

Softwarearchitektur

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

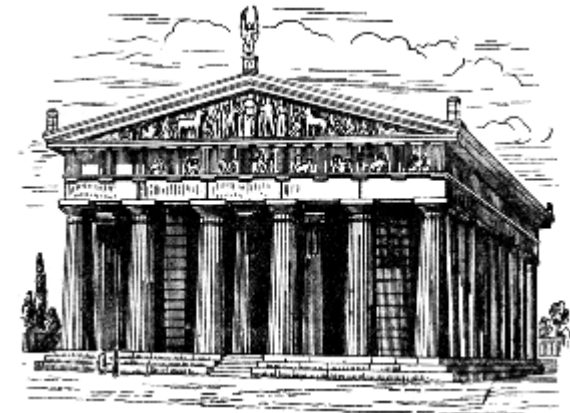
10. Betriebliche Informationssysteme – Teil 3

Vorlesung Wintersemester 2008 / 09

Technische Universität München

Institut für Informatik

Lehrstuhl von Prof. Dr. Manfred Broy



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

Inhalt

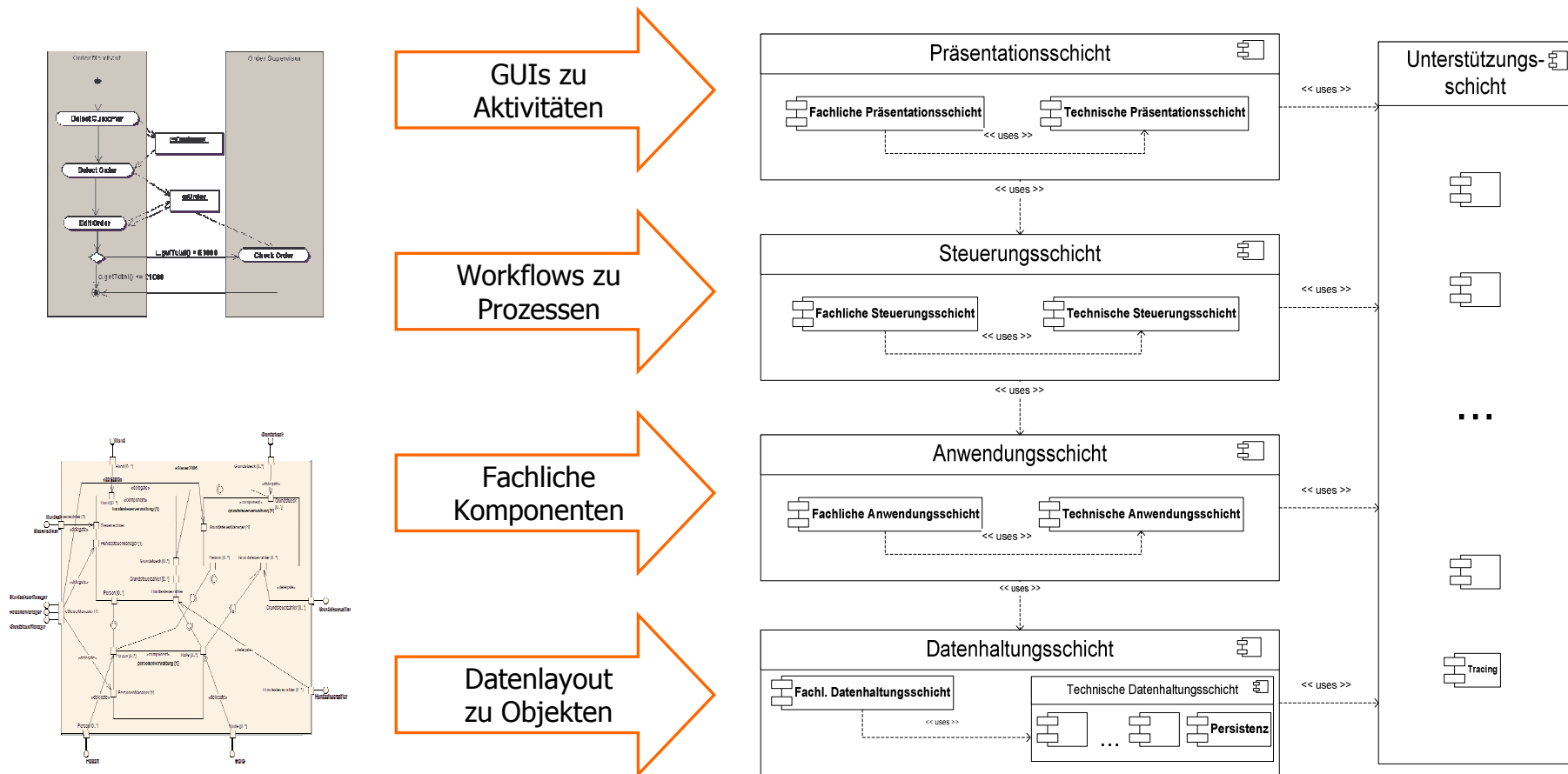
- Rekapitulation Teil 1 und 2
- Steuerungsschicht
 - Aufgaben und Charakteristika
 - Dienste und Middleware
- Präsentationsschicht
 - Aufgaben und Charakteristika
 - Model-View-Controller-Architekturen
- Unterstützungsschicht
- Zusammenfassung
- Literaturhinweise

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.



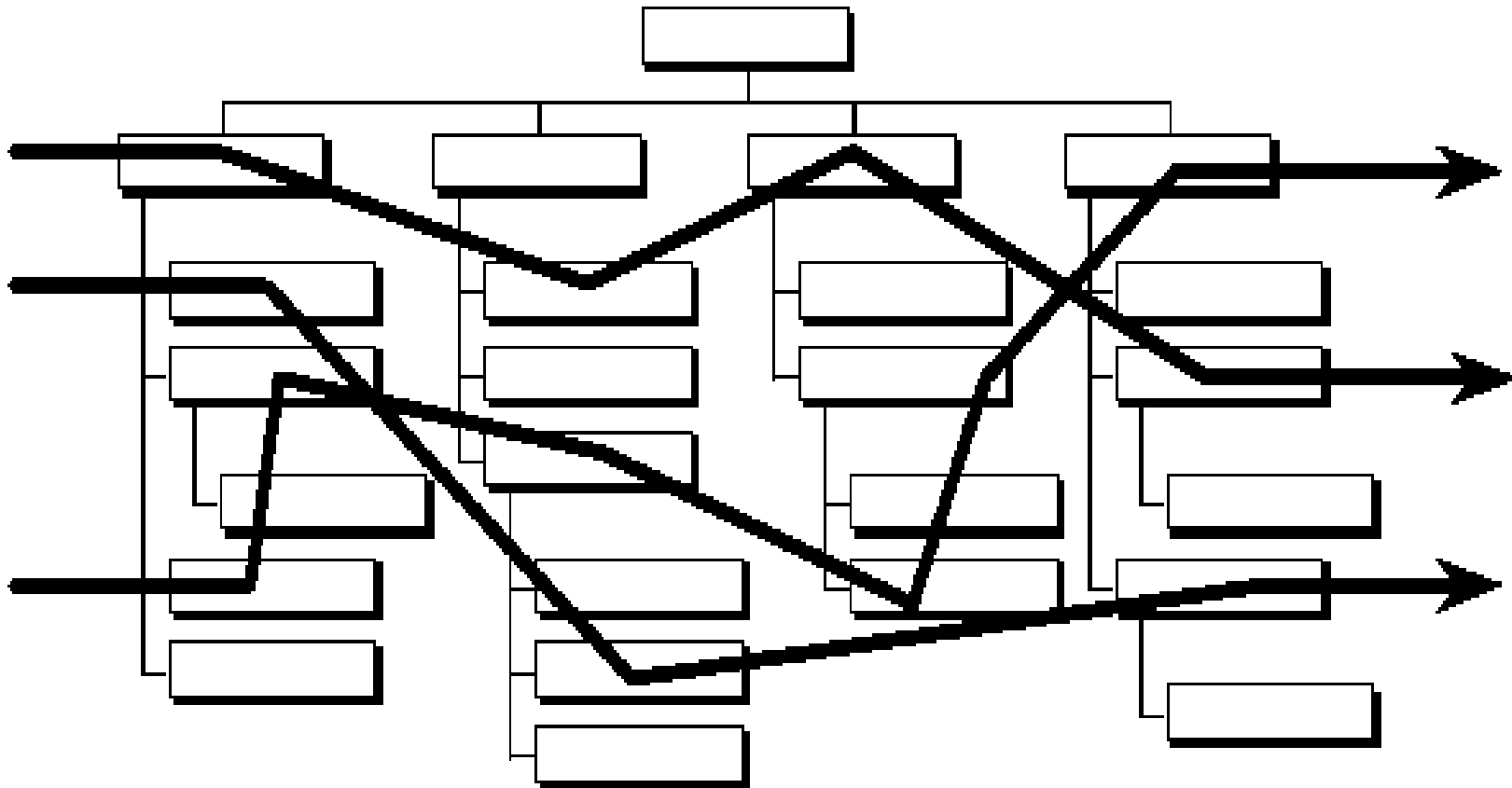
Inhalt

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

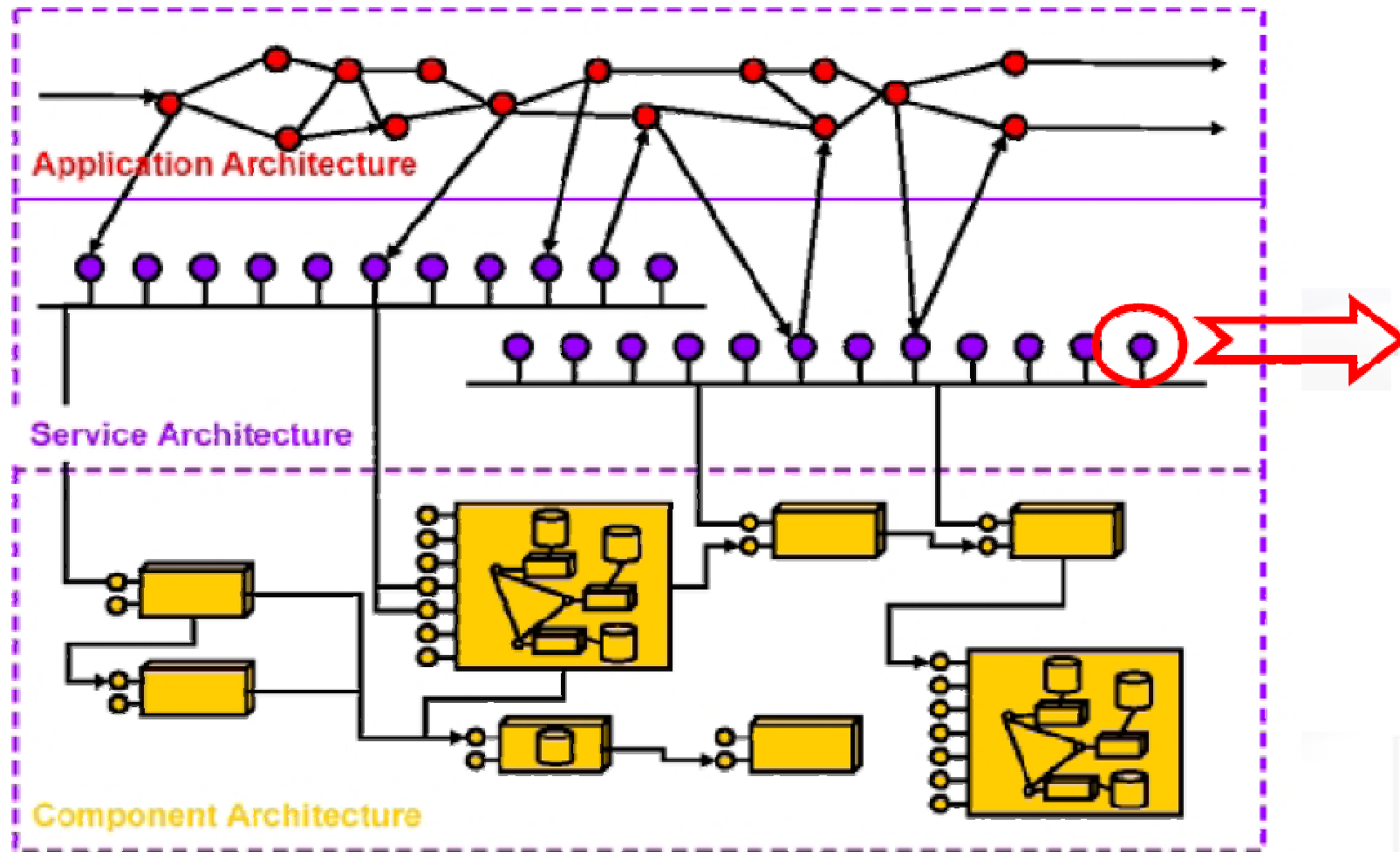
Aufgabe der Steuerungsschicht

- 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

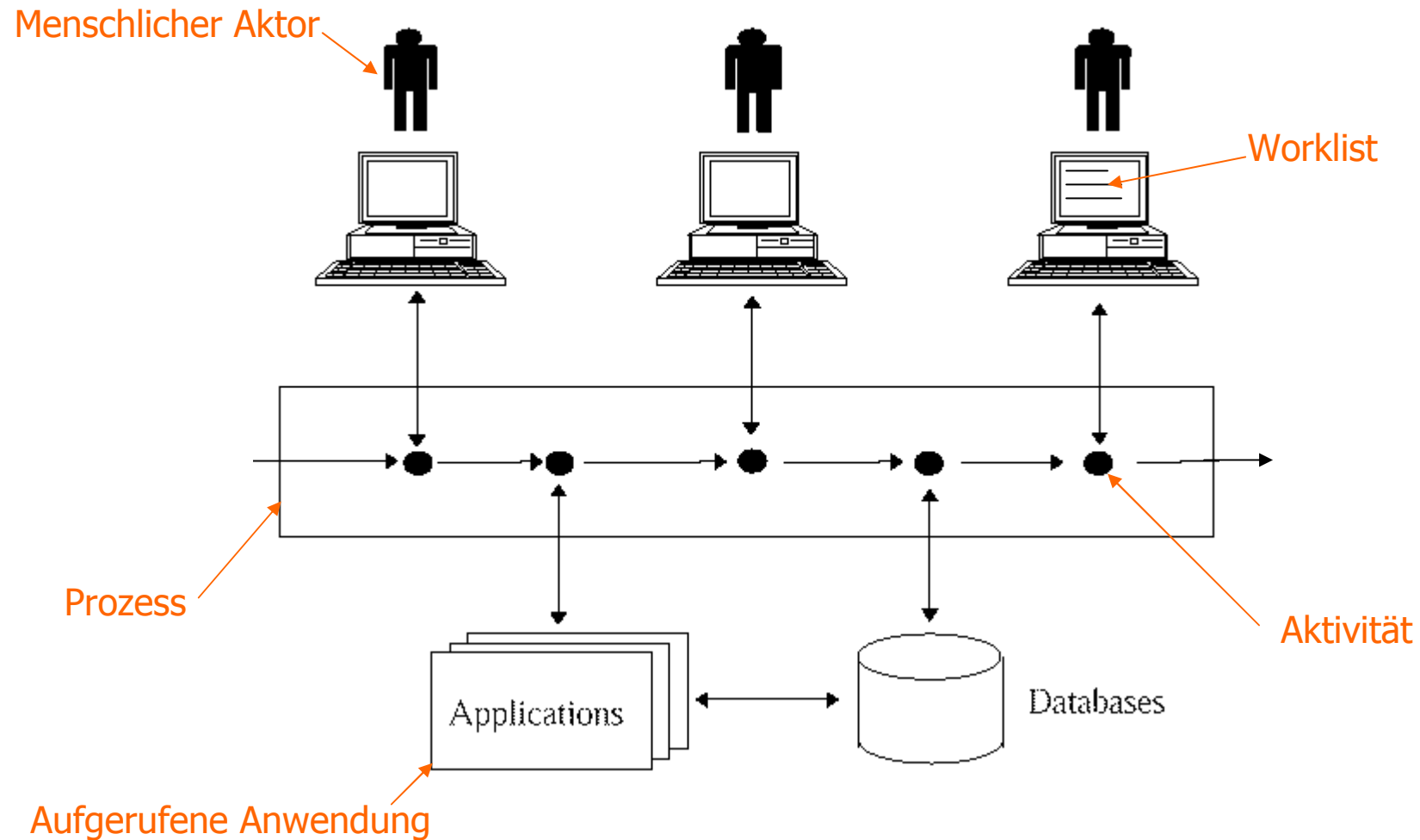
Geschäftsprozesse und Organisationen



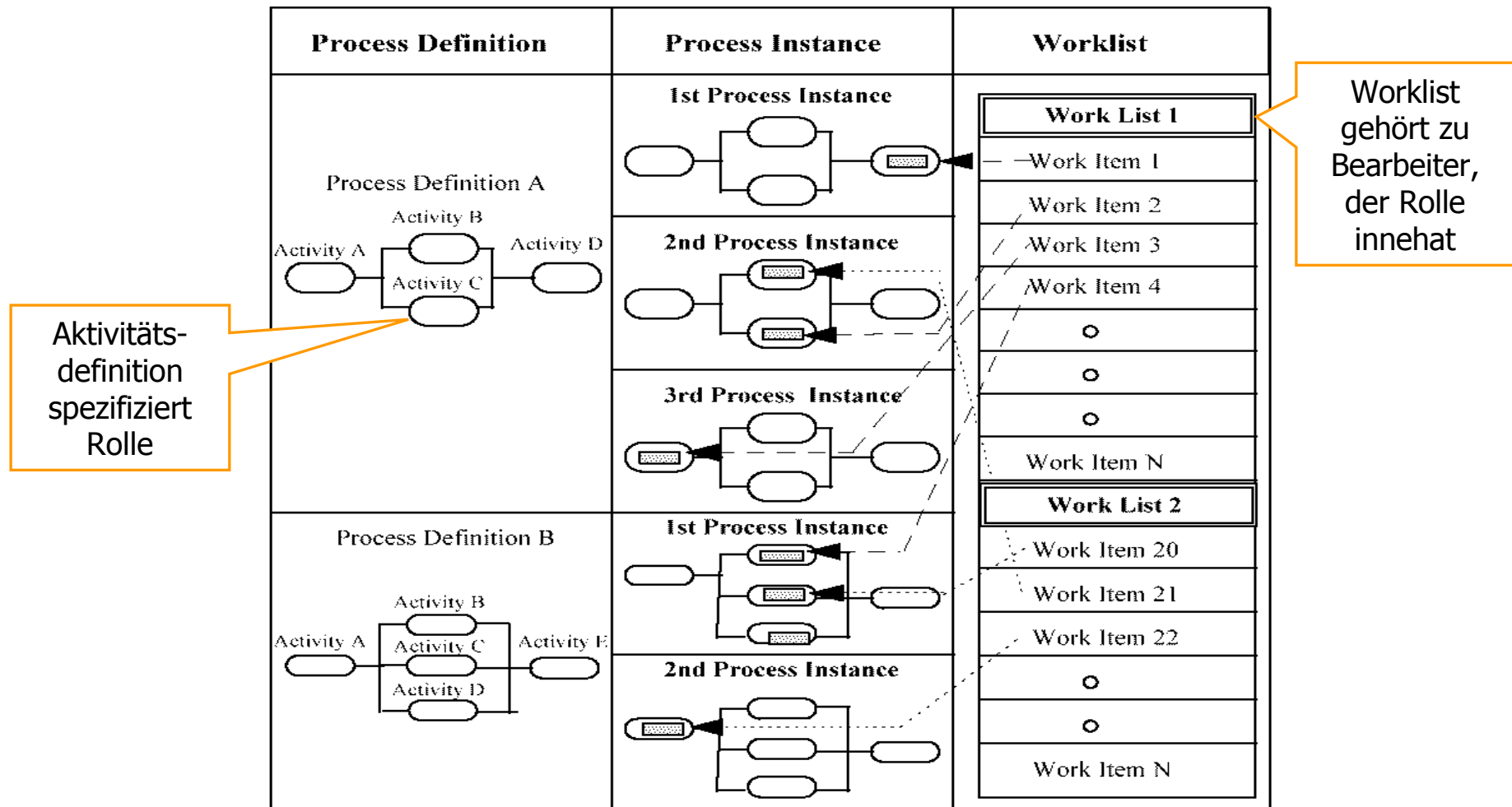
Services als Anwendungs-Bausteine



Workflow Management

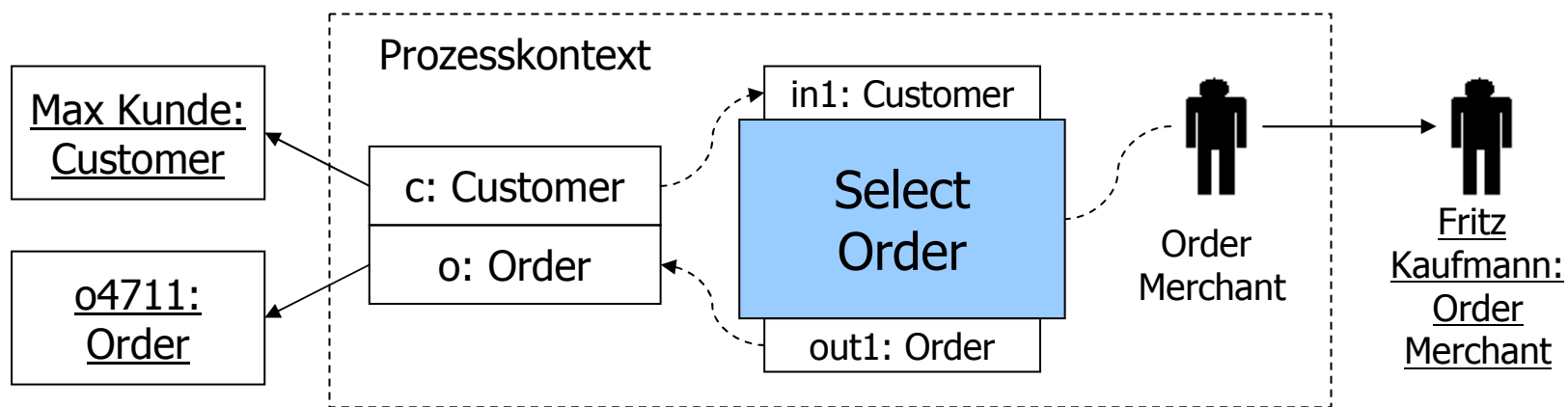


Workflow: Verwaltung von Abläufen und Zuständen

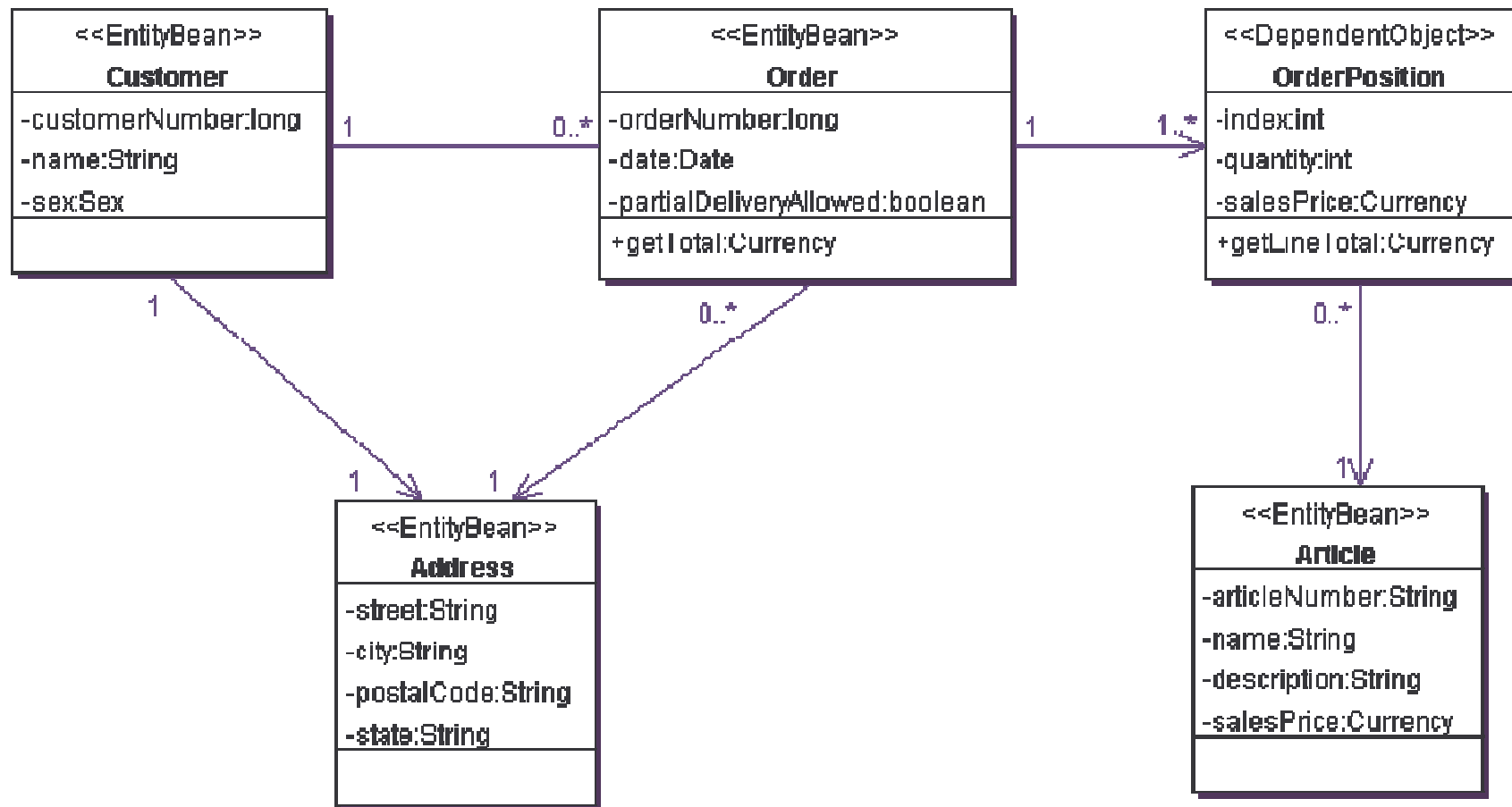


Workflow: Kontext- und Ressourcenverwaltung

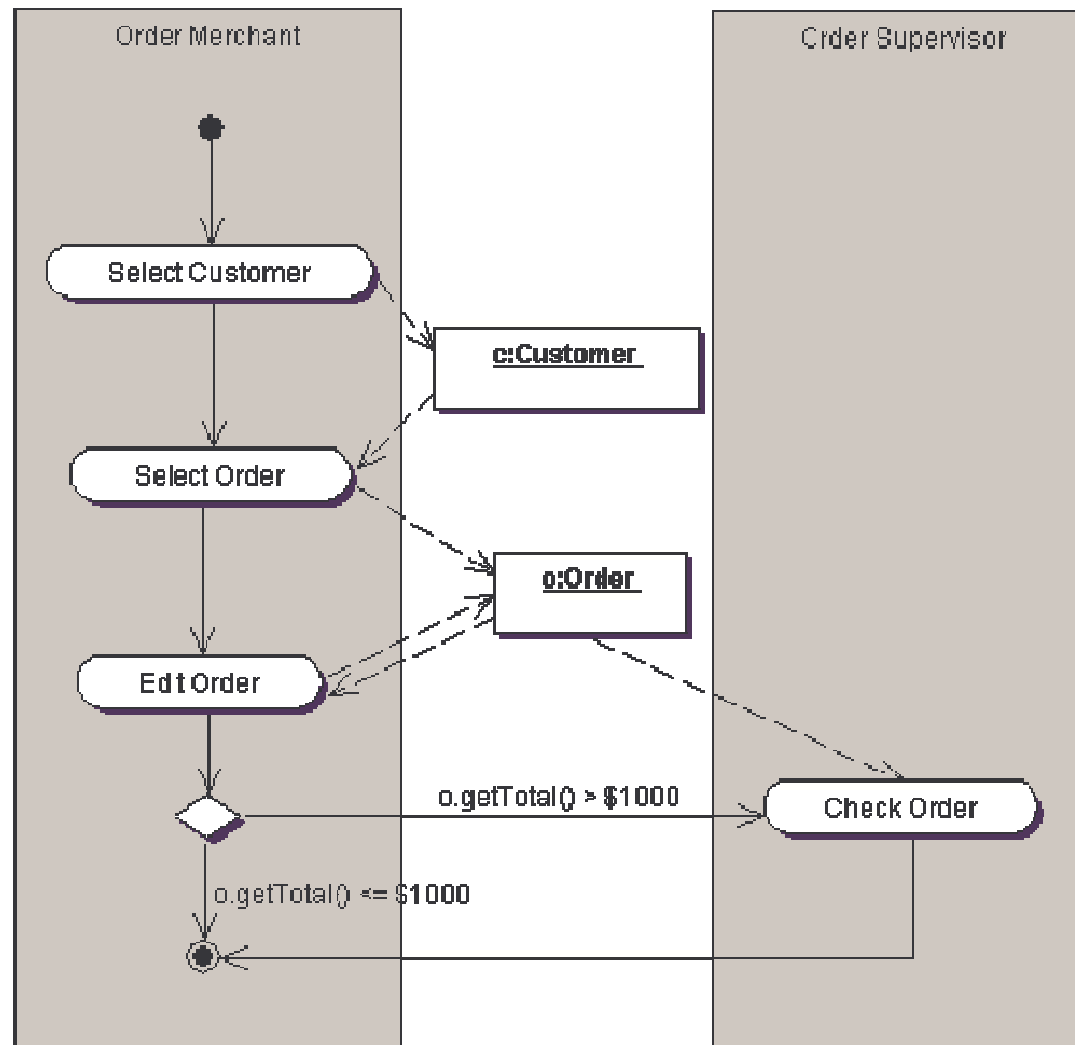
- Aktivitäten sind möglichst kontextlos (→ Service-Orientierung)
 - spezifizieren lediglich Ein-/Ausgabeschnittstelle
 - damit einsetzbar im Kontext beliebiger Prozesse
- Workflow-Engine stellt Kontext zur Verfügung
 - Versorgt Ein-/Ausgabeschnittstelle mit aktuellen Kontext-Daten
 - Per Value: Skalare 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)

The screenshot shows the Domino Client interface for an order system. The window title is "Domino Client (jclav@HighQ: Endverbraucher)". The menu bar includes "Server", "Process", "Properties", "View", "Window", and "Help". The toolbar contains "Login...", "Refresh", "Worklist", "Process Templates", and "Content...".

On the left side, there is a "Worklist" pane showing a tree view with "olav" and "employee". Below it, a "Process-definitions" pane shows a tree view with "article:" (containing "NewArticle" and "ModifyArticle"), "order" (containing "NewOrder" and "ModifyOrder"), and "customer" (containing "NewCustomer" and "ModifyCustomer").

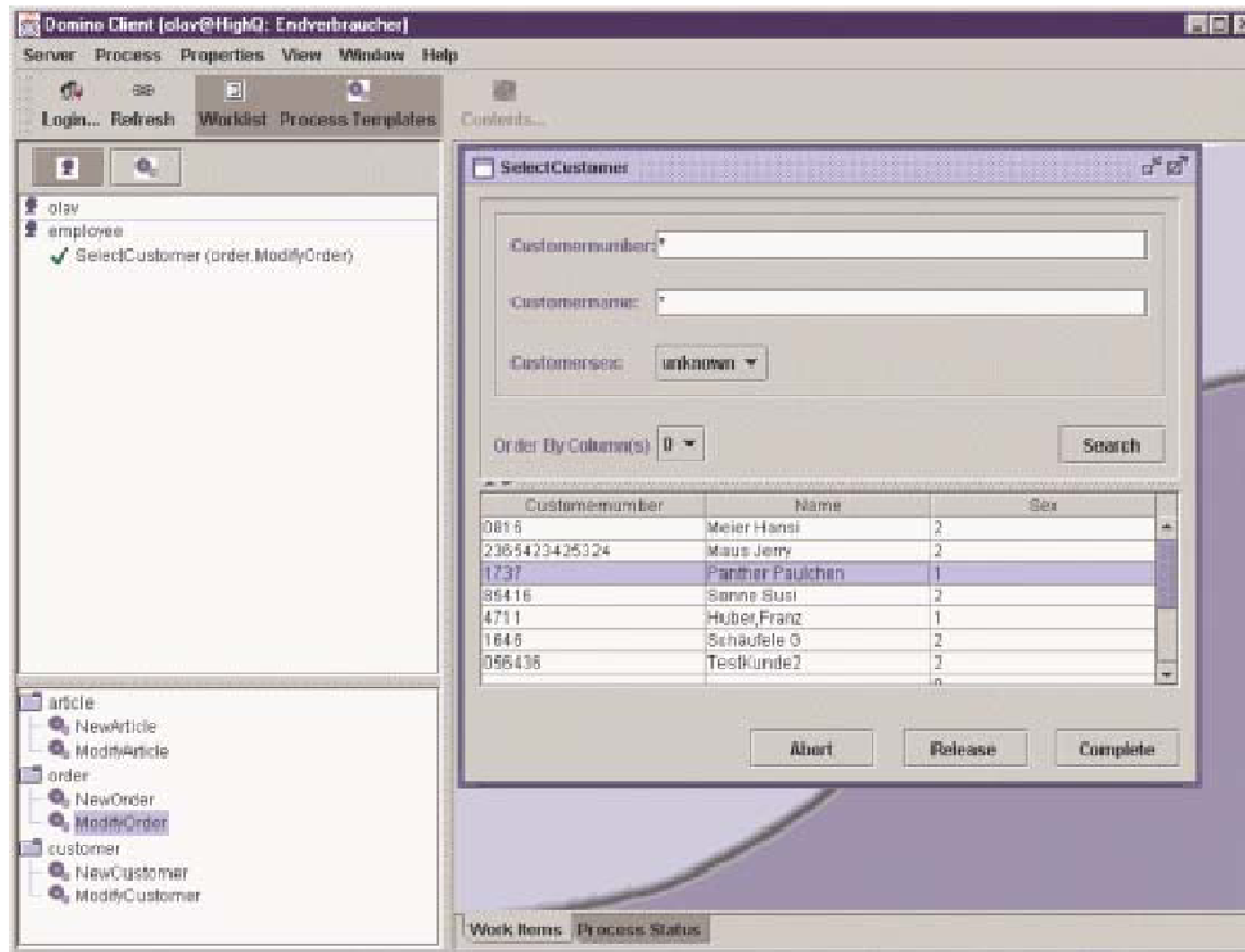
The main area on the right is the "Arbeitsbereich" (workspace), which is currently empty and displays a large, abstract graphic. A small dialog box is open in the center, titled "Process Name:", with a text input field containing "order.ModifyOrder" and "OK" and "Cancel" buttons.

At the bottom of the workspace, there are two tabs: "Work Items" and "Process Status".

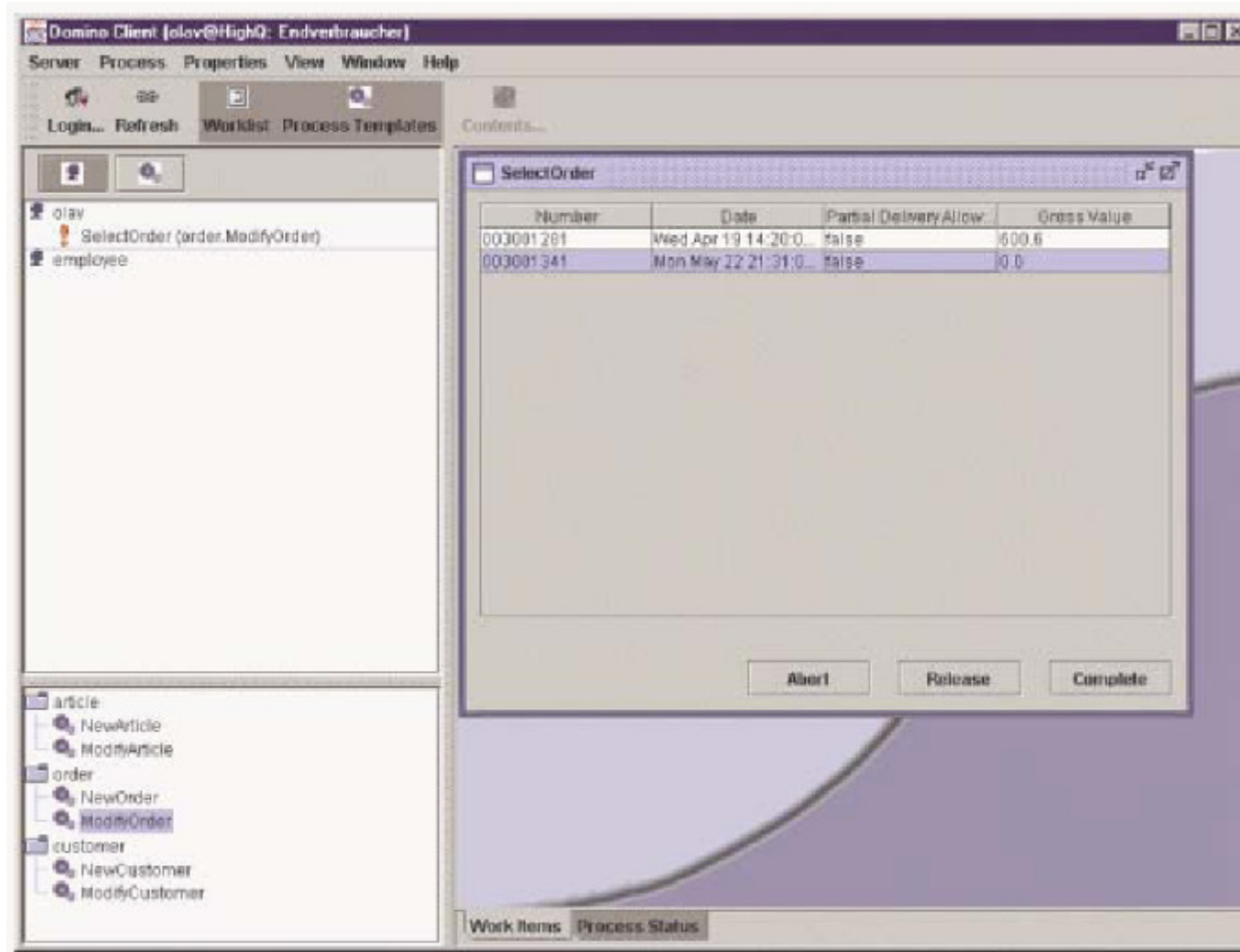
Callouts in orange boxes point to specific elements:

- "Worklist (Aktivität und Ablauf)" points to the left-hand tree view.
- "Arbeitsbereich: aktive Aktivitäten" points to the main workspace area.
- "Prozessdefinitionen" points to the bottom-left tree view.

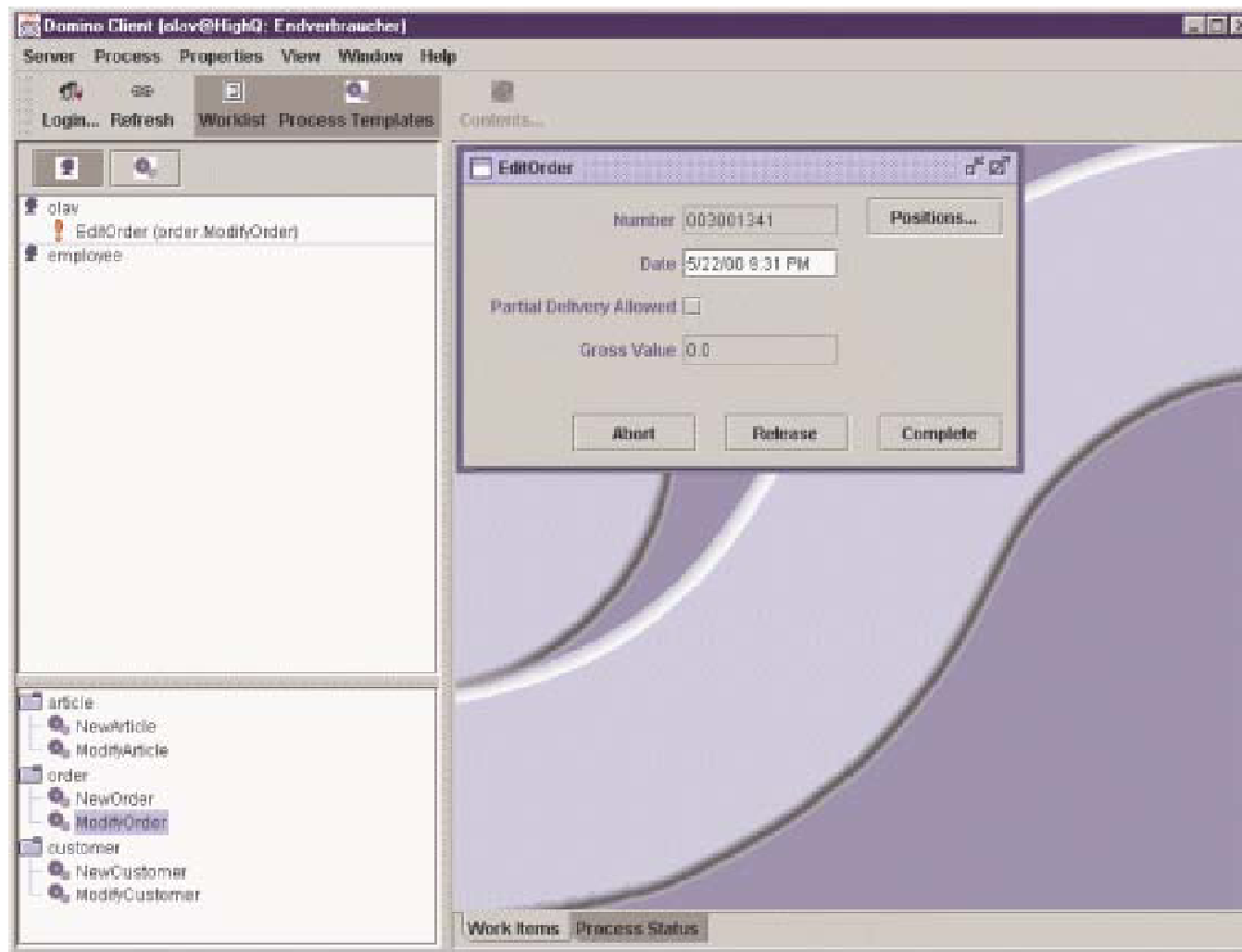
Beispiel: Order-System (2)



Beispiel: Order-System (3)



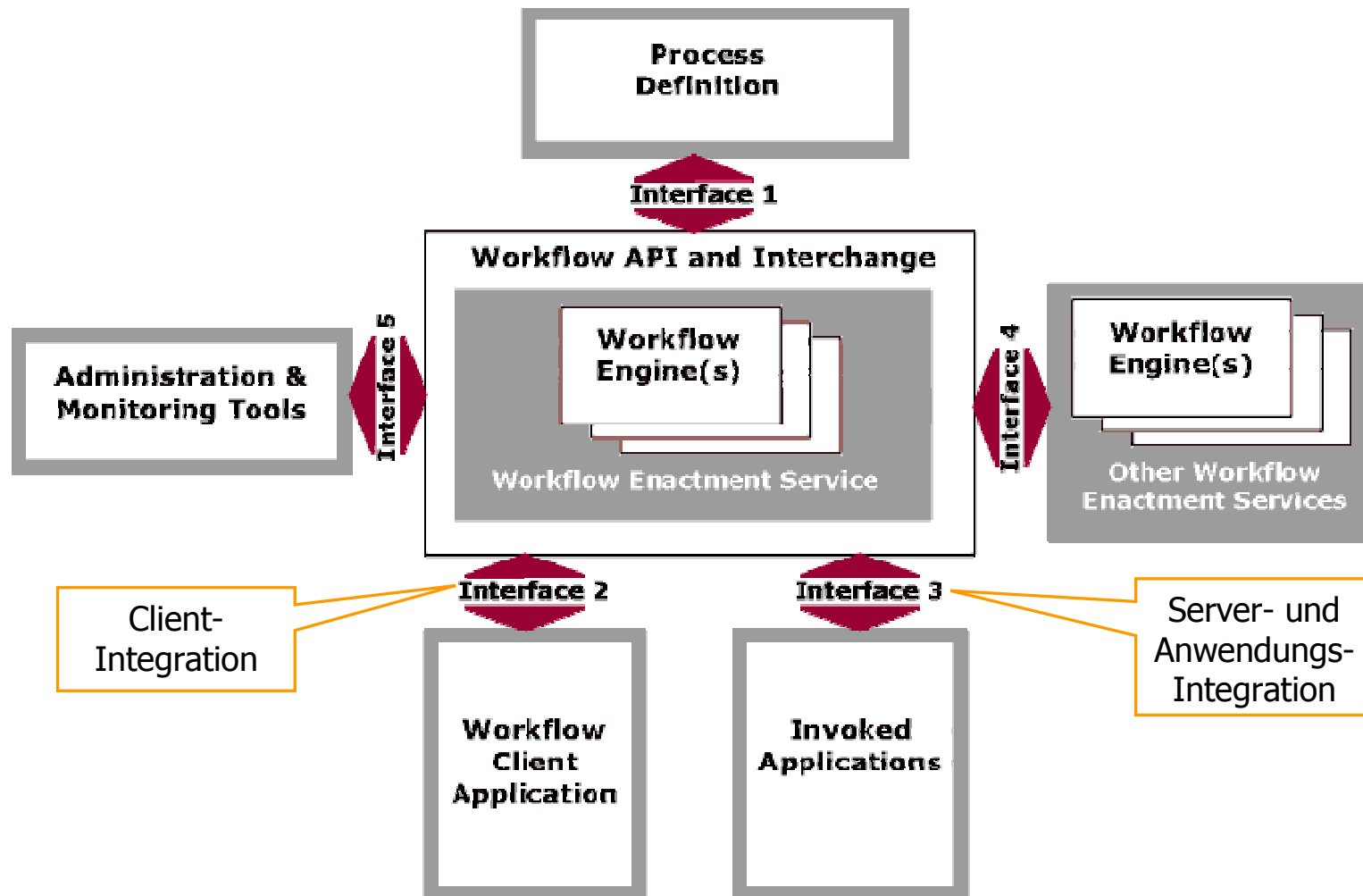
Beispiel: Order-System (4)



Middleware zum Workflow Management

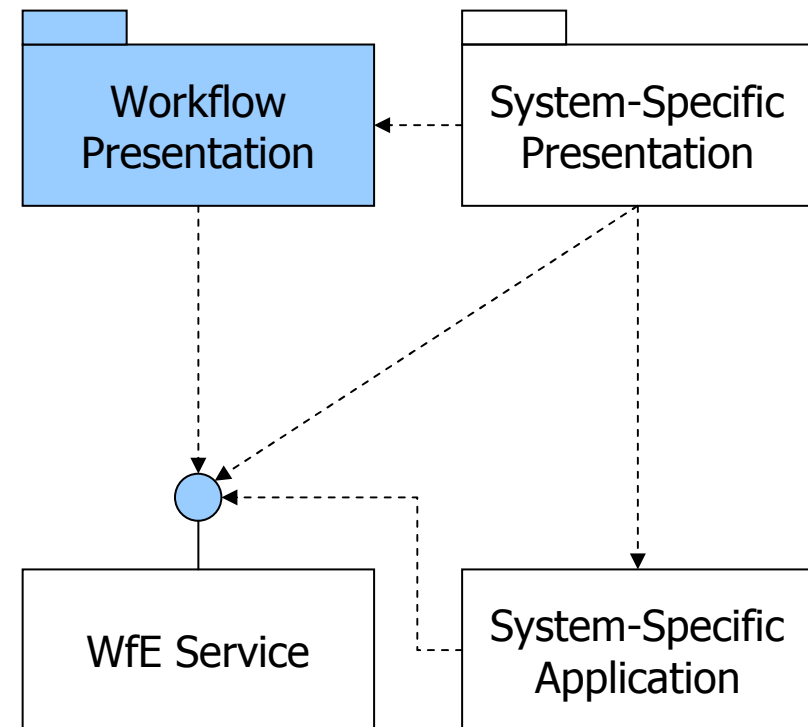
- Realisierung der Steuerungsschicht
 - Abläufe als Programmcode
 - ± Nur bei kleinen Systemen tragfähig
 - Verwaltung der Prozesse durch Workflow Engine
 - + Nutzung von mächtigen Mechanismen (s. nächste Folien)
 - + Konfiguration und Modellierung statt Programmierung
 - Ggf. Integration mit System 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-Engine als Standardisierungsansatz im Umfeld Web Services, oft integrierter Dienst von ESBs (Enterprise Service Busses)

WfMC Workflow Reference Model



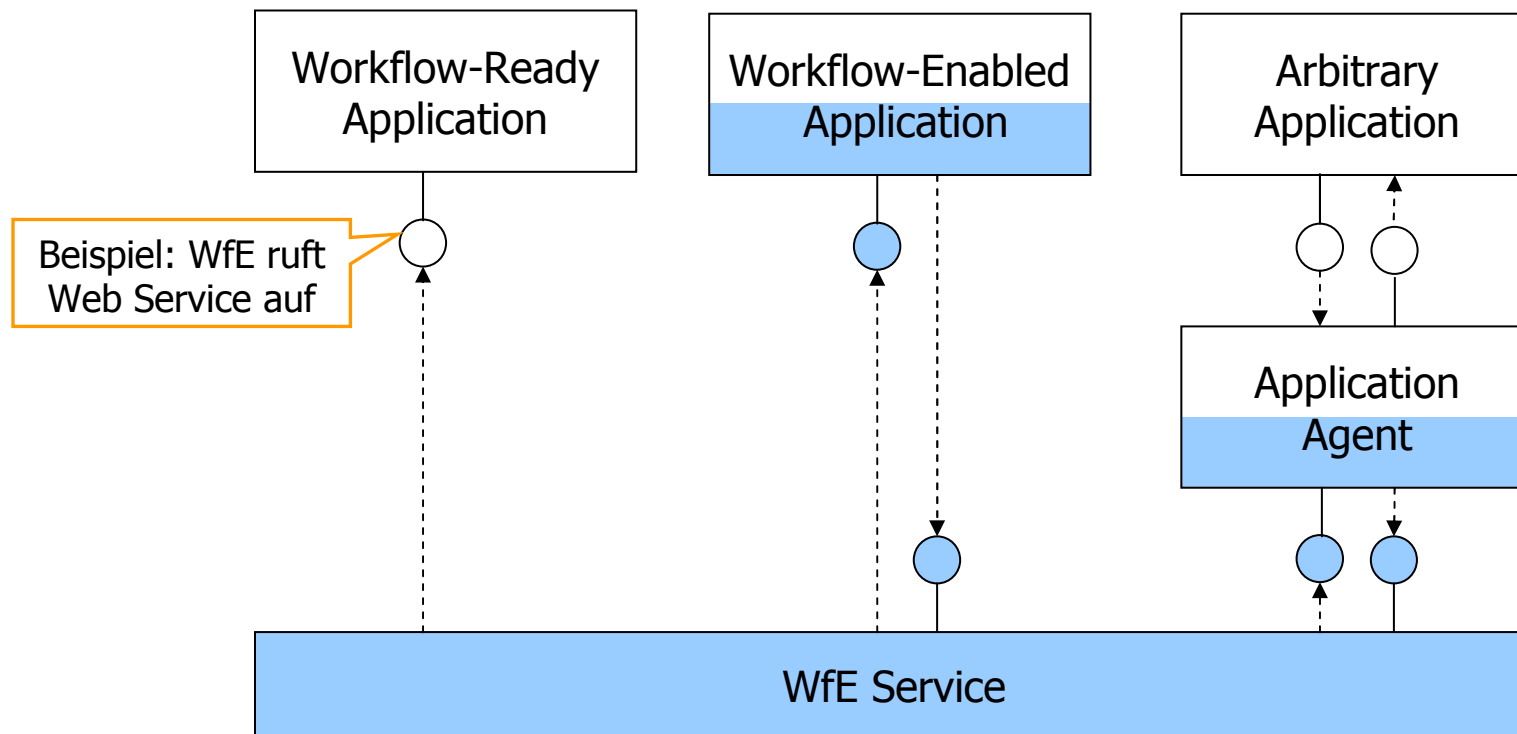
Workflow Client Application (Interface 2)

- 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



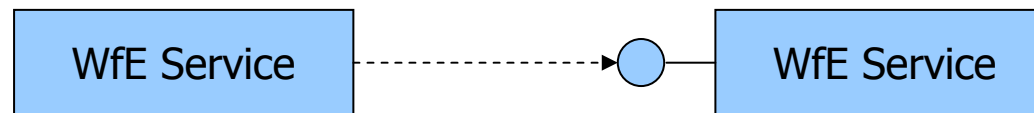
Invoked Application (Interface 3)

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



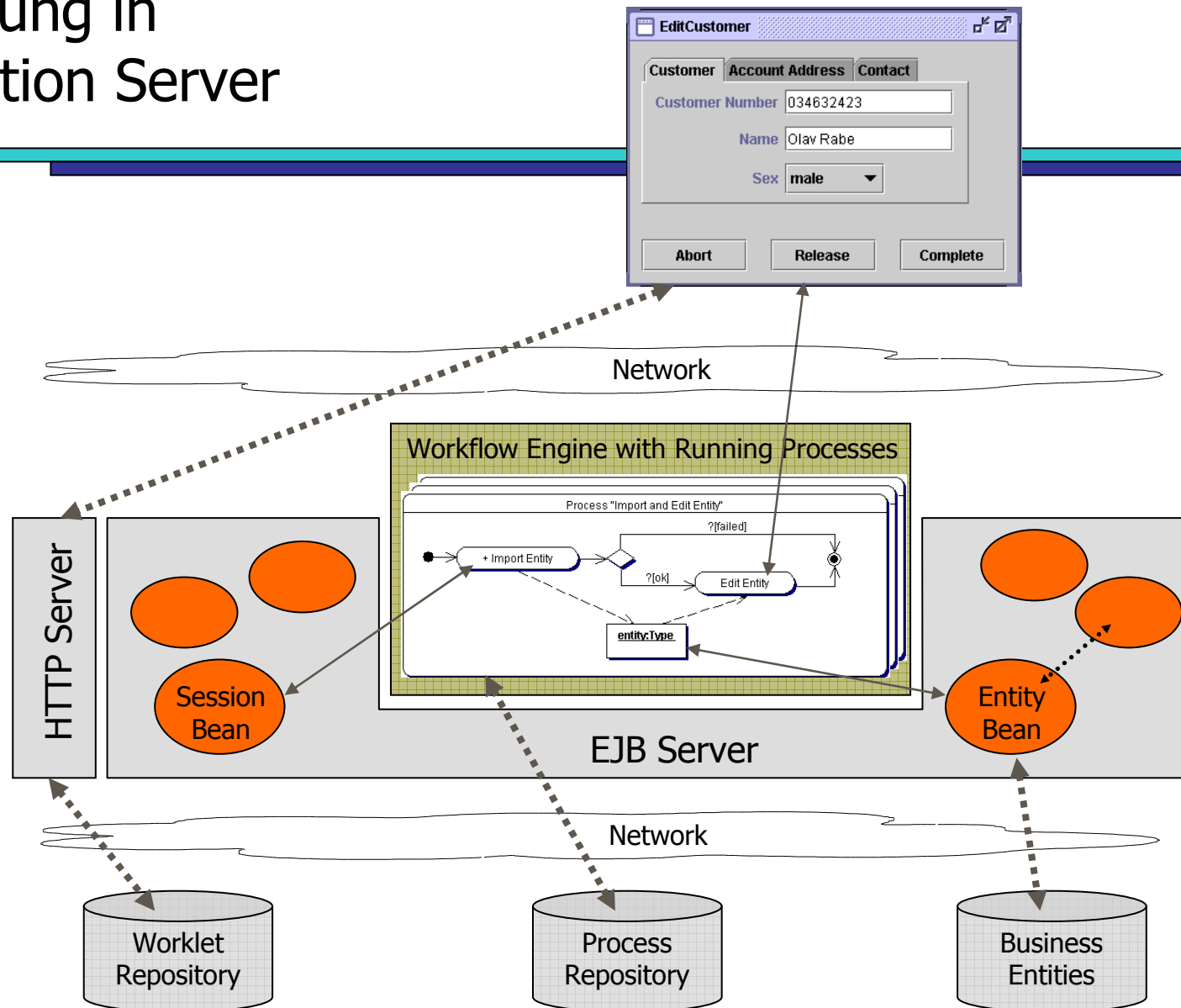
Workflow Interoperability (Interface 4)

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

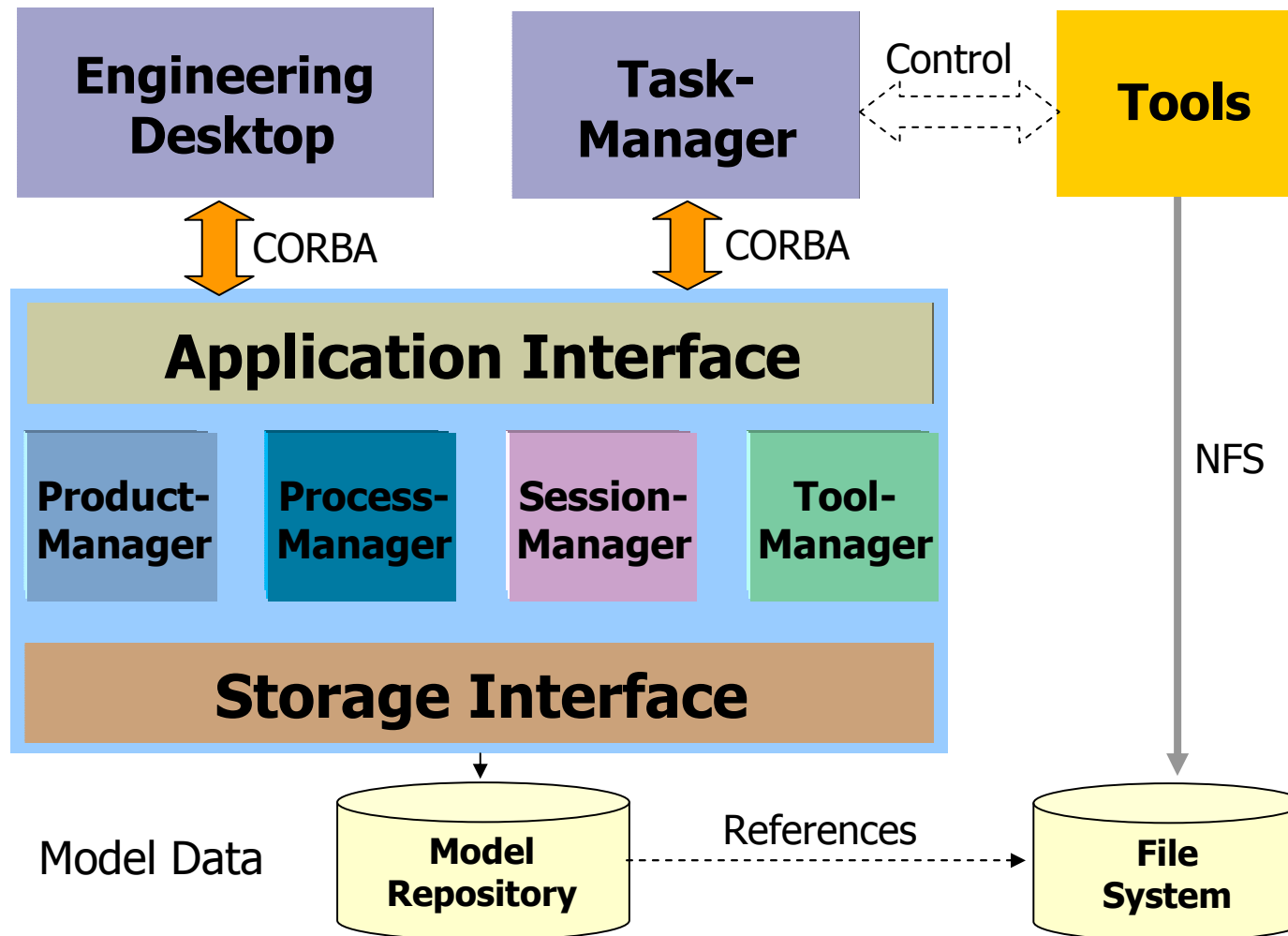


- Standardisierter Ansatz: BPEL, Business Process Execution Language
 - Kommunikation über Web-Services in XML-Form
 - Aufgerufene Anwendungen stellen Web-Services bereit
 - Aufruf von Prozessen erfolgt über Web-Services
 - Primitive zum Senden, Empfangen, Antworten auf Nachrichten
 - Ablaufsteuerungsfunktionen: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

Einbettung in Application Server



Beispiel: Architektur eines Engineering-Workflow-Systems



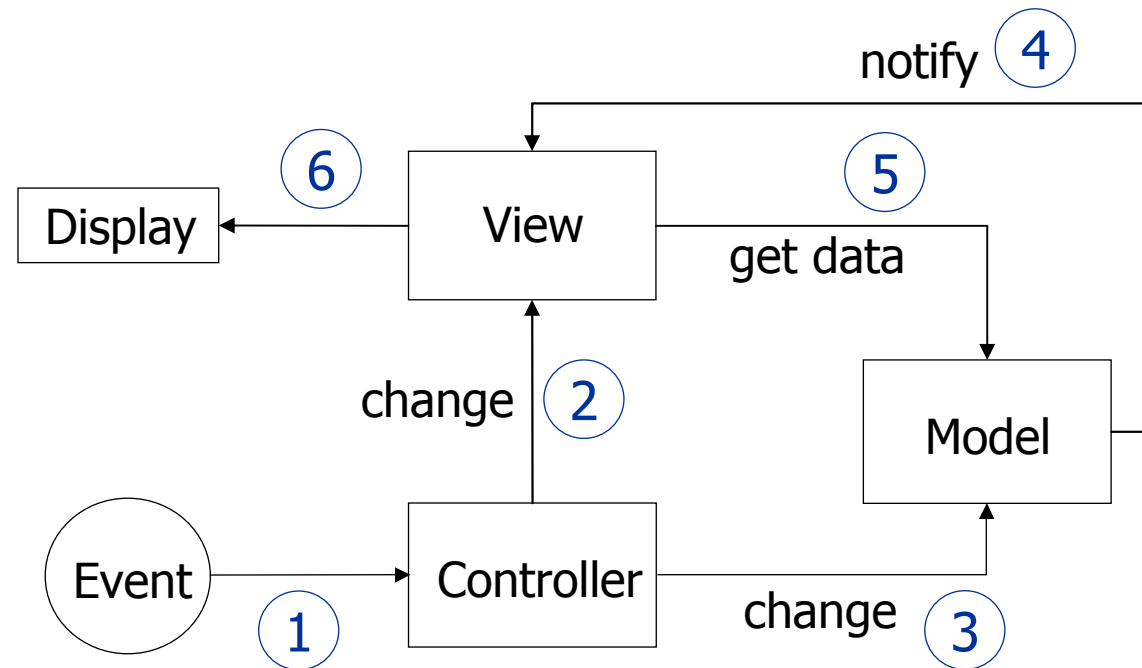
Inhalt

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

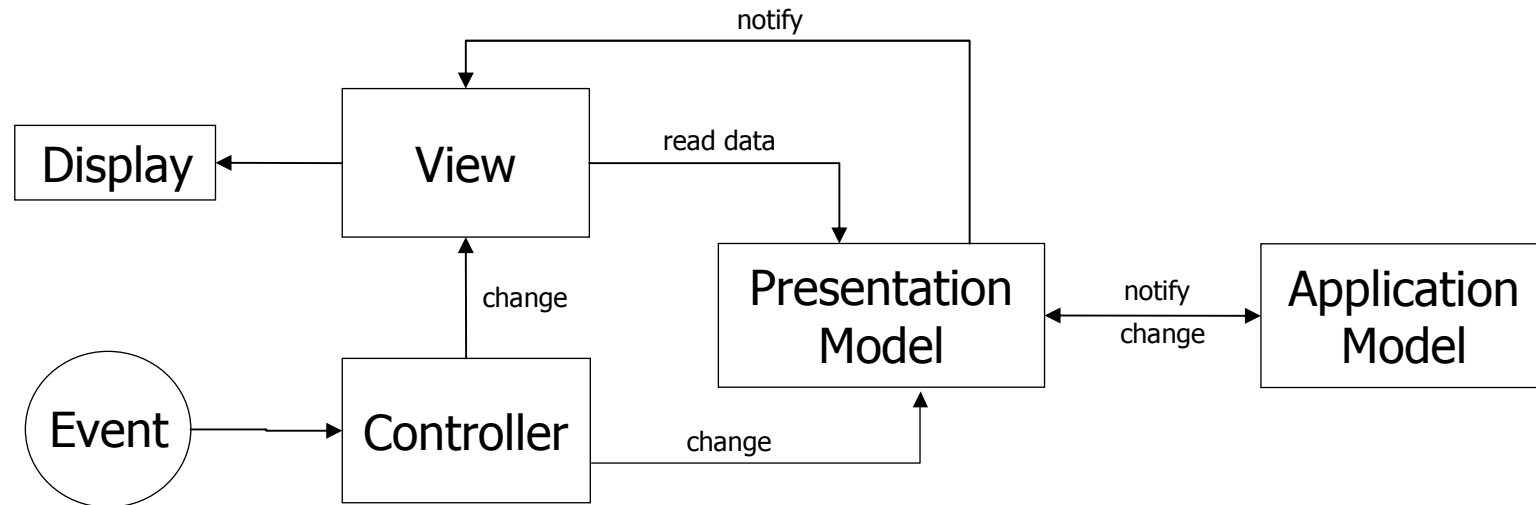
Aufgabe der Präsentationsschicht

- 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



Entwurfsmuster: Aufteilung des Models



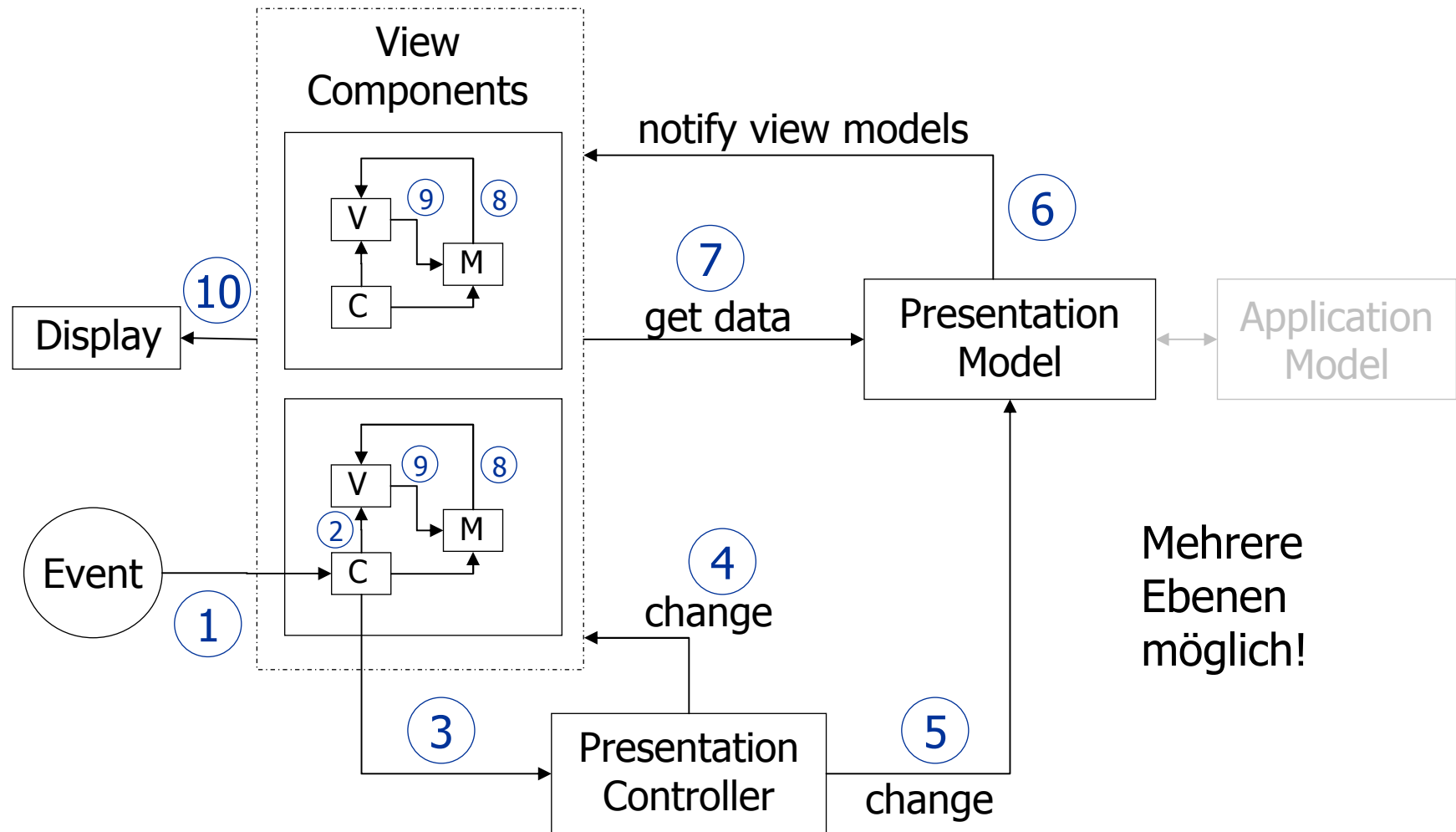
Presentation Model

- Bereitet Anwendungsdaten so auf, dass Views und Controller leicht damit arbeiten können (z.B. TreeModel)
- Enthält möglichst wenig Anwendungslogik (z.B. nur Prüfungen)

Application Model

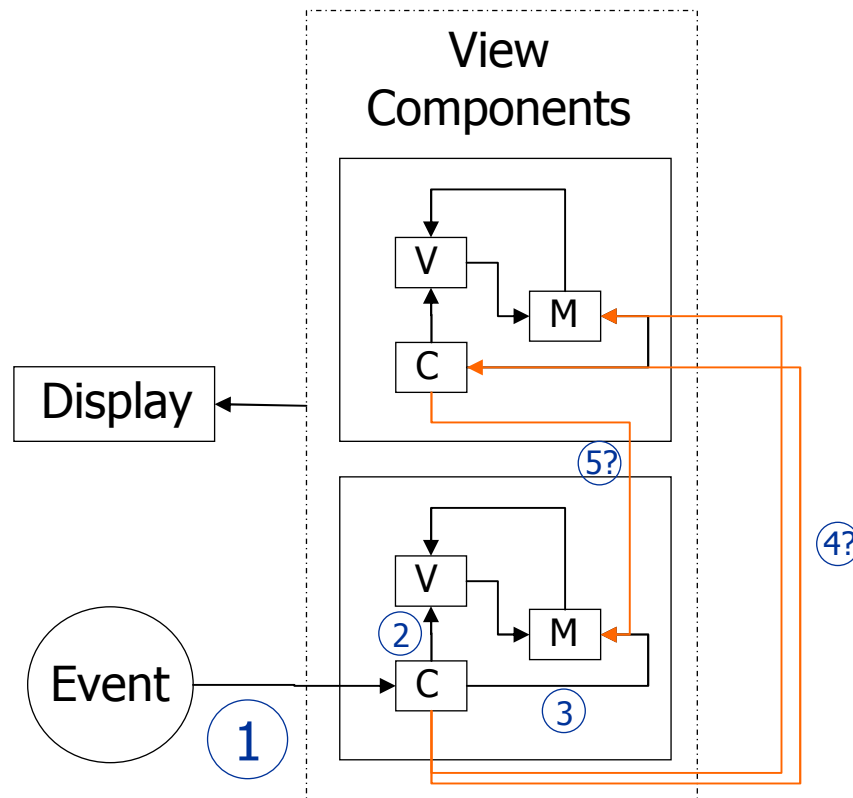
- Bildet Anwendungsfunktionalität in allgemeiner Form ab
- Enthält die Anwendungslogik

GUI-Komponenten bei hierarchischem MVC



Anti-Pattern: Spagetti-Controller

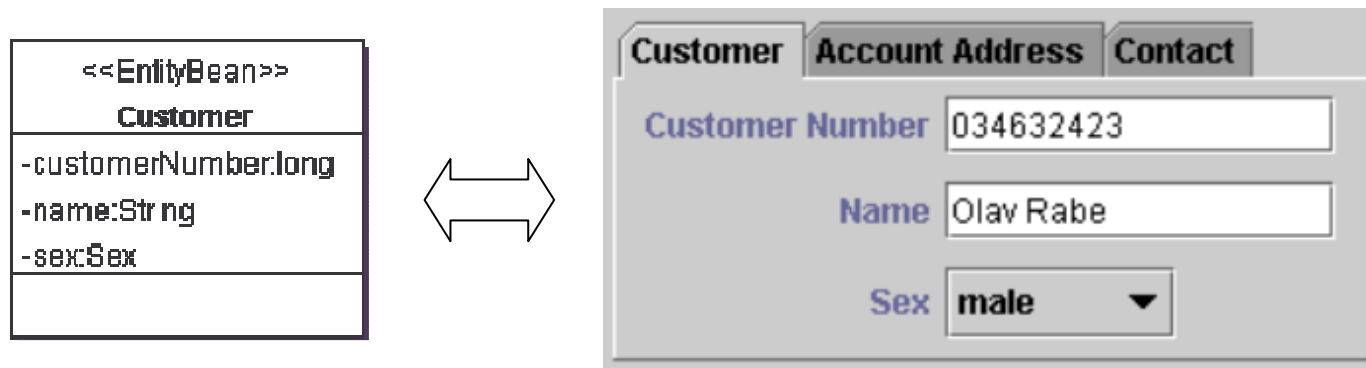
- Presentation Controller und/oder Presentation Model fehlen
- Controller und Models ändern andere Controller und Models



- Schlechte Wartbarkeit und Erweiterbarkeit, da View-Komponenten nicht mehr unabhängig voneinander sind
- Hohe Wahrscheinlichkeit für Fehler, insbesondere wenn sich Schleifen beim Aufruf ergeben

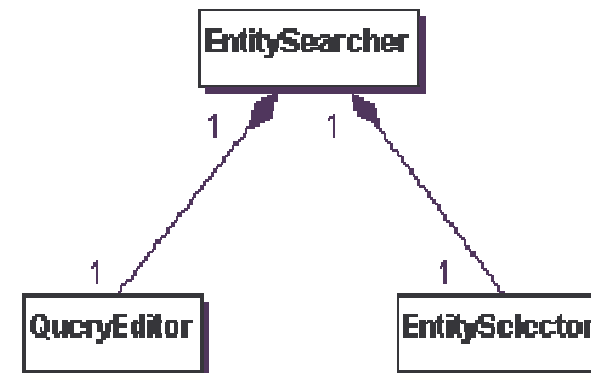
Generische GUI-Komponenten (1)

- 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)

Customer number	Name	Sex
0816	Meier Hansi	2
7365473475374	Maus Jerry	?
1737	Panther Paulchen	1
85416	Sonne Susi	2
4711	Huber, Franz	1
1646	Schäufele G	?



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

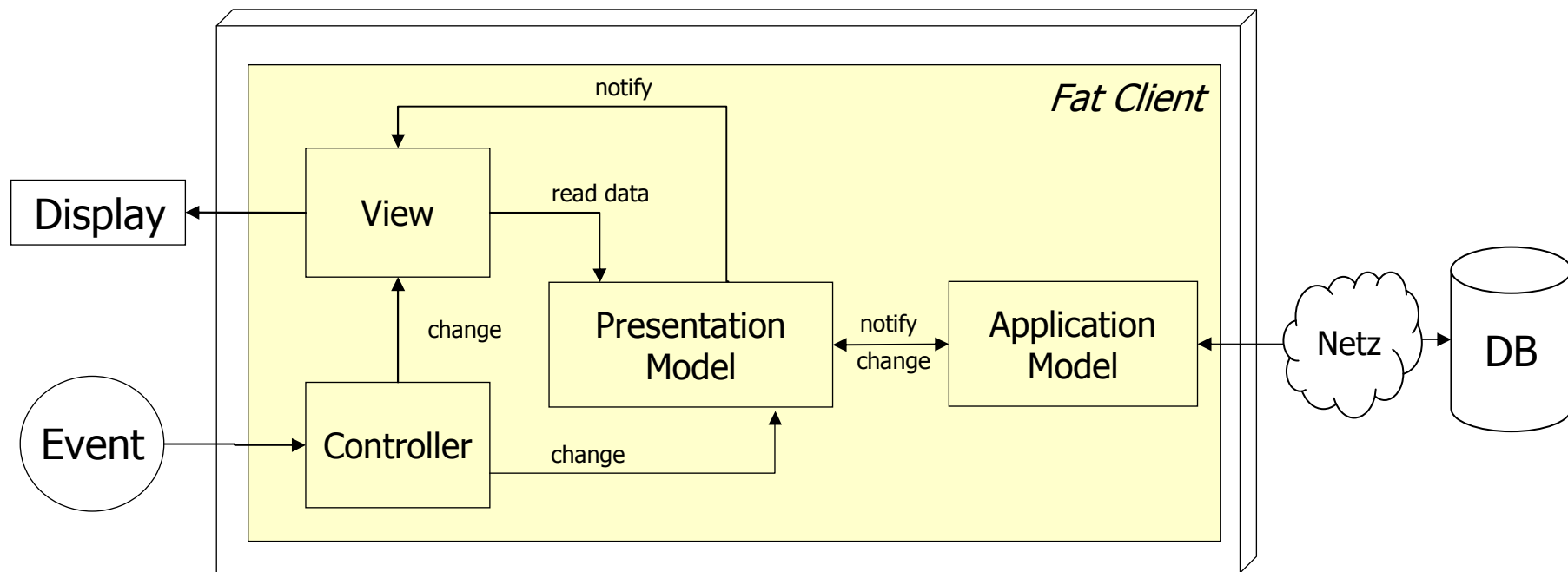
Umsetzung und Verteilung der MVC-Architektur

Das MVC-Muster kann auf vielfältige Weise umgesetzt werden. Sie unterscheiden sich insbesondere bei der Verteilung.

Bei der Auswahl sind folgende Architekturtreiber wichtig:

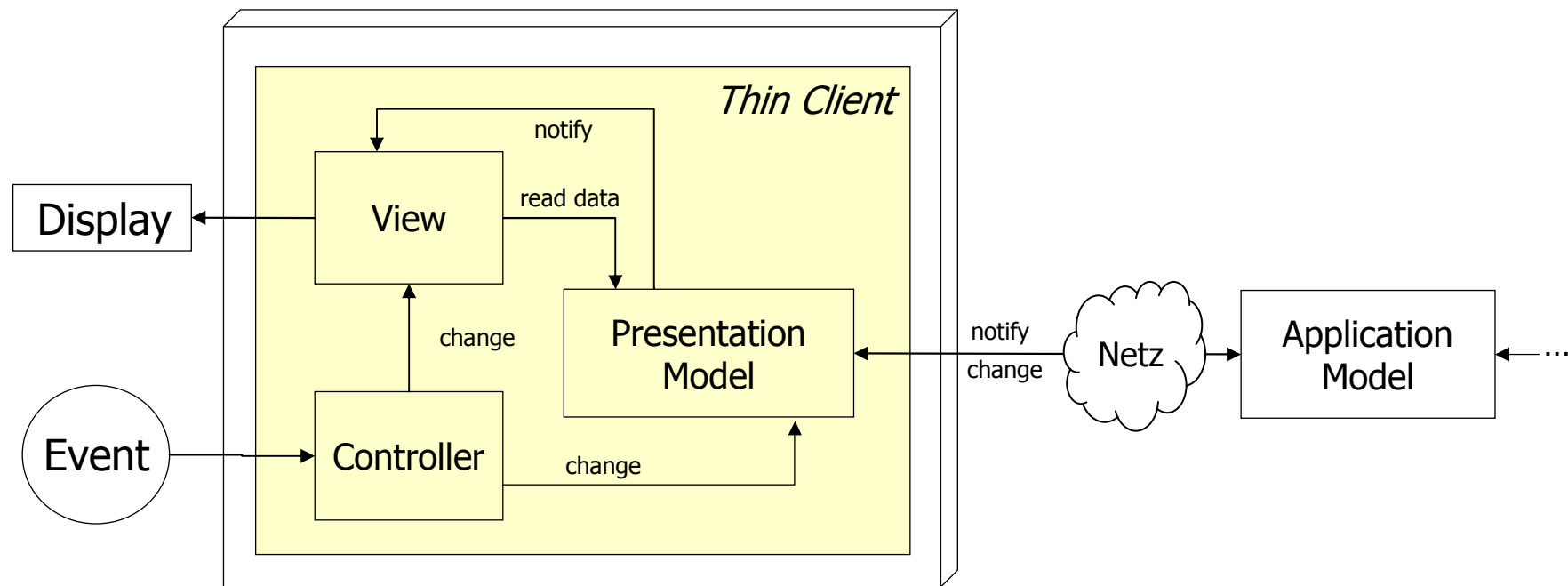
- Geforderte **Reaktivität und Performanz** der GUI
 - Wie sind Latenz und Bandbreite zwischen Client und Server?
- Notwendigkeit der **Offline-Verfügbarkeit**
 - Kann GUI auch ohne Server ausgeführt werden?
- Anforderungen bei **Installation und Deployment**
 - Welche Basis-Software ist nötig? Wie erfolgt die Verbreitung?
- **Aufwand für Entwicklung** und Test
 - Wie aufwändig ist die Erstellung von GUIs? Wie steil ist die Lernkurve?
- Erweiterbarkeit und **Zukunftssicherheit**
 - Gibt es das Framework auch morgen noch?

Architektur beim Fat Client



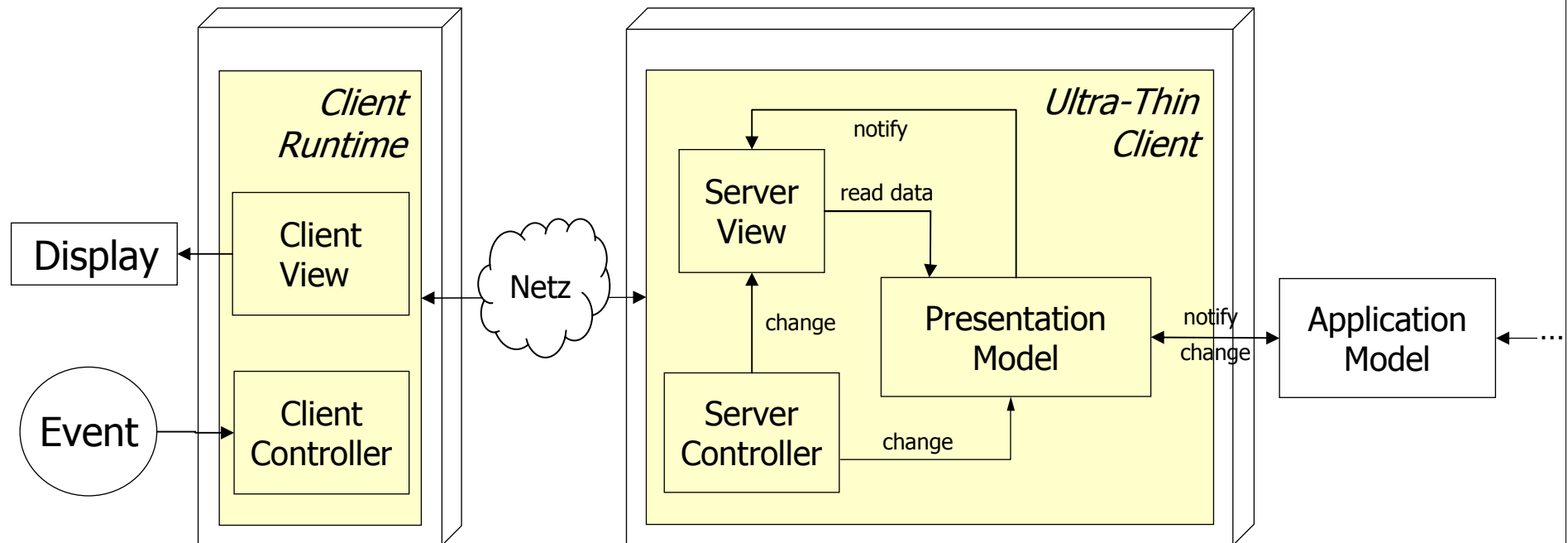
- Fat Client erfordert universell programmierbare Ablaufumgebung
 - Client wird als vollständige Anwendung installiert (Java Swing, Eclipse RCP).
 - Client wird in bestehende Runtime geladen (Java Applet, JavaFX, Adobe AIR, Microsoft Silverlight, einige AJAX-Anwendungen)
- Ansonsten sämtliche Möglichkeiten gegeben
 - Offline-Verfügbarkeit, Reaktivität, einfache Implementierung etc.

Architektur beim Thin Client



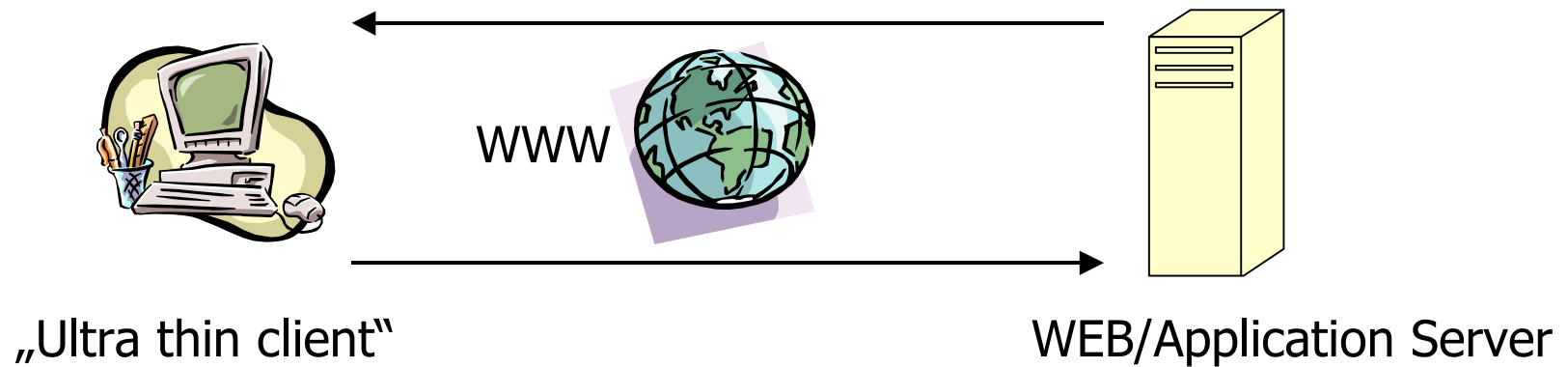
- Thin Client erfordert universell programmierbare Ablaufumgebung
- Zugriff auf Anwendungsfunktionalität über Netzkommunikation
 - Niedrigere Performanz wegen nötiger Netz-Roundtrips
 - Höhere Sicherheit, da Funktionalität vom Server erbracht wird
 - Keine Offline-Verfügbarkeit, da Funktionalität auf Client fehlt

Architektur beim Ultra-Thin Client

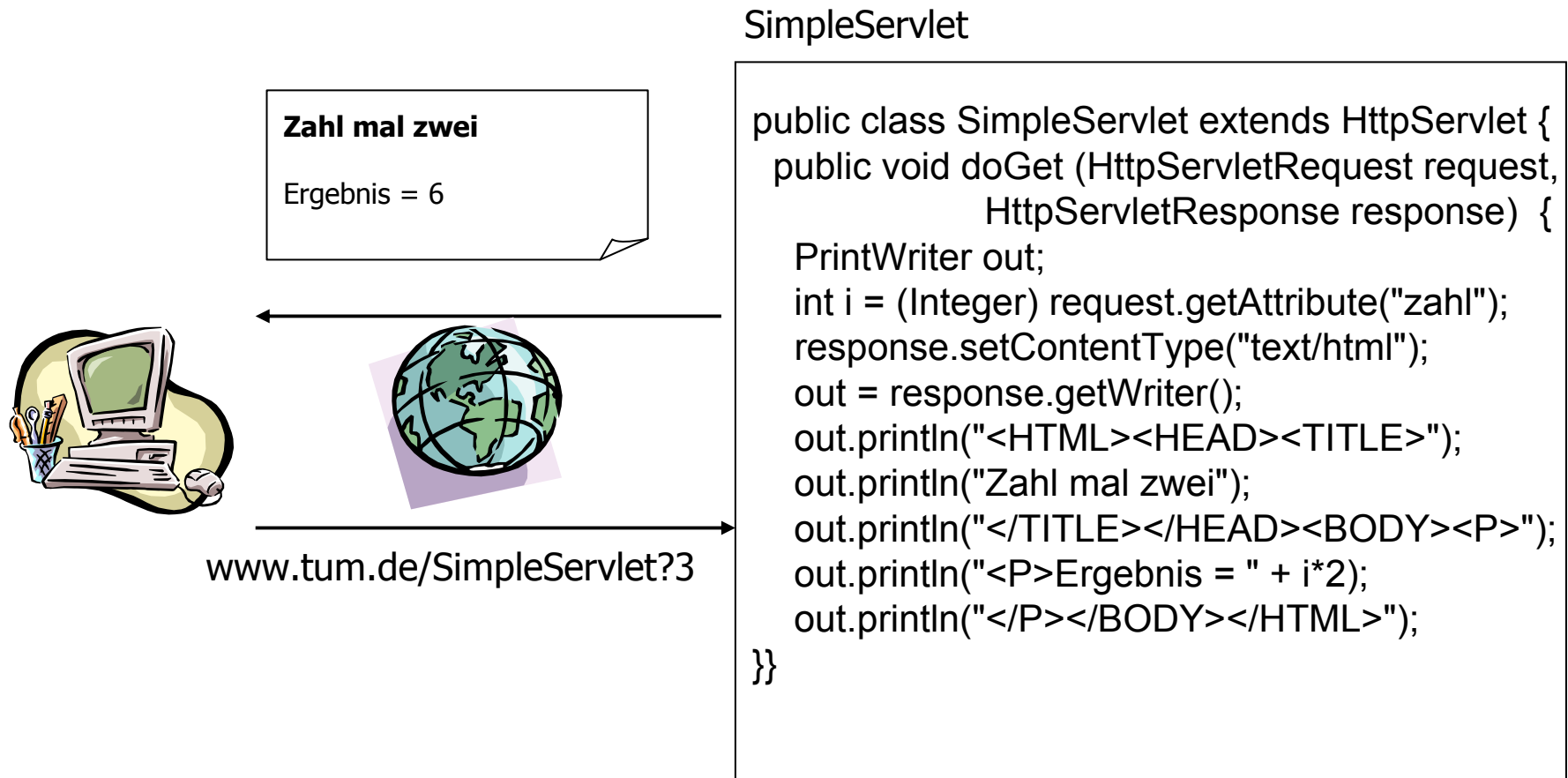


- Ultra-Thin Client erfordert nur minimale Ablaufumgebung
 - Z.B. Web-Browser für Anzeige von Web-Seiten, Terminal Server, X Server
- Zugriff auf Präsentationslogik über Netzkommunikation
 - Noch niedrigere Performanz wegen vieler nötiger Netz-Roundtrips
 - Hohe Sicherheit, da gesamte Funktionalität vom Server erbracht wird
 - Keine Offline-Verfügbarkeit, da Funktionalität auf Client fehlt

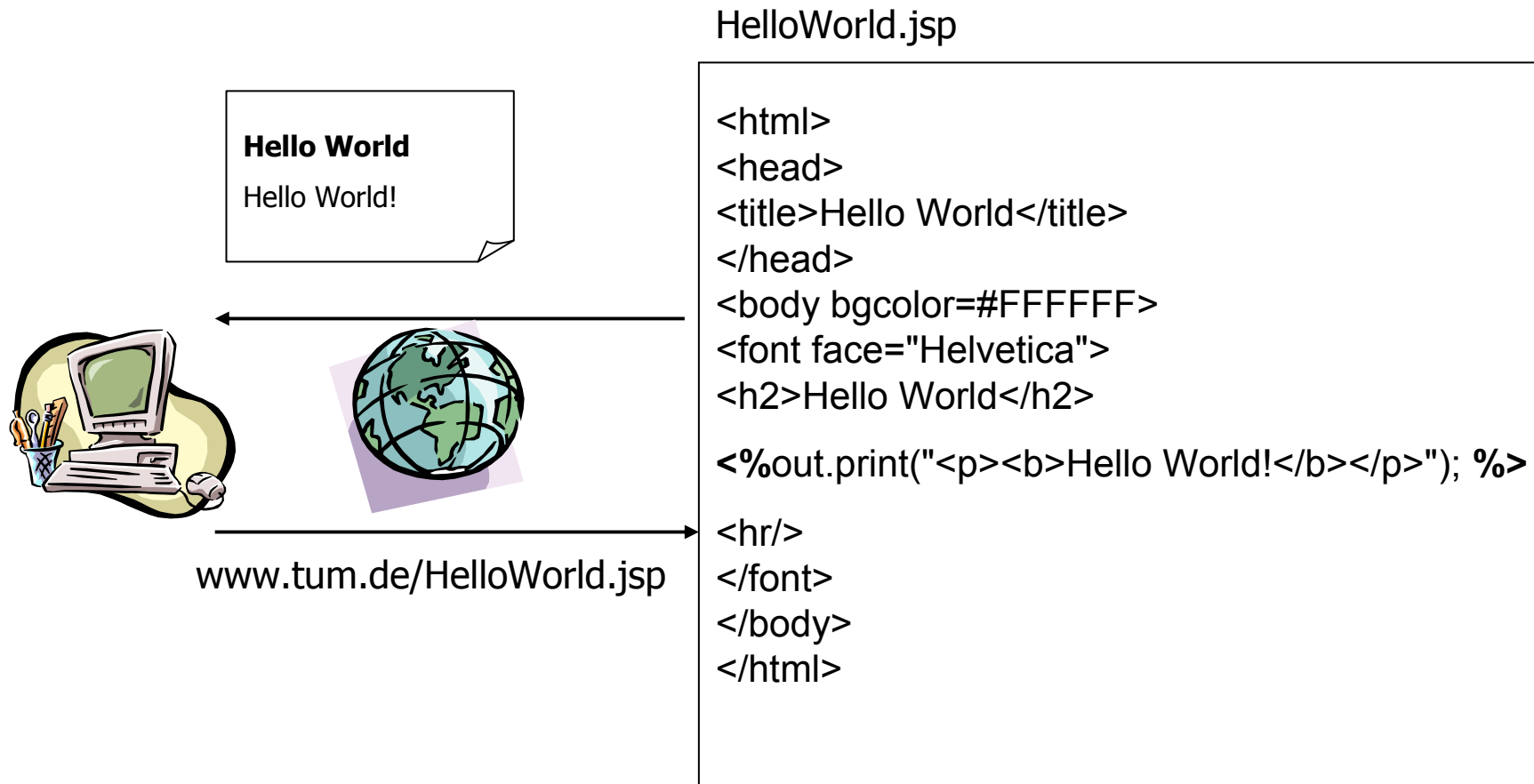
Präsentationsschicht bei "traditionellen" Web-GUIs



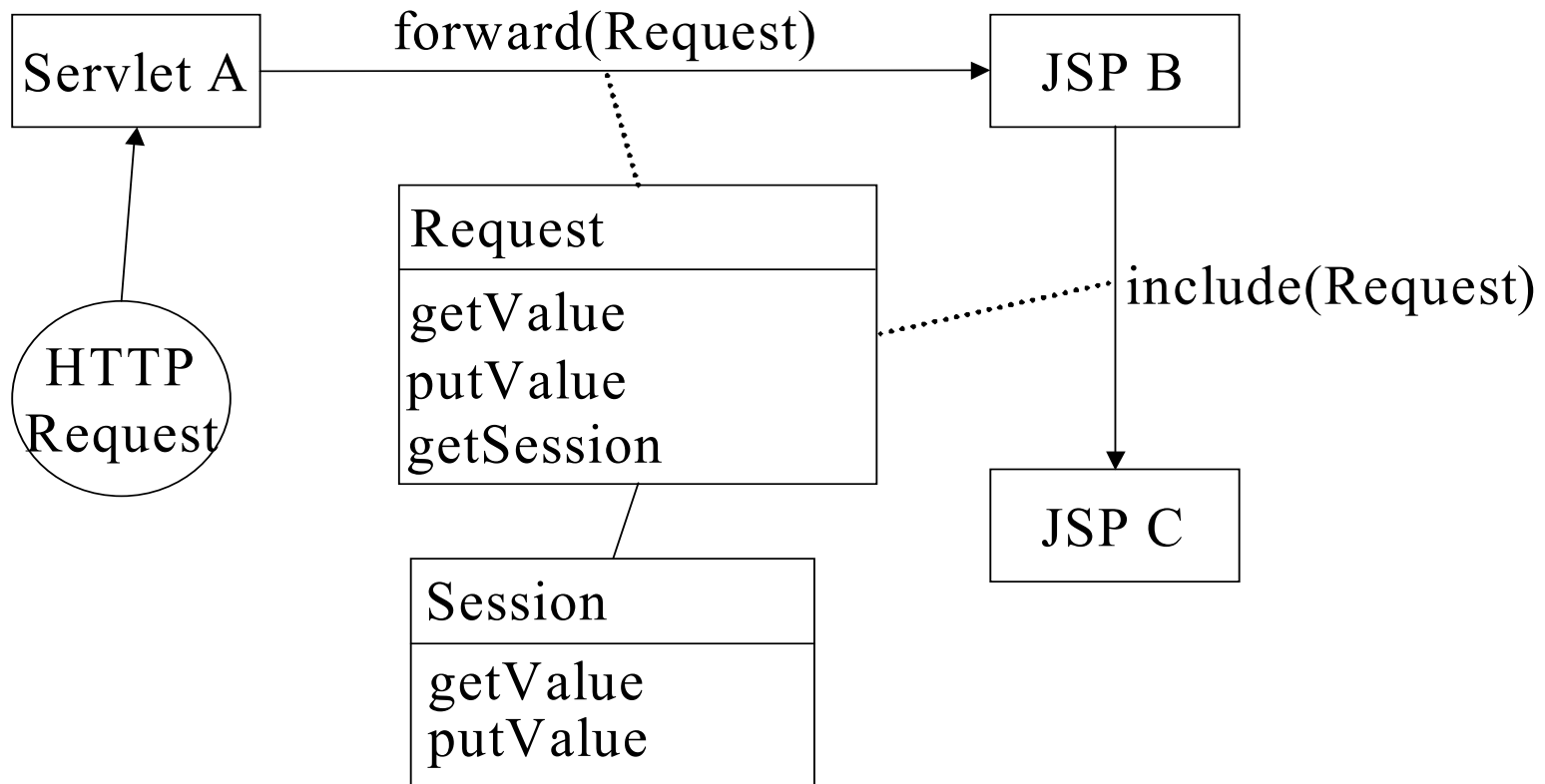
Basistechnik: Java Servlets



Basistechnik: JavaServer Pages

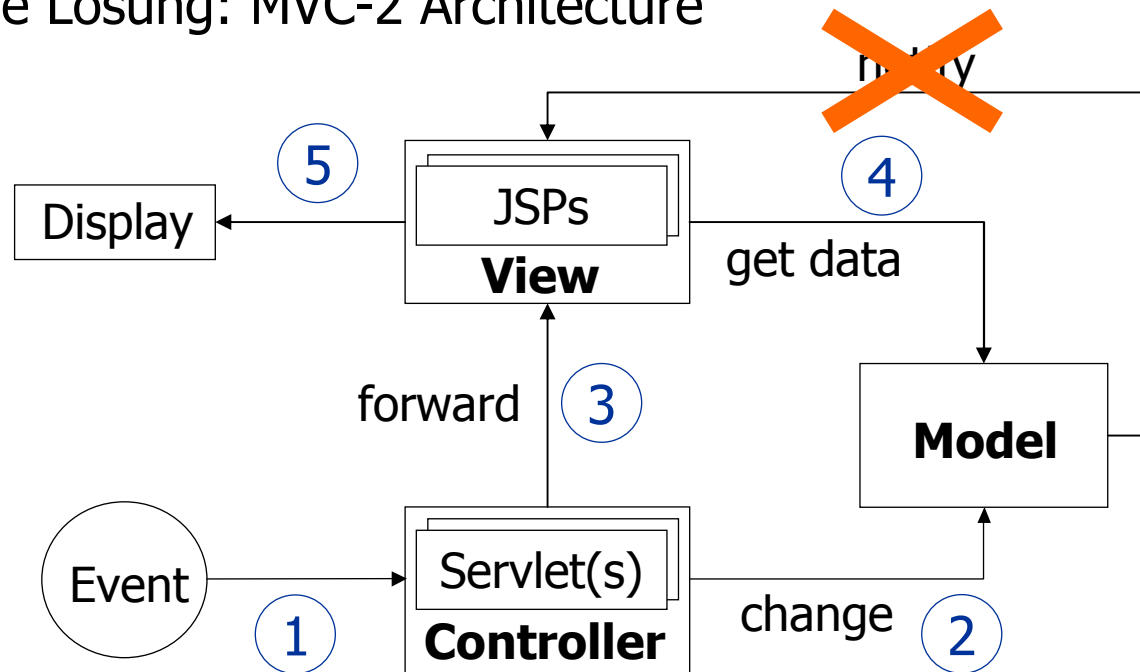


Basistechnik: Forwarding und Including

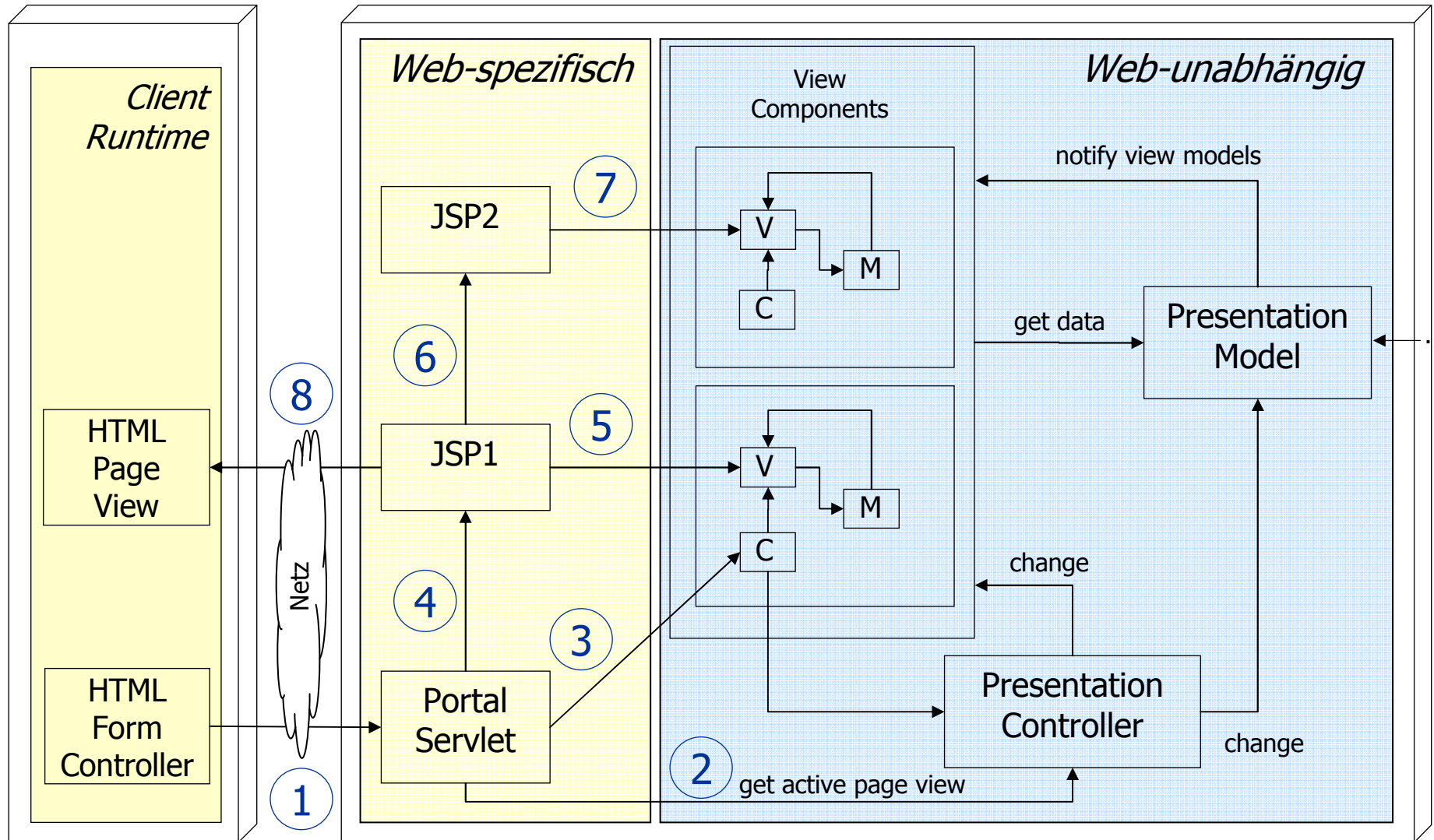


MVC bei Ultra-Thin Clients

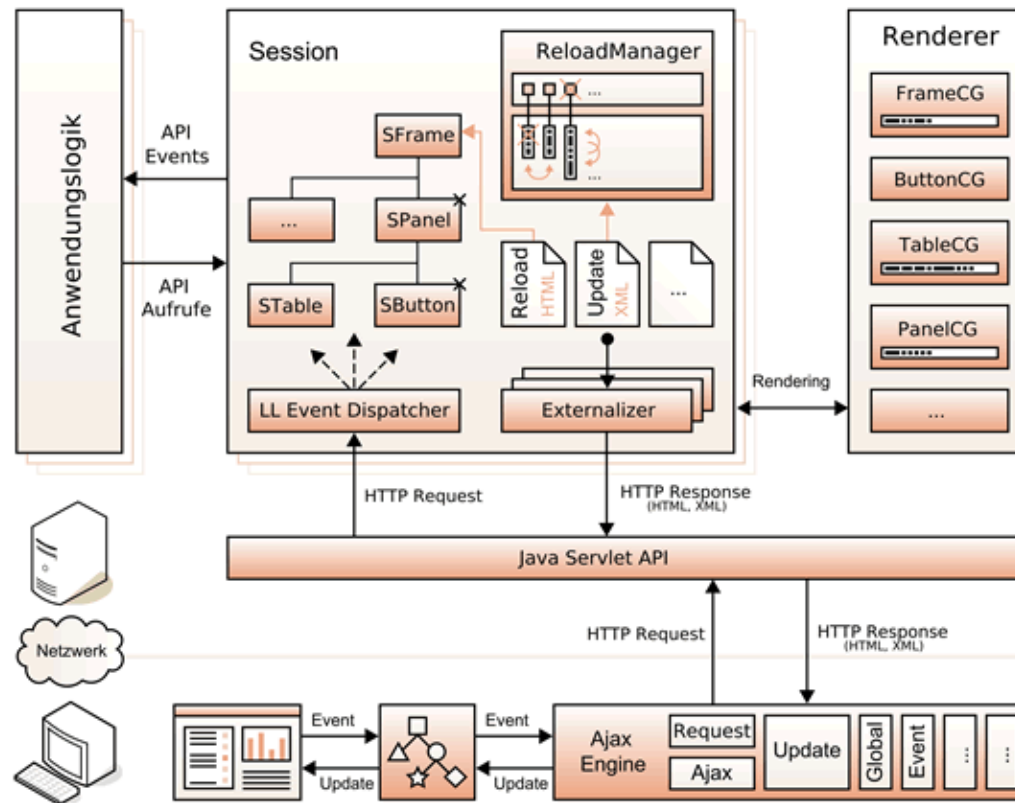
- Direkte Umsetzung von MVC scheitert, weil
 - HTML/HTTP ist seitenorientiert und zustandslos
 - Kommunikation über HTTP-Requests statt Events
 - Kommunikation wird immer vom Browser initiiert
- Mögliche Lösung: MVC-2 Architecture



Multi-Channel-Architektur beim Ultra-Thin-Client



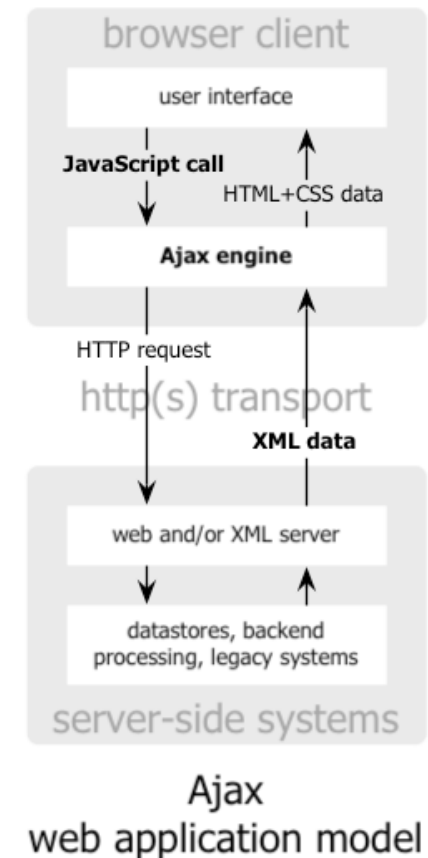
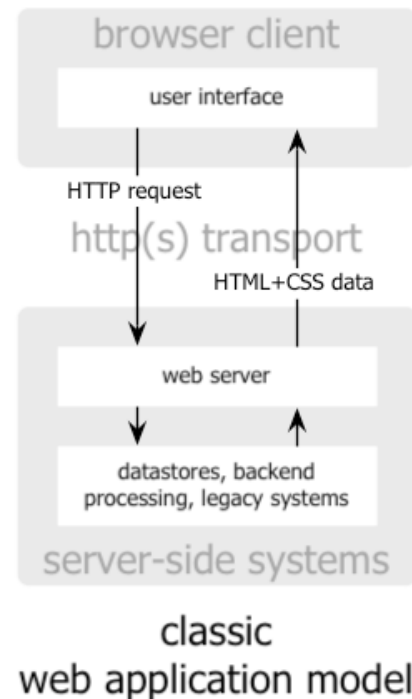
Beispiel für Multi-Channel Framework: wingS



- Bietet Swing-Programmiermodell für das Web
- Nutzt in einer neuen Version zusätzlich AJAX-Ansätze
- Ähnliche Ansätze: Rich AJAX Platform (RAP) im Eclipse Framework, Echo2

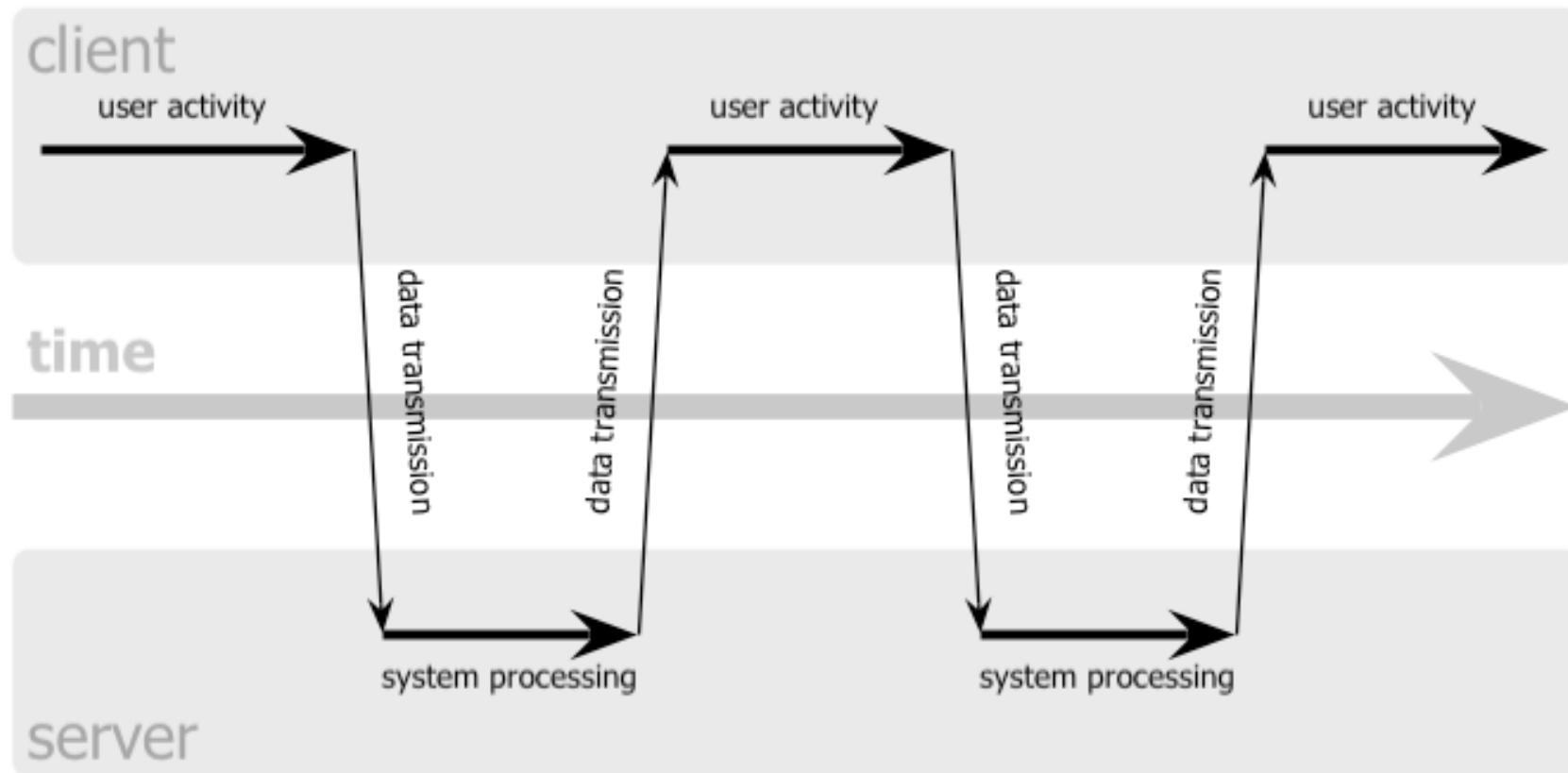
AJAX – Asynchronous JavaScript and XML

- Ansatz für besser bedienbare, reaktive Web-Anwendungen
- Browser als Thin-Client
 - JavaScript als Verarbeitungssprache (muss aktiviert sein!)
 - XHTML + CSS + DOM als GUI-Framework
 - Datenrepräsentation und Manipulation mit XML + XSLT
 - Datenaustausch über Format XMLHttpRequest
- Vorteile
 - JavaScript ist universell
 - Es gibt viele Bibliotheken
- Nachteile
 - JavaScript als Sprache erschwert Programmierung und Debugging (Ausnahme: Google Web Toolkit)



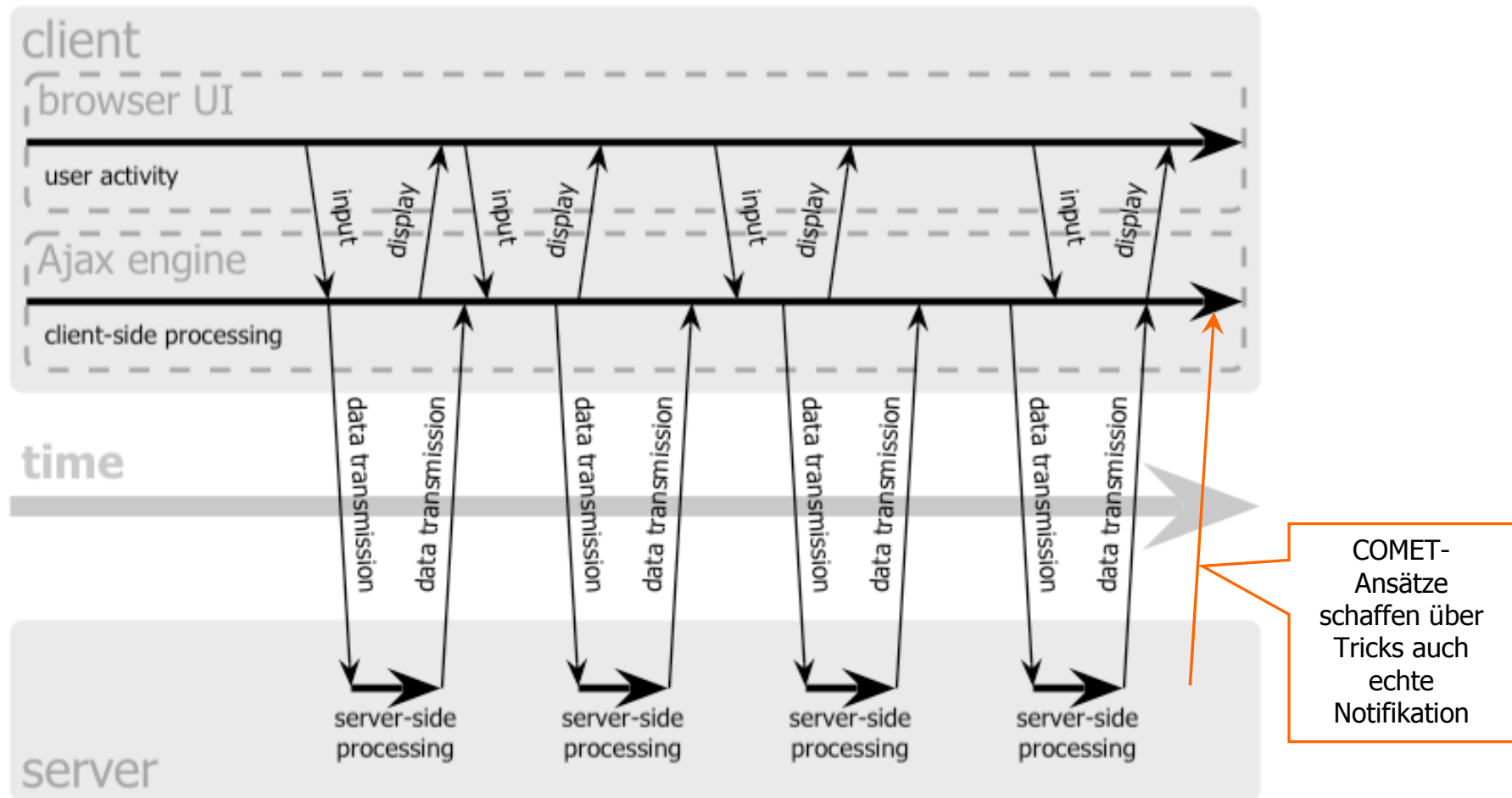
Verarbeitungsmodell bei klassischer Web-Applikation

Synchrone Aufrufe → Benutzer muss warten



Verarbeitungsmodell bei AJAX-Applikation

Asynchrone Aufrufe → Benutzer kann weiterarbeiten



AJAX-Frameworks

- AJAX-Programmierung „von Hand“ ist extrem aufwändig
 - Nutzung von JavaScript als Programmiersprache
 - Entwicklungsumgebungen nicht sehr ausgereift
 - Viele kleine browserspezifische Fehler und Unterschiede
 - Erstellung von XMLHttpRequest-Objekten, Reaktion auf Antworten, Manipulation der XML-DOM-Repräsentation im Browser ist umständlich
- Es gibt eine große Vielzahl von AJAX-Frameworks
 - Verstecken die unhandlichen Basis-Mechanismen hinter Schnittstellen:
 - Bereitstellung von GUI-Widgets
 - Einfachere Kommunikation mit dem Server
 - Programmierung dann i.W. wie bei „traditionellen“ GUIs (→Thin Client oder sogar →Fat Client), nur mit Hilfe der Sprache JavaScript.
 - Beispiele: Prototype, ExtJs, SmartClient und viele mehr.
- Google Web Toolkit als Compiler-Ansatz
 - Entwicklung in (Subset von) Java mit gewohnten Werkzeugen
 - Compiler übersetzt Anwendung in JavaScript

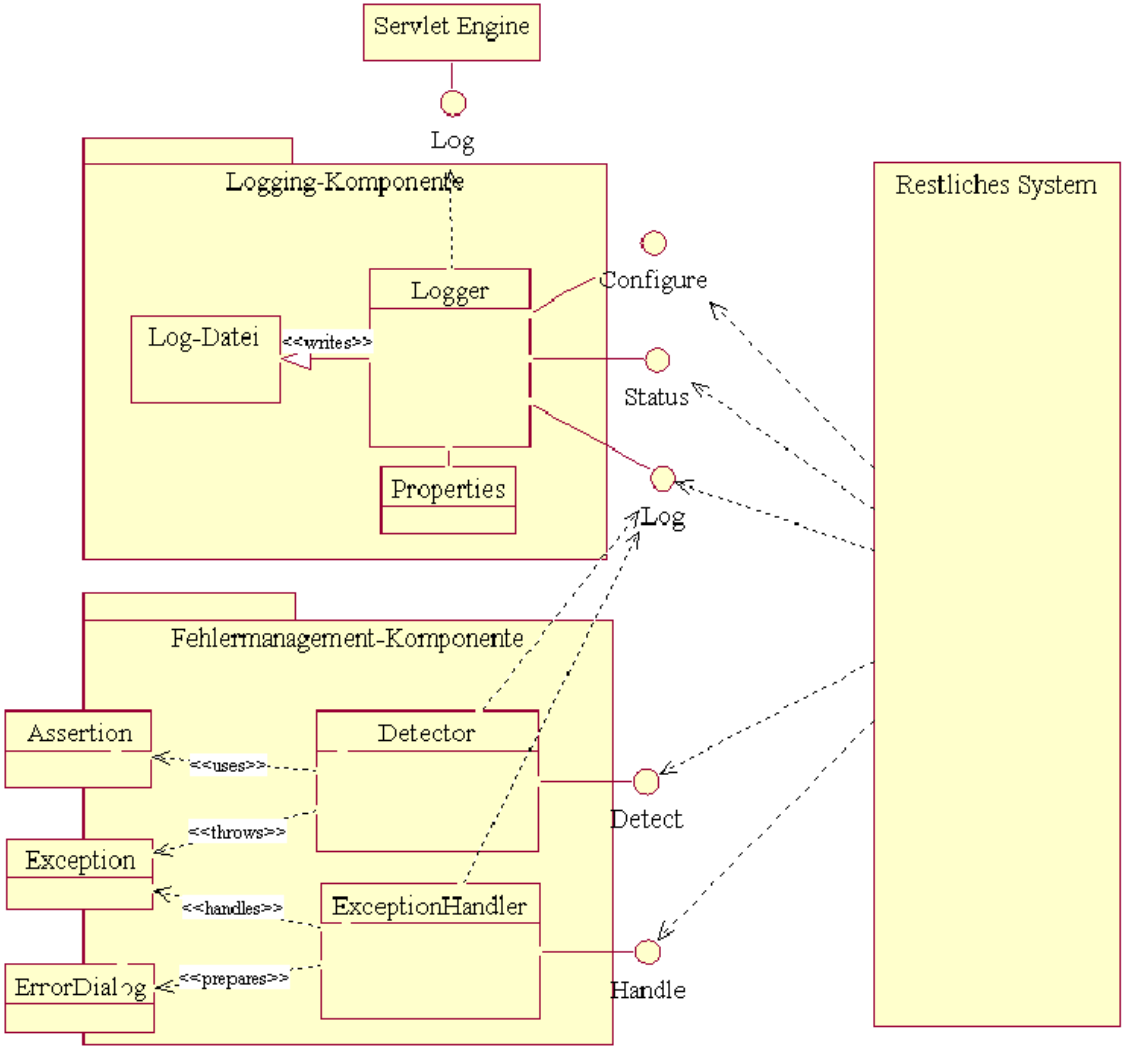
Inhalt

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

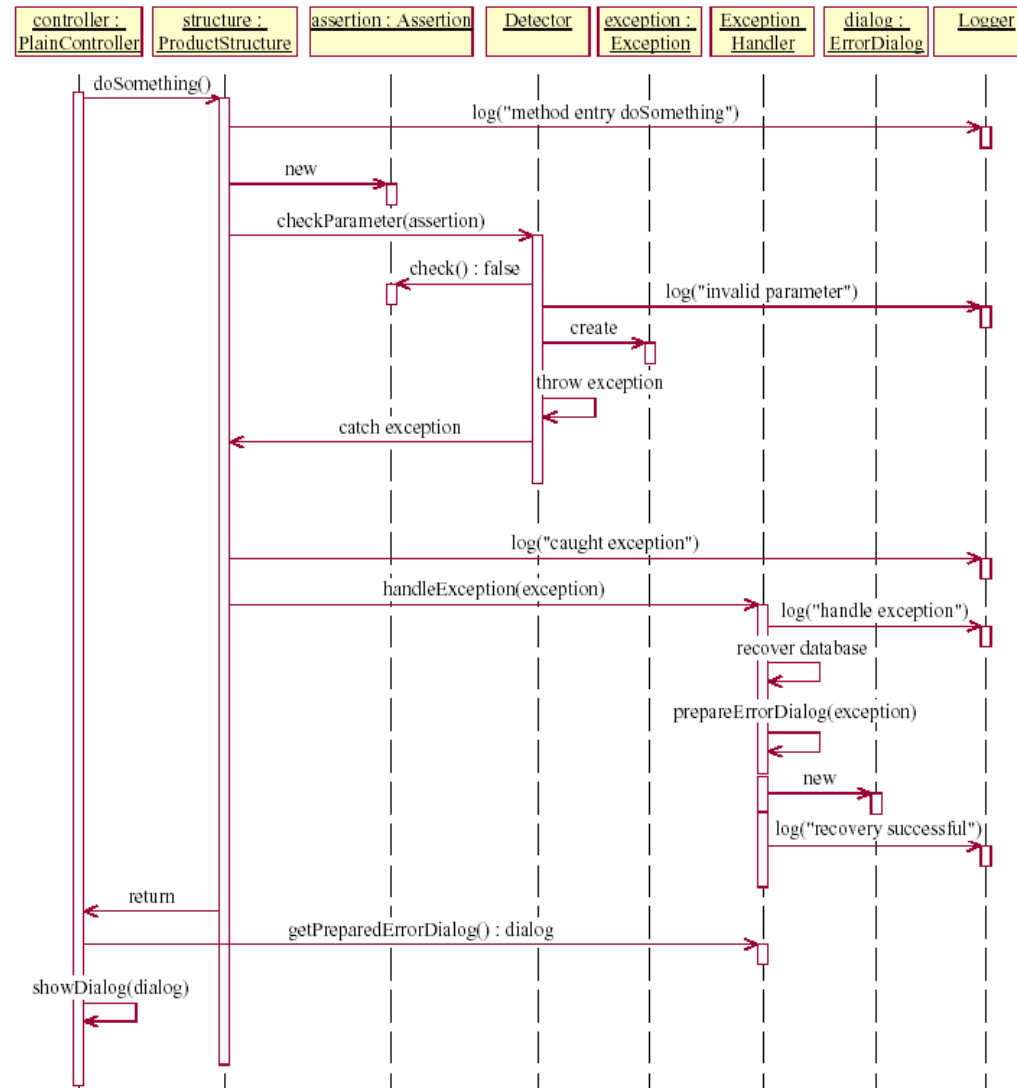
Unterstützungsschicht

- 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-Aufrufen im restlichen Code (oft gut über Aspects realisierbar oder vom Container übernommen)

Beispiel: Basiskomponenten für Fehlermanagement und Logging

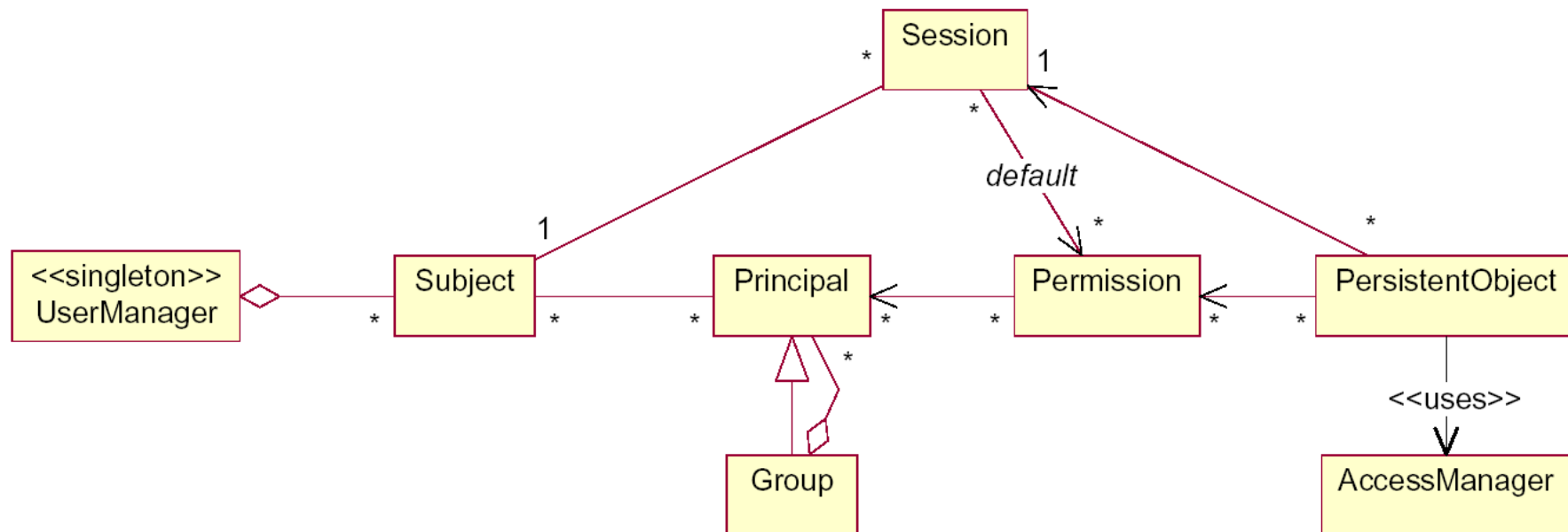


Beispiel: Aufruf von Fehlermanagement und Logging



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 Fat- und 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.
- [Fow] <http://martinfowler.com/eaDev/PresentationModel.html>
- [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.springframework-work.org>
- [WfMC] *Workflow Management Coalition Homepage*, <http://www.wfmc.org>