

Softwarearchitektur

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

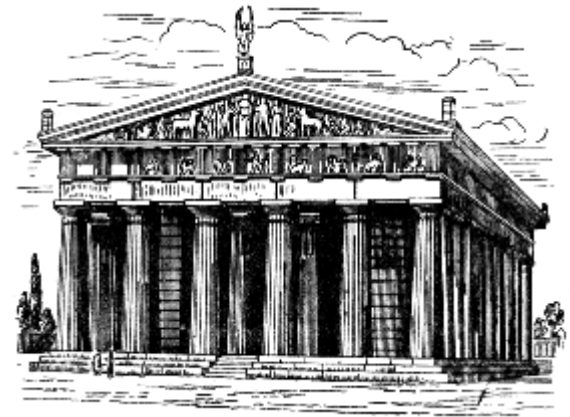
6a. Ausprägungen und Wiederverwendung

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

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- Ausblick auf Komponenteninfrastrukturen: CORBA
- Zusammenfassung
- Literaturverzeichnis

- *Nächste Stunde: Wiederverwendung per*
 - *Architekturmuster*
 - *Referenzarchitektur*
 - *Produktlinienarchitektur*

Inhalt

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- Ausblick auf Komponenteninfrastrukturen: CORBA
- Zusammenfassung
- Literaturverzeichnis

Wiederverwendung als Weg aus der Softwarekrise?

- Seit vielen Jahren sieht man in der Wiederverwendung von Software-Artefakten die Lösung zu vielen Problemen der Softwareentwicklung
 - bessere Qualität durch Nutzung eines „gereiften“ Produkts
 - geringere Kosten durch mehrfache Verwendung
 - kürzere Entwicklungszeiten aufgrund weniger Eigenentwicklung
- Beispiel Kostenreduktion:

Reuse rates	0%	25%	50%
Time [m.m.]	81,5	45	32
# Developers	8	6	5
Costs per l.o.c. [DM]	70	37	28
Lines per m.m.	165	263	370
Savings [DM]	-	~ 400.000	~ 560.000
Savings	-	~ 45%	~ 60%

From: C. Jones, *The Impact of Reusable Modules and Functions*, in *Programming Productivity*, Mc Graw Hill, 1986

Warum Wiederverwendung oft nicht klappt...

■ Technische Probleme

- Wiederverwendung meist nur auf Code-Ebene
- Fehlerhaftes Subtyping verhindert Wiederverwendung
- Vernachlässigte Dokumentation und Spezifikationen
- Korrektheit der wiederverwendeten Teile

■ Organisatorische Probleme

- Wiederverwendung selten geplant, zu spät gemacht
- Motivation und Anreize für Wiederverwendung
- Kaum Handelsplätze für Software, Anbieter und Nutzer finden nur schwer zusammen
- Aufwand für Wiederverwendung u. U. höher als Nutzen

zu flexibel, Anpassung
sehr aufwändig



zu starr festgelegt
auf einen Anwendungsfall

Wo Wiederverwendung erfolgreich ist...

- **Ausnahmen**
 - Bibliotheken (MFC, Swing, ...)
 - Datenbanken
 - Kommunikationsprotokolle
 - Transaktionsmonitore
 - Middleware (CORBA, COM, J2EE)
- **Erfolgsfaktoren**
 - Technische Erfolgsfaktoren
 - präzise definierte, möglichst standardisierte Schnittstellen
 - abgeschottete Implementierung (Geheimnisprinzip)
 - niedrige Einsatzschwelle: wenig Konfiguration, verständliche Dokumentation, ...
 - Organisatorische Erfolgsfaktoren
 - Systematisches Vorgehen bei der Entwicklung Nutzung
 - Anreize und Motivation für die Wiederverwendung

Kommerzieller Marktplatz für Komponenten

http://www.componentsource.com/

Advanced Search | Shopping Cart | My Account | Contact Us

Logon | Register | Sign In

ComponentSource
Site Map Stores: All Products | .NET | COM | MTS | Java | J2EE | VCL | C++ | Tools | Other

The world's largest
COMPONENT MARKETPLACE
for software development

Browse & Buy

- Categories
- Publishers
- Product A-Z
- Best Sellers
- Top Downloads
- New Products
- New Versions
- My Account
- Shopping Cart
- Previous Orders
- Quotes
- Shopping Guide
- Payment Options
- Support

Featured Products

Imaging Pro
More Info | Buy Now
Add robust image processing to your applications.
LEADTOOLS Raster

Imaging Pro is an API, C++ Class libs, ActiveX and VCL toolkit with 80+ raster file formats (JPEG, GIF, TIFF...), 70+ Image Transforms and Filters, extensive display options, TWAIN scanning, printing, thumbnail browser, internet and database imaging functions and much more.

TX Text Control ActiveX
More Info | Buy Now
Build your own word processor. TX Text Control ActiveX gives your application the ability to read, edit, and create documents in Microsoft Word formats including Word 2000.

Spread
More Info | Buy Now
Add spreadsheet functionality to your applications. Use FarPoint's Spread 6 to easily incorporate advanced grid and high-level spreadsheet features into your applications.

What's on the site

New Visitor
If you are new to the ComponentSource Web site, the following links may help you find the information you are looking for.

Other Sections...

- About Components
- About Us
- Build Components
- News Center
- SAVE-IT™ Enterprise Base
- Sell Components

Browse Stores

All Products .NET, .NET Remoting, .NET Ready (COM), C++, C++ Builder, COM, Component Building Tools, Delphi, DLL & Libraries, J2EE, Java, JavaBeans, JBuilder, Linux, MTS, Tools, VBA, VCL, Visual Basic, Visual Studio .NET, Visual Studio, Windows CE

Stop Press

Get ComponentOne V8 Power!
Give your software projects V8 power from ComponentOne. Now available for immediate download are new Version 8 updates for the following best selling ComponentOne products: True DBGrid Pro 8.0, VSFlexGrid Pro 8.0, VSView 8.0 Classic Edition, VSView 8.0 Reporting Edition, ComponentOne Chart 8.0, ComponentOne WebChart 8.0, ComponentOne Query 8.0, TrueDataControl 8.0 and True DBLIST 8.0

New ComponentOne Studio Enterprise
Add grid, reporting, charting, data manipulation and user interface capabilities to your .NET, ASP.NET and ActiveX applications. ComponentOne Studio Enterprise is a combination of three individual ComponentOne Studio subscriptions delivering you 30 components.

Top Downloads

Developer Express XtraGrid .NET Suite
Buy Now | Other Grid Components
Add a bound or unbound, native .NET grids to your Visual Studio .NET applications. Developer Express XtraGrid .NET Suite is a 100% C# Grid, CardView, and Editors Library built specifically for Visual Studio .NET.

Sax ActiveX SmartUI
Buy Now | Other User Interface Components
Build a complete application user interface with a single comprehensive component. Sax ActiveX SmartUI provides all the functionality required for a modern application front-end (Office2000/Win2000) including: MenuBars, ToolBars, StatusBars, OutlookBars, TreeViews, ListBoxes, OptionLists, PropertyLists, Property Toolboxes, flat Tabstrips and much more.

AddFlow
Buy Now | Other Diagramming Components
Create interactive flowcharts and workflow diagrams. AddFlow is an ActiveX component that is useful each time you need to display and use relationships between objects in your application.

ASPxGrid
Buy Now | Other Grid Components
Add Outlook style grids to your ASP.NET applications. The ASPxGrid by Developer Express is a 100% C# based grid control for ASP.NET.

RAD XML Persistent Tools
Buy Now | Other Data Storage Components
With almost no coding create sophisticated TreeViews, and VB-Like data-entry Property Dialogs, that are XML-ready and can share their data through built-in multi-user XML-Database engine.

YARCHART XGantt LC
Buy Now | Other Charting and Graphing Components
Add interactive Gantt diagram functionality to your desktop or web application. YARCHART XGantt LC is an ActiveX control which lets you graphically display, edit and print your data for project or resource planning in Gantt diagrams or control panels.

DockStudioXP
Buy Now | Other Toolbar Components
Provide Visual Studio .NET or Microsoft XP/2000 style toolbars and dockable windows in your applications. DockStudioXP allows you to implement docking flexibility with no complex manual coding.

ComponentOne Studio for .NET
Buy Now | Other Button and Cursor Design Components
Maintain a complete collection of components for all aspects of application development. ComponentOne Studio for .NET provides existing .NET tools, any new .NET tools and updates released within your 1 year subscription period, plus email support.

ImagXpress

Product Search

All Products
Enter Search Go

SALES & SUPPORT

- Call Toll Free
- Email Sales
- Email Support

Sponsored Links

Best Sellers

- Janus GridEX
- InfraGrids NetAdvantage Suite
- ActiveReports for .NET
- ActiveReports
- Janus Controls Suite

View All

Microsoft .net msdn dev2dev .www.sun.com Sun

New Products

- PowerTCP SSL Sockets for .NET
- Transcender WANCert/Security (640-442)

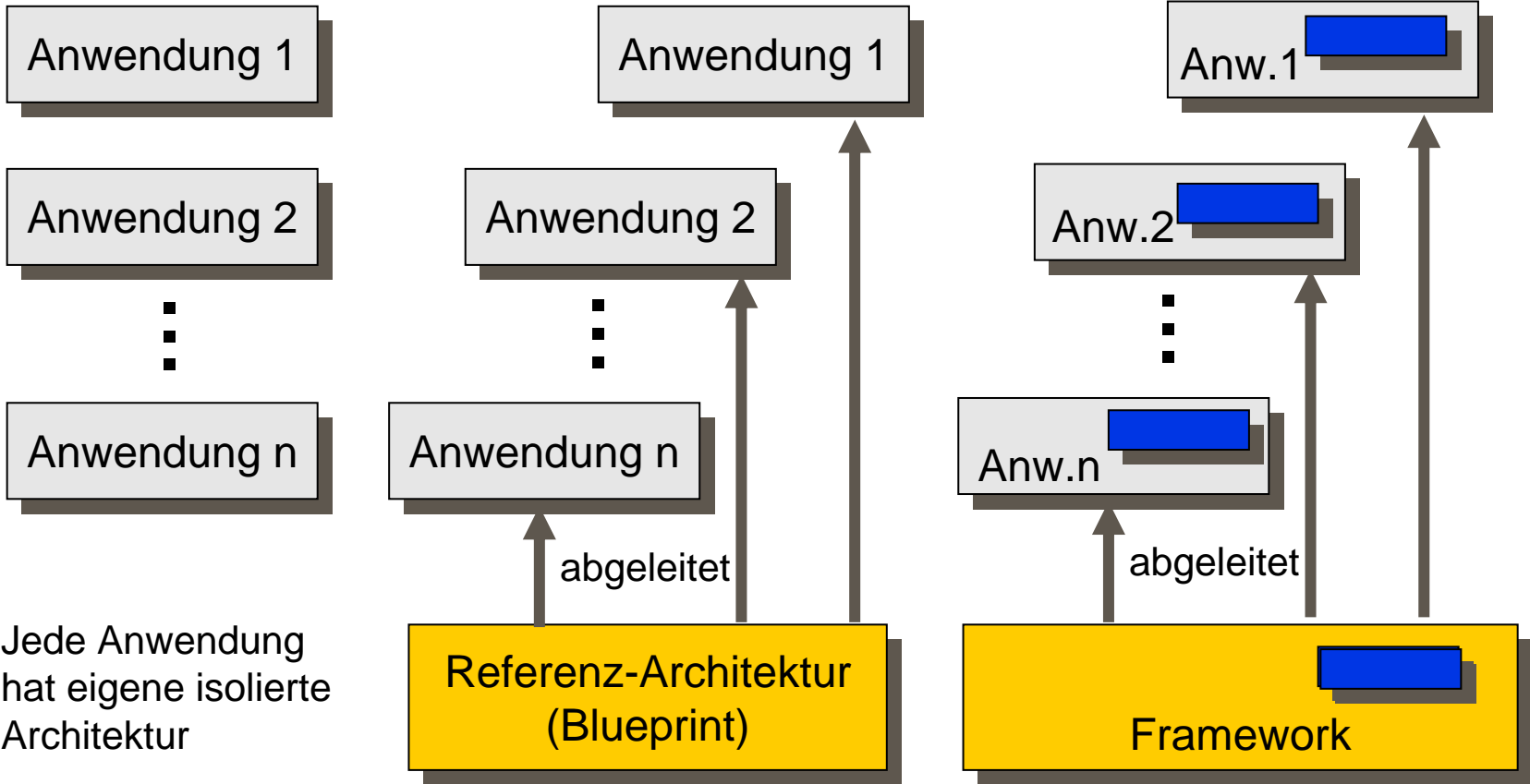
Wiederverwendung von Architekturen

- Beobachtung: mit wachsender Komplexität eines Systems ist das Zusammenspiel der Komponenten kritischer als ihre Funktionalität.
- Die Vision einer „plug-and-play“ Softwareentwicklung kann nur real werden, wenn neben einzelnen Komponenten auch die Organisation und das Zusammenspiel der Komponenten (also die Architektur) einfach wiederverwendet werden können
- Im Bereich Softwarearchitektur ist Wiederverwendung an vielen Stellen wünschenswert:
 - Spezifikation (z.B. Schnittstellen, Protokolle, Strukturen, etc.)
 - Implementierung (z.B. elementare Komponenten einer Architektur)
 - Konzepte und Zusammenhänge (z.B. Architekturtreiber)
 - Methodik (z.B. „Best Practice“, Cookbook-Ansätze)
- Diese Artefakte verbessern nicht zuletzt die Kommunikation unter den Entwicklern („klassische Drei-Schichten-Architektur“)

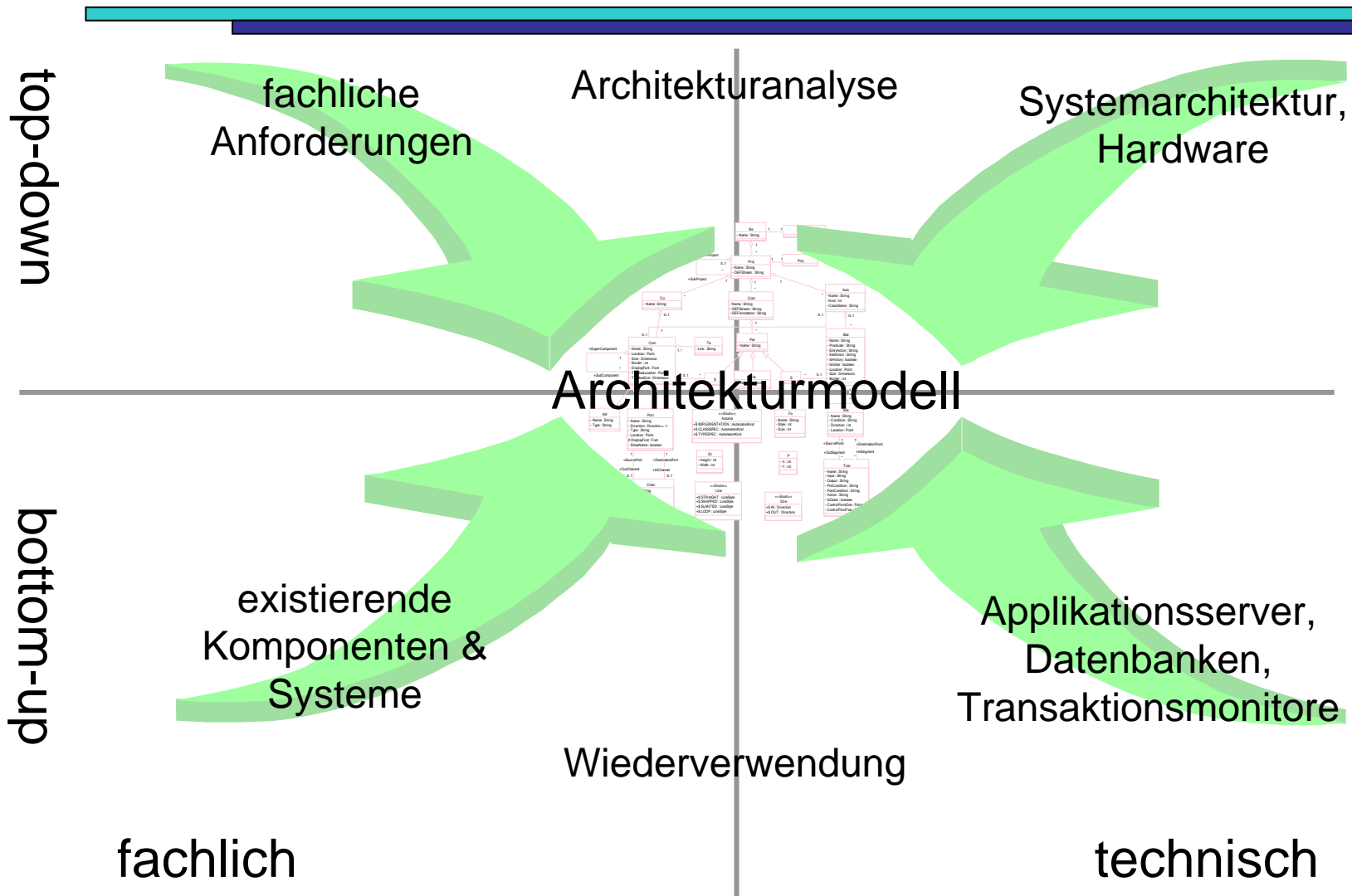
Übersicht

- Der Softwarearchitekt kann bereits heute bewährte Ideen, Konzepte und Umsetzungen bedarfsgerecht für „seine“ Architektur wiederverwenden, beispielsweise in Form von:
 - konkreten Referenzarchitekturen
 - gattungsspezifisch (z.B. Informationssysteme)
 - domänenspezifisch (z.B. Überwachungssysteme)
 - Architekturmustern (problembezogene Darstellung eines Lösungsaspekts ähnlich zu Entwicklungsmustern)
 - Schnittstellen- und Protokollstandards
 - Frameworks (einfach zu ergänzendes Grundgerüst einer Anwendung)
 - Middleware und Komponententechnologien

Rolle von Referenzarchitekturen und Frameworks



Systematische Wiederverwendung



Inhalt

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- Ausblick auf Komponenteninfrastrukturen: CORBA
- Zusammenfassung
- Literaturverzeichnis

Frameworks: Definition

- Definition: ein Framework ist ein „halbfertiges“ Anwendungssystem
- Frameworks
 - definieren in ihrem Anwendungsbereich die Softwarearchitektur
 - definieren die Regeln, nach denen sie zur vollständigen Anwendung weiterentwickelt werden
 - enthalten eine wiederverwendbare, allgemeingültige und durch das Zusammenspiel einzelner Instanzen des Frameworks vordefinierte Verarbeitungslogik
 - beeinflussen den gesamten Software-Entwicklungsprozess
 - erlauben die Wiederverwendung von Spezifikation und Code
 - realisieren das „Hollywood-Prinzip“ („Don't call us – we call you!“)

Blackbox und Whitebox-Frameworks

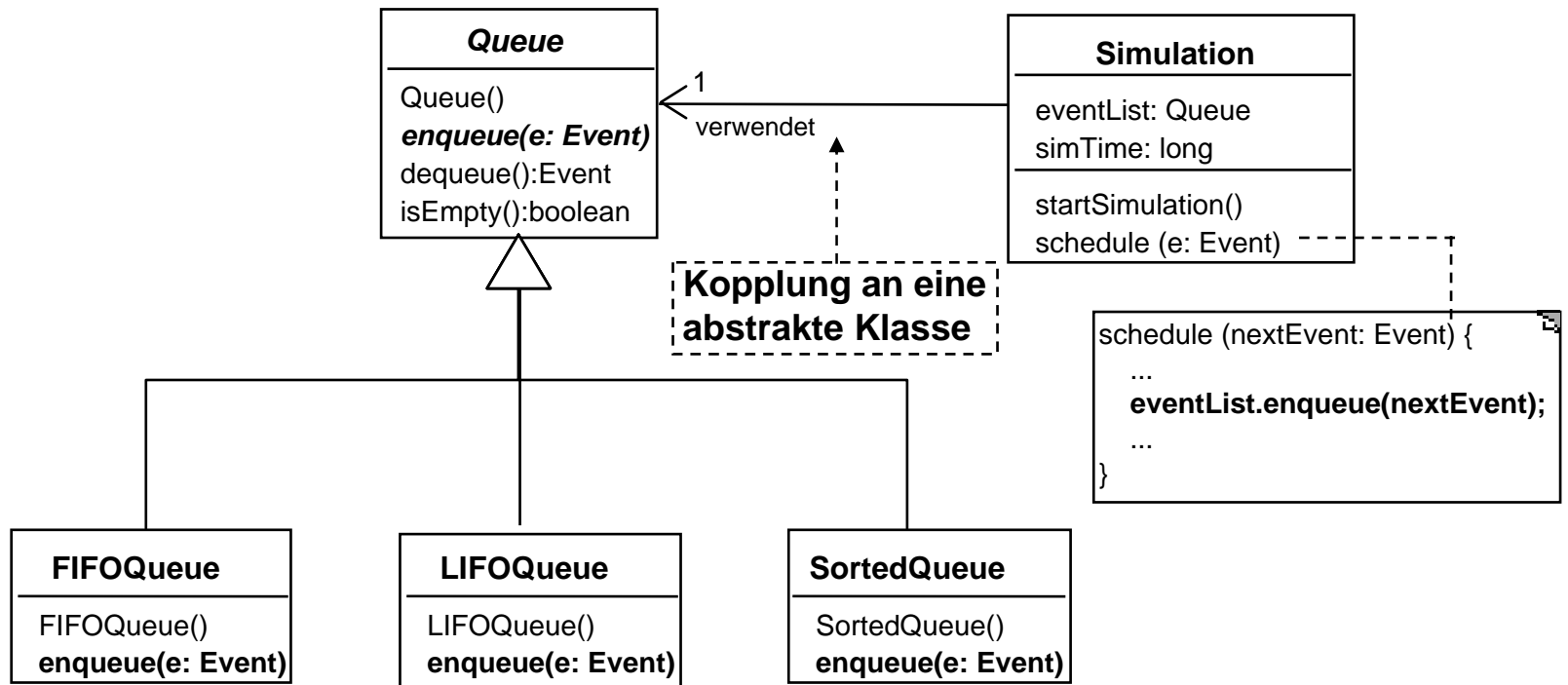
- Anpassung durch Vererbung oder Komposition
 - Whitebox-Frameworks erlauben Wiederverwendung durch Ableiten von abstrakten Klassen
 - Anwender muss Verhalten abstrakter Klassen implementieren
 - Blackbox-Frameworks ermöglichen Wiederverwendung durch Parametrisierung oder Komposition von Klassen
 - basieren auf einer Menge schon implementierter Klassen
 - Anwender bestimmt Verhalten durch Einfügen bestimmter Klassen
- Interaktion mit dem Framework
 - “calling” Frameworks sind aktiv, d.h. beinhalten den zentralen Programmablauf und rufen andere Teile der Anwendung auf
 - “called” Frameworks sind passiv und werden von anderen Teilen der Anwendung aufgerufen (eher selten)

Objektorientiert oder komponentenbasiert

- Traditionelle, objektorientierte Frameworks basieren stark auf dem Prinzip der Erweiterung und Wiederverwendung durch (Code-) Vererbung. Dies erweist sich in der Praxis als nachteilig:
 - „Vererbungsschnittstellen“ sind meist unpräzise
 - durch Vererbung ergeben sich starke Bindungen im Code
- Durch die Verwendung von Delegation anstelle von Vererbung ergeben sich flexible, objektorientierte Frameworks.
- Auf der Ebene von Komponenten wird ebenfalls der Mechanismus der Delegation genutzt – es ergeben sich komponentenorientierte Frameworks oder kurz Komponentenframeworks.

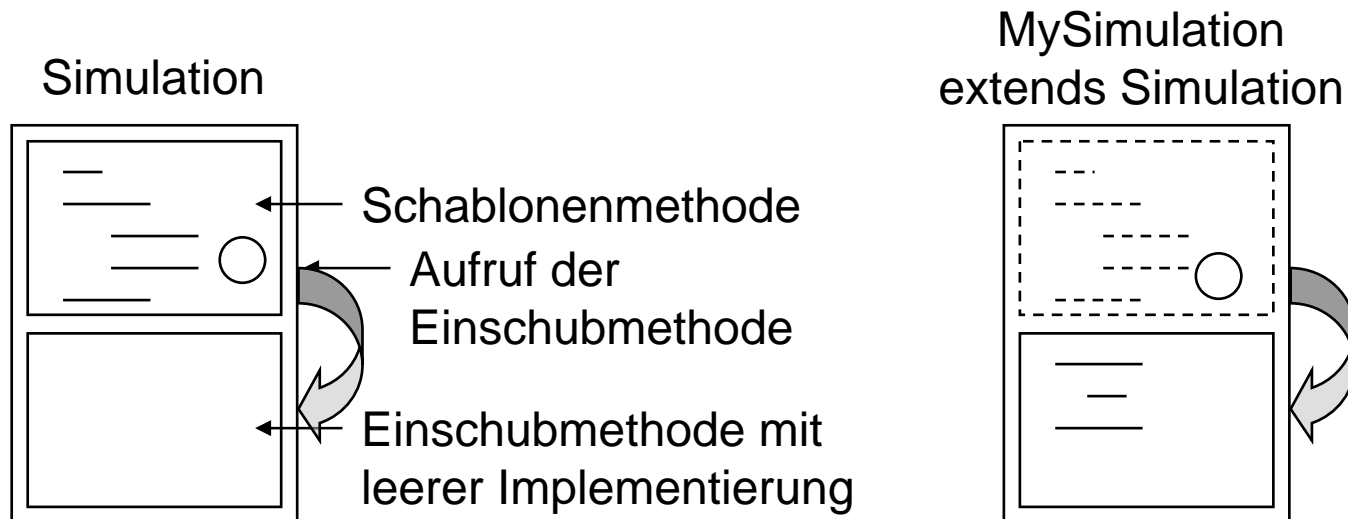
Abstrakte Kopplung [Pre97]

- abstrakte Klassen definieren eine Schnittstelle, zu der abgeleitete, implementierte Klassen kompatibel sind (Polymorphismus)



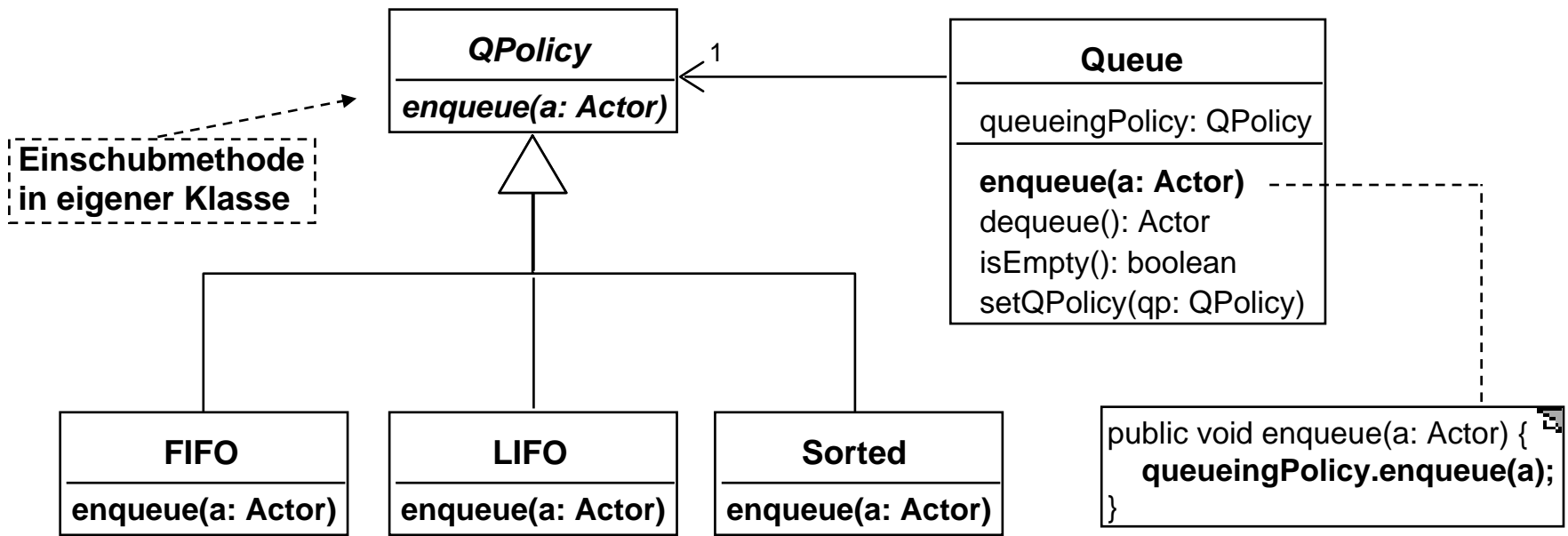
Einschubmethoden / Hot Spots [Pre97]

- Flexibilität durch Platzierung von "Einschubmethoden"-Aufrufen in "Schablonenmethode"
- Grundidee: Verhalten eines Objektes verändern, ohne den Quellcode der zugehörigen Klasse ändern zu müssen.



Einschubklassen [Pre97]

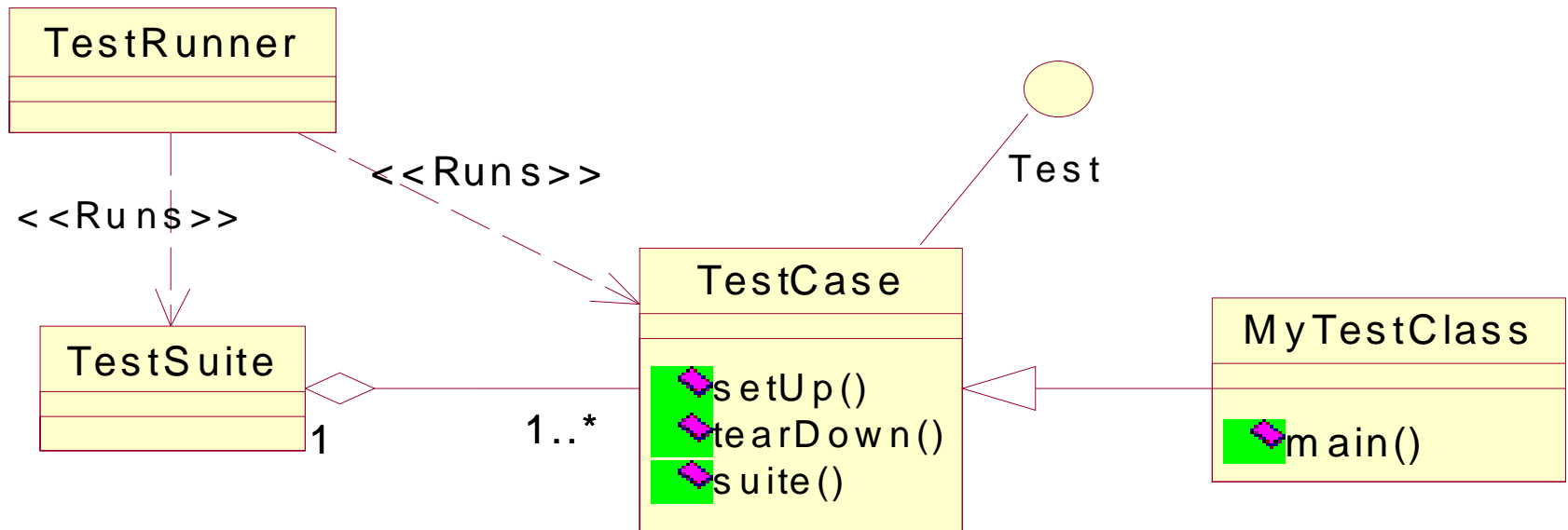
- Problem : Schablonen- und Einschubmethode in einer Klasse erlauben keine Anpassung zur Laufzeit!
- Lösung : Komposition statt Vererbung. Einschubmethode in eigener Klasse kapseln (Einschubklasse)



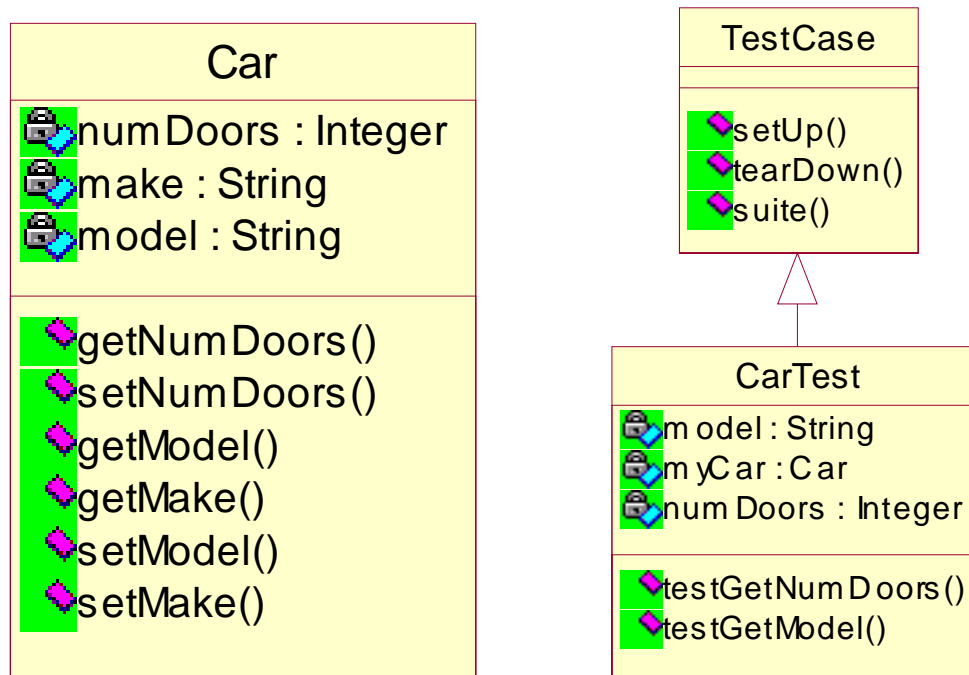
Beispiel: JUnit

- JUnit ist ein handliches Test-Framework in Java
- Eine Reihe von erweiterbaren Klassen realisiert die Testfunktionalität (Fehler zählen, auswerten, darstellen, Testfälle durchspielen)
- Es bietet eine einfache, doch komfortable Oberfläche
- Grundlage einer Infrastruktur für "extreme testing" (nightly build, nightly test)
- Entwickelt von
 - Kent Beck, kentbeck@compuserve.com
 - Erich Gamma, erich_gamma@acm.org

JUnit: Klassendiagramm



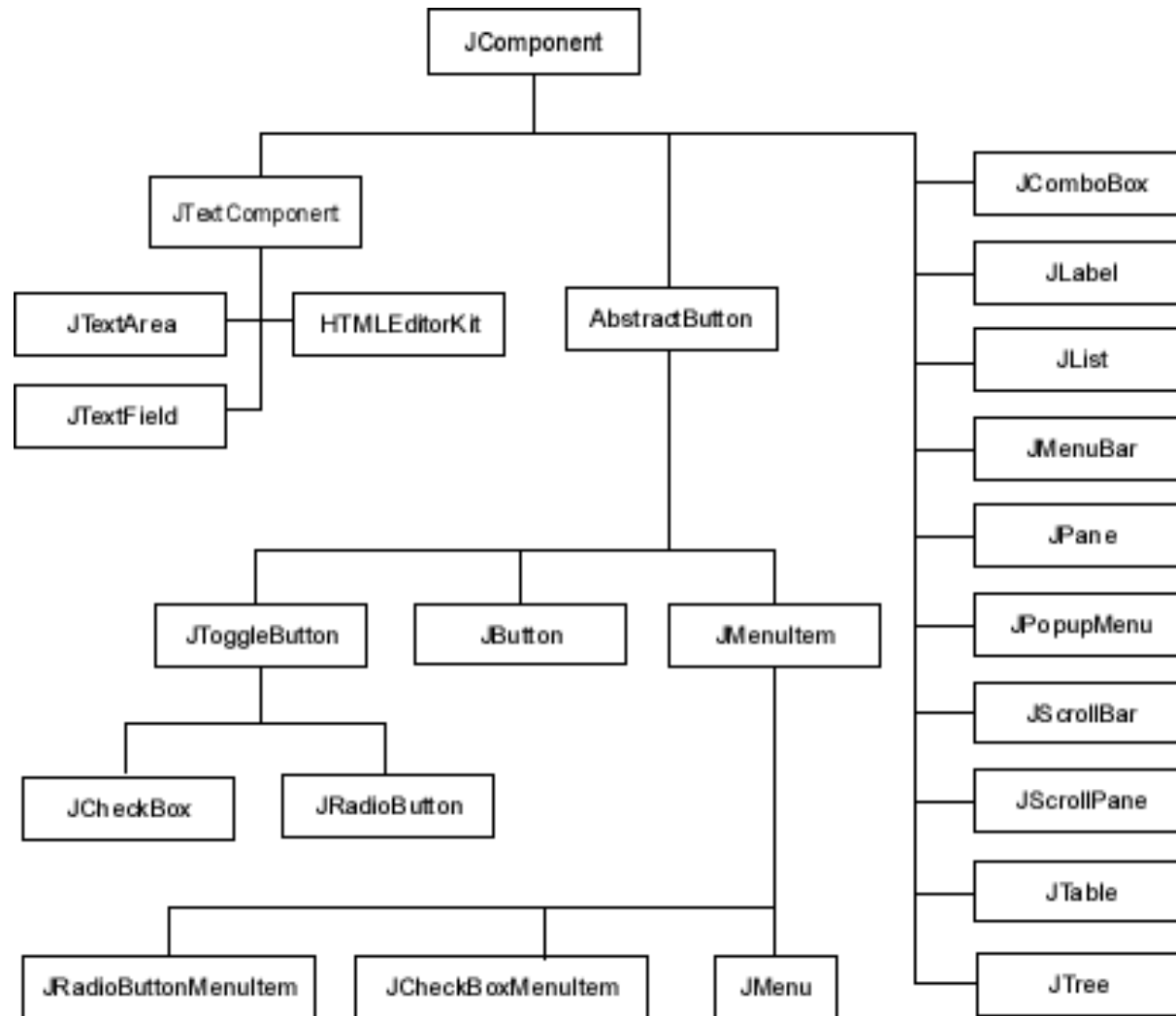
JUnit: Testklasse und Testfall



JUnit: Realisierung Testfall-Klasse „CarTest“

```
public void setUp() {
    myCar = new Car ();
    numDoors = 4;
    model = "BMW 318i";
}
public void testGetNumDoors() {
    assert (numDoors == myCar.getNumDoors());
}
public void testGetModel() {
    assert(myCar.getModel().equals(model));
}
public static void main(String[] args) {
    junit.textui.TestRunner.run(suite());
}
```

Beispiel: Das Swing-Framework



Beispiel: Nutzung von Swing

```
1: import java.awt.event.*;
2: import javax.swing.*;
3:
4: public class Framework extends JFrame {
5:     public Framework() {
6:         super("Application Title");
7:         // Hier die Komponenten einfügen
8:     }
9:
10:    public static void main(String[] args) {
11:        JFrame frame = new Framework();
12:
13:        WindowListener l = new WindowAdapter() {
14:            public void windowClosing(WindowEvent e) {
15:                System.exit(0);
16:            }
17:        };
18:        frame.addWindowListener(l);
19:
20:        frame.setVisible(true);
21:    }
22: }
```


Entkopplung von Swing

- Die wünschenswerte Entkopplung von Objektmodell und Swing wird über das Adapter-Pattern realisiert

```
public class AddressBook{
    List personList;

    public int getSize(){...}
    public int addPerson(...){...}
    public Person getPerson(...){...}
}
```

```
public interface TableModel
{
    // return number of rows
    getRowCount();

    ...
}
```

```
public class AddressBookTableAdapter
implements TableModel
{
    AddressBook ab;

    public AddressBookTableAdapter(
        AddressBook ab ) {
        this.ab = ab;
    }

    //TableModel impl
    public getRowCount(){
        ab.getSize();
    }

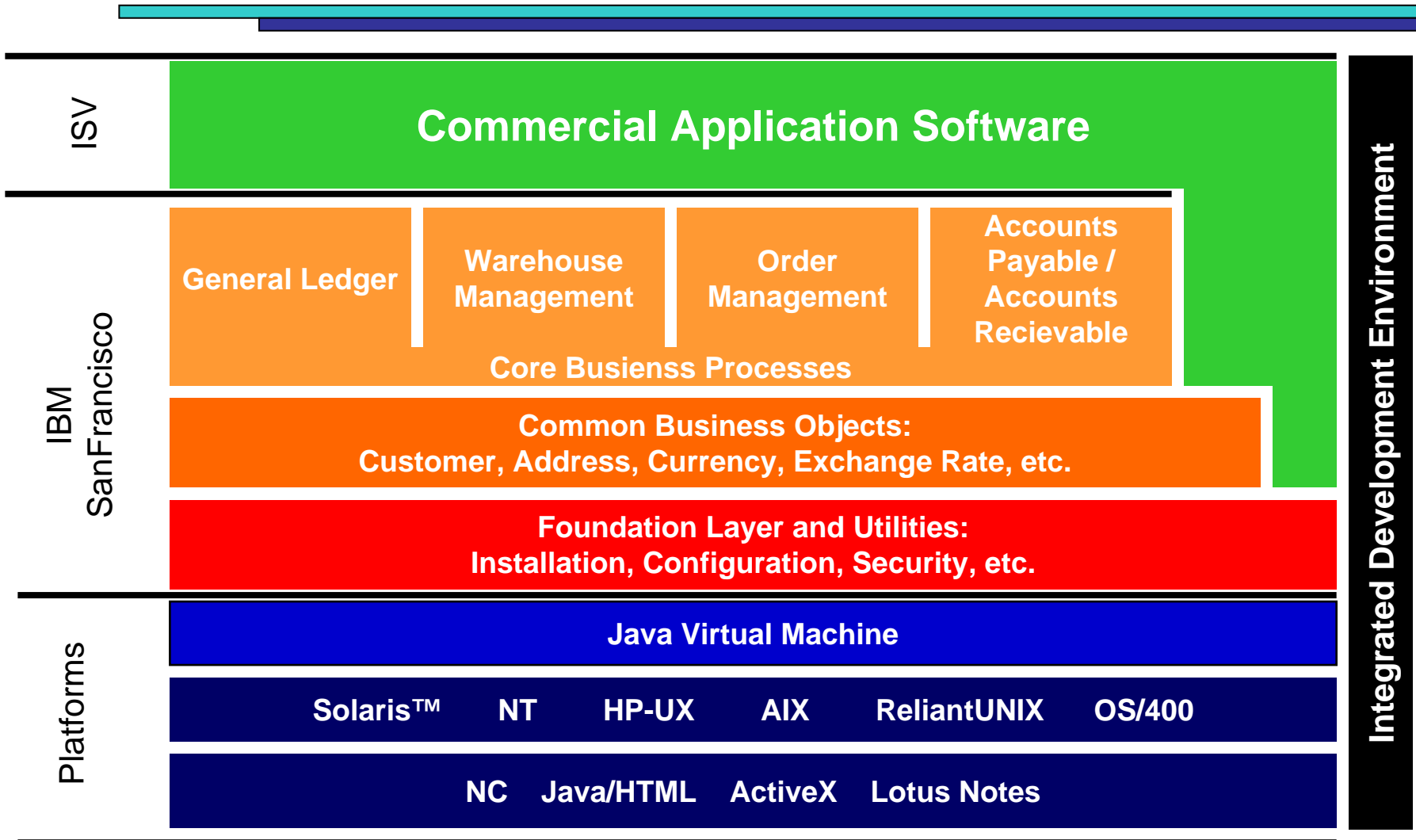
    ...
}
```

Ausblick: IBM's San Francisco



- Vielversprechender Ansatz zur Standardisierung von Geschäftsobjekten und –prozessen im Rahmen eines Frameworks für Informationssysteme
- Das Framework enthält
 - Gängige Geschäftsobjekte, die domänenübergreifend benötigt werden
 - Breites Spektrum an Konstrukten für unterschiedliche Geschäftsfelder
 - Schnittstellen zu zentralen Geschäftsprozessen des Finanzmanagements
 - übliche Funktionalitäten eines Informationssystems
- San Francisco wurde inzwischen vom Markt genommen...

San Francisco: Architektur

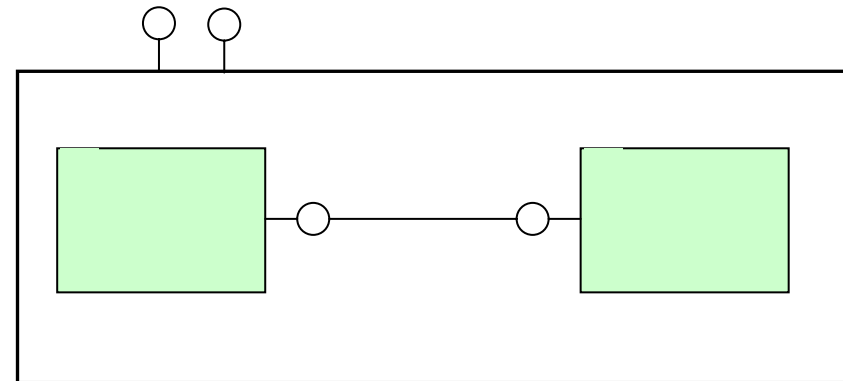


Bewertung

- Beobachtung anhand des San Francisco Frameworks
 - wie viele andere Frameworks strebt auch SF nach „alleiniger Herrschaft“ in der Anwendung
 - während der eigentliche Nutzen und die Kompetenz von SF im Aufbau standardisierter Fachmodelle und Geschäftsprozesse liegt, werden technische Aspekte adressiert, die evtl. andere Frameworks besser beherrschen
 - Wünschenswert wäre eine Möglichkeit, mehrere spezialisierte Frameworks miteinander kombinieren zu können.
 - Vorteil: Reduzierte Komplexität der Einzellösungen
 - Nachteil: Zusammenspiel und Aufteilung des Kontrollflusses komplizierter
 - -> aktuelles Forschungsthema

Komponentenframeworks

- Ein Komponentenframework ist eine besondere Komponente, deren Realisierung freie Stellen („plugs“) vorsieht zum Einfügen bestimmter, bedarfsabhängiger Teilkomponenten
- In der Blackbox-Sicht werden nur die Plugs der Komponente dargestellt
- In der Glassbox-Sicht werden zudem interne Komponenten des Frameworks repräsentiert, die einen Teil seiner Realisierung darstellen



Bewertung: Frameworks

- Hohe Wiederverwendung nicht nur von Design, sondern gleichfalls von Artefakten der Implementierung
- sehr kurze Entwicklungsdauer für Anwendungen
- weniger fehleranfällig
- Führen langfristig zu einer Standardisierung des jeweiligen Anwendungsbereichs
- Konzentration auf die wesentlichen Teile der Anwendung
- hoher Einarbeitungsaufwand für den Anwender
- geben evtl. Programmiersprache vor
- nur für einen bestimmten Problembereich anwendbar
- hoher Entwicklungsaufwand für das Framework
- Frameworks „reissen Kontrolle an sich“

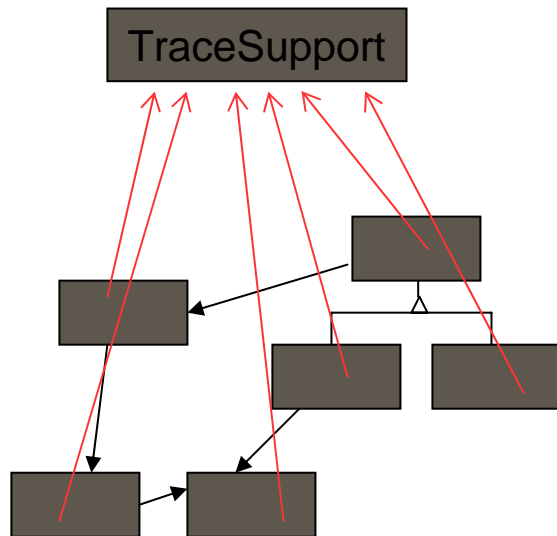
Inhalt

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- Ausblick auf Komponenteninfrastrukturen: CORBA
- Zusammenfassung
- Literaturverzeichnis

Aspect-oriented programming (AOP) [KHH+94]

- Verallgemeinerung von Architektursystemen:
 - Architektursysteme sind spezielle Aspektsysteme: Ihre beiden Aspekte sind anwendungsspezifische Funktionalität und Kommunikation
 - Weitere Aspekte: Synchronisation, Verteilung, Persistenz etc.
- Transparenz durch aspektbeschreibende Sprachen
- Basiskommunikation: ein Aspekt
- Generierung: „Weben“ aus Aspektspezifikationen
- Schnittstellen: Join points, an denen Aspekte verwoben werden
- Adaption:
 - Interne Adaption: Aspekte bilden Teile von Komponenten, d.h. interne Adaption durch Austausch von Aspekten
 - Externe Adaption: Weben von Klebecode um Komponenten herum

AOP: Beispiel



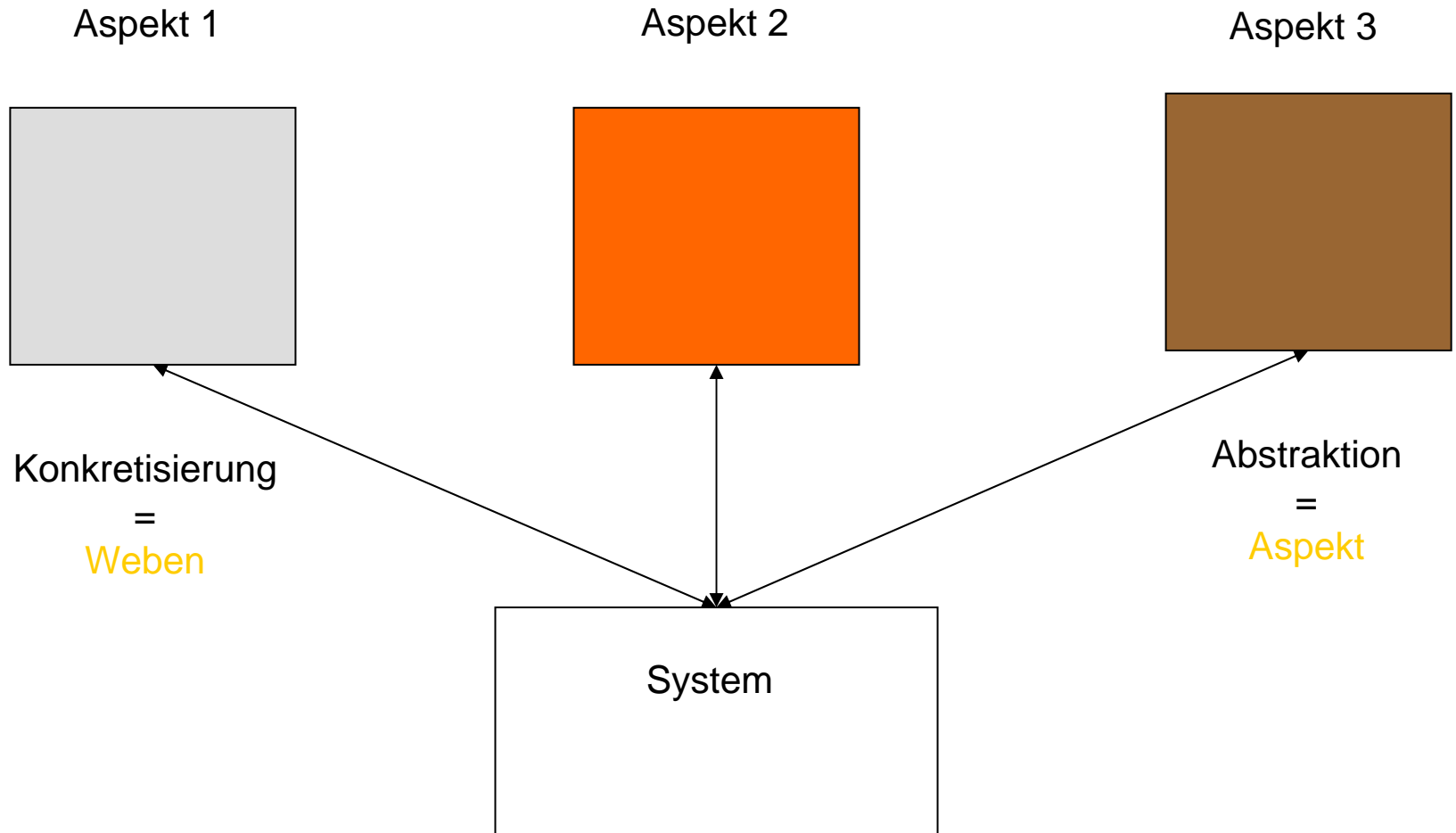
```
class TraceSupport {
    static int TRACELEVEL = 0;
    static protected PrintStream stream = null;
    static protected int callDepth = -1;

    static void init(PrintStream _s) {stream=_s;}

    static void traceEntry(String str) {
        if (TRACELEVEL == 0) return;
        callDepth++;
        printEntering(str);
    }
    static void traceExit(String str) {
        if (TRACELEVEL == 0) return;
        callDepth--;
        printExiting(str);
    }
}
```

```
class Point {
    void set(int x, int y) {
        TraceSupport.traceEntry("Point.set");
        _x = x; _y = y;
        TraceSupport.traceExit("Point.set");
    }
}
```

Aspekte als Sichten



Merkmale von Aspekten

- Die Anweisungen zur Protokollierung treten in mehreren Funktionen an bestimmten Stellen auf (Cross-Cutting)
- Sie können mit herkömmlichen Sprachmitteln aus den Funktionen nicht völlig herausgelöst werden (Modularisierung)
- Die Vorausplanung von Aspekten fällt schwer (Pre-Planning Problem)

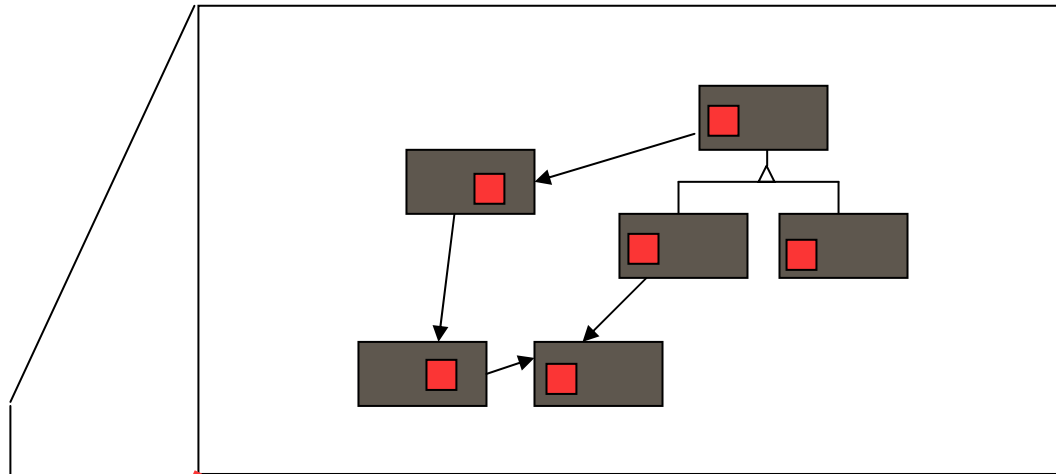
Probleme

- Orthogonale Aspekte
 - Keine Interaktionen zwischen den Aspekten
 - Weben einfach
- Nicht-orthogonale Aspekte
 - Z.B. Funktionalität und Synchronisation, Funktionalität und Verteilung, Protokolle und Synchronisation etc.
 - Getrennte Spezifikation unmöglich
 - Weben unklar
- Aspekt als Sicht
 - Abstraktion von Systemeigenschaften
 - Umkehrungsfunktion nicht immer
 - Eindeutig
 - Widerspruchsfrei mit anderen Aspekten

AOP: Hierarchische Organisation

- Funktionen, Prozeduren und Objekte unterstützen die hierarchische Organisation von Systemen
- In der Regel bestimmen die fachlichen Anforderungen die Hierarchie
- Aspekte liegen "quer" zu diesen Hierarchien
- ☞ AOP ist für alle verallgemeinerten prozeduralen Sprachen möglich

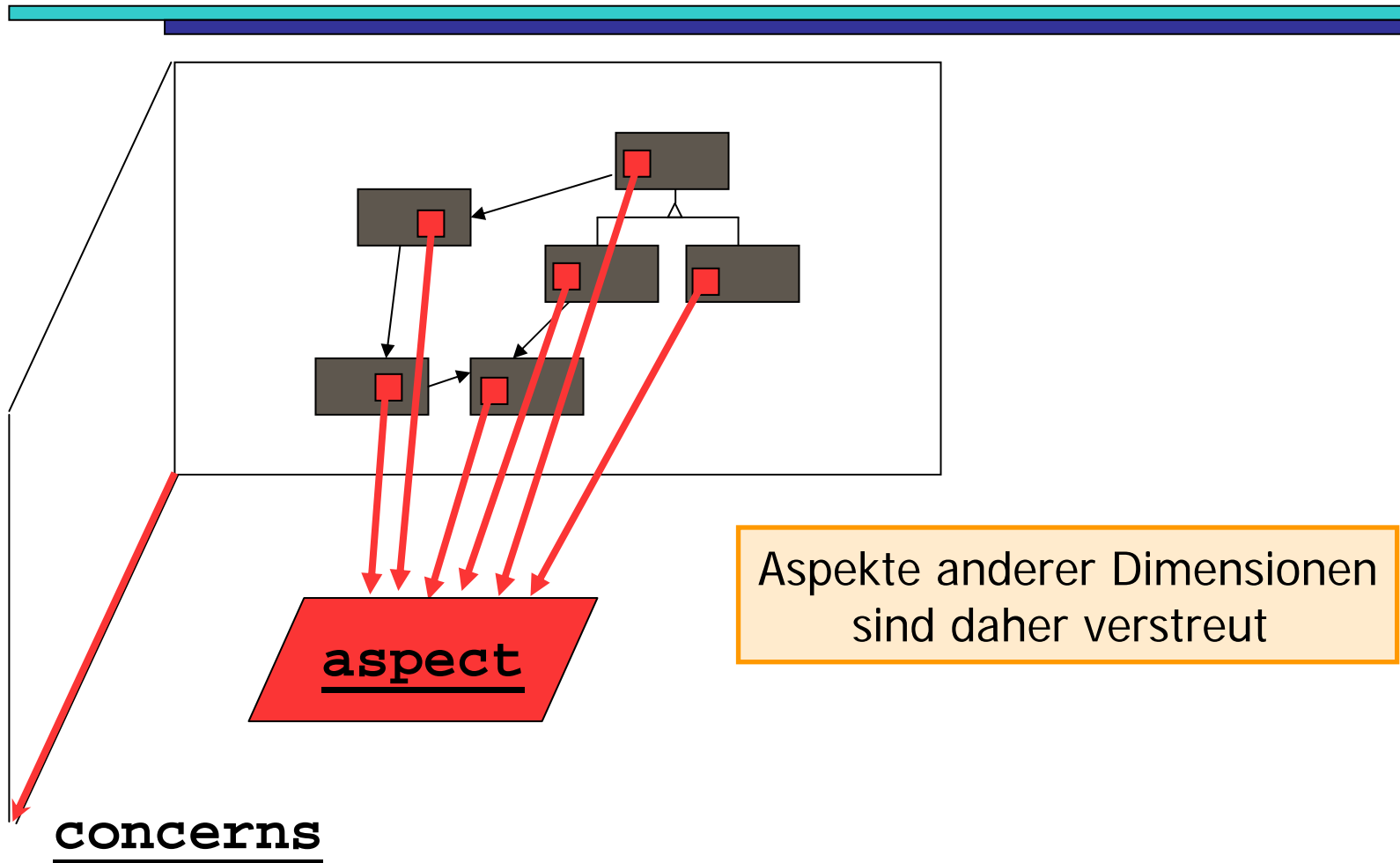
AOP: Opening a New Dimension in Software Development



Die hierarchische Zerlegung ist selten für mehr als eine "Dimension" optimal.

concerns

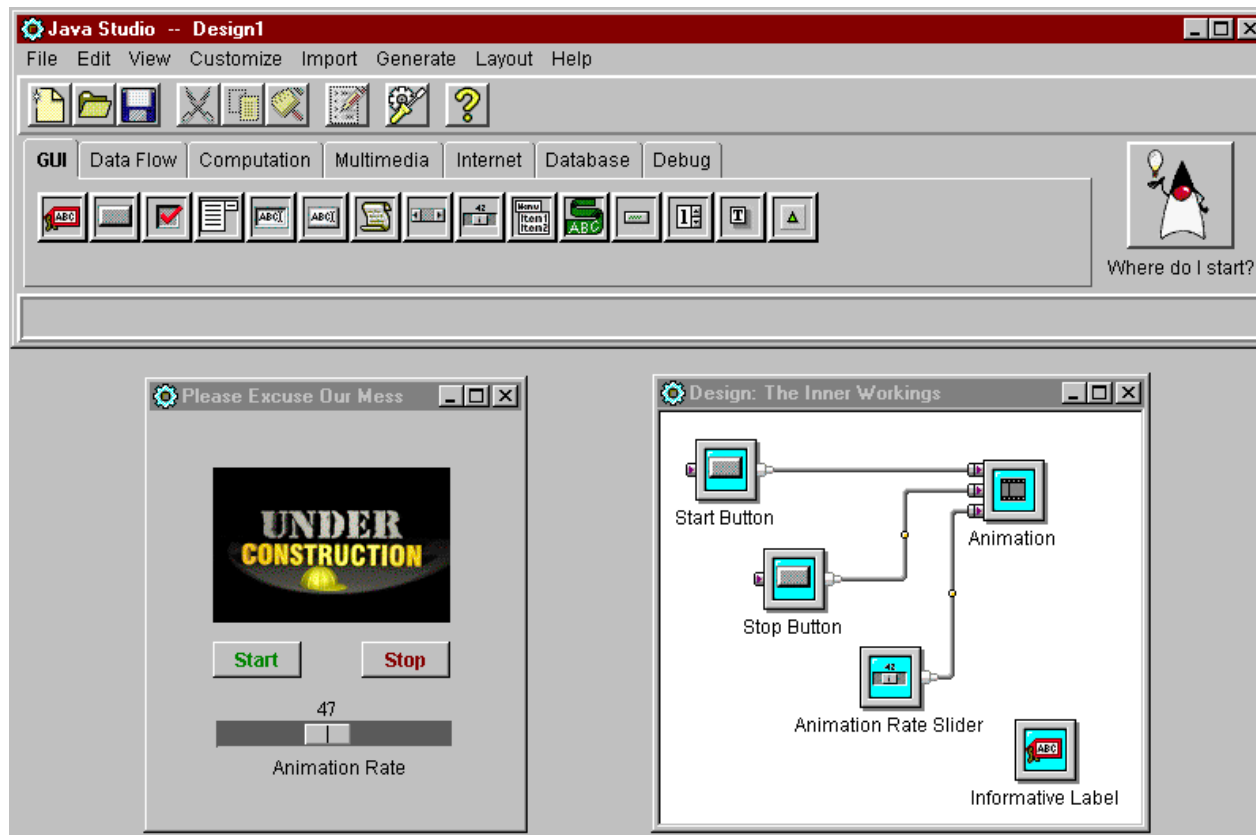
AOP: Opening a New Dimension in Software Development



Inhalt

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- **Ausblick auf Komponenteninfrastrukturen: CORBA**
- Zusammenfassung
- Literaturverzeichnis

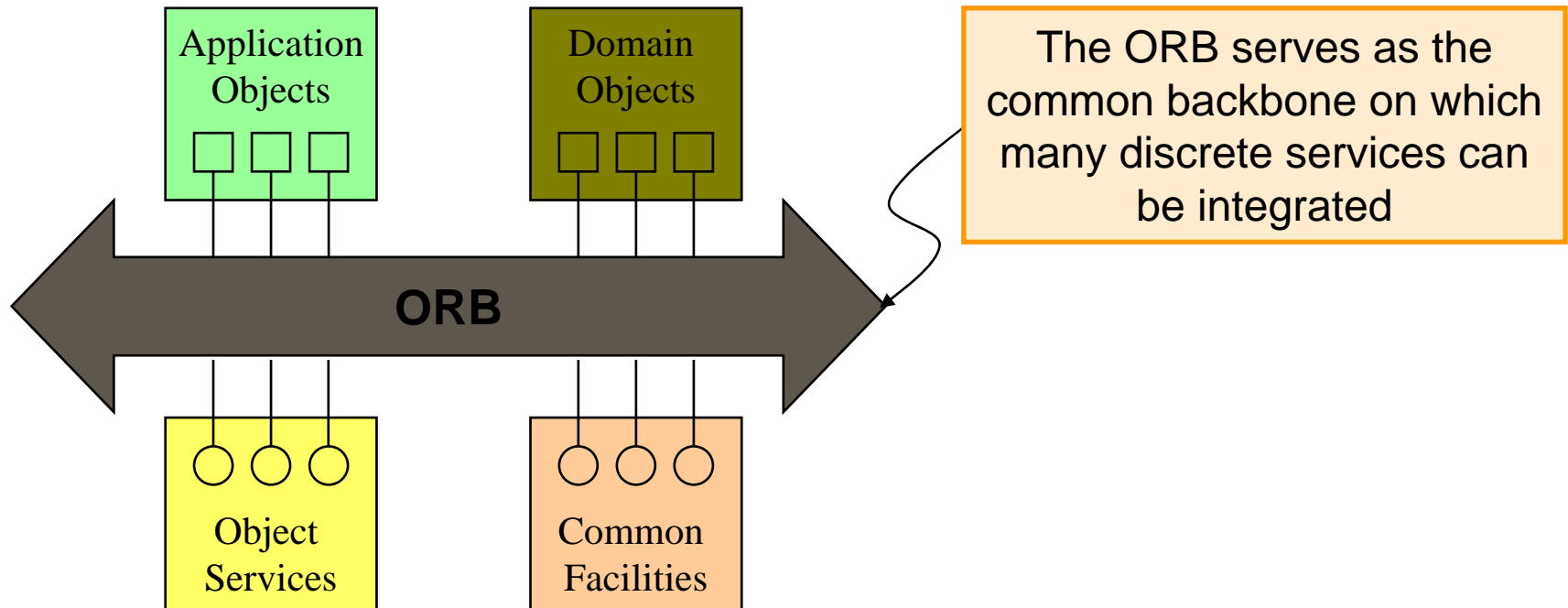
Plug-and-Play-Entwicklung



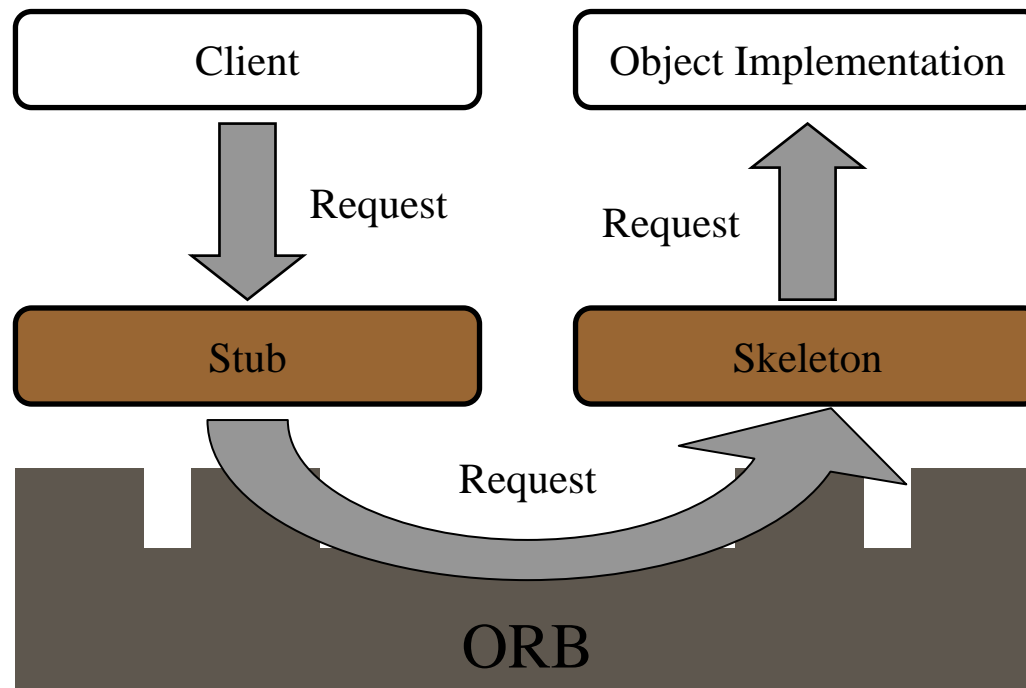
Beispiellösung für die Verteilung: CORBA

- Common Object Request Broker Architecture
- Vorteile:
 - Offener Standard für APIs und Protokolle, veröffentlicht durch die Object Management Group
 - Verfügbar für UNIX, Windows NT, Windows95 und OS/390 (MVS)
 - Vielfältige ausgereifte Produkte mehrerer Anbieter am Markt (z.T. mit integrierter Transaktionssicherung)
 - Hohe Sicherheit
 - Integration in Netzwerk Management Produkte (z.B. Tivoli TME10)
 - Inzwischen gute Integration mit anderen Standardprodukten

Ausflug: Object Management Architecture



Funktionsweise eines Object Request Brokers



Eigenschaften des Object Request Brokers

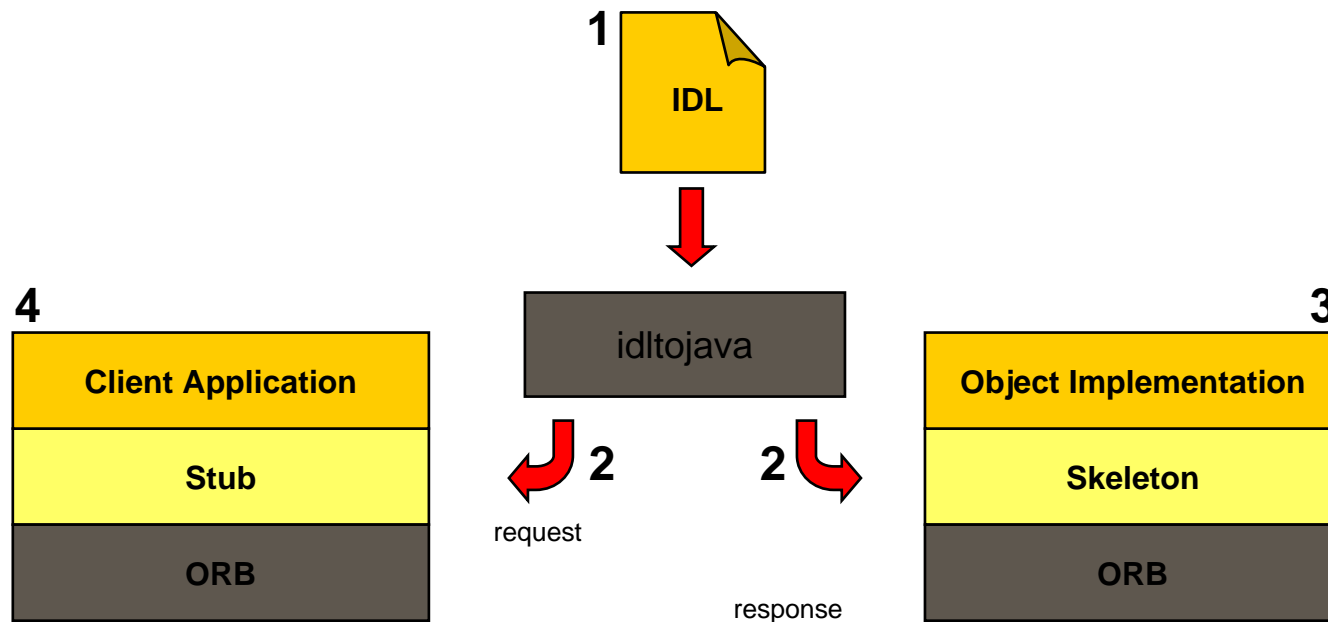
- Statische und dynamische Methodenaufrufe:
 - Methodenaufrufe statisch beim Kompilieren definieren oder dynamisch zur Laufzeit „entdecken“.
- Freie Wahl von Programmiersprachen:
 - Aufrufe durch Sprache Ihrer Wahl unabhängig von Implementationssprache des Servers durch Trennung von Schnittstelle und Implementierung.
 - Sprachenunabhängige Datentypen.
- Selbstbeschreibendes System:
 - Interface Repository als Verzeichnis der Schnittstellen, die von Servern angebotene Methoden und deren Parameter beschreiben (in IDL).
- Ortstransparenz:
 - Vermittelt Aufrufe zwischen Objekten innerhalb eines Prozesses oder Prozessen, die an verschiedenen Orten im Netz unter verschiedenen Betriebssystemen laufen vollkommen transparent.

MVC Beispiel: CORBA IDL

```
module MVC {  
  interface Observer {  
    oneway void update ();  
  };  
  
  interface Observable {  
    void register (in Observer obs);  
    void unregister (in Observer obs);  
  };  
  
  interface Controller : Observer {  
    void userInput (in String input);  
  };  
}
```

```
interface Model : Observable {  
  void stateChange (in String newState);  
};  
  
interface View : Observer {  
  void userOutput ();  
};  
};
```

MVC Beispiel: CORBA IDL und Java



steps:

- 1 write the IDL file
- 2 compile with idltojava (stubs/skeleton generated automatically)
- 3 write object implementation (servant)
- 4 write client application

CORBA Services (1)

- Der Naming Service bildet von Menschen vergebene Namen auf Objektreferenzen ab.
- Der Life Cycle Service stellt Operationen zur Erzeugung, zum Kopieren, Verschieben und Löschen von Objekten bereit.
- Der Event Service ermöglicht es Objekten, ihr Interesse an bestimmten Ereignissen dynamisch registrieren zu lassen.
- Der Trader Service stellt „Gelbe Seiten“ für Objekte zur Verfügung. Er erlaubt Objekten, ihre Dienste zu veröffentlichen.
- Der Transaction Service ermöglicht es verschiedenen verteilten Objekten, an einer atomaren Transaktion teilzunehmen. Er unterstützt auch verschachtelte Transaktionen.
- Der Concurrency Control Service ermöglicht Objekten den Zugriff auf gemeinschaftlich genutzte Ressourcen mit Hilfe von Sperren (Locks).
- Der Security Service stellt einen ganzen Satz von Sicherheitsmechanismen bereit. Er unterstützt z. B. Authentifizierung und Zugriffskontrolle.

CORBA Services (2)

- Der Persistent Object Service ermöglicht es Objekten, auch dann dauerhaft zu existieren, wenn die Applikation, die das Objekt erzeugt hat, beendet wurde.
- Der Externalization Service definiert Schnittstellen, um Objekte in einen Datenstrom umzuwandeln und sie aus einem Datenstrom zurückzuholen.
- Der Query Service definiert Operationen für Objektanfragen mit SQL3 und OQL ähnlichen Sprachen.
- Der Collection Service ermöglicht es, Objekte in Gruppen zu manipulieren. (Arrays, Bäume, Stacks, ...)
- Der Relationship Service ermöglicht die Definition von Beziehungen zwischen Objekten.
- Der Time Service stellt z. B. Interfaces zur Verfügung, um sich in einer verteilten Umgebung miteinander zu synchronisieren oder zeitabhängige Ereignisse zu definieren.
- Der Licensing Service lässt Sie z. B. die Nutzungsdauer von Komponenten messen und entsprechend abrechnen.
- Der Property Service erlaubt es, einem Objekt dynamisch eine Eigenschaft zuzuweisen (z. B. einen Titel oder ein Datum).

Seit CORBA 3.0

- Java und Internet Integration:
 - Ein Java-zu-IDL-Mapping für die automatische Generation von IDL stubs und skeleton aus Java-Objekten.
 - Firewall-Spezifikation für Aufrufe von CORBA-Objekten durch Firewalls.
- QoS-Kontrolle:
 - Erlaubt Klienten und Objekten verschiedene QoS-Parameter zu kontrollieren: Prioritäten, Deadlines, Time-To-Live.
- CORBAcomponents:
 - Umgebung definierbar, in der alle Komponenten persistent, transaktional und sicher sind.
- Asynchronous and Messaging Invocation (AMI): Diese Spezifikation definiert eine Anzahl von asynchronen Aufruf-Modi für CORBA (z. B. Warteschlangen)
- Minimum, Fault-Tolerant and Real-Time CORBA.
- Objekte, die by value übergeben werden können.

Inhalt

- Wiederverwendung
 - Motivation
 - Erfolgsfaktoren
 - Wiederverwendung von Softwarearchitekturen
- Frameworks
 - Objektorientierte Frameworks
 - Komponentenframeworks
- Aspect-oriented Programming
- Ausblick auf Komponenteninfrastrukturen: CORBA
- Zusammenfassung
- Literaturverzeichnis

Zusammenfassung

- Die pragmatische Wiederverwendung von Softwarearchitekturen in Form von Design, Implementierung und Erfahrung ist essentielle Voraussetzung für die effektive Softwareentwicklung
- Frameworks stellen eine „halbfertige Anwendung“ dar und ermöglichen die einfache Wiederverwendung einer Architektur
- Bislang konnten sich nur wenige objektorientierte Frameworks etablieren, teilweise aufgrund des nicht unerheblichen Lernaufwands (siehe IBM's San Francisco)
- Komponentenframeworks nutzen moderne Konzepte und sind die „besseren“ Frameworks
- Aspect-oriented Programming ermöglicht die Wiederverwendung (meist) orthogonaler Aspekte. Selten jedoch finden sich Anwendungsmöglichkeiten, die über Logging und Transaktionsmanagement hinausgehen.

Literaturhinweise

- [HNS99] Christine Hofmeister, Robert Nord, Dilip Soni: *Applied Software Architecture*, Addison Wesley – Object Technology Series, 1999.
- [DW98] Desmond Francis D'Souza, Alan Cameron Wills: *Objects, Components, and Frameworks With UML: The Catalysis Approach*, Addison Wesley Publishing Company, 1998
- [KHH+94] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, William G. Griswold: *An Overview of AspectJ, ECOOP 2001, 2002*
- [Pre97] Wolfgang Pree: *Komponentenbasierte Softwareentwicklung mit Frameworks*, dpunkt Verlag, 1997.