operationen zur Erzeugung, zum Kopieren, Verschieben und Löschen von Objekten bereit.
- **Collection Service** ermöglicht es, Objekte in Gruppen zu manipulieren. (Arrays, Bäume, Stacks, …)
- **Relationship Service** ermöglicht die Definition von Beziehungen zwischen Objekten.
- **Property Service** erlaubt es, einem Objekt dynamisch eine Eigenschaft zuzuweisen (z. B. einen Titel oder ein Datum).
- **Concurrency Control Service** ermöglicht Objekten den Zugriff auf gemeinschaftlich genutzte Ressourcen mit Hilfe von Sperren (Locks).
- **Security Service** stellt Sicherheitsmechanismen bereit. Er unterstützt z. B. Authentifizierung und Zugriffskontrolle.
- **Time Service** stellt z. B. Interfaces zur Verfügung, um sich in einer verteilten Umgebung miteinander zu synchronisieren oder zeitabhängige Ereignisse zu definieren.
- **Licensing Service** erlaubt es, die Nutzungsdauer von Komponenten zu messen und entsprechend abzurechnen.

Inhalt

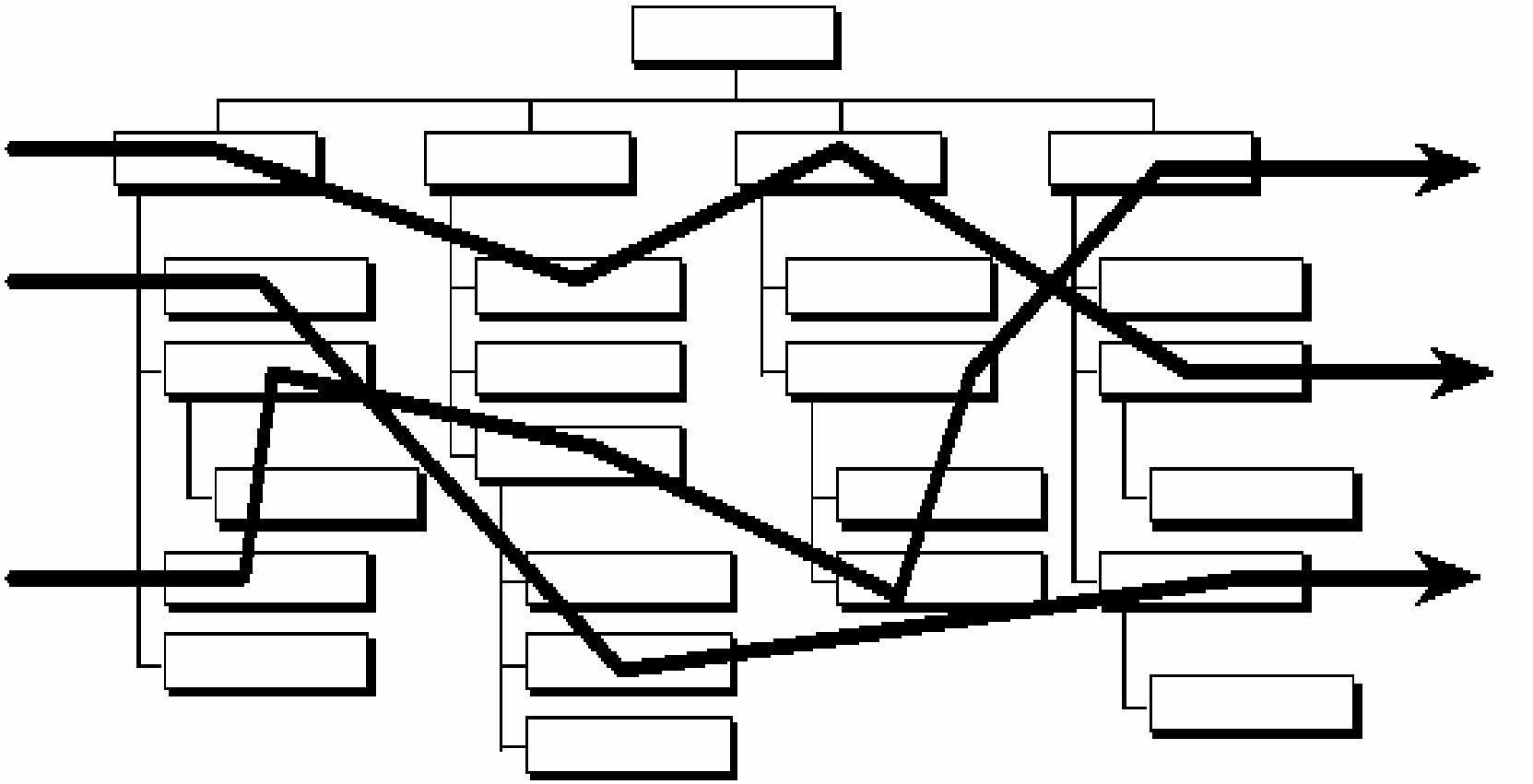
- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

Aufgabe der Steuerungsschicht

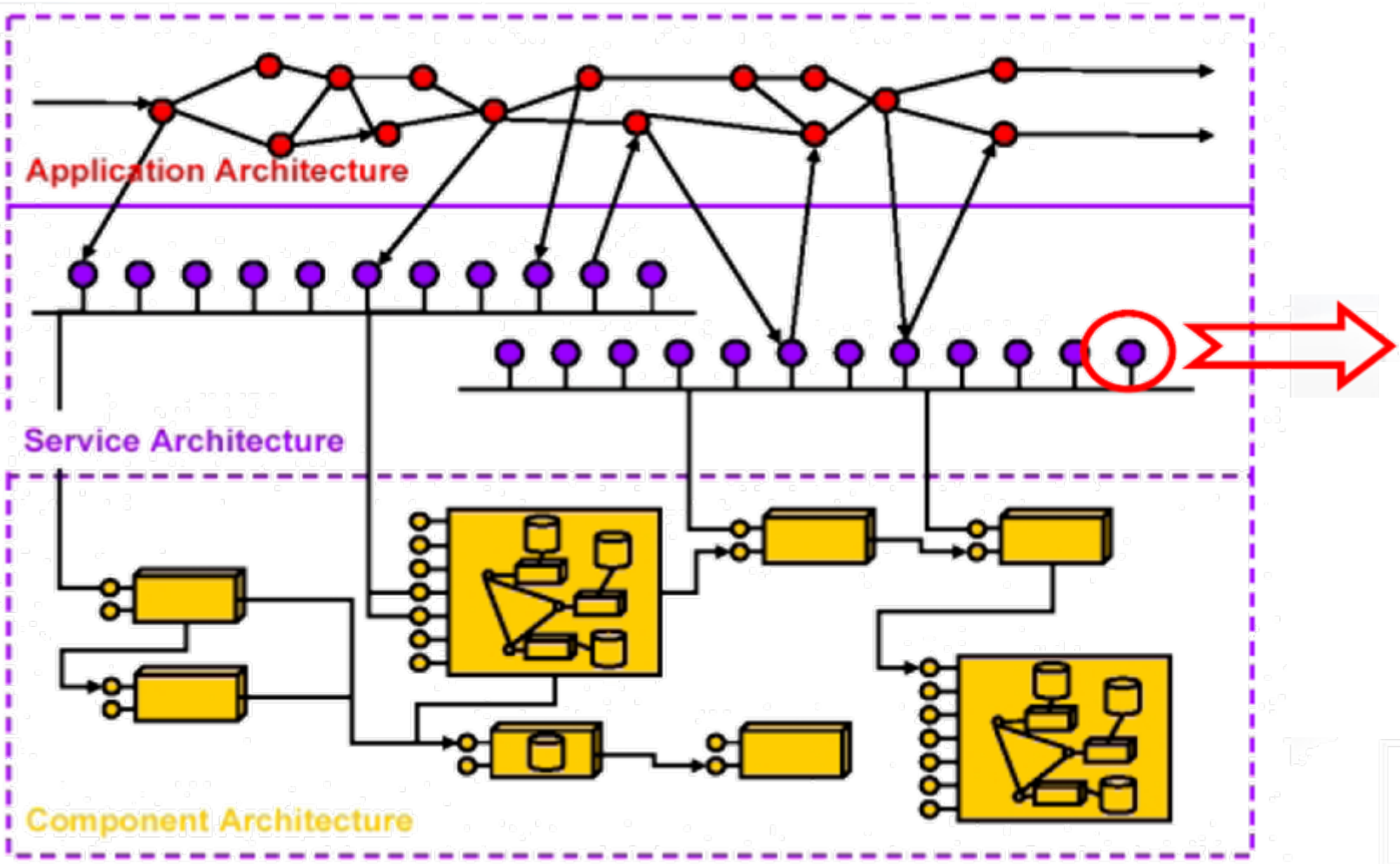
- Ziele einer expliziten Steuerungsschicht
  - Agile Unternehmen, Anpassung durch rekonfigurierbare Prozesse
  - Neue Geschäftsprozesse aus vorhandenen Aktivitäten/Services bauen
  - Einfache Kommunikation mit Partnern und Kunden
  - Einfaches Outsourcing von Teilprozessen

- Funktionen der Steuerungsschicht
  - Bietet Ablaufumgebung für die spezifizierten Geschäftsprozesse
  - Steuert den Ablauf der Aktivitäten
    - Aufruf fachlicher Funktionalität
    - Anzeige der Dialoge der Präsentationsschicht
  - Verwaltet Prozesskontext: Daten und Ressourcen
- In vielen Systemen mit Anwendungsschicht zusammengefasst

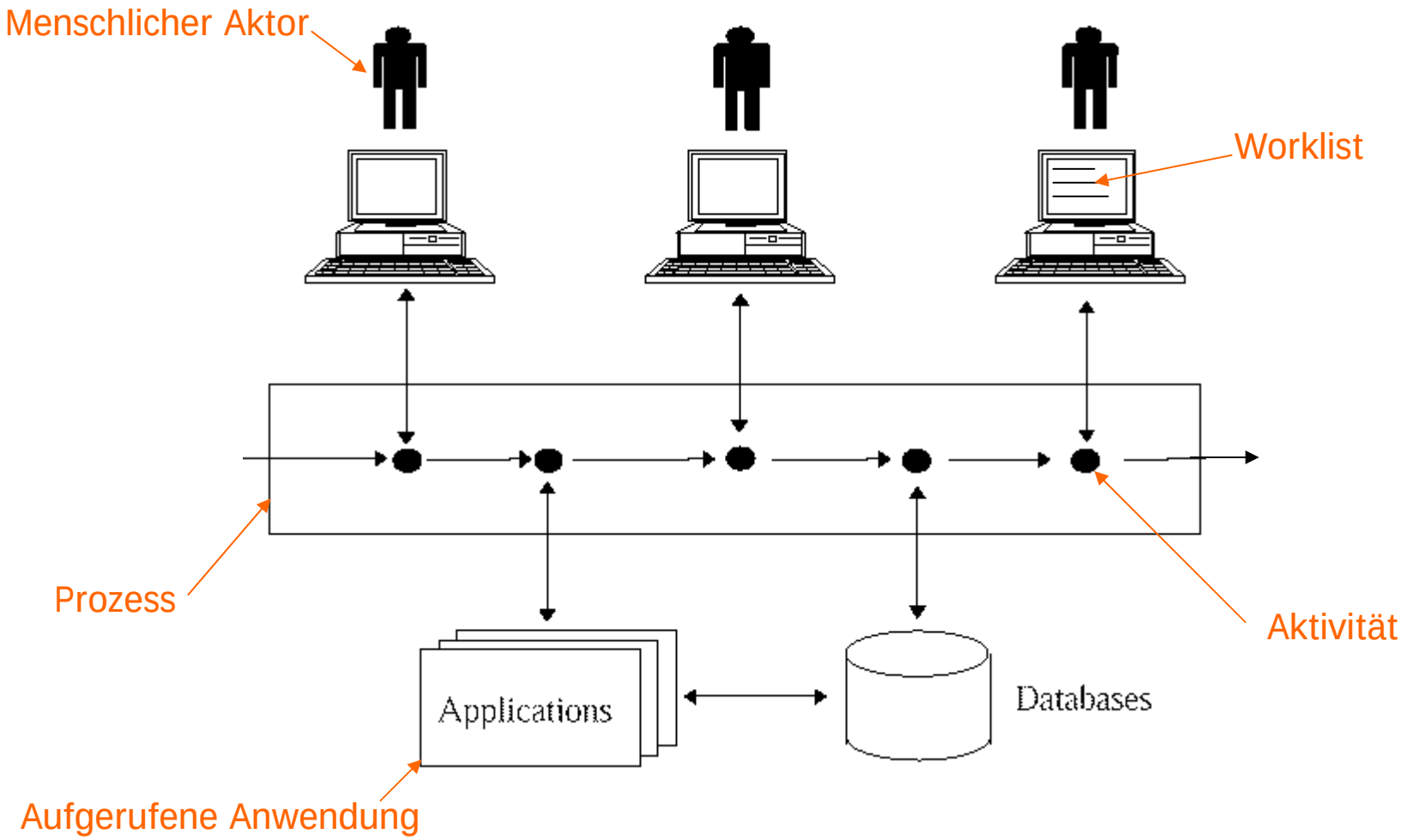
Geschäftsprozesse und Organisationen



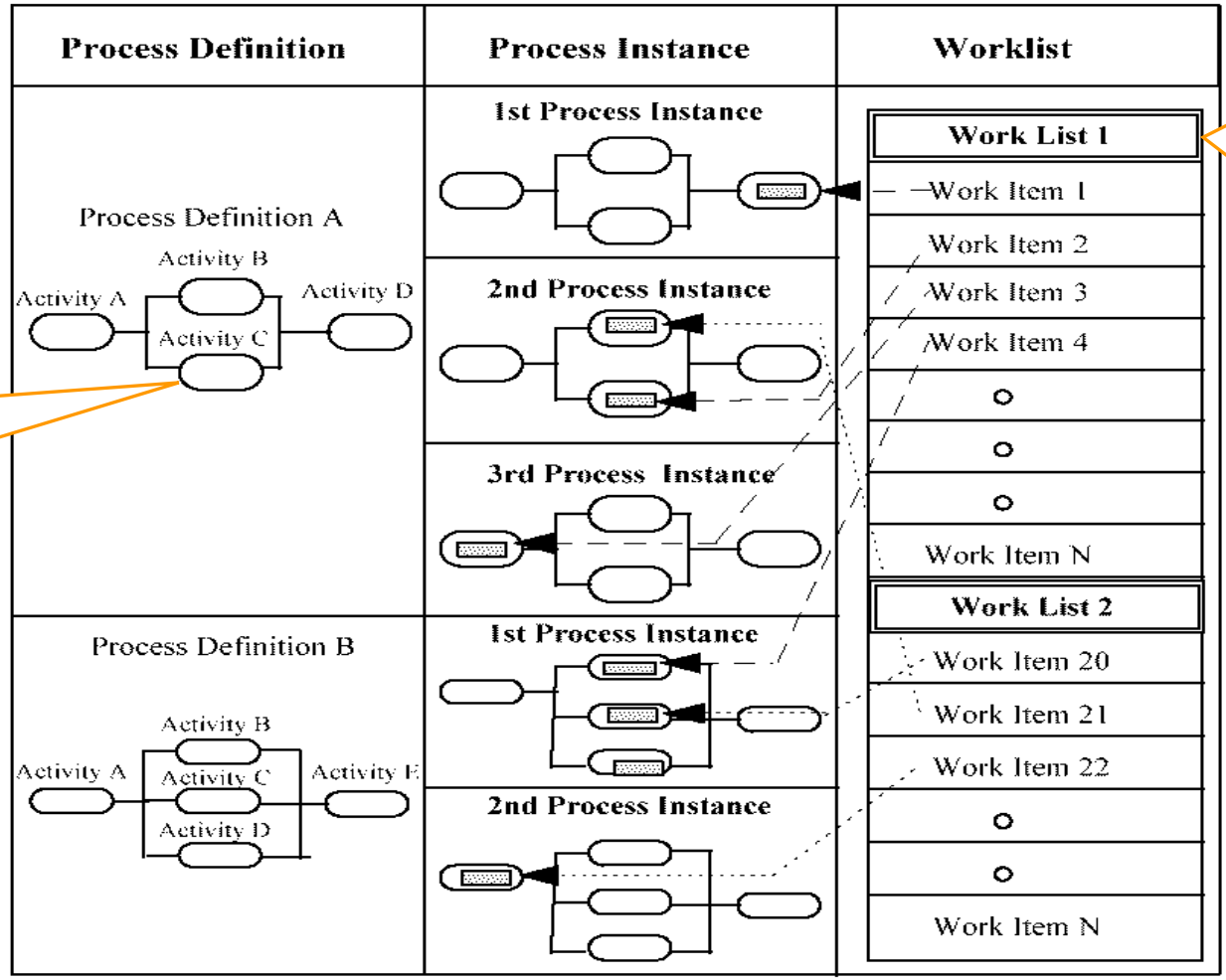
Services als Anwendungs-Bausteine



Workflow Management



Workflow: Verwaltung von Abläufen und Zuständen

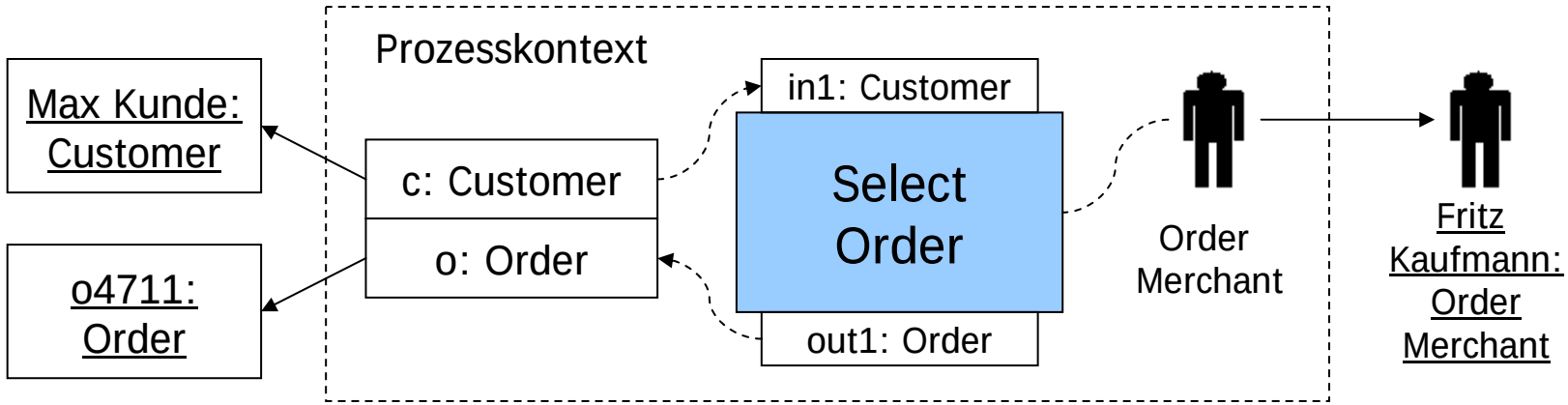


Aktivitäts-definition spezifiziert Rolle

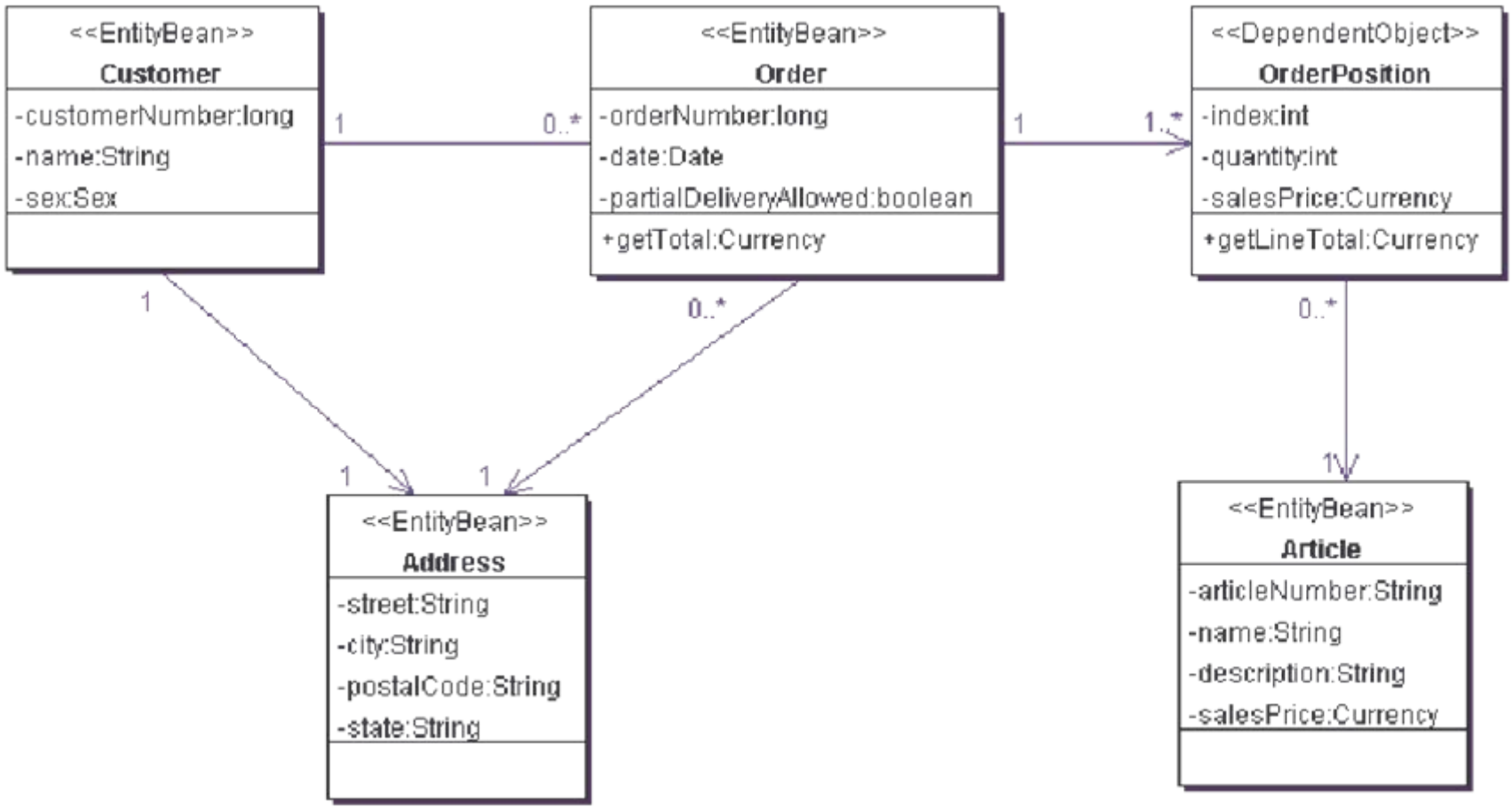
Worklist gehört zu Bearbeiter, der Rolle innehat

Workflow: Kontext- und Ressourcenverwaltung

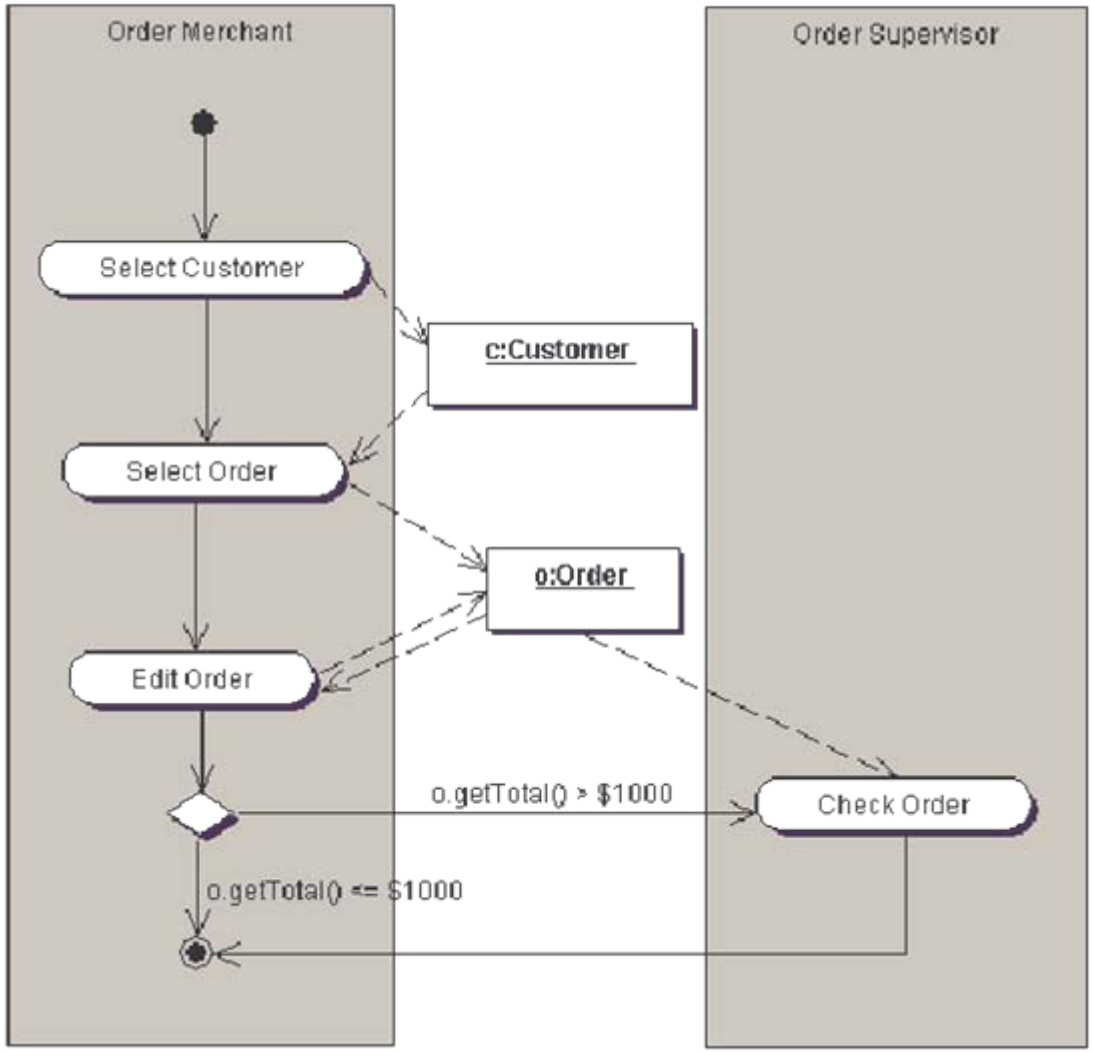
- Aktivitäten sind möglichst kontextlos
  - spezifizieren lediglich Ein-/Ausgabeschnittstelle
  - damit einsetzbar im Kontext beliebiger Prozesse
- Workflow-Engine stellt Kontext zur Verfügung
  - Versorgt Ein-/Ausgabeschnittstelle mit aktuellen Kontext-Daten
    - Per Value: Skalare Daten, Value-Objekte, Dokumente, Dateien
    - Per Reference: IDs von Geschäftsobjekten, URLs von Dokumenten
  - Ordnet Bearbeiter als Ressourcen gemäß Rollenspezifikation zu



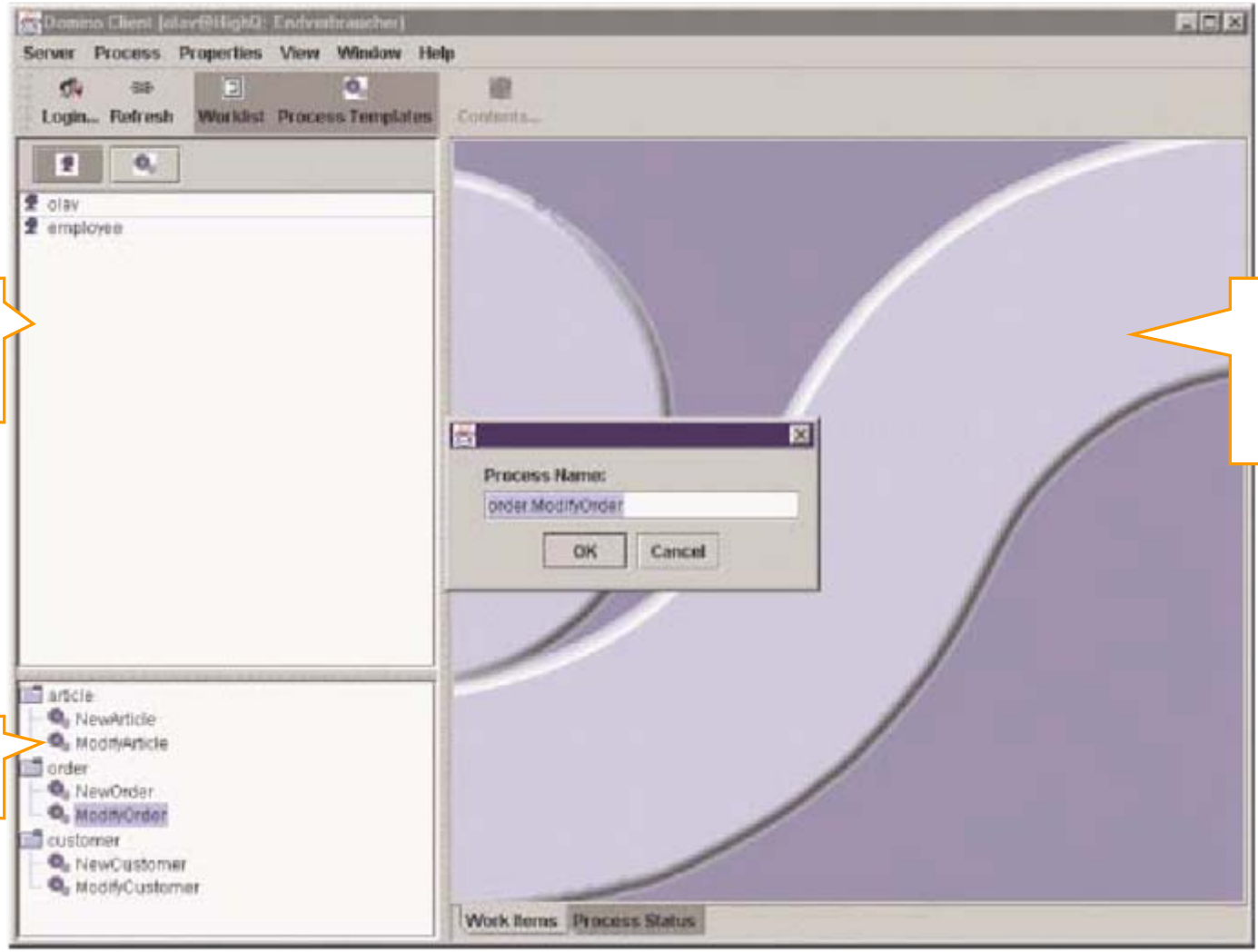
Beispiel: Order-System



Beispiel: Order-System



Beispiel: Order-System (1)

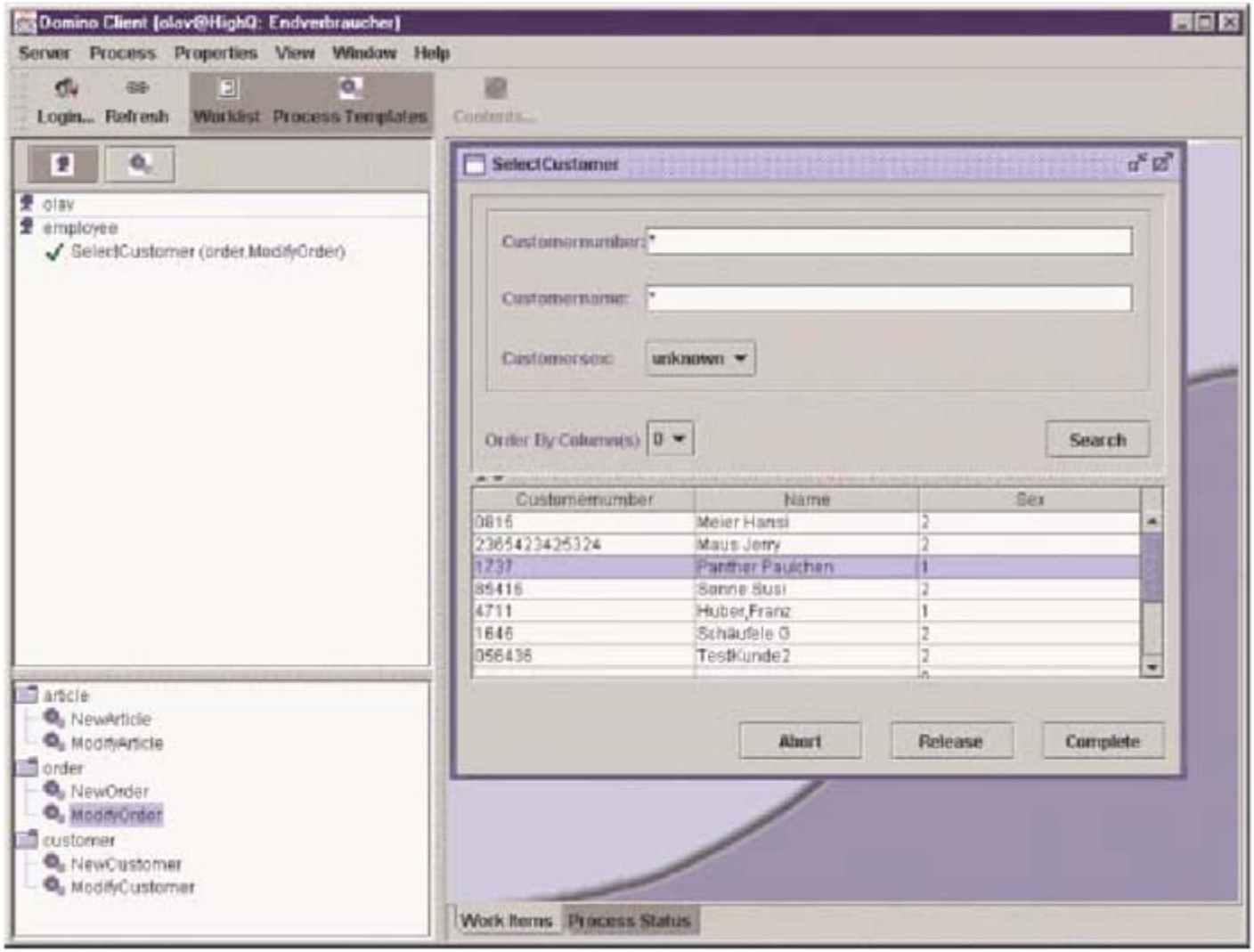


Worklist (Aktivität und Ablauf)

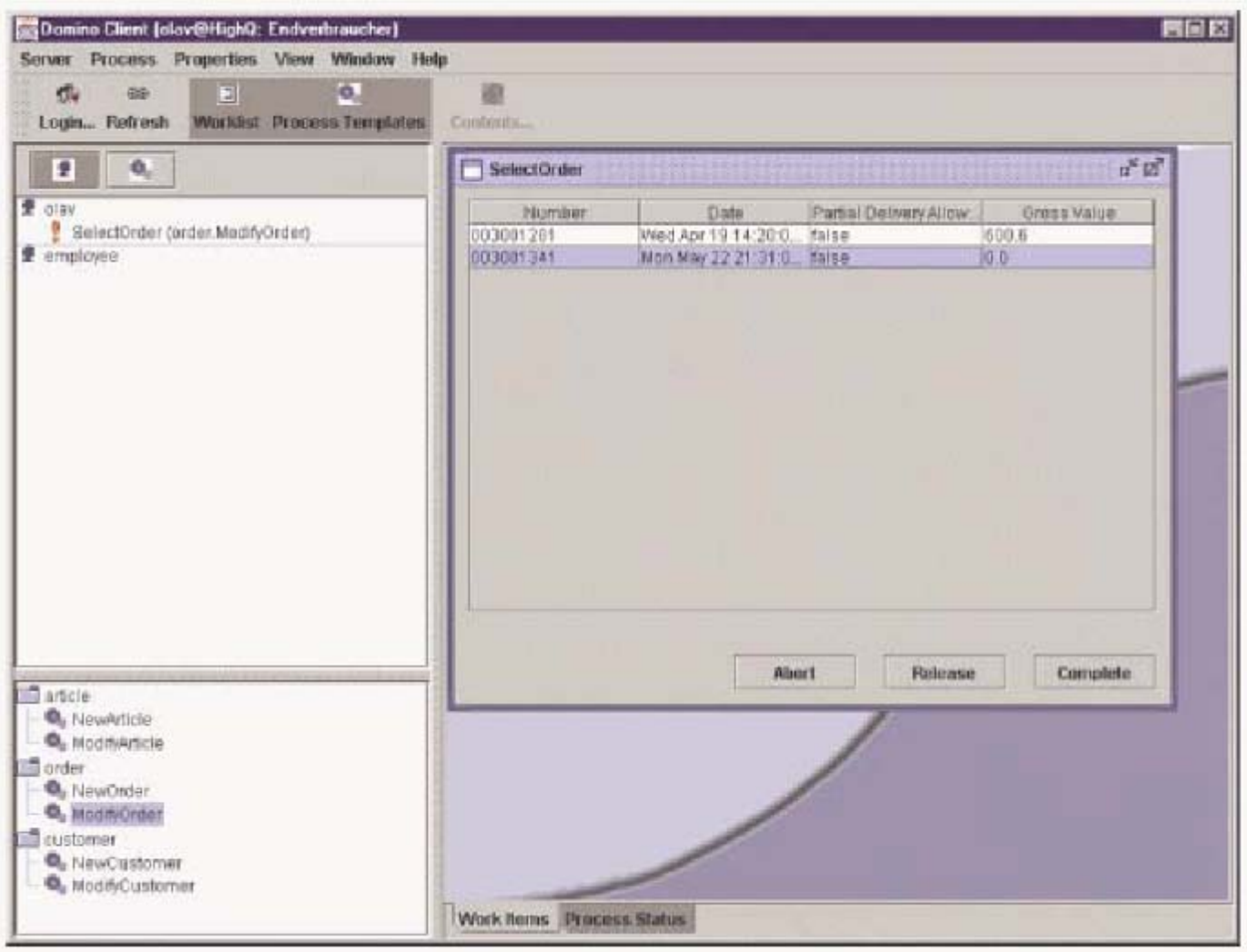
Prozess-definitionen

Arbeits-bereich: aktive Aktivitäten

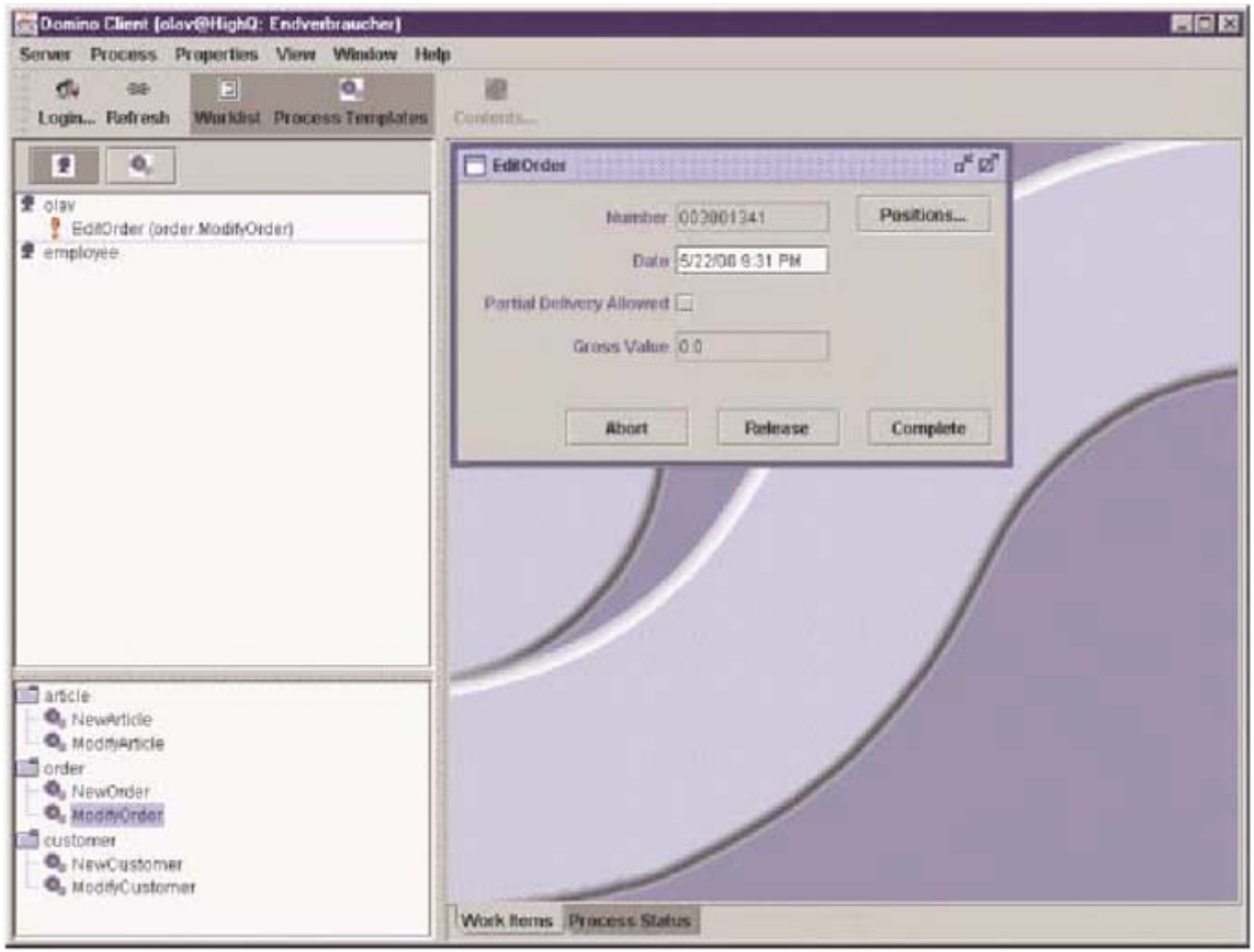
Beispiel: Order-System (2)



Beispiel: Order-System (3)



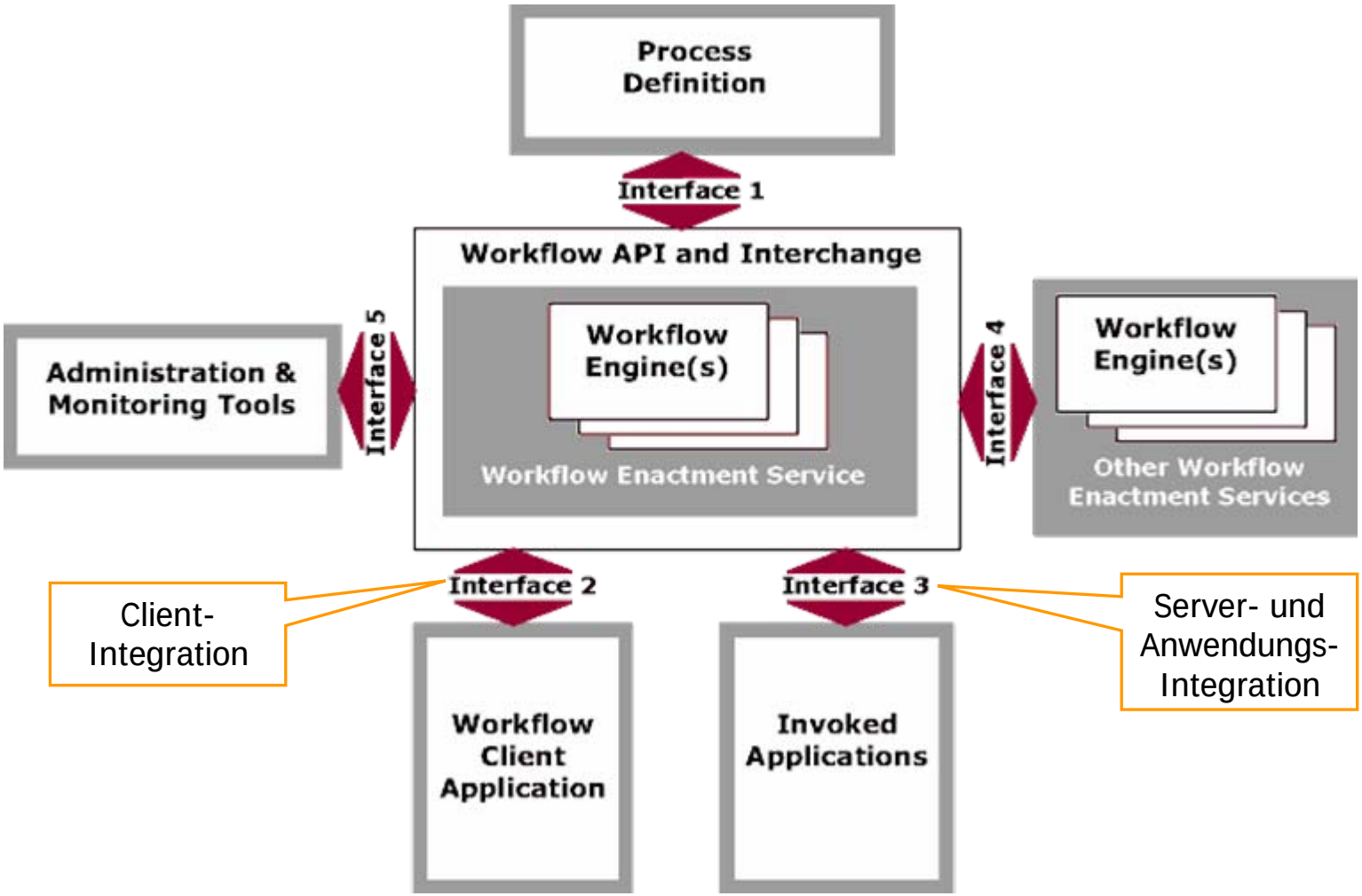
Beispiel: Order-System (4)



Middleware zum Workflow Management

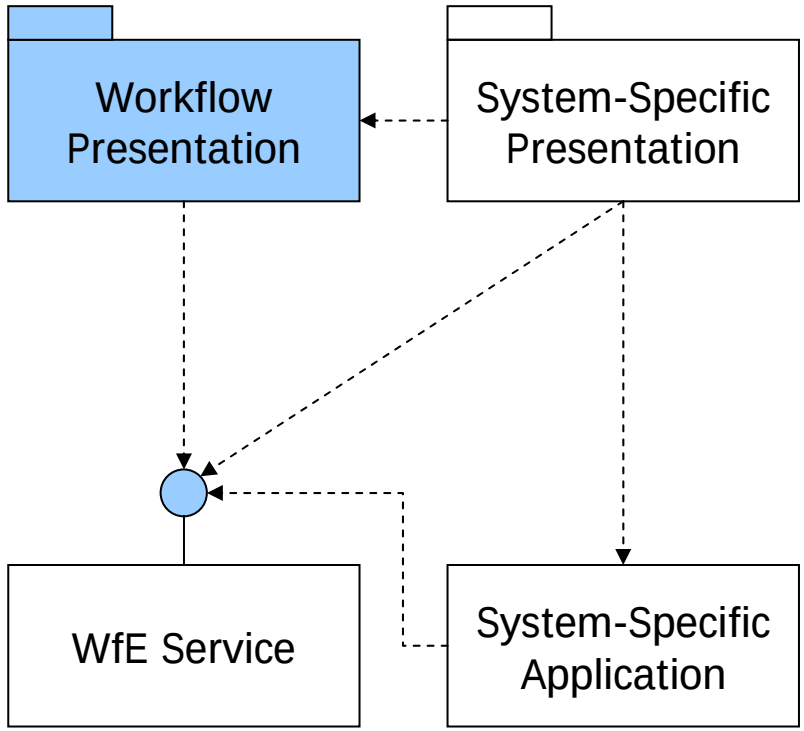
- Realisierung der Steuerungsschicht
  - Abläufe als Programmcode
    - ± Nur bei kleinen Systemen tragfähig
  - Verwaltung der Prozesse durch Workflow Engine
    - \+ Nutzung von mächtigen Mechanismen (s. nächste Folien)
    - \+ Konfiguration und Modellierung statt Programmierung
    - \- Ggf. Integration mit System zu leisten
    - ± Open-Source-Systeme bisher mäßig erfolgreich (Bsp. jBPM/JBOSS)
- Sehr heterogener Markt, Standards nur in Teilbereichen
  - WfMC – Workflow Management Coalition – Traditionelle Schnittstellen
  - OMG Workflow Facility – Prozesse und Aktivitäten als CORBA-Objekte
  - BPEL als Standardisierungsansatz im Umfeld Web Services

WfMC Workflow Reference Model



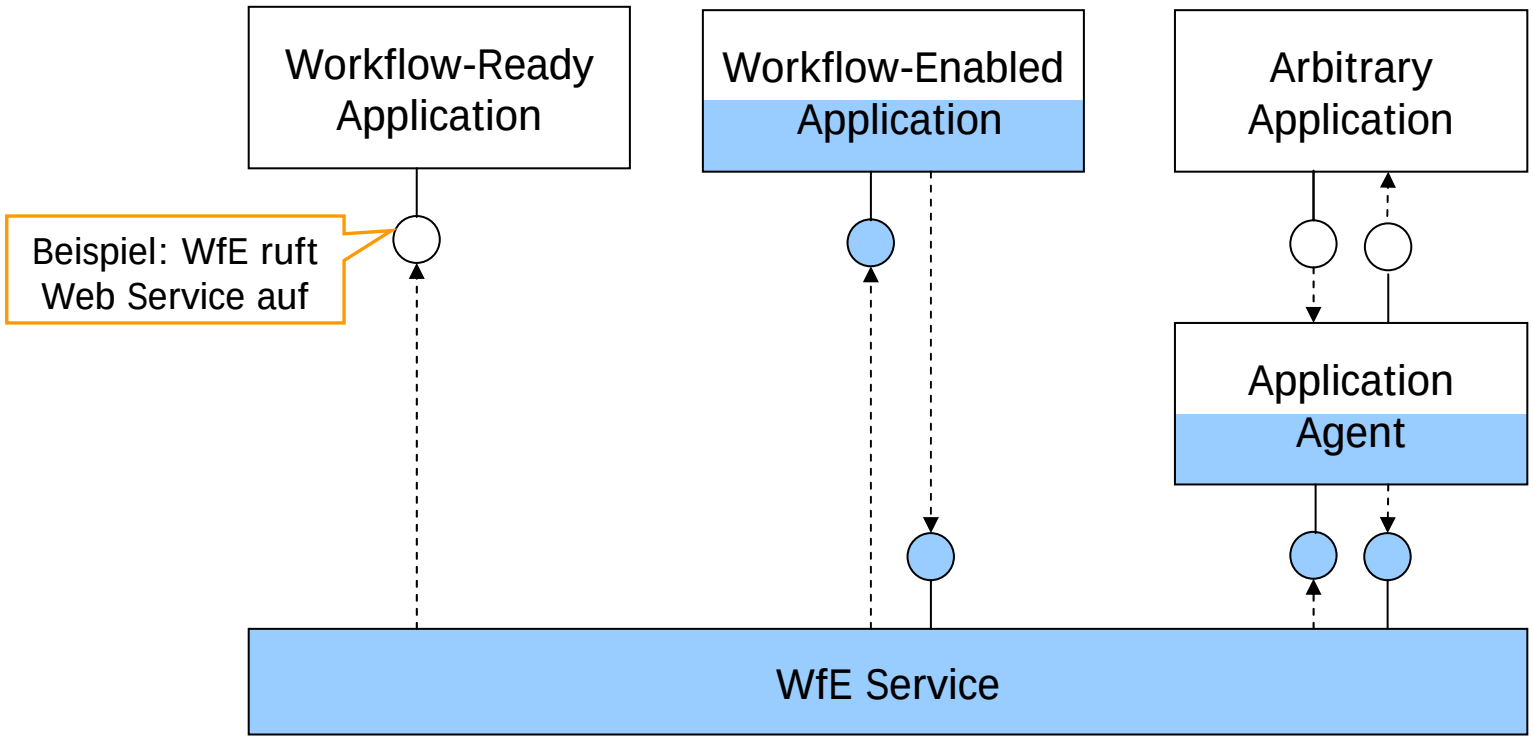
Workflow Client Application (Interface 2)

- Workflow Presentation
  - GUI-Komponentenbibliothek für Workflow-Anwendungen mit
    - Anzeige der Worklist
    - Anzeige startbarer Prozesse
    - Ablaufvisualisierung
  - Workflow-Rahmenanwendung als Basis für „Worklets“
- Workflow Client Application Interface
  - Sitzungsmanagement
  - Prozess- und Aktivitätskontrolle
    - Starten/Abbrechen von Prozessen
    - Durchführen von Aktivitäten
  - Worklist-Manipulation
    - Anzeige der Worklist
    - Annehmen von Aktivitäten
  - Status-Abfrage von Prozessen



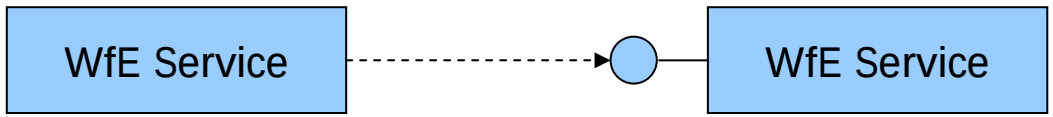
Invoked Application (Interface 3)

- Aufruf von Anwendungen umfasst
  - Start, Parameterversorgung, Überwachung, ggf. Re-Start
  - Zuordnung zu Bearbeiter und ggf. dessen Client-Rechner



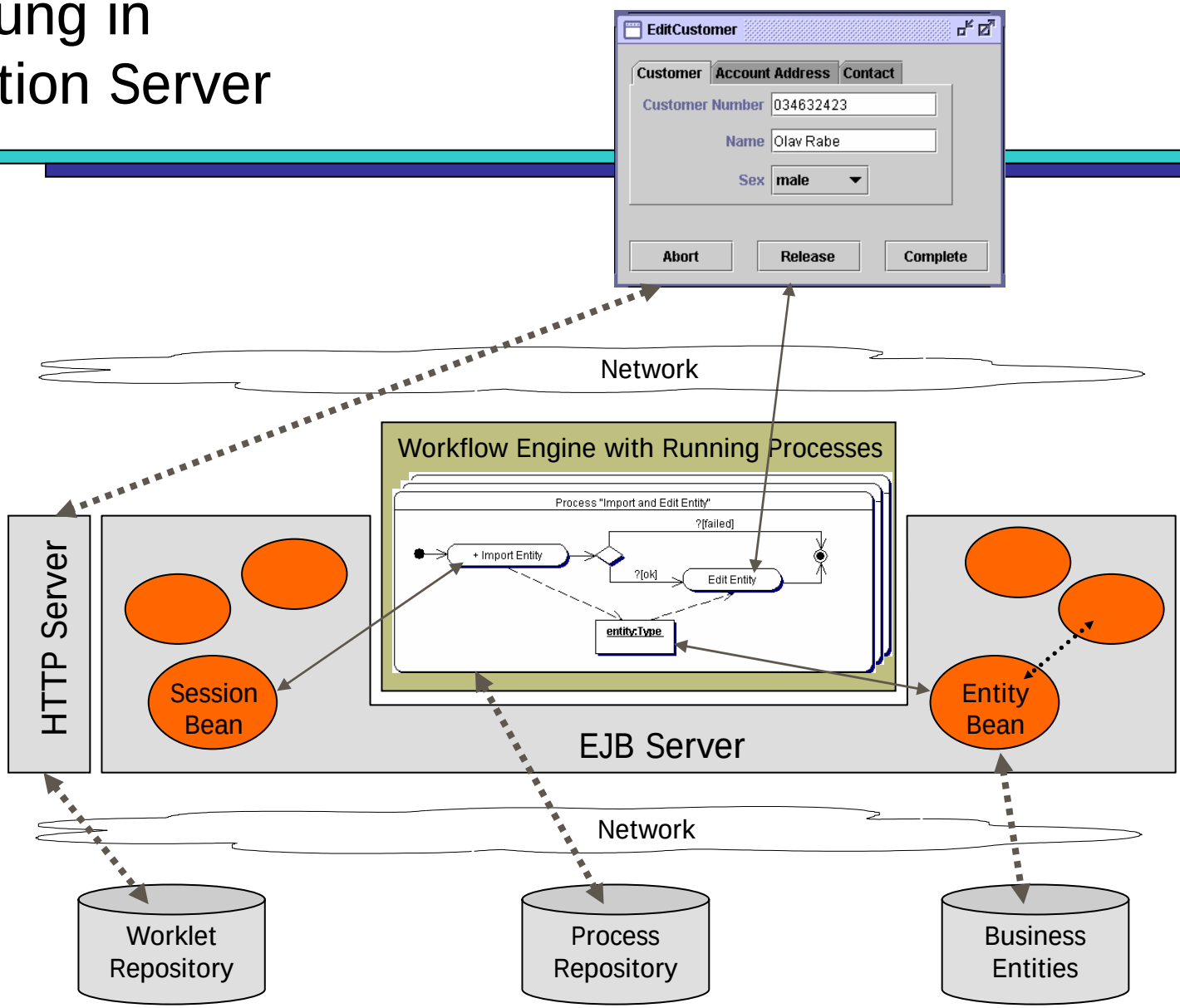
Workflow Interoperability (Interface 4)

- Verteilung der Workflow-Steuerung
  - Auslagerung und Aufruf von Teilprozessen
  - Verteilte Ablaufsteuerung und Statusabfrage
  - Transfer der Workflow-Kontextdaten
  - Senden von Prozessdefinitionen

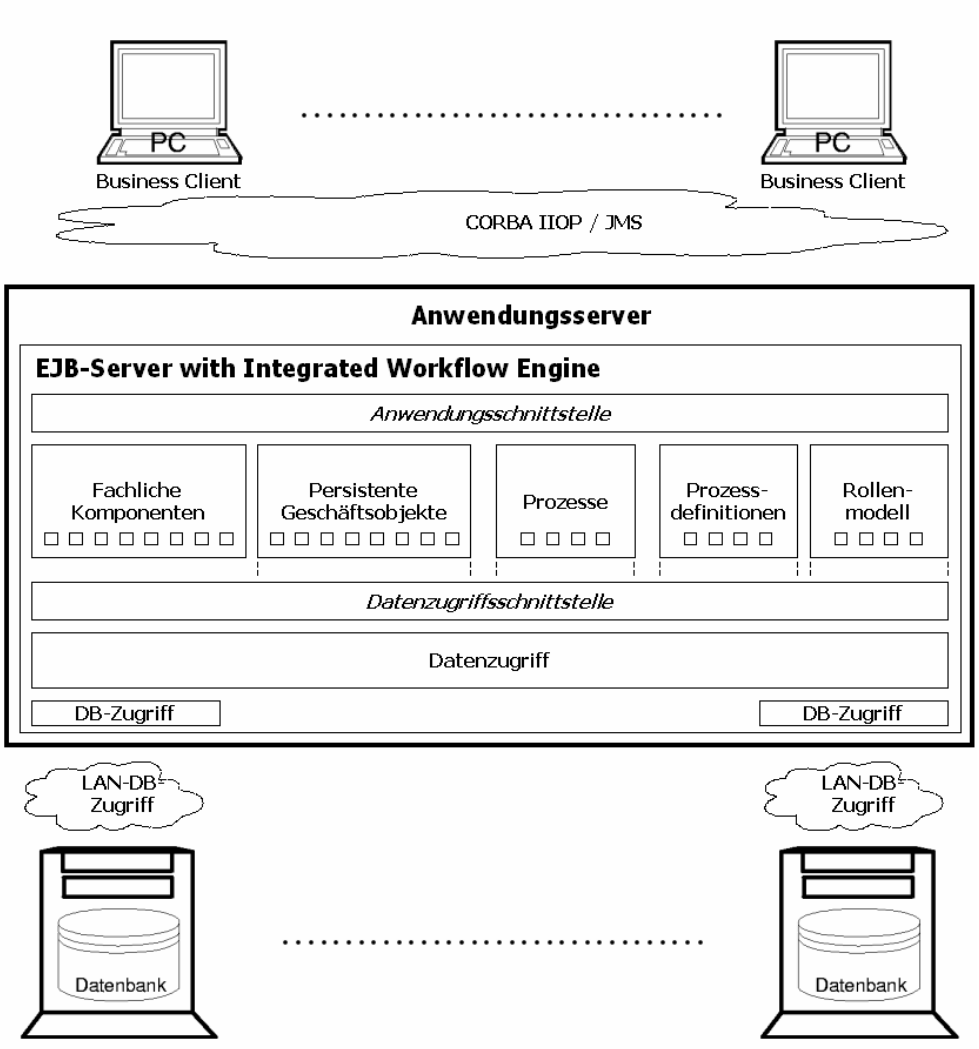


- Standardisierter Ansatz: BPEL, Business Process Execution Language
  - Kommunikation über Web-Services in XML-Form
    - Aufgerufene Anwendungen stellen Web-Services bereit
    - Aufruf von Prozessen erfolgt über Web-Services
  - Primitive zum Senden, Empfangen, Antworten auf Nachrichten
  - Ablaufsteuerungsfunktionen: Schleifen, Bedingungen
  - Rückgängigmachen bereits durchgeführter Aktivitäten durch Kompensation

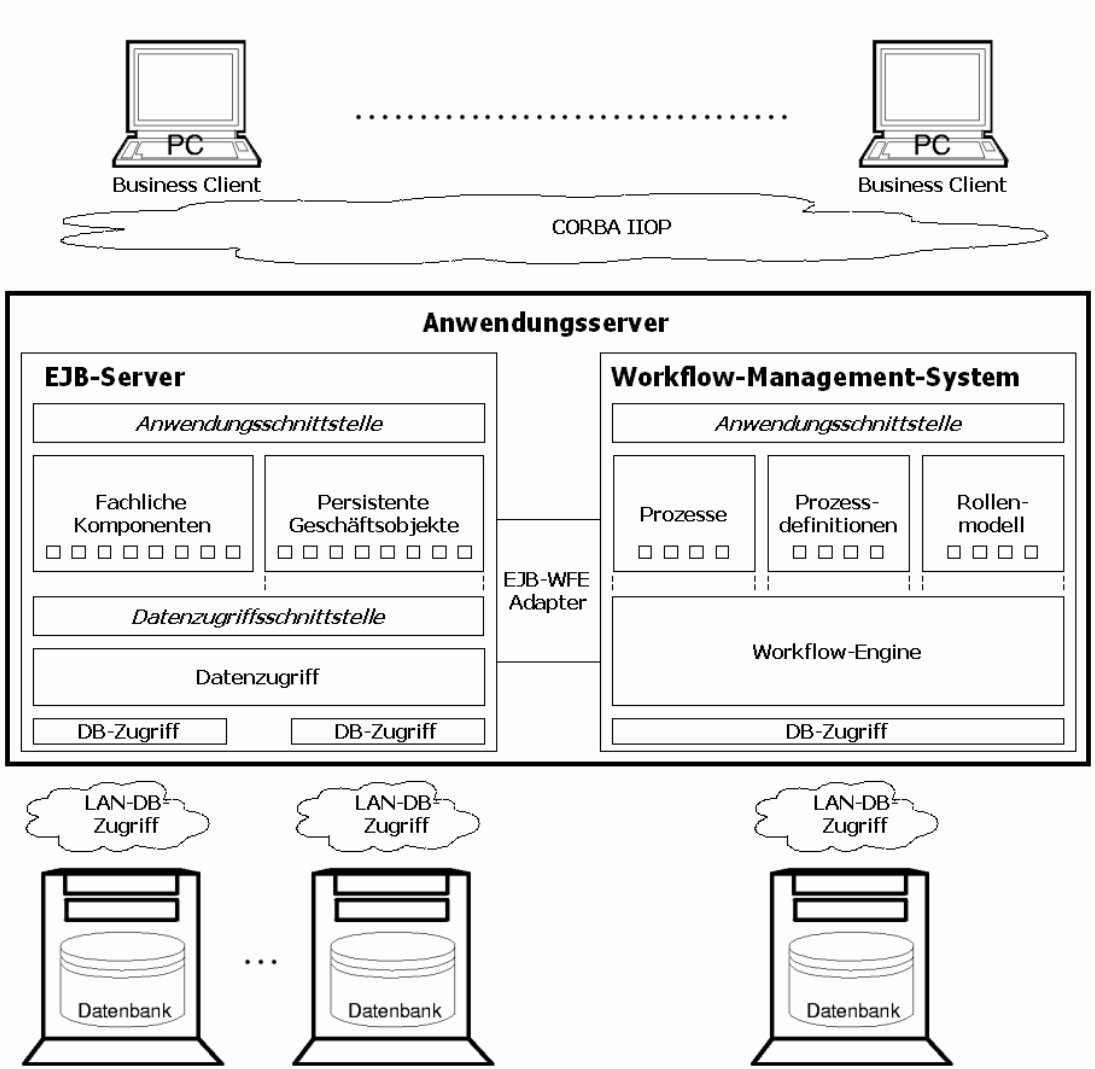
Einbettung in Application Server



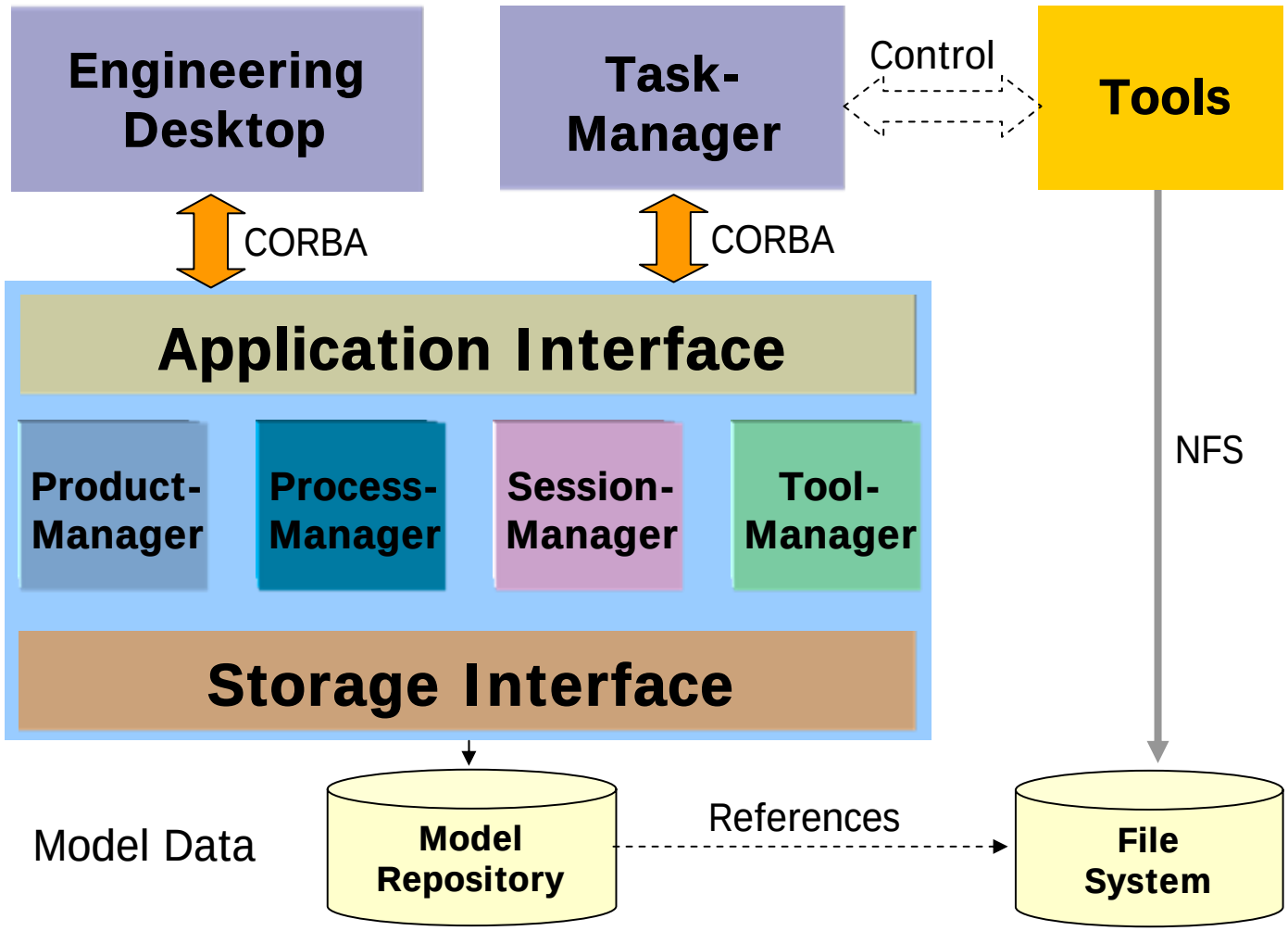
Beispiel: Schematische Architektur eCommerce-System



Beispiel: Reale Architektur des eCommerce-Systems



Beispiel: Architektur eines Engineering-Workflow-Systems



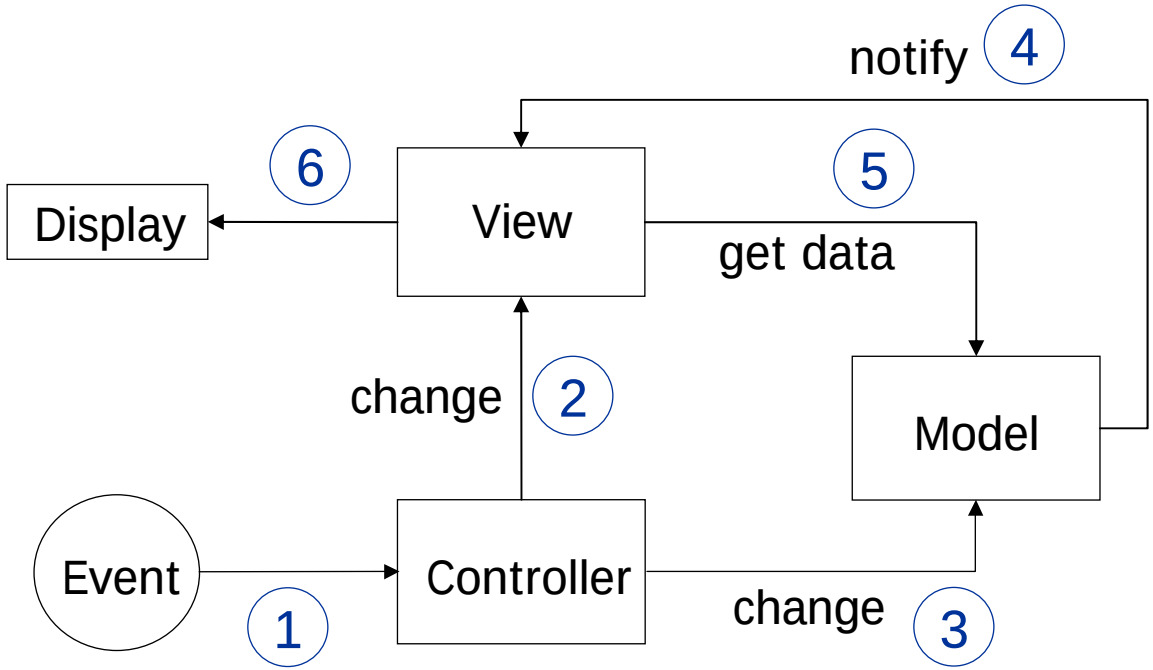
Inhalt

- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

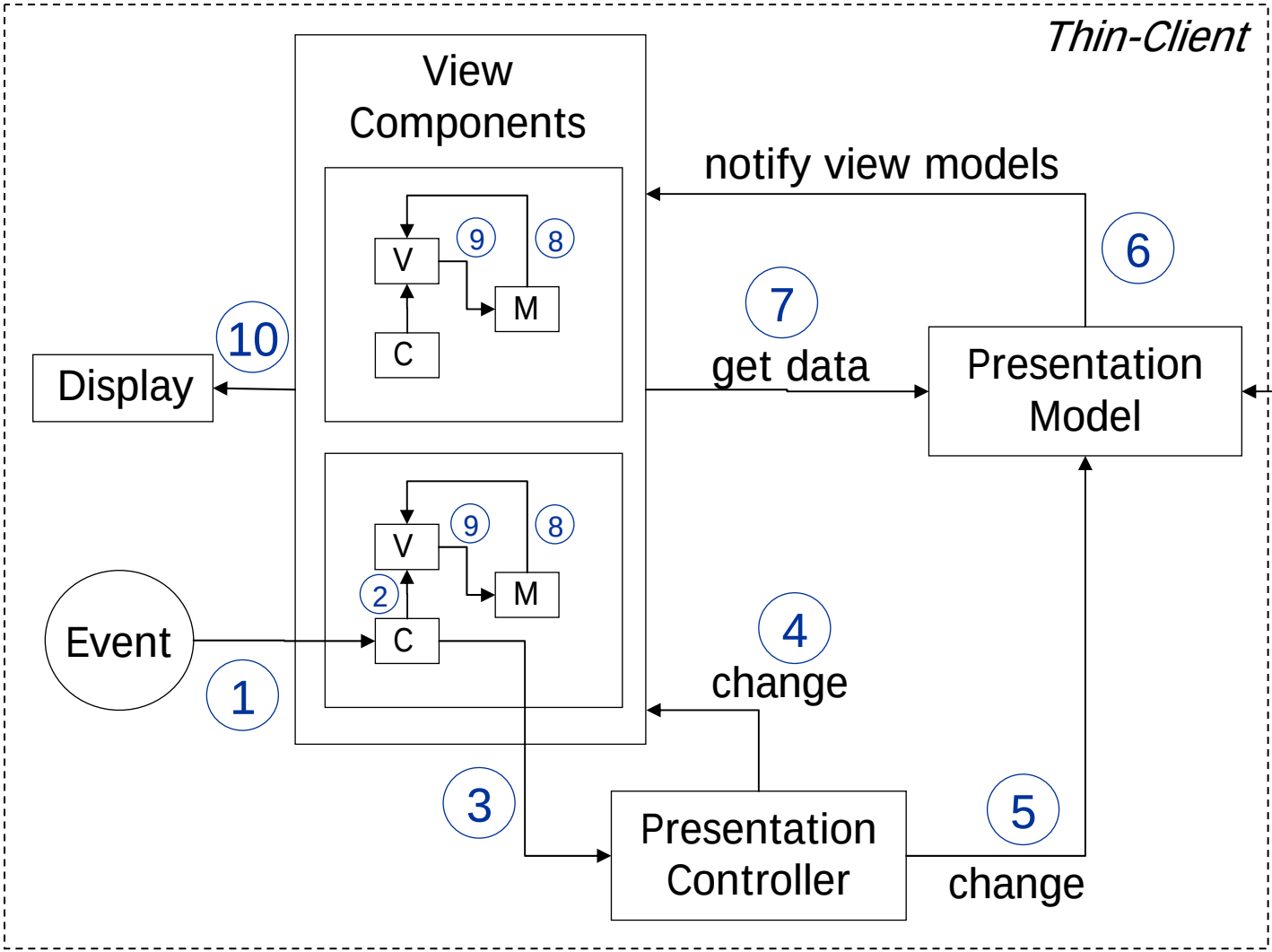
Aufgabe der Präsentationsschicht

- Zeigt Anwendungsdaten an
  - Greift lesend auf die Anwendungsschicht zu
  - Aktualisiert die Anzeige
- Steuert die Interaktion mit dem Benutzer
  - Zeigt graphische Elemente der Bedienoberfläche an
  - Erkennt Benutzeraktionen auf unterschiedlichen Kanälen
  - Verwaltet den Kommunikationszustand
  - Ruft Anwendungsfunktionalität auf
- Führt selbständig einfache Prüfungen durch
  - Prüfung auf richtiges Eingabeformat (z.B. Eingabe einer Zahl)
  - Prüfung auf Eingabebereiche (z.B. Eingabe einer Zahl kleiner 10)

Rekapitulation: Model-View-Controller-Muster



Hierarchisches MVC beim Thin-Client



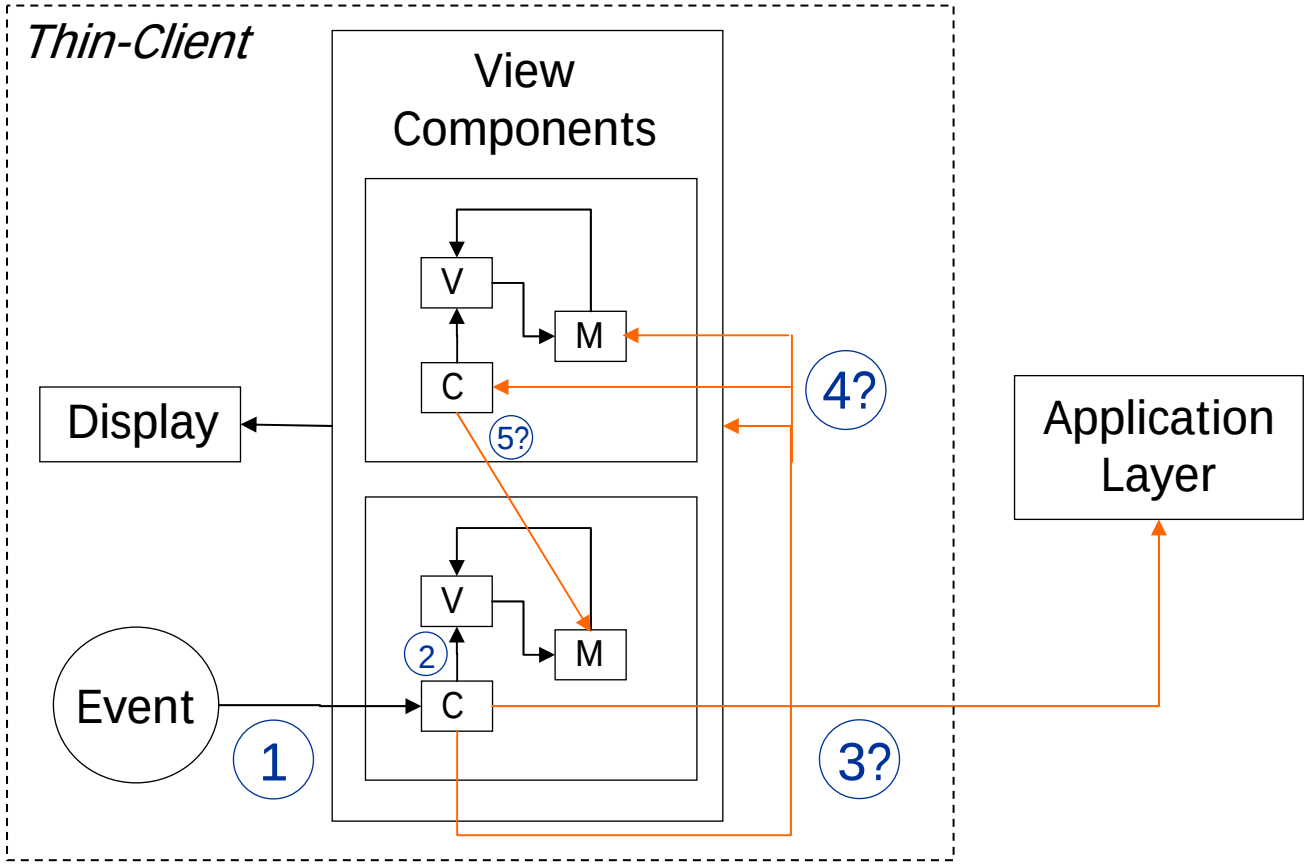
Erfordert Lauf-zeitumgebung auf Client

Beispiele:

- Java Client
- Java-Applet

Anti-Pattern: Spagetthi-Controller

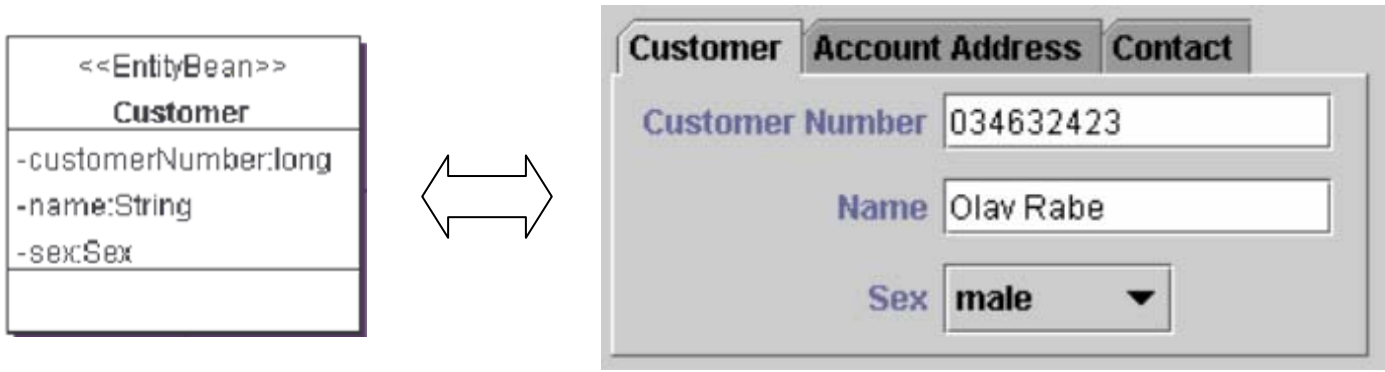
- Presentation Controller und/oder Presentation Model fehlen
- View-Controller ändern gegenseitig Views, Controller und Models



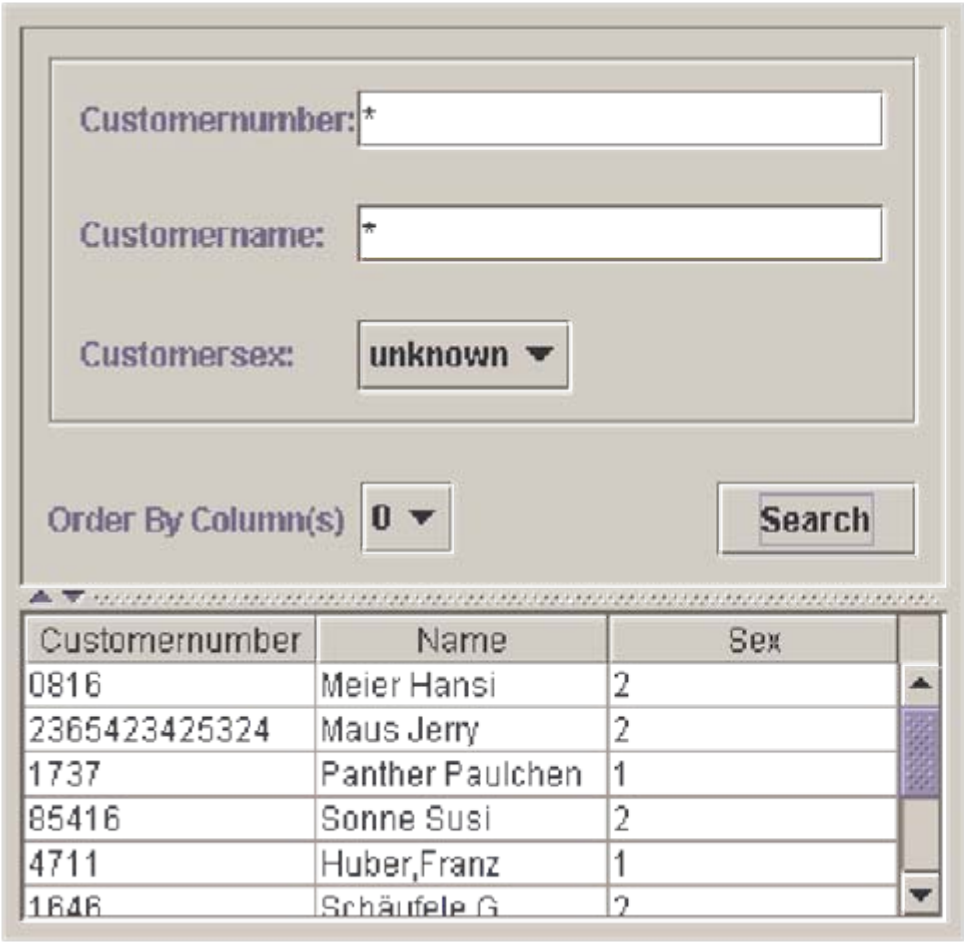
Generische GUI-Komponenten (1)

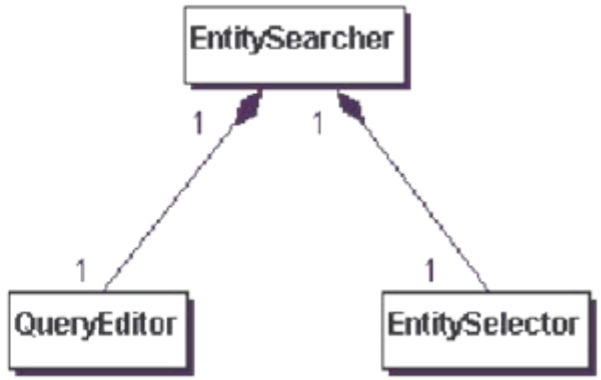
- Bindung an beliebige Model-Objekte
  - GUI-Komponenten stellen über Reflection die Attribute und Operationen der Model-Objekte fest und zeigen diese dynamisch an

- Beispiel: Entity Editor / Viewer
  - Editiere / zeige Attribute und aggregierte Objekte eines Geschäftsobjekts internationalisiert an



Generische GUI-Komponenten (2)





- Beispiel: Generische Abfrage-Komponente
  - Erlaubt Suche nach allen Attributen eines Objekts
  - Kann ggf. konfigurierbar sein, um Attribute auszu-schließen etc.

Präsentationsschicht bei Ultra-Thin-Clients

„Ultra thin client“

WEB/Application Server

Basistechnik: Java Servlets

SimpleServlet

```
public class SimpleServlet extends HttpServlet {
  public void doGet (HttpServletRequest request,
                     HttpServletResponse response)  {
    PrintWriter out;
    String title = "Simple Servlet Output";
    response.setContentType("text/html");
    out = response.getWriter();
    out.println("<HTML><HEAD><TITLE>");
    out.println(title);
    out.println("</TITLE></HEAD><BODY>");
    out.println("<H1>" + title + "</H1>");
    out.println("<P>Output from SimpleServlet.");
    out.println("</BODY></HTML>");
}}
```

Simple Servlet

Output from SimpleServlet.

www.tum.de/SimpleServlet

Basistechnik: JavaServer Pages

HelloWorld.jsp

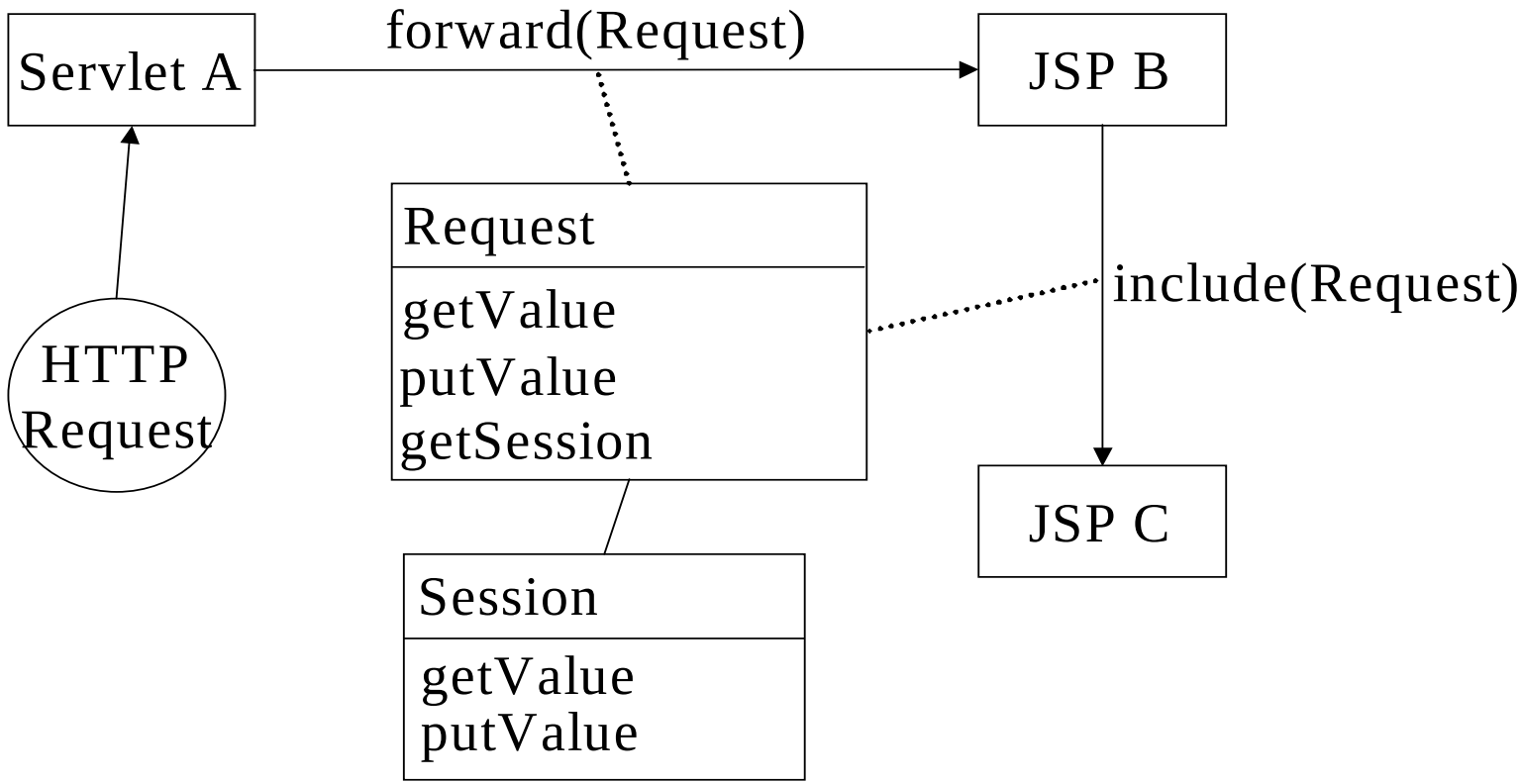
```
<html>
<head>
<title>Hello World</title>
</head>
<body bgcolor=#FFFFFF>
<font face="Helvetica">
<h2><font color=#DB1260>Hello World</font>
</h2>

<%out.print("<p><b>Hello World!</b>"); %>

<hr>
</font>
</body>
</html>
```

Hello World

Hello World!

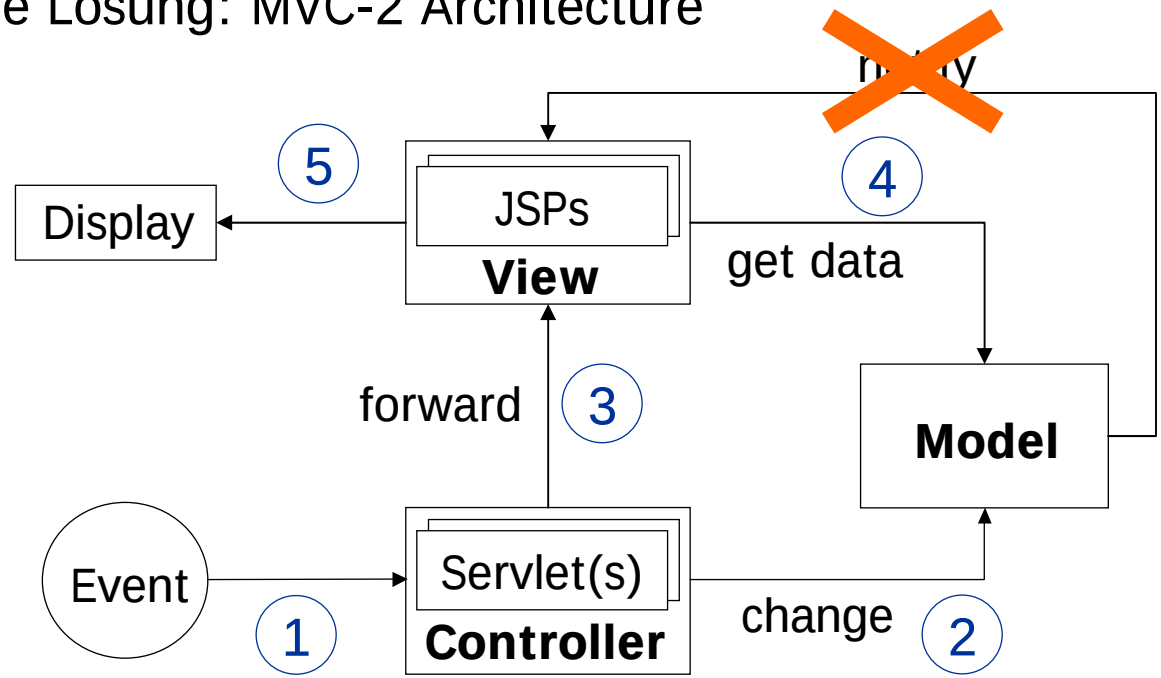
www.tum.de/HelloWorld.jsp

Basistechnik: Forwarding und Including

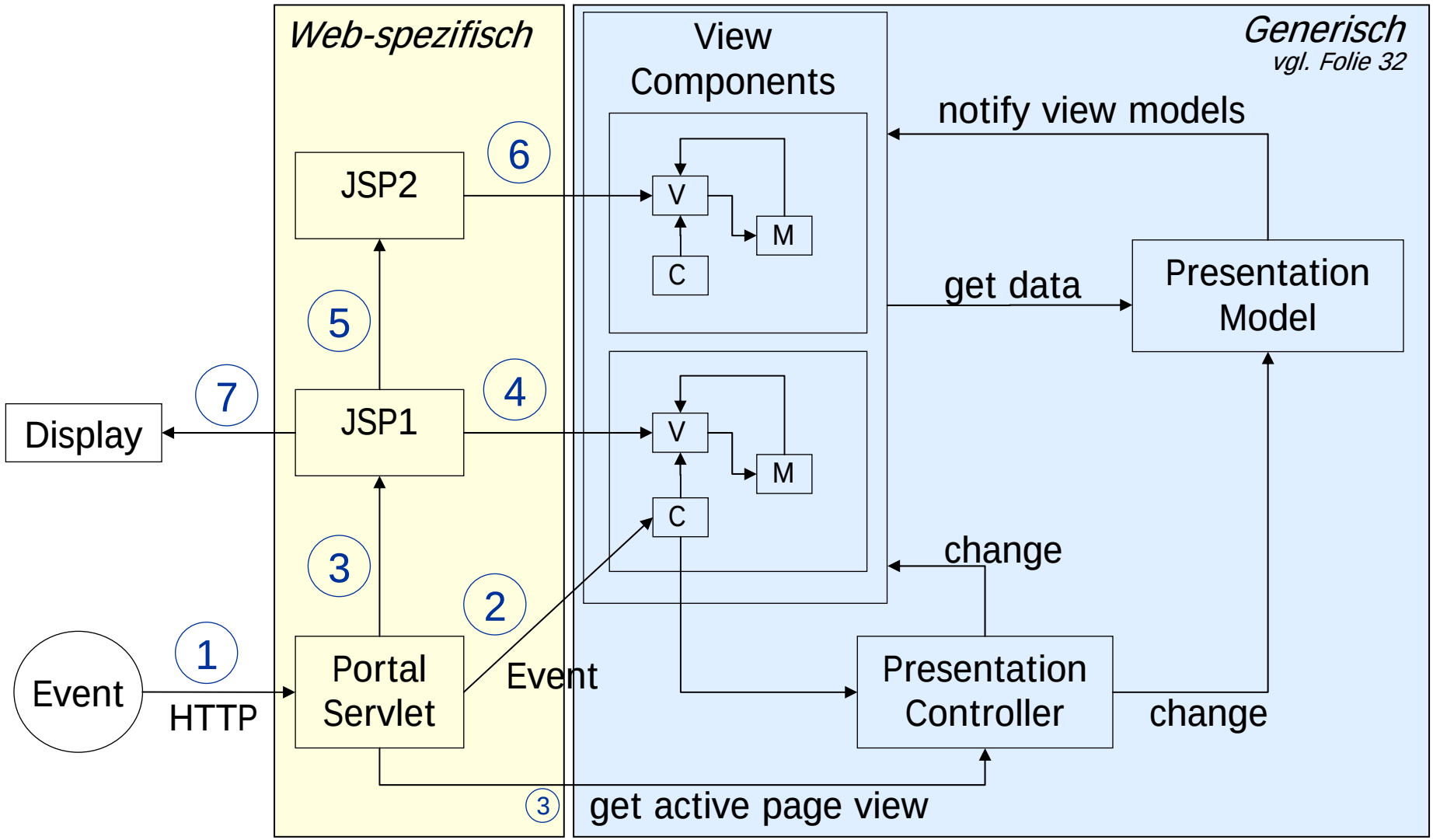


MVC bei Ultra-Thin Clients

- Direkte Umsetzung von MVC scheitert, weil
  - HTML/HTTP ist seitenorientiert und zustandslos
  - Kommunikation über HTTP-Requests statt Events
  - Kommunikation wird immer vom Browser initiiert
- Mögliche Lösung: MVC-2 Architecture

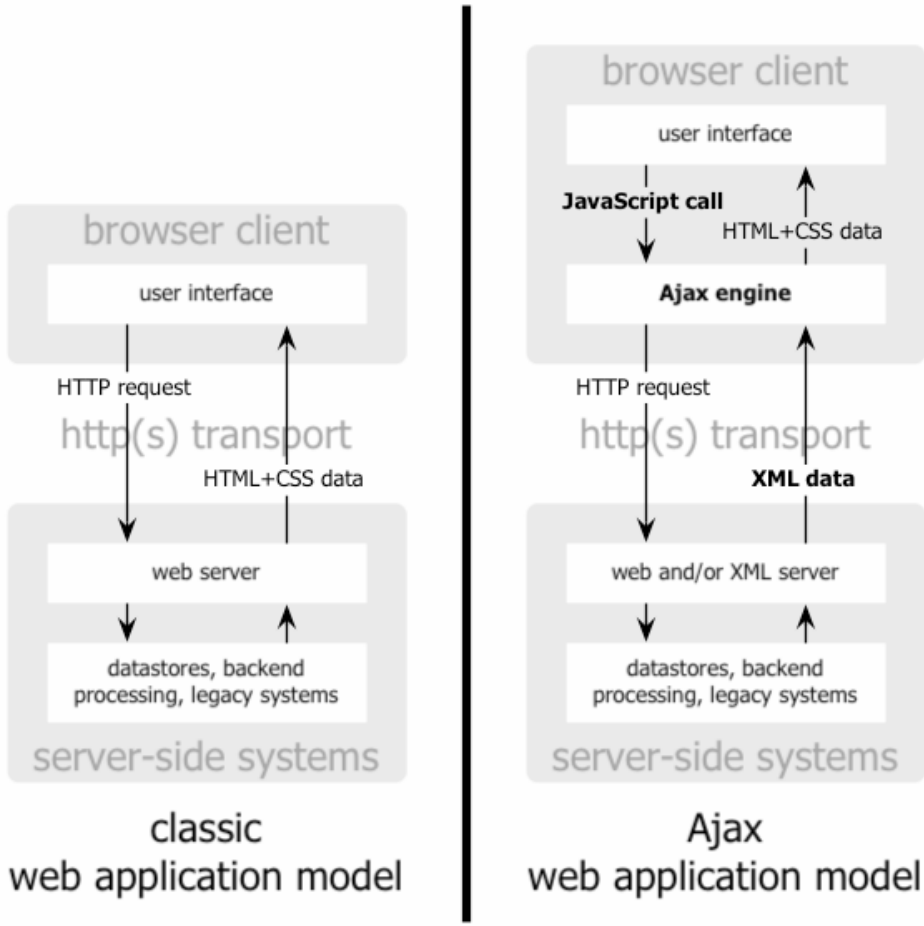


Multi-Channel-Architektur beim Ultra-Thin-Client



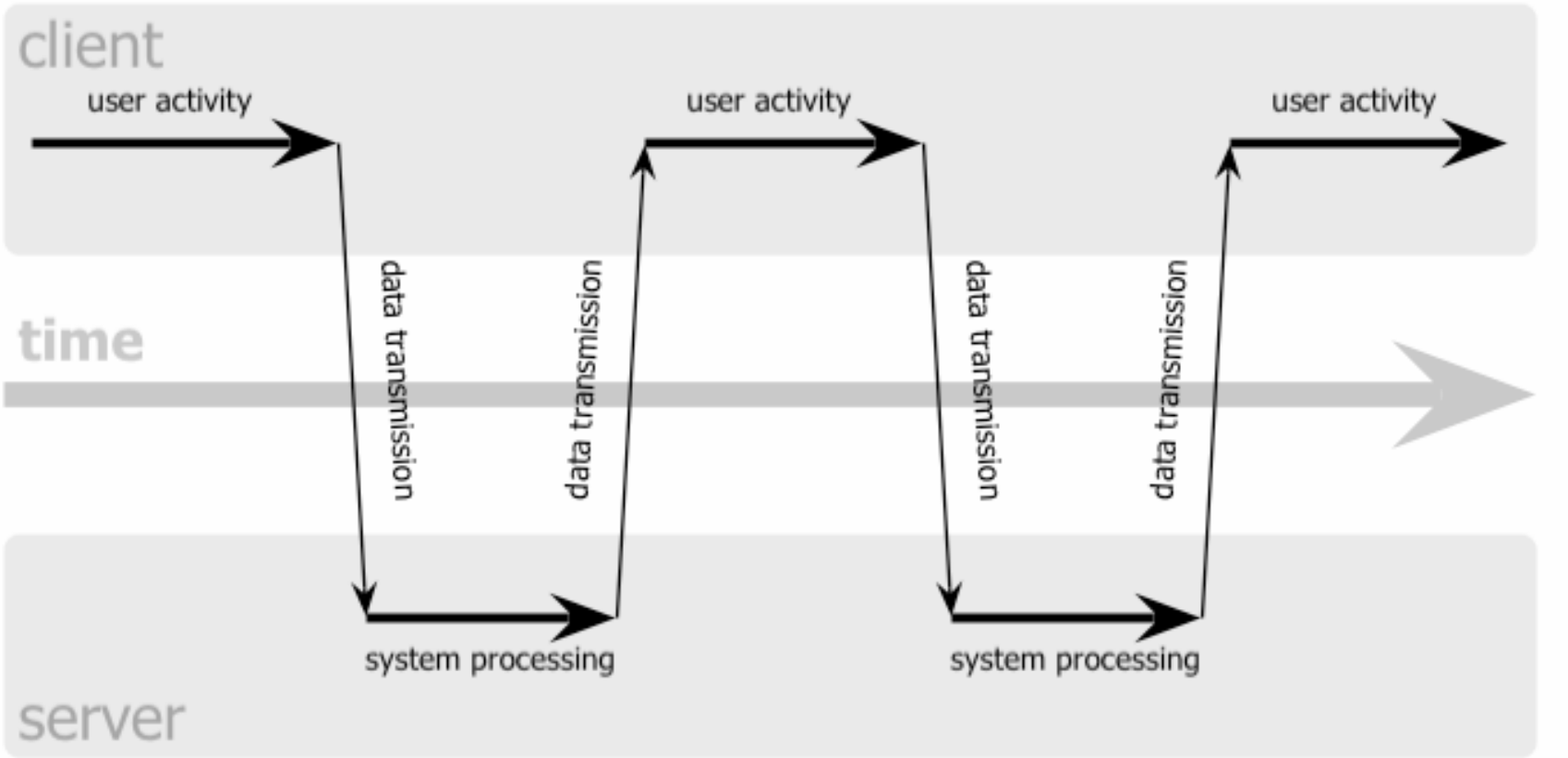
AJAX – Asynchronous JavaScript and XML

- Ansatz für besser bedienbare, reaktive Web-Anwendungen
- Browser als Thin-Client
  - JavaScript als Verarbeitungs-sprache (muss aktiviert sein!)
  - XHTML + CSS + DOM als GUI-Framework
  - Datenrepräsentation und Manipulation mit XML + XSLT
  - Datenaustausch über Format XMLHttpRequest
- Technisch teilweise schwierig
  - Zurück-Button und direkte URLs nur mit Tricks behandelbar
  - Erste Frameworks existieren



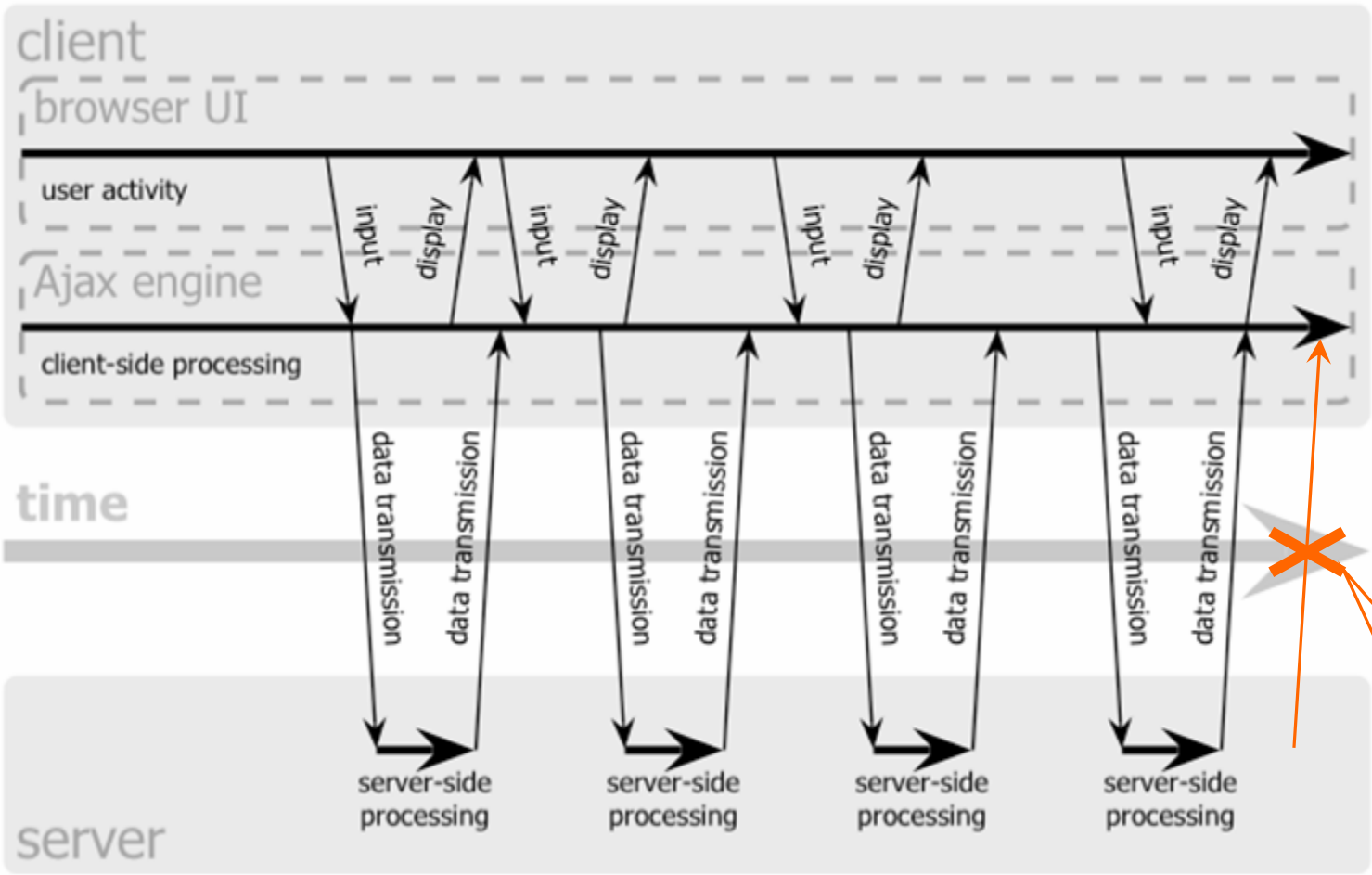
Verarbeitungsmodell bei klassischer Web-Applikation

Synchrone Aufrufe → Benutzer muss warten



Verarbeitungsmodell bei AJAX-Applikation

Asynchrone Aufrufe → Benutzer kann weiterarbeiten



Welche Technik wann?

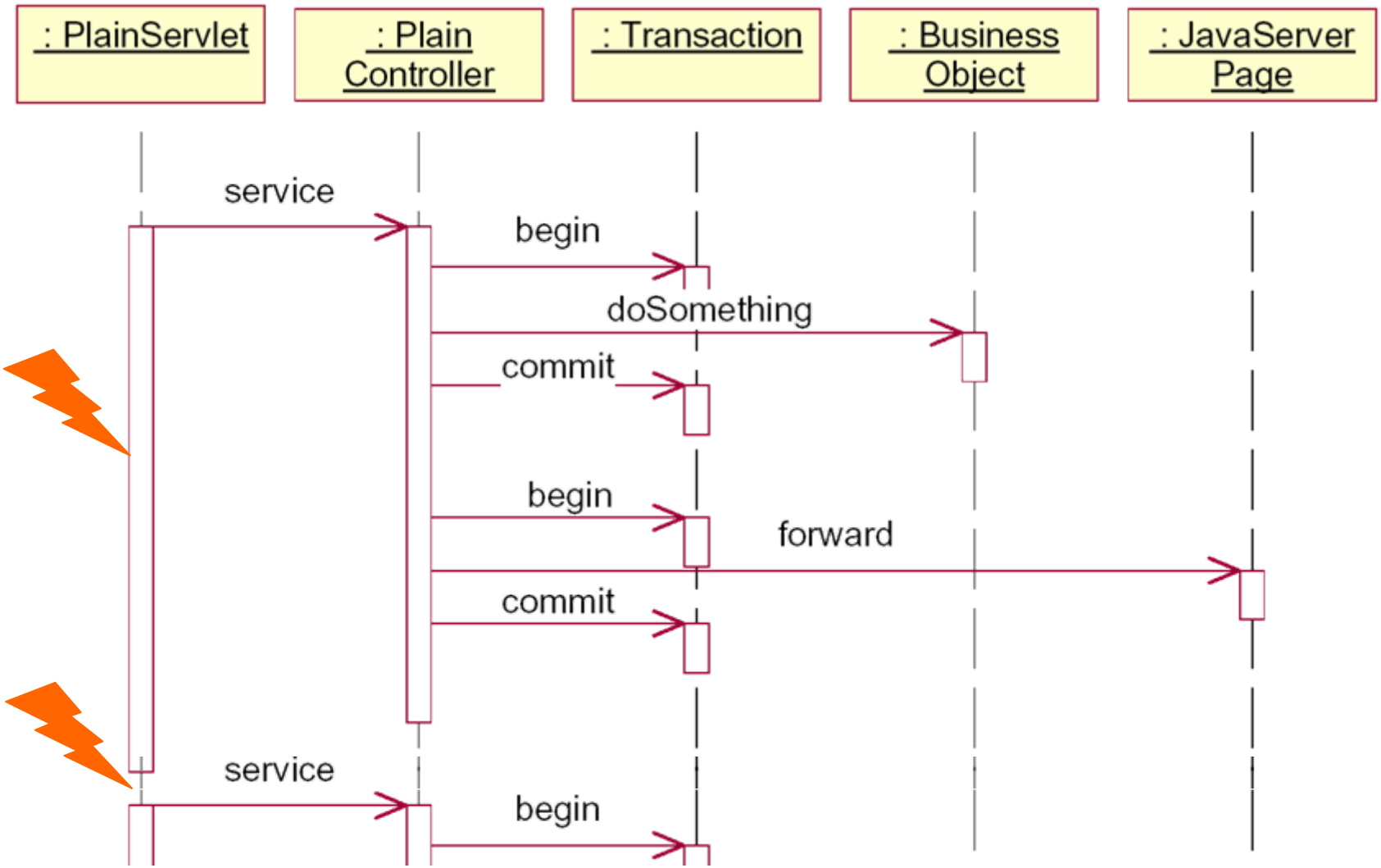
	Nativer Client	Applets	AJAX-Client	Web-Client
Mächtigkeit und Features	++ alles möglich	+ fast alles möglich	o vieles möglich	- nur seiten-orientierte Anwendungen
Performanz, Reaktivität	++	++	+	-
Installation, Deployment	- Installation	o JRE nötig	+ JavaScript nötig	++ Browser nötig
Programmier-Aufwand	++ viele Tools und Bibliotheken	++ entsprechend nativem Client	o relativ neu, erste Tools	++ viele Tools und Bibliotheken

Transaktionen bei Web-Anwendungen

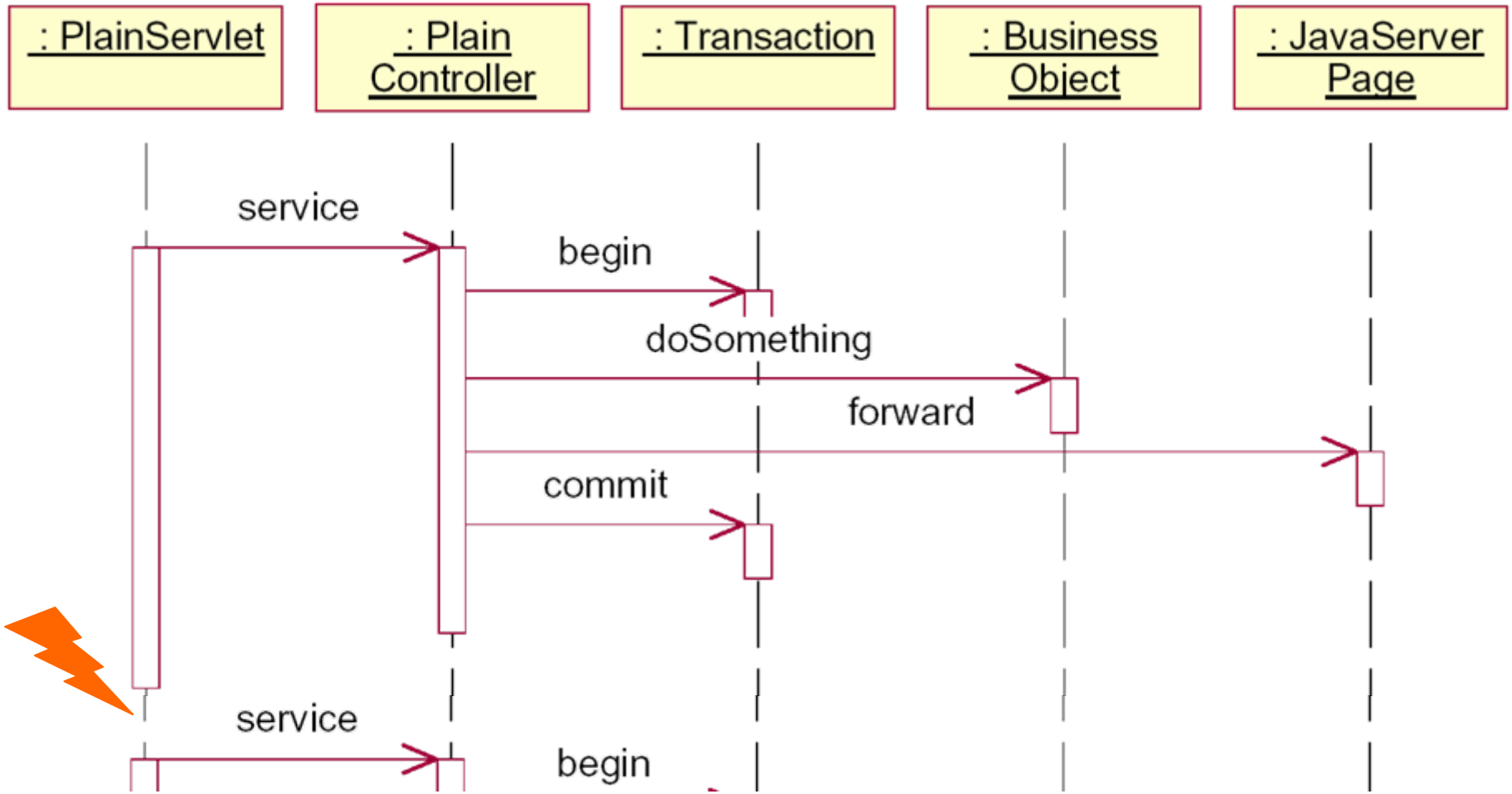
- Typisches Muster bei Web-Anwendung
  - Anwendung zeigt Formulardaten an (forward zu JSP)
  - Anwender liest Formulardaten
  - Anwender füllt Formular aus
  - Anwender schickt Antwort ab (Aufruf service-Methode bei Servlet)

- Was ist bei der weiteren Bearbeitung zu beachten?
  - Lesen und Schreiben von Daten muss transaktional erfolgen
  - Zwischen Formular-Anzeige und Abschicken der Antwort dürfen sich die Daten, auf denen das Formular beruht, nicht ändern!
  - Frage: Wie Transaktionen schneiden?

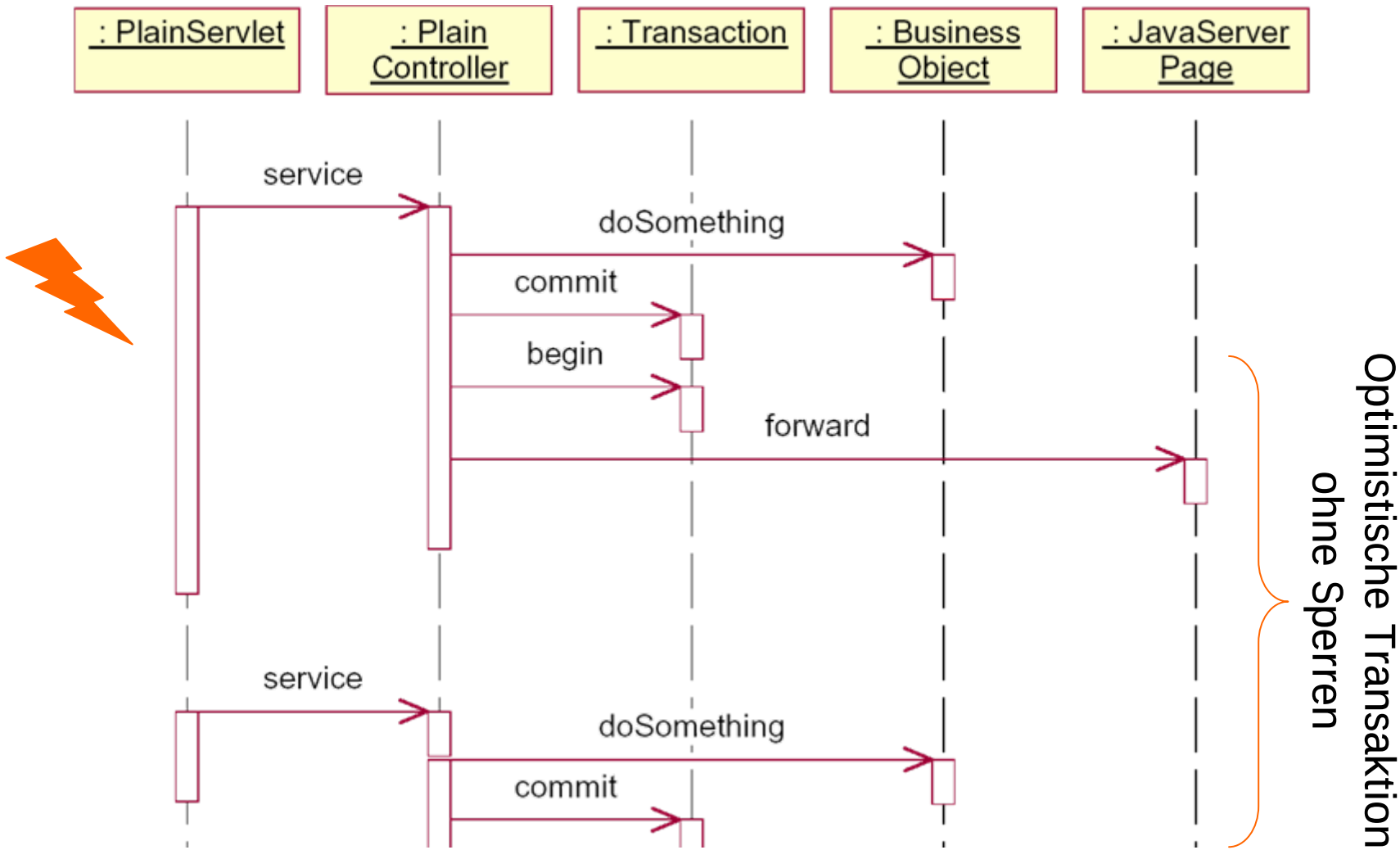
Benutzerinteraktion und Transaktionen (1)



Benutzerinteraktion und Transaktionen (2)



Client-Session und –Transaktionen (3)



Inhalt

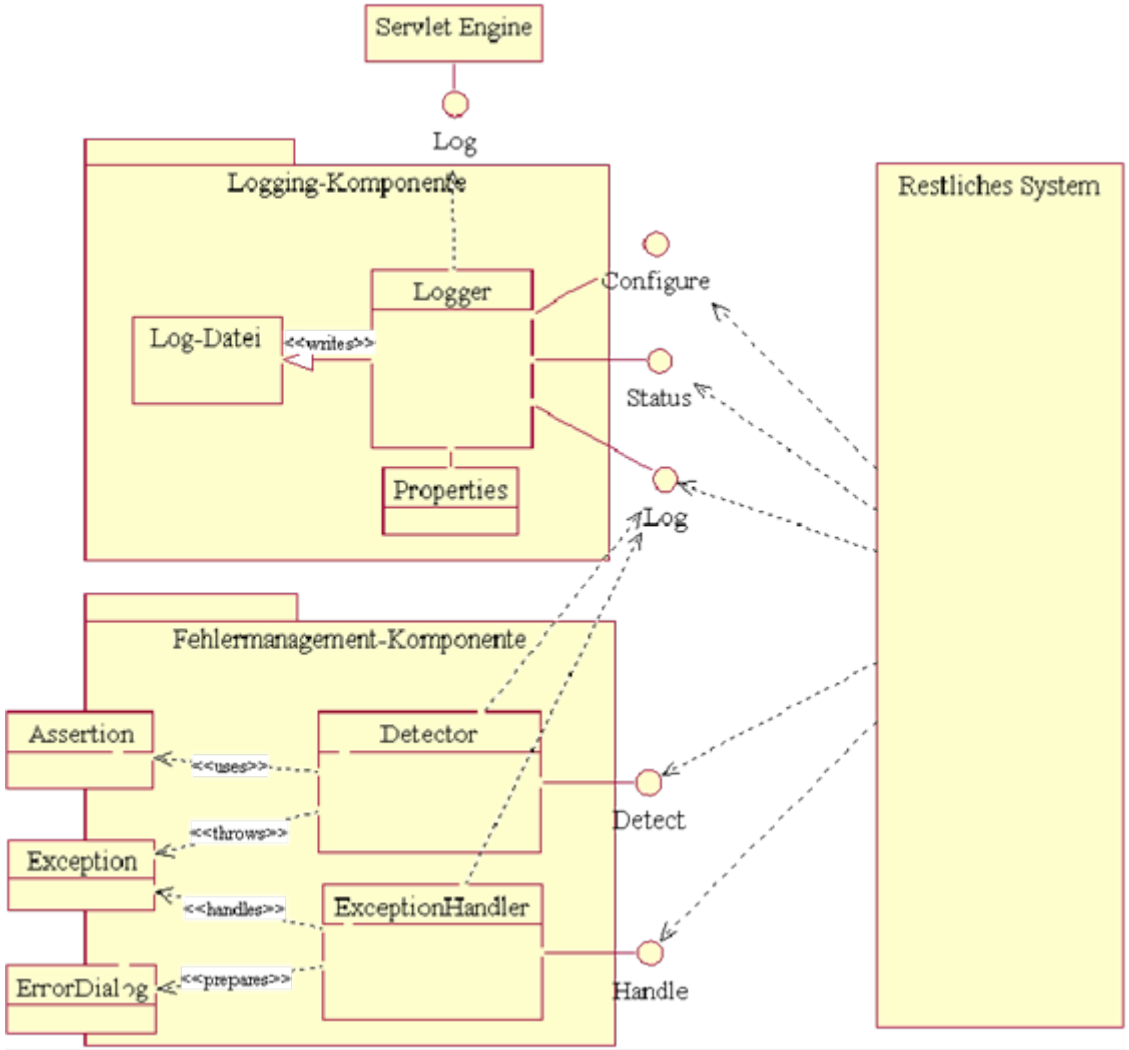
- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

Unterstützungsschicht

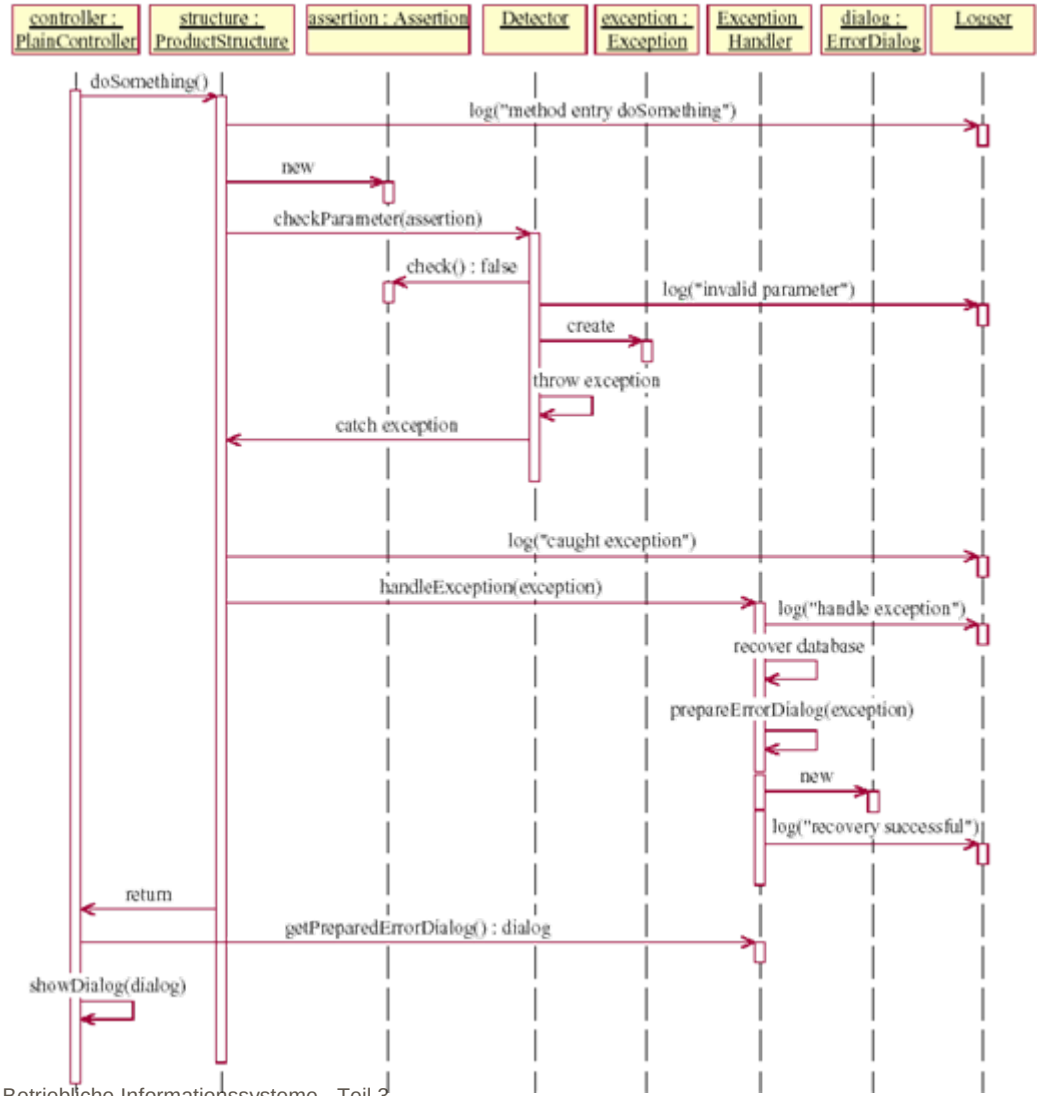
- Die Unterstützungsschicht bietet Querschnittsmechanismen, die in allen restlichen Schichten relevant sind. Dazu gehören insbesondere:
  - Fehlermanagement
  - Verwaltung von Konfigurationsinformationen
  - Authentifizierung und Autorisierung
  - Internationalisierung

- Die Architektur basiert typischerweise auf
  - Basiskomponenten für gemeinsame Funktionalität
  - Standard-Aufrufe im restlichen Code (oft gut über Aspects realisierbar)

Beispiel: Basiskomponenten für Fehlermanagement und Logging

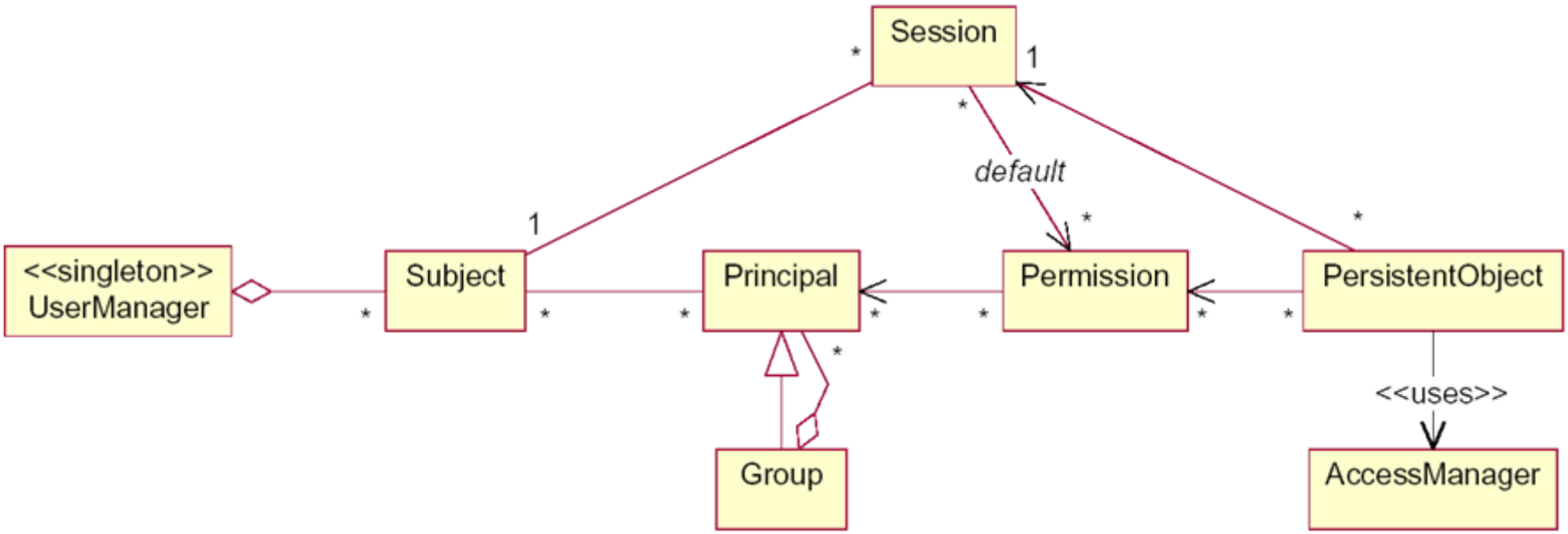


Beispiel: Aufruf von Fehlermanagement und Logging



Beispiel: Authentifizierung und Autorisierung (JAAS)

- Anreicherung von Geschäftsobjekten (PersistentObject) um eine Access Control List (Liste von Permission-Objekten)
- Mitführen des Benutzers (Principal) in einem Security Context (Session-Objekt)



Inhalt

- Rekapitulation Teil 1 und 2
- Steuerungsschicht
  - Aufgaben und Charakteristika
  - Dienste und Middleware
- Präsentationsschicht
  - Aufgaben und Charakteristika
  - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

Zusammenfassung

- Die Steuerungsschicht verwaltet Abläufe und Zustände sowie Kontext und Ressourcen von Geschäftsprozessen.
  - Sie setzt typischerweise auf den Services der Anwendungsschicht auf.
  - Als Middleware kommen Workflow Engines zum Einsatz.
- Die Präsentationsschicht steuert die Interaktion mit dem Benutzer und zeigt GUI-Elemente sowie Anwendungsdaten an.
  - Als Architektur kommt typischerweise eine Variante des Model-View-Controller-Architekturmusters zum Einsatz.
  - Grundlegend lassen sich Thin-Clients (Verarbeitungslogik läuft in Laufzeitumgebung auf dem Client) und Ultra-Thin-Clients (Verarbeitungslogik läuft auf dem Server) unterscheiden.
- Die Unterstützungsschicht bietet querschnittliche Mechanismen, beispielsweise für Fehlermanagement, Logging, Internationalisierung etc.

Literaturhinweise

- [Bi02] Adam Bien: *J2EE Patterns*, Addison-Wesley, 2002.
- [BMR+] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: *Pattern-Oriented Software Architecture*, Wiley, 1996.
- [EJB3] Sun Microsystems: *J2EE Website*. http://java.sun.com/j2ee, 2005.
- [jBPM] *jBPM WfE Homepage*, http://www.jboss.com/products/jbpm
- [Sie02] J. Siegel, *An Overview Of CORBA 3.0*, Object Management Group, 2002.
- [Spr05] Rob Harrop, Jan Machacek: *Pro Spring*, apress, 2005.
- [Spring] *Spring Framework Homepage*, http://www.springframe-work.org
- [WfMC] *Workflow Management Coalition Homepage*, http://www.wfmc.org