Architektur II
Patterns (Entwurfsmuster)

TU-Wien, Sommersemester 2004

Rudolf Lewandowski

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Überblick

- Einführung
  - Was sind Patterns (Entwurfsmuster)
  - Herkunft, Begriffe, Beispiele, Arten
- Beispiele
  - MVC und JSP Model 2 Architektur
  - Produktserver
- Literatur

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Einführung

- **Christopher Alexander**
  - „Bau“-Architekt mit dem Anspruch, ästhetisches und gutes Design erklärbar und erlernbar zu machen.
  - **1977**: A Pattern Language: Towns, Buildings, Construction, Oxford University Press,
  - **1979:** Timeless Way of Building, Oxford University Press
- **OOPSLA'87, Kent Beck, Ward Cunningham**
  - "Using Pattern Languages for Object-Oriented Programs".
- **1995: Design Patterns** von Gamma, Helm, Johnson, Vlissides

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Einführung
Definition

- Ein Entwurfsmuster (Pattern) ist ein Problem mit seiner Lösung in einem Kontext.

- In allen möglichen Ingenieurdisziplinen ist es üblich, Entwurfshandbücher zu benutzen, die Lösungen zu bestimmten Problemen bereithalten.

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Problem / Lösung / Kontext

- Kontext
  - Bau von Mehrparteienhäusern. Große Balkone kosten viel Geld, bringen wenig Miete und werden daher oft klein gebaut und nicht benutzt. Alle wünschen sich einen hellen Balkon.
- Problem
  - Wie baut man einen Balkon so, dass er auch benutzt wird?
- Lösung
  - Man baue ihn mit einer Tiefe von mindestens 2 Metern, so dass man darauf bequem einen Tisch und Stühle für mehrere Personen unterbringen kann.

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

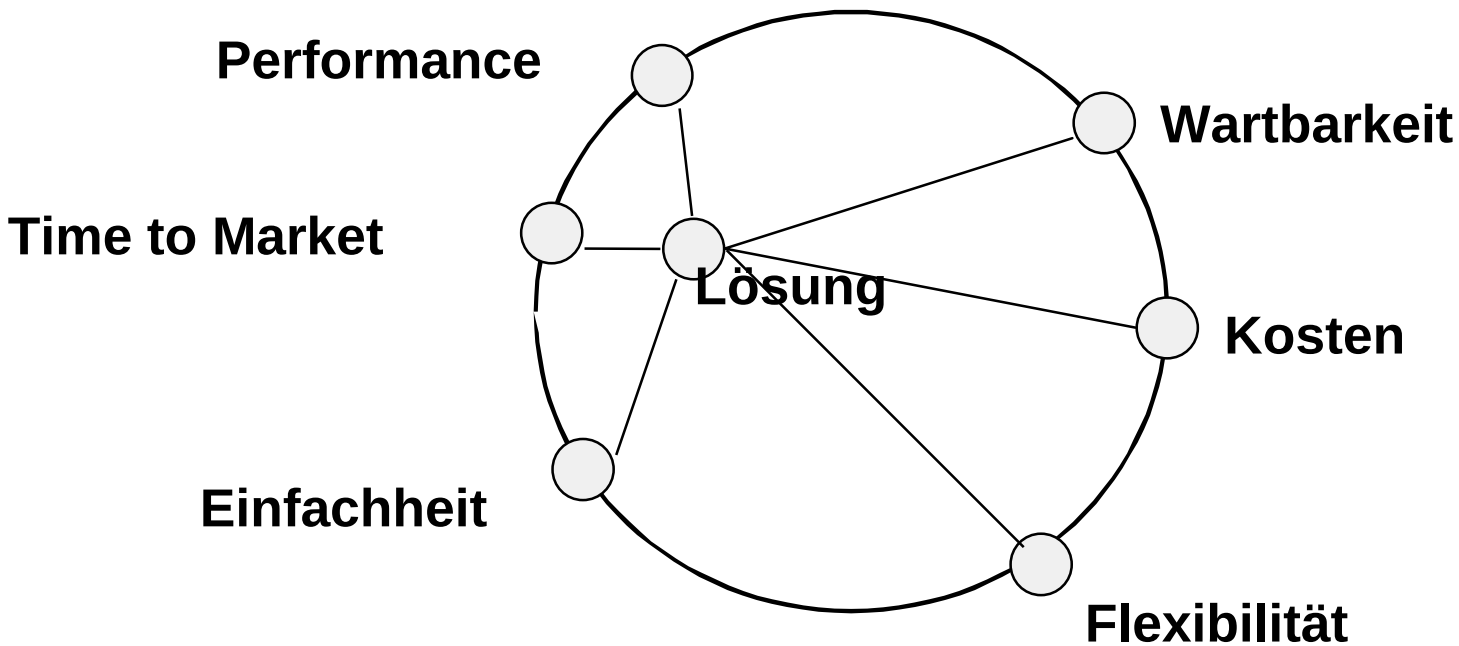
Kontext enthält Zielkonflikte, die der Designer abwägen muss

- Kosten **gegen** Benutzbarkeit
- Lichteinfall für den Nachbarn unter mir **gegen** Benutzbarkeit

- Bei Software sind Zielkonflikte zwischen nichtfunktionalen Eigenschaften typisch:
  - Einfachheit **gegen** Wartbarkeit
  - Performance **gegen** Kosten
  - Lese-Performance **gegen** Schreib-Performance
  - Batch-Performance **gegen** Online-Performance
  - ... und viele mehr

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Gut geschriebene Patterns machen neben der Nennung einer Lösung für ein Problem die Zielkonflikte explizit und zeigen die Konsequenzen, die eine Lösung hat, auf ...



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Was ist ein Pattern und was nicht ...

- Patterns sind bewährte Lösungen, die beschrieben werden, weil sie in der Praxis unabhängig voneinander an vielen Stellen zu finden sind
- Ein eigenes Design, das man einmal gemacht hat und das man in der Form von Patterns beschreibt, ist noch lange kein Pattern (es fehlen die sog. **Known Uses**)
  - Eine Lösung zu einem häufig auftretenden Problem ist noch lange kein gut geschriebenes Pattern, wenn die Zielkonflikte und die Konsequenzen der Lösung nicht erläutert sind (es fehlen die sog. **Forces** und **Consequences**)

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Wesentliche Arten von Patterns

- Analyse Patterns
  - Beschreiben wiederkehrende Lösungen in fachlichen Objekt- und Datenmodellen. Beispiel: Rolle.
- Architektur-Patterns
  - Beschreiben Muster auf der Ebene von Software-Architekturen: Beispiele: Schichten, Pipes&Filters, Repository, Blackboard, ...
- Design-Patterns
  - Beschreiben Interaktionen von Klassen auf der Ebene von technischem Design: Beispiele: PAC, MVC, Observer, Factory, ...
- Idioms
  - Beschreiben Muster für guten Code in einer speziellen Programmiersprache: Beispiel Letter/Envelope, Counted Reference, ..

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

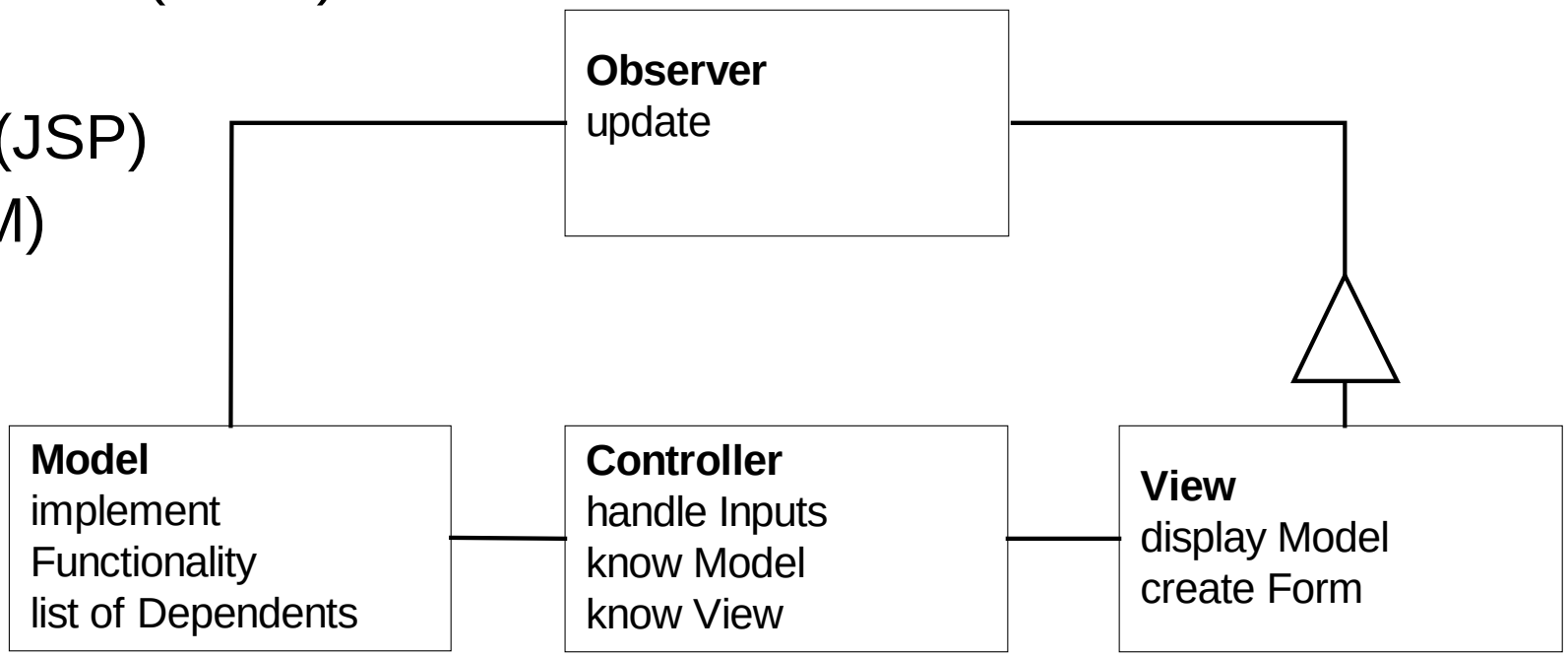
Beispiel MVC

- Problem
  - Userinterfaces ändern sich oft.
  - Die selbe Information ist in verschiedenen Fenstern in unterschiedlicher Form darzustellen **gegen** Komplexität der dabei benötigten Frameworks. Verschiede Gruppen von Benutzer brauchen unterschiedliche Aufbereitungen.
  - Abwägen von konsistenten Sichten auf ein Model **gegen** Performance Probleme durch exzessive Anzahl von Updates.
  - Portieren einer Anwendung auf eine andere Plattform sollte nicht Überarbeitung der gesamte Anwendung bewirken.
  - ...

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

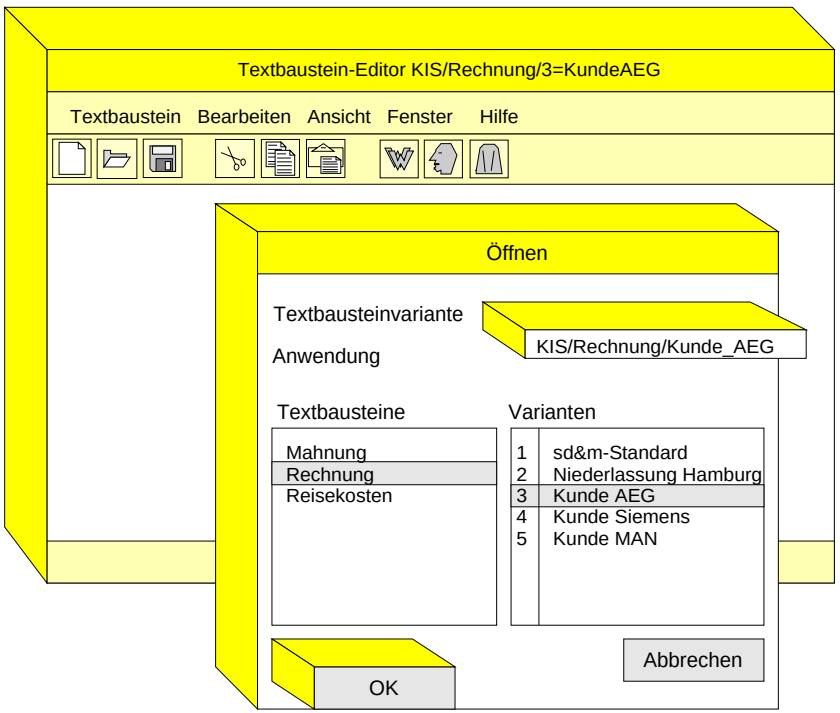
MVC

- Smalltalk 80
- IBM Smalltalk (Motiv)
- VB, MFC
- Catalina (JSP)
- MFS (IBM)



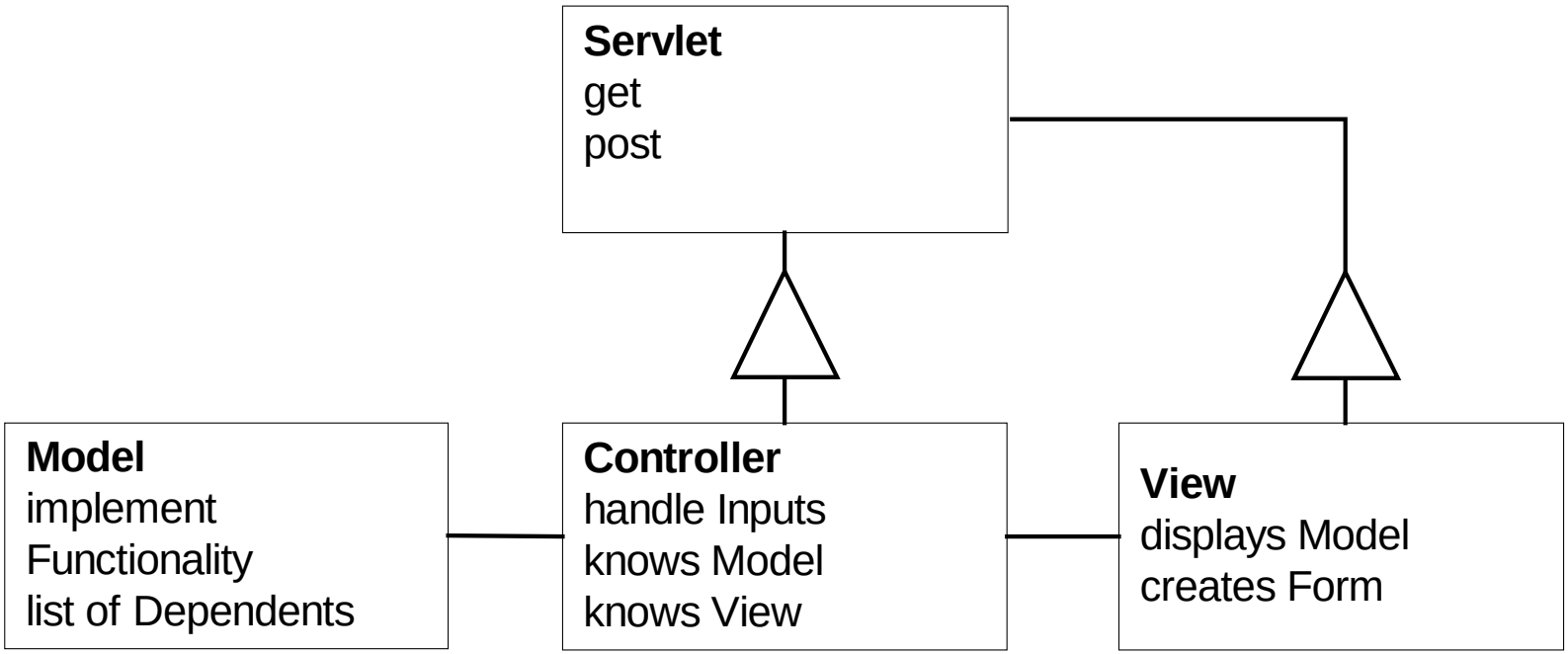
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC View
Local Userinterface



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC Servlet



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC View
Distributed Userinterface

```
<html>
<head>
<title>Text des Titels</title>
</head>
<body>
Text, Verweise, Grafikrefer
</body>
</html>
```

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Servlet Architecture

```
public class SimpleExampleServlet extends HttpServlet {
public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, java.io.IOException {
        PrintWriter out = res.getWriter();
        HttpSession session = req.getSession(true);
        out.println("<HTML><HEAD><TITLE>Shopping...</TITLE></HEAD><BODY>");
        out.println("<H1>Your Order was:</H1>");

        java.util.Enumeration enum = req.getParameterNames();
        while (enum.hasMoreElements()) {
                String element = (String) enum.nextElement();
                out.println("<p>Element " + element + " is: " + req.getParameter(element) + "</p>");
        }
        out.println("Your Session is: " + session.getId());
        out.println("</BODY></HTML>");
}}
```

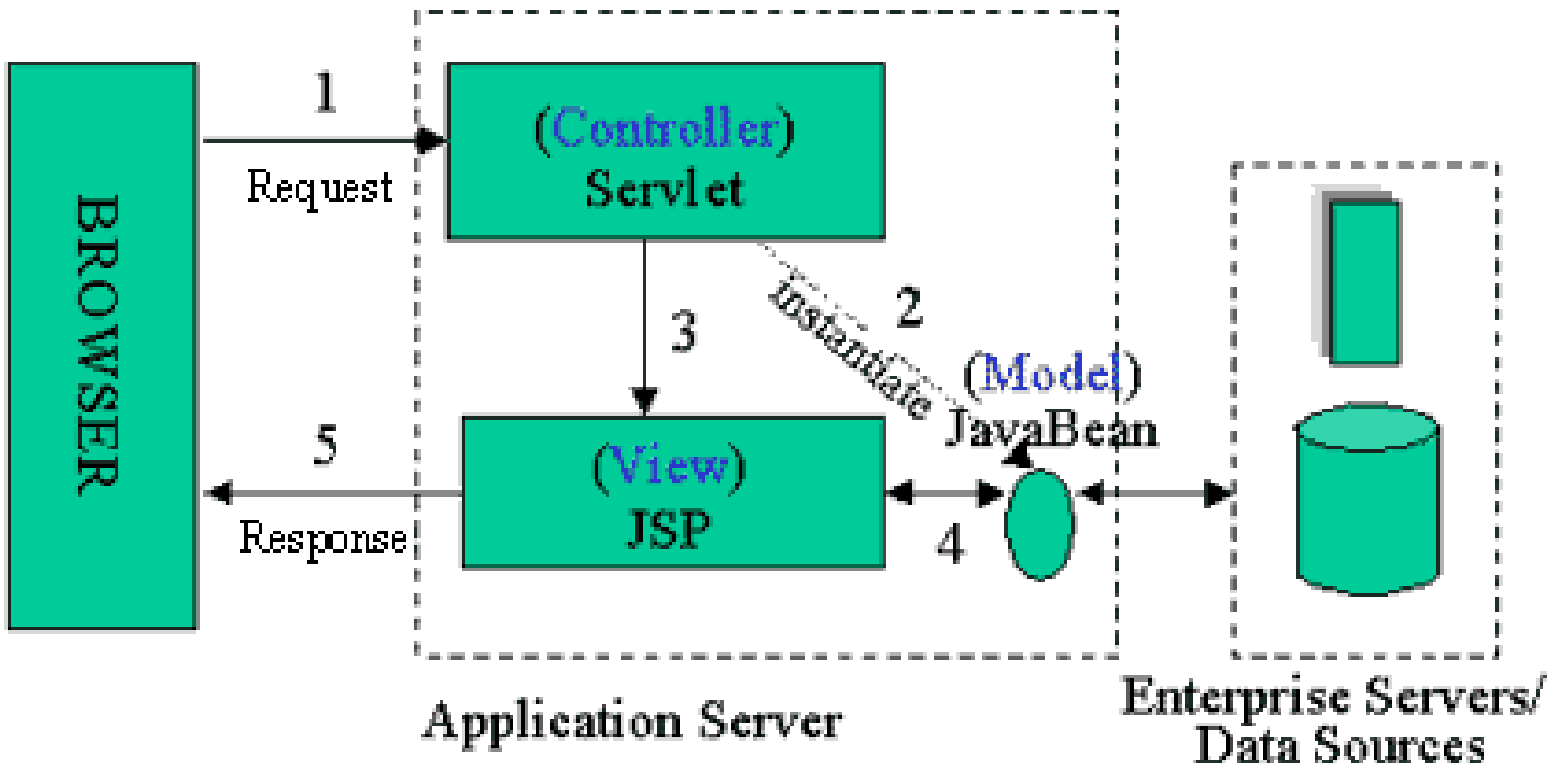

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

JSP Model 1 Architecture (inside out)

```
<HTML>
<HEAD>
<TITLE>Shopping...</TITLE>
</HEAD>
<BODY>
<%@ page language="java" import="java.util.*" %>
<H4>Mixing Java and Code</H4>
<% if (Calendar.getInstance().get(Calendar.AM_PM) == Calendar.AM) {%>
Good Morning <BR>
<% } else { %>
Good Afternoon <BR>
<% } %>
</BODY>
</HTML>
```


© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

JSP Model 1 Architecture (MVC Web Architecture)



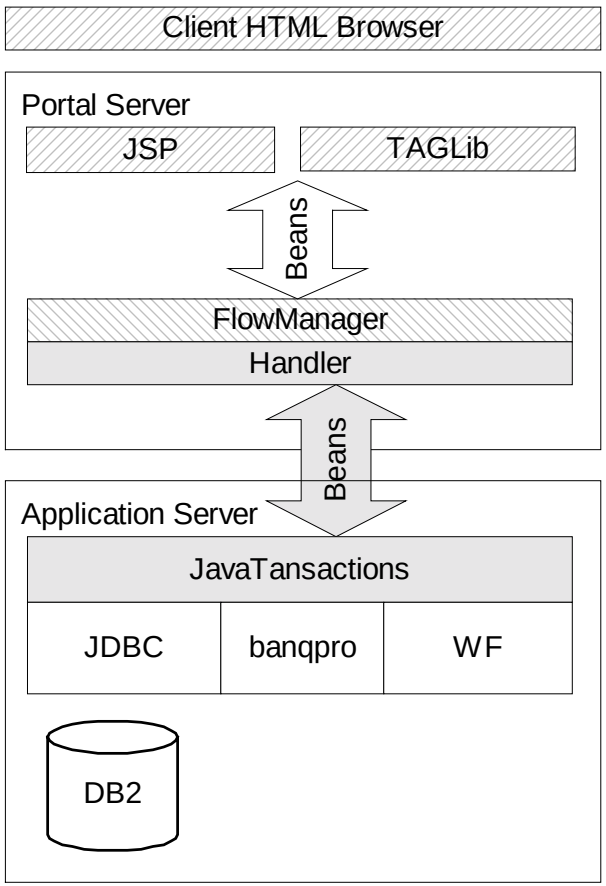
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

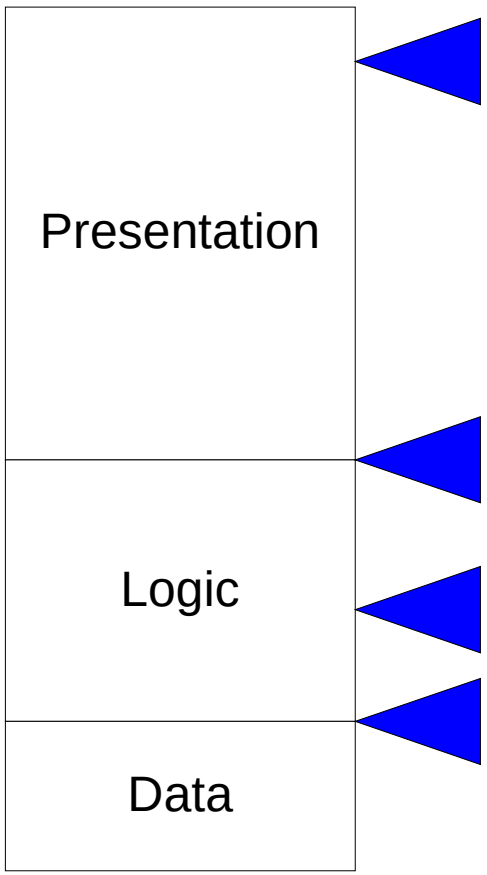
MVC View

- View mittels GUI Builder Tool erzeugt
  - Graphical Builders
  - JSP Compiler
  - XMLC Compiler
- Zur Erstellung von Views ist wenig Programmiererfahrung notwendig
- Generierung von Klassen die zur Laufzeit Fenster/Formulare auf dem gleichen Rechner oder remote über HTML anzeigen.
- Oftmals Composition aus Widgets oder Tags
- Benutzereingaben informieren den Controller (Event Dependent)

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

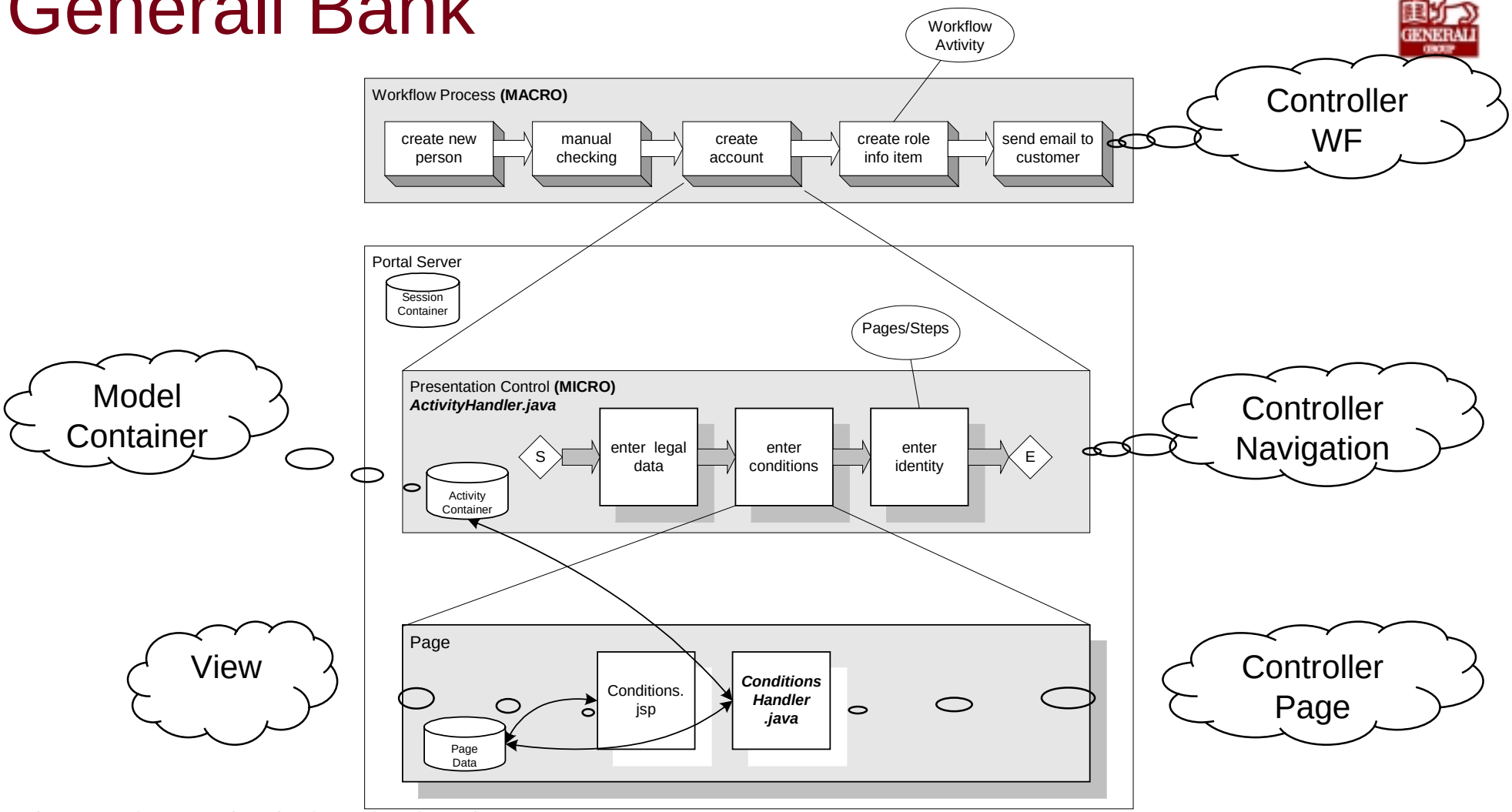
MVC
Beispiel: Generali Bank





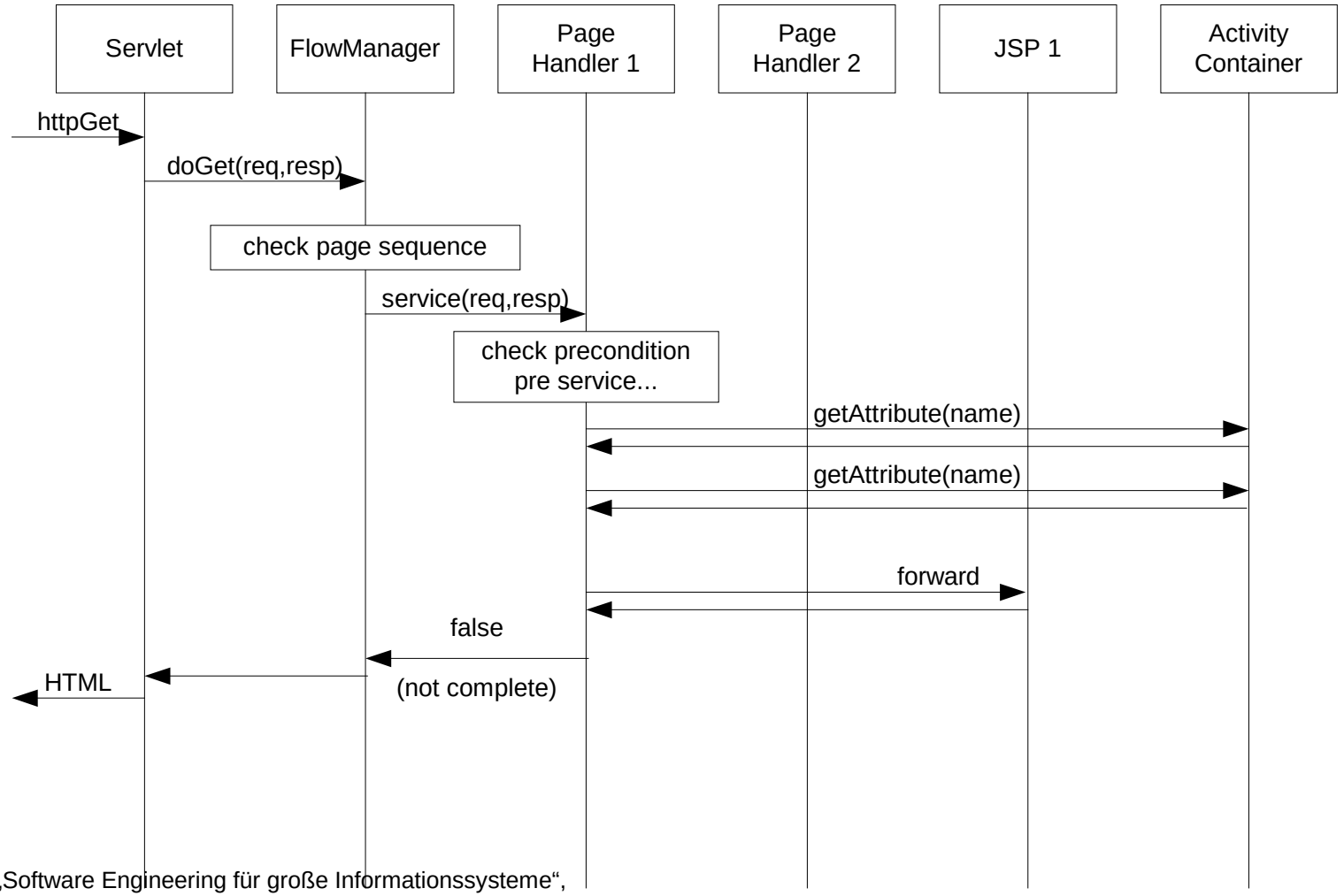
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC
Generali Bank



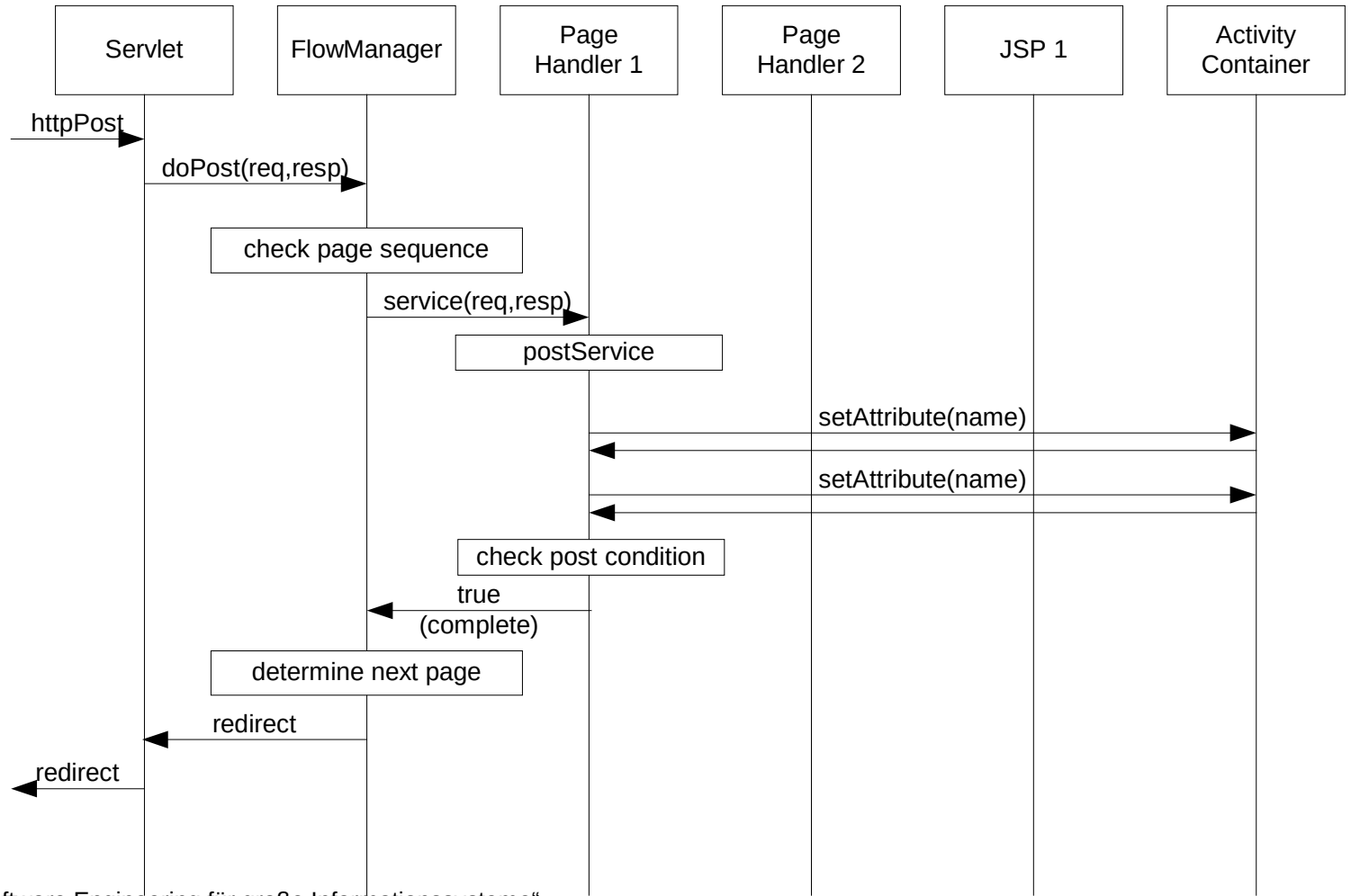
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC
Generali Bank



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

MVC
Generali Bank



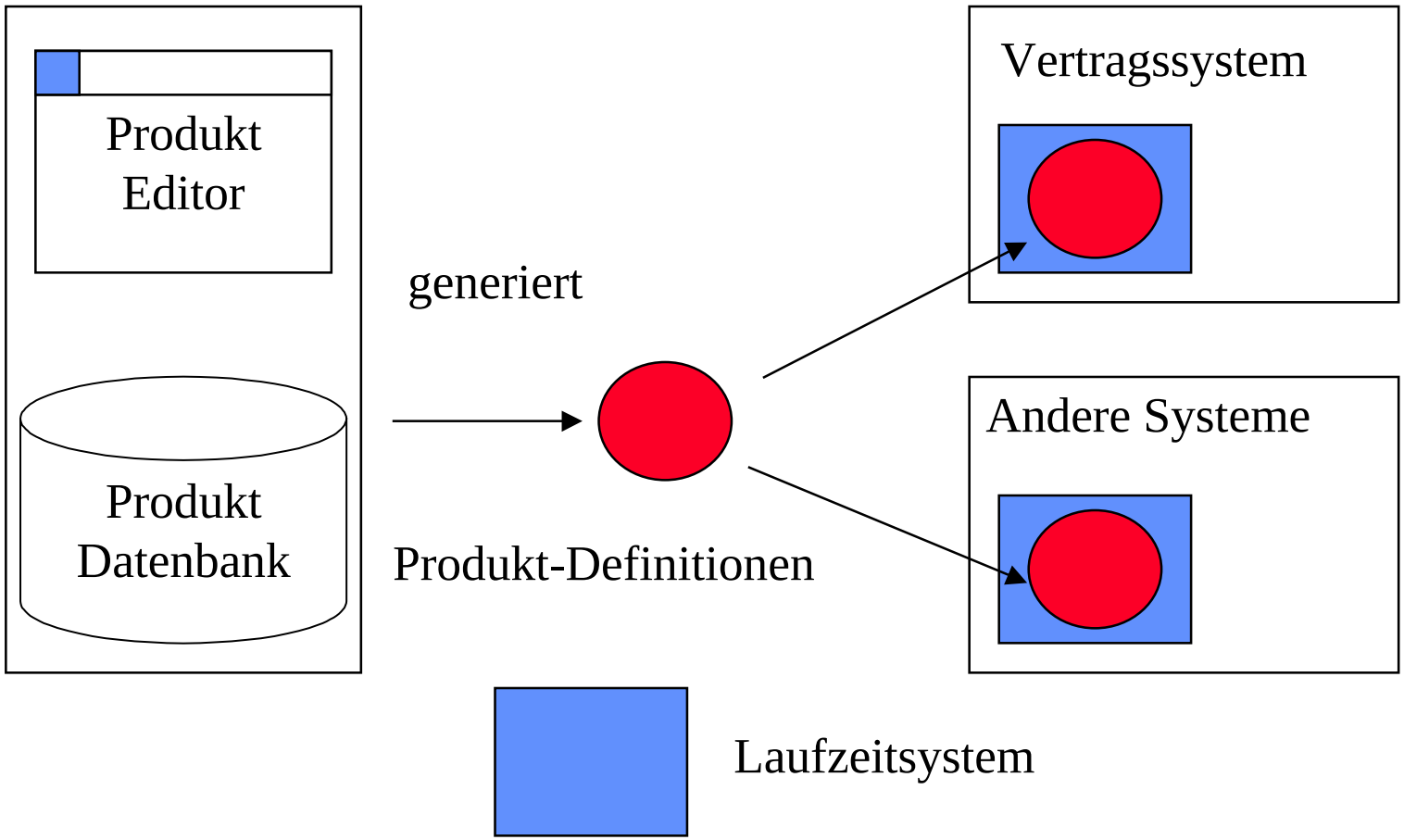
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Beispiel Produktserver

- Problem
  - Neue Versicherungsprodukte sollen innerhalb von Tagen im DV System verfügbar sein. Wie baut man ein System, das dies möglich macht?
  - Faktoren, die das Design beeinflussen (exemplarisch ...)
  - Plattformunabhängigkeit **gegen** Kosten: Produktwissen muss plattformunabhängig zur Verfügung stehen und in vielen Teilsystemen.
  - Kapselung des Produktwissens **gegen** optimales Design von User Interfaces: Gut designte Benutzungsschnittstellen enthalten ebenfalls Produktwissen - eine vollständige Auslagerung in einen Produktserver führt zu User Interfaces, die nicht optimal zu benutzen sind.
  - ... Viele viele weitere ...

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Produktserver Lösung



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

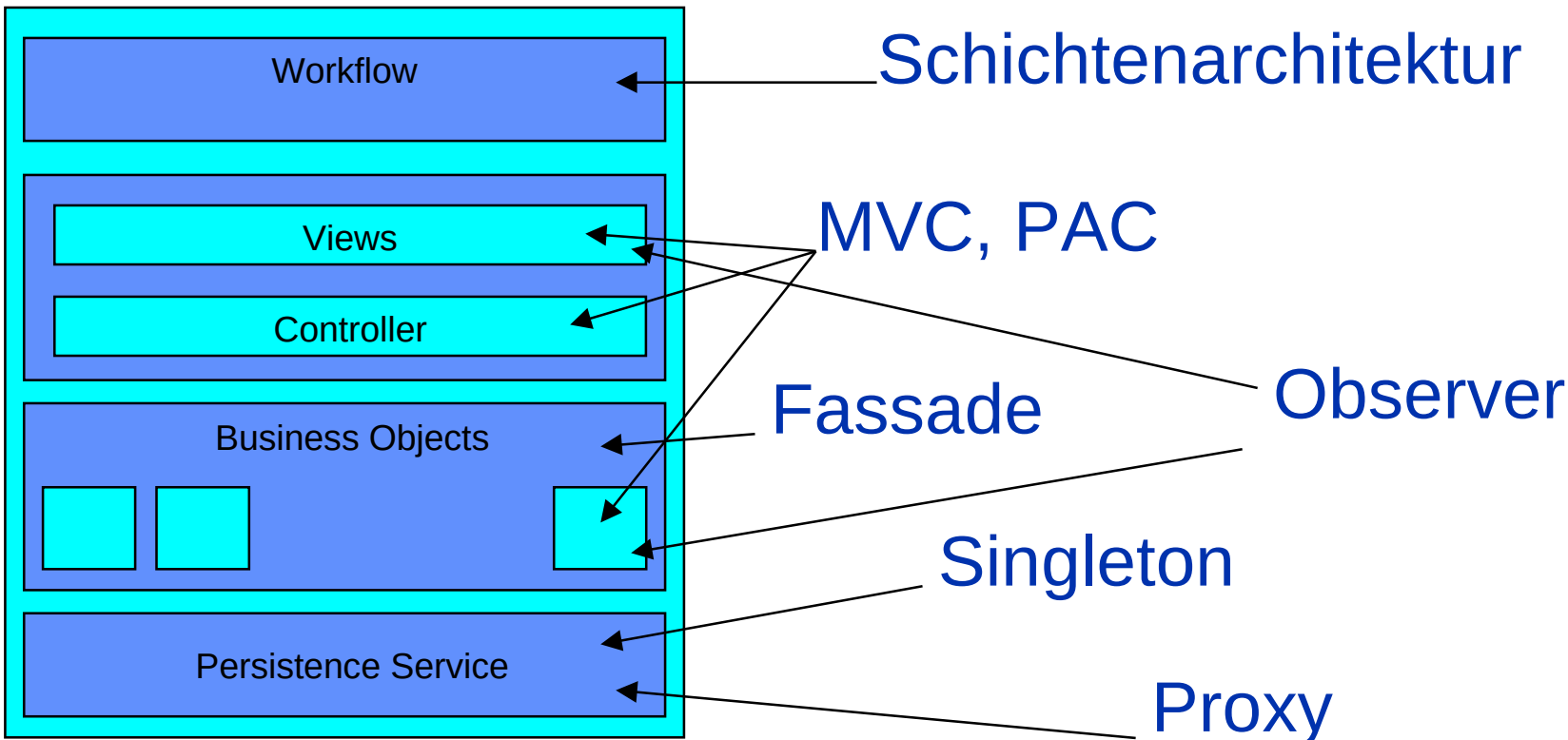
Produktserver

- Auswirkungen (exemplarisch ...)
  - Produktwissen wird plattformunanbhängig zur Verfügung gestellt
  - Benutzungsschnittstelle selten voll aus dem Produktwissen zu generieren - also hier noch Programmierung bei der Einführung neuer Produkte
  - ... Viele weitere ...

- Zum Nachlesen im Web unter ..
  - Some Patterns for Insurance Systems, Wolfgang Keller, PLoP 1998, http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Phoenix
Design- und Architektur-Patterns



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Phoenix
Fachliche Patterns

Some Patterns for Insurance Systems, Wolfgang Keller, PLoP 1998,
http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/

- Verwenden wir, um den versicherungsfachlichen Teil der Facharchitektur zu beschreiben
- Zum Beispiel
  - Wertkette der Versicherung
  - Produktserver
  - Außendienst-Shell
  - Produktbaum
  - Objekt / Ereignis / Leistung
  - Gespiegelte Stückliste ...
  - ... Weitere

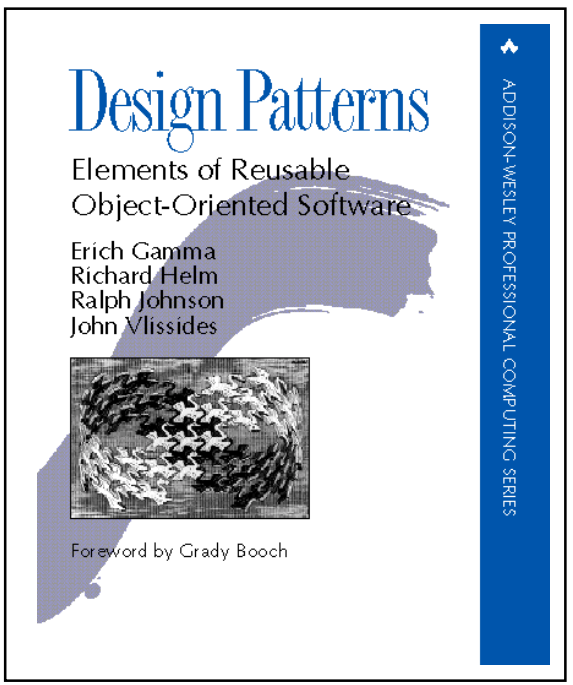
© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

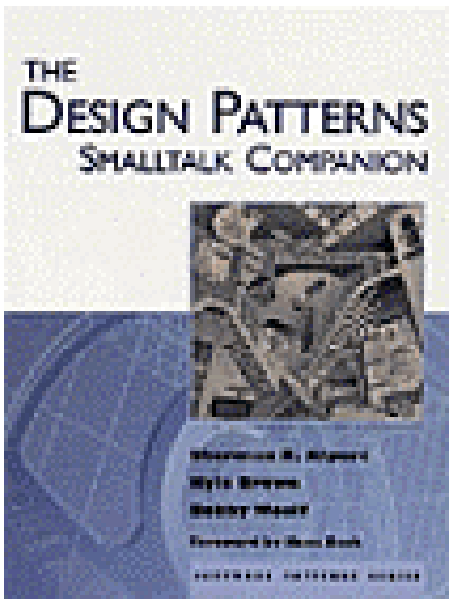
Design Patterns

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy
- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Momento
- Observer
- State
- Stategy
- Template Method
- Visitor

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Design- und Architektur-Patterns

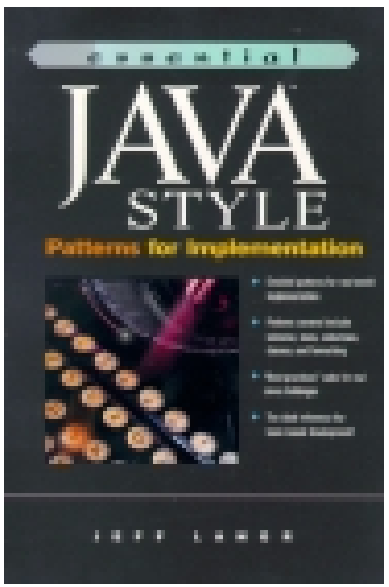




© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Programmier-Idioms, Design-Patterns

- Im Smalltalk- und Java Bereich gibt es mehrere gute Bücher, die das Erstellen eigener Programmierrichtlinien weitgehend ersetzen ..



© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Patterns

- Patterns sind nicht die Lösung aller Design Probleme
  kein Silver Bullet
- Patterns machen aus einem Anfänger keinen Spitzendesigner
- Patterns sind keine fertige Spezifikation
- Patterns ersetzen kein DV-Konzept und auch kein sauberes Design. Aber sie unterstützen beides

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Patterns

- Patterns schaffen eine **Sprache für Design**
- Das Problem und die Kräfte werden explizit gemacht - nicht nur die Lösung.
- Man bekommt fertige Skizzen für gutes Design
- Man kann sein eigenes Design besser dokumentieren
- Man wird schneller ein guter Designer

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Anhang:
Gängige Bücher

- Architektur-Patterns
  - Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern Oriented Software Architecture - A System of Patterns, Wiley 1996.
  - Mary Shaw, David Garlan: Software Architecture, Perspectives of an Emerging Discipline, Prentice Hall 1998 (nicht in Pattern Form - aber gut).
  - Martin Fowler: Patterns of Enterprise Application Architecture, Addison Wesley 2003.
- Design-Patterns
  - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns, Elements of Reusable Object-oriented Software, Addison-Wesley 1995
  - Sherman Alpert, Kyle Brown, Bobby Woolf: The Design Patterns Smalltalk Companion, Addison Wesley 1998.

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004

Bücher

- Idioms / Smalltalk
  - Kent Beck, Smalltalk Best Practice Patterns, Prentice-Hall 1996.
  - Suzanne Skublics, David Thomas, Edward Klimas: Smalltalk with Style, Prentice-Hall 1995.
- Idioms / C++
  - Jim Coplien, Advanced C++ Programming Styles and Idiomd, Addison Wesley, 1991.
  - viele weitere ....
- Analyse-Patterns.
  - Martin Fowler: Analysis Patterns; Addison Wesley Longman, 1997.
  - David C. Hay, Data Model Patterns, Dorset House Publishing, 1995.

© 2004, Bernhard Anzeletti, Wolfgang Keller, Rudolf Lewandowski, Klaudius Messner, All rights reserved, Technische Universität Wien, SS2004