

Quasar Client Architektur

Hochschule für Angewandte Wissenschaften

Hamburg, 20.05.2011



GEMEINSAM. ENERGIEN FREISETZEN.



Capgemini – der weltweit größte Dienstleister für Consulting, Technology und Outsourcing mit europäischem Ursprung

Geografische Verteilung der Capgemini-Mitarbeiter¹



Zusammenarbeit

Zusammenarbeit ist das zentrale Element der Capgemini-Philosophie und gleichzeitig die Säule unseres Leistungsangebots. Sie trägt zur Wertschöpfung bei, begrenzt Risiken, optimiert Kapazitäten und richtet die Organisation auf Zielerreichung aus. Wir nennen dies Collaborative Business Experience™.

Servicebereiche

- Consulting Services (Capgemini Consulting)
- Technology Services
- Outsourcing Services
- Local Professional Services (Sogeti)

Globale Sektoren

- Energy & Utilities, Chemicals
- Financial Services
- Manufacturing, Retail & Distribution
- Public Sector
- Telecom, Media & Entertainment

Fakten¹

- 8,371 Mrd. EUR Umsatz
- 90.516 Mitarbeiter weltweit, davon 36.236 im globalen Liefernetzwerk Near- und Offshore
- Mehr als 300 Standorte in mehr als 30 Ländern
- Börsennotiert in Paris als *Cap Gemini S.A.*

1 Stand: 31. Dezember 2009

Mit unseren Services liefern wir unseren Kunden maßgeschneiderte Lösungen – nachhaltig, zuverlässig und kostenoptimiert

**Consulting Services
(Capgemini Consulting)**

- Strategy and Transformation Consulting
- Technology Transformation
- Operations Transformation

Technology Services

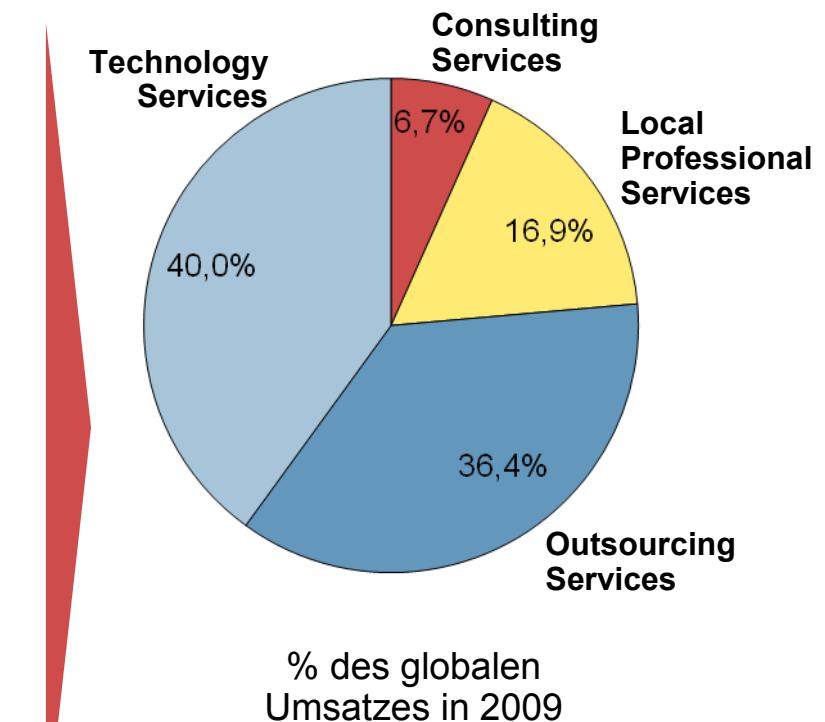
- Business Technology
- Process Consulting & Package Based Solutions
- Custom Solution Development
- Application Management
- Business Information Management

Outsourcing Services

- Business Process Outsourcing
- Infrastructure Outsourcing
- Governance, Service Desk & Management

**Local Professional Services
(Sogeti)**

- Software Quality Management & Testing Services
- Infrastructure Services



Technology Services steht für leistungsfähige Prozess- und Softwarelösungen, die die Wettbewerbsfähigkeit unserer Kunden erhöhen

Kunden

Unsere Kunden sind namhafte Unternehmen aller Branchen sowie öffentliche Institutionen, deren Erfolg von anspruchsvollen Prozess- und Softwarelösungen abhängt.

Ihr Nutzen besteht in erhöhter Wettbewerbsfähigkeit durch

- Differenzierung bei unternehmenskritischen Lösungen
- Effizienzverbesserung bestehender Lösungen

Leistungsangebot

- Business Technology
- Process Consulting & Package Based Solutions
- Custom Solution Development
- Application Management
- Business Information Management

Kompetenzen

- Umsetzungsorientierte Beratung
- Management komplexer IT-Projekte
- Software-Engineering
 - Gestaltung anspruchsvoller IT-Architekturen
 - Implementierung von Standardsoftwarepaketen (insbesondere SAP®- und Oracle-Pakete)
 - Rightshore®
 - Forschung & Innovation
- Collaborative Business Experience™

Rightshore® ist eine eingetragene Marke von Capgemini

Standorte in Deutschland, Österreich, Schweiz



● Rightshore®-Standorte für die Region Deutschland, Österreich und Schweiz

Branchen

- Automotive
- Energy & Utilities, Chemicals
- Financial Services
- Manufacturing, Retail & Distribution
- Public Sector
- Telecom, Media & Entertainment

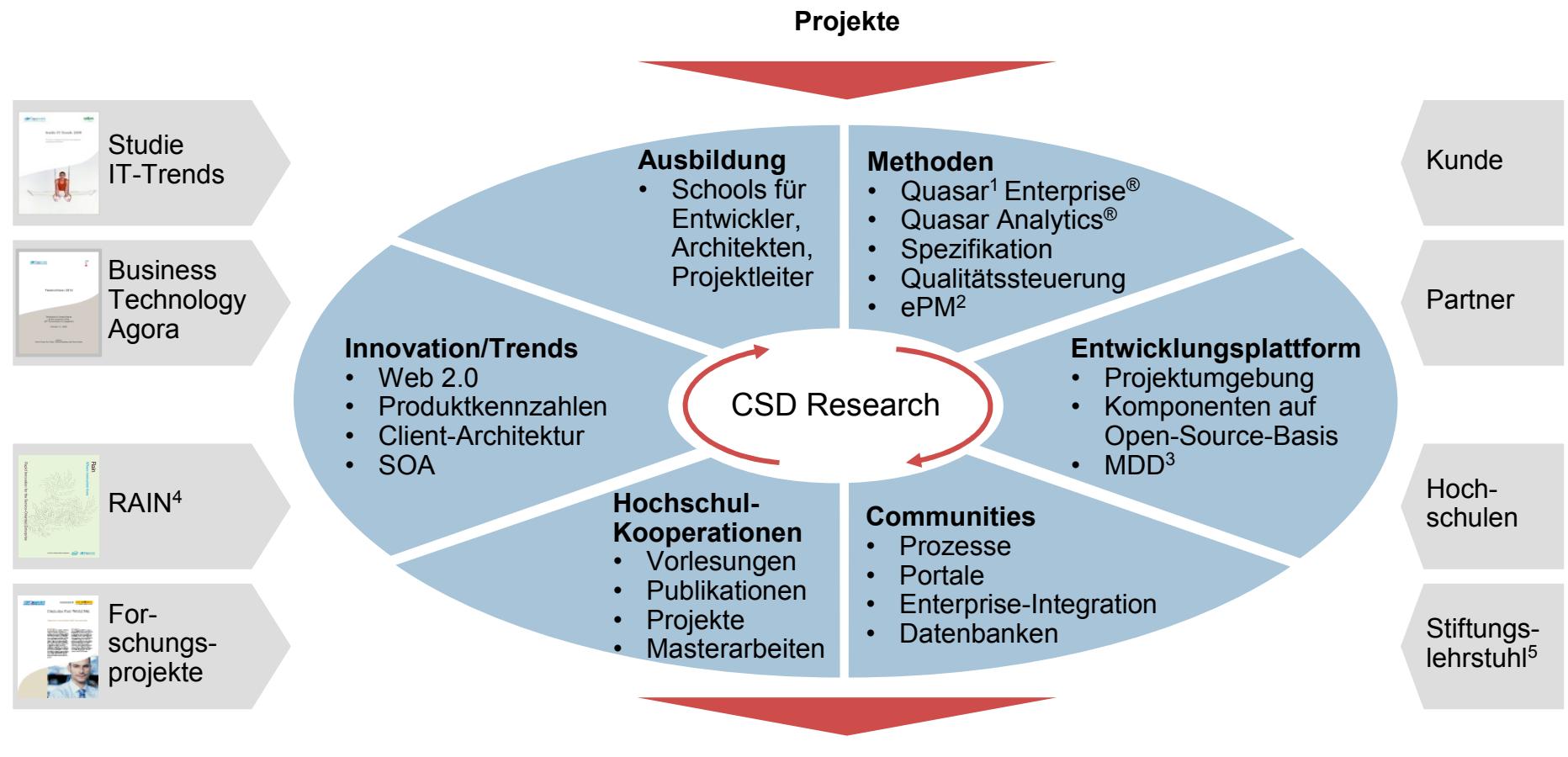
Capgemini kombiniert seine hohe fachliche Kompetenz mit Branchen-Know-how

Ausgewählte Referenzkunden



CSD Research entwickelt unsere Software-Engineering-Kompetenz stetig weiter und stellt Projekten aufbereitetes Wissen bereit

Wir gestalten die IT der Zukunft mithilfe von Innovationen und unserer systematischen Vorgehensweise



1 Quasar = Qualitäts-Software-Architektur

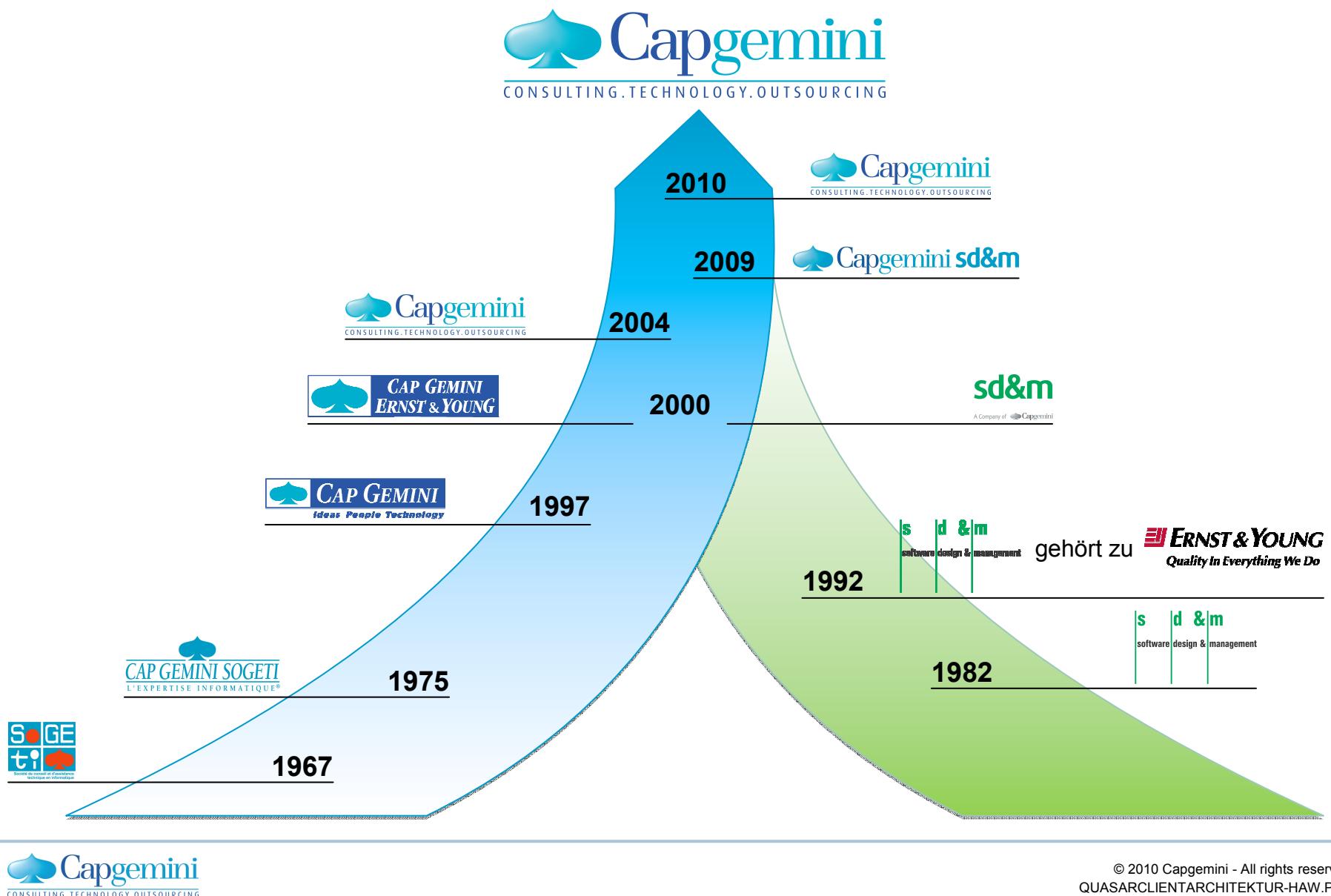
2 ePM = Effizientes und effektives Projektmanagement

3 MDD = Model Driven Development

4 RAIN = Rapid Innovation - globale, serviceorientierte Umgebung für die Auseinandersetzung mit der Zukunft

5 Stiftungslehrstuhl Global Software Engineering an der TU München am Lehrstuhl Informatik

Der Weg von sd&m in die Capgemini Gruppe



AGENDA

- Motivation: Client-Anforderungen
- Quasar Client Architektur
- Quasar Client in der Praxis

AGENDA

- Motivation: Client-Anforderungen
- Quasar Client Architektur
- Quasar Client in der Praxis

Motivation

Moderne Client Anwendungen zeichnen durch eine hohe Komplexität aus

Herausforderung

**Hohe Anzahl und
vielfältige Strukturierung
der Anzeige-Elemente**

Hohe Interaktivität

Details

- komplexe Fachlichkeit erfordert zahlreiche Bedienelemente
- Strukturierung der Elemente durch stabiles und ergonomisches Layout
- Intuitive Steuerung erfordert “Vernetzung” der Bedienelemente
- Fachlichkeit gibt komplexe Datenstrukturen kombiniert mit unterschiedlichsten Statusübergängen vor



Beispiel aus einem Projekt:

The screenshot displays a car configuration application window titled "MBKS-PKW II Version VKLII 126 - Neues Geschäft 1". The main menu bar includes "Geschäft", "Verkaufsinformationen", "Verwaltung", "Formulare", "Fenster", and "?". The toolbar contains icons for file operations, search, and help.

The top navigation bar shows tabs: "Fahrzeug" (selected), "Kunde", "Angebot", and "Bestellung".

The left sidebar has sections for "Konfigurator", "Bestandsuche", and "Modifikation". It features a preview image of a silver CLS 500 Coupé and its price: 70.122,00€.

The central part of the screen is a "Lack" (Paint) configuration section. A table lists paint options with their codes, descriptions, prices, and lease costs:

Code	Beschreibung	Preis (€)	Leas...
000	Lack wird später festgel...	0,00	
040	Schwarz	0,00	
960	Alabasterweiß	0,00	
189	Smaragdschwarz metallic	997,60	
197	Obsidianblack metallic	997,60	
359	Tansanitblau metallic	997,60	
544	Carneolrot metallic	997,60	
600	Wasserfallrosa metallic	997,60	

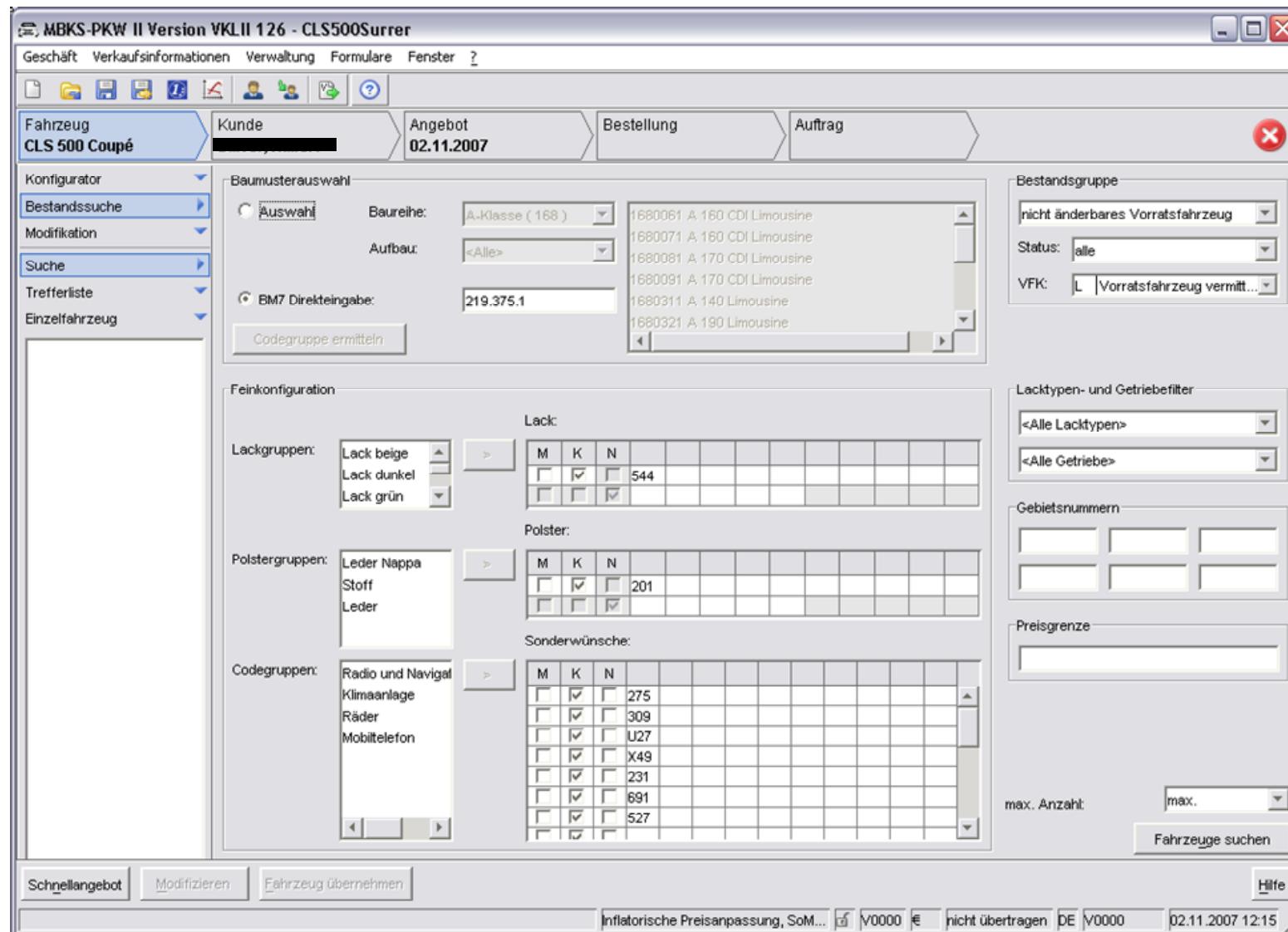
The selected option is "Carneolrot metallic" (997,60€). Below this, there is a "Direkteingabe" (Direct input) field with a color swatch and a preview image of the car's exterior and interior in red.

At the bottom of the main window, there are buttons for "Übernehmen" (Accept), "Drucken" (Print), and "Abbrechen" (Cancel). The status bar at the bottom right shows "Inflatorische Preisanpassung, SoM... 10/000 € nicht übertragen DE 10/000 02.11.2007 12:03".

Beispiel aus einem Projekt:

The image displays two windows side-by-side. On the left is the 'MBKS-PKW II Version VKLII 126 - CLS500Surfer' application window. It shows a navigation bar with 'Geschäft', 'Verkaufsinformationen', 'Verwaltung', 'Formulare', 'Fenster', etc. Below is a form for a 'CLS 500 Coupé'. The 'Kunde' field is filled with a redacted name, and the 'Angebot' date is set to '02.11.2007'. Under 'GFZ-Erhalt', it says '2 - Ersatz eines Fremdfabrikats'. Under 'Gebrauchtfahrzeugdaten', there's a checked checkbox for 'Gebrauchtfahrzeugdaten'. Other fields include 'GFZ-Vereinbarung: B - Festpreis' and 'GFZ-Übergabe: 1 - Bei Lieferung Neufahrzeug'. A section for 'Angaben zum Vorbesitz' lists 'Vorauswahl Fabrikat: DMW', 'Typ: 177W', '7er Limousine', and other details like 'Kennzeichen:' and 'Festpreis netto:'. At the bottom, there are buttons for 'WinLeas Drucken' and 'Angebot drucken'. A status bar at the bottom shows a warning message about GFZ-Übergabe and a timestamp '02.11.2007 12:12'. On the right is a 'Angebotsmappe - Druckvorschau' window titled 'Persönliches Angebot für Herrn'. It features the Mercedes-Benz logo, a red CLS 500 Coupé, and the text 'Mercedes-Benz CLS 500 Coupé'. Below the car is a placeholder for 'Überreicht durch:'. The bottom of the window has buttons for 'Drucken...', 'Alle Drucken...', and 'Schließen'.

Beispiel aus einem Projekt:



Typische Client-Anforderungen (1/4)

Clients von BIS liegen vielfältige und komplexe Anforderungen zugrunde

Personalisierung

- **Bedienmodi**, z.B. Anfängermodus und Expertenmodus
- **Layoutanpassungen**, z.B. Größe des Fensters, Größe der Bereiche innerhalb des Fensters, Schriftgrößen, Farben...
- Konfiguration von **Toolbar** und **Menü** („Favoriten“)
- Definition von **Tastaturkürzeln**
- Benutzerabhängige **Vorbelegung** von editierbaren Feldern
- Änderbares „**Look & Feel**“ der Anwendung

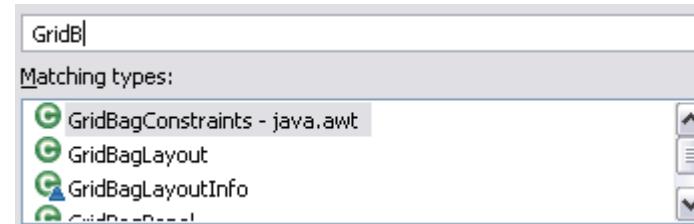
Typische Client-Anforderungen (2/4)

Clients von BIS liegen vielfältige und komplexe Anforderungen zugrunde

Komplexe Steuerelemente

- Auswahllisten
- Editierbare Tabellen
- Editierbare Bäume
- Baum-Tabellen
- Date Picker
- „Narrow down“ Widgets
- ...

TreeTable			
Name	Size	Type	Modified
etc	0	Directory	30-Aug-98
export	0	Directory	08-May-98
home	0	Directory	19-Aug-98
milne	0	Directory	31-Aug-98
.Bugtraqrc	1,911	File	24-Jul-98
.Xauthority	203	File	14-Aug-98
.ctables	0	Directory	10-Aug-98
.cshrc	420	File	19-Aug-98
cshrc	294	File	08-Aug-98



Typische Client-Anforderungen (3/4)

Clients von BIS liegen vielfältige und komplexe Anforderungen zugrunde

Validierungen

- Syntaktisch vs. semantisch
- Auslöser: Tastendruck, Fokuswechsel, Speichern...
- Markierung fehlerhafter Felder
- Unterscheidung: Fehler vs. Warnung
- Markierung von Reitern
- Anzeige der Fehlermeldung
- „Jump-To-Error“

Last Name	<input type="text"/>
First Name	<input type="text"/> X
Age	<input type="text"/> ! 134

Typische Client-Anforderungen (4/4)

Clients von BIS liegen vielfältige und komplexe Anforderungen zugrunde

Anforderung	Details
Daten-Export	<ul style="list-style-type: none">• Export in Datei (CSV, PDF, XML...)• Export zu Nachbarsystem (z.B. Excel)• Drucken
Berechtigungen	<ul style="list-style-type: none">• Funktionale Berechtigungen• Datenabhängige Berechtigungen
Internationalisierung	<ul style="list-style-type: none">• Statische Inhalte• Dynamische Daten• Lokalisierung bei Auslieferung, beim Login, zur Laufzeit...
Weitere Anforderungen	<ul style="list-style-type: none">• Fokusmanagement• Portal-Integration• Hilfe-System• ...

Software-Ergonomie (1/2)

Client-Anforderungen müssen den Regeln der Software-Ergonomie genügen

- Die **Software-Ergonomie** beschäftigt sich mit der Gebrauchstauglichkeit, also der Benutzerfreundlichkeit von Computer-Programmen.
- Die **ISO 9241** definiert in Grundsätze für die Gestaltung und Bewertung einer Schnittstelle zwischen Benutzer und System:
 - Aufgabenangemessenheit
 - Selbstbeschreibungsfähigkeit
 - Steuerbarkeit
 - Erwartungskonformität
 - Fehlertoleranz
 - Individualisierbarkeit
 - Lernförderlichkeit

Software-Ergonomie (2/2)

Client-Anforderungen müssen den Regeln der Software-Ergonomie genügen

- **Kurzzeitgedächtnis:** 7 ± 2 Chunks (Menü, Auswahllisten...)
- **Gruppierung:** Abstand statt Rahmen
- **Schriftarten:** Fixed Font + Serifen für Dateneingabe
(z.B. „Illustrierte“ vs. „Illustrierte“)
- **Farben:** Dezent; Kontraste; Bedeutung (z.B. Rot = Warnung)
- **Beschriftung** von Aktions-Widgets muss sprechend sein
(z.B. „[Speichern] [Nicht speichern]“ vs. „[Ja] [Nein]“)
- **Erwartungskonformität** (z.B. Windows Look & Feel)
- **Kurze Antwortzeiten** (z.B. per Splash-Screen oder Status-Balken)
- **Intuitive Benutzbarkeit** („Don't make me think“)

Realisierungsaufwand (1/2)

Client-Aufwand wird oft unterschätzt. Beispiel aus Capgemini-Projekt.

- Beispiel:

Komplexität des Dialogs	Schätzung	Realer Aufwand
Leicht	0,25 BT	1,09 BT
Mittel	0,73 BT	2,19 BT
Schwer	2,48 BT	4,37 BT

- Bei **jedem** Dialog fallen mehrere Realisierungsaufgaben an:
 - *Dialogdaten* (Model)
 - *Dialogdarstellung* (View)
 - *Dialoglogik* (Controller)
 - *Test* (manuell + evtl. automatisiert)
- Bei großer Anzahl von Dialogen: **MDD** zur Aufwandsminimierung

Realisierungsaufwand (2/2)

Spezifikation kann den Aufwand stark beeinflussen

Beispiel

Status-Balken

- *Schwer:* Realer Fortschritt
- *Leicht:* Pseudo-Fortschritt

Cancellation

- *Schwer:* Genereller Abbrechen-Knopf für Server Aufrufe
- *Leicht:* Sonderbehandlung für lang andauernde Aufrufe

Bei der Spezifikation niemals die Machbarkeit außer Acht lassen

Typische Probleme bei der Client-Entwicklung (1/2)

Erfahrungen aus zahlreichen Projekten zeigen die Unterschätzung des Clients

Problem	Details
Unterschätzter Aufwand	<ul style="list-style-type: none">• Aufwand bei GUI-lastigen Systemen häufig $\geq 60\%$• Spezifikation ohne Berücksichtigung der Machbarkeit
Unterschätzte Bedeutung	<p>„Was zählt ist der Anwendungskern (AWK). Die Dialoge machen wir nebenbei.“</p>
Hoher Zeitdruck	<ul style="list-style-type: none">• Screenshots schon für Spezifikation benötigt• Große Anzahl Dialoge => Entwicklung parallel zu Basisfunktionen
Hohe Komplexität durch vielfältige Dimensionen	<ul style="list-style-type: none">• Berechtigung• Internationalisierung• Personalisierung• ...

Typische Probleme bei der Client-Entwicklung (2/2)

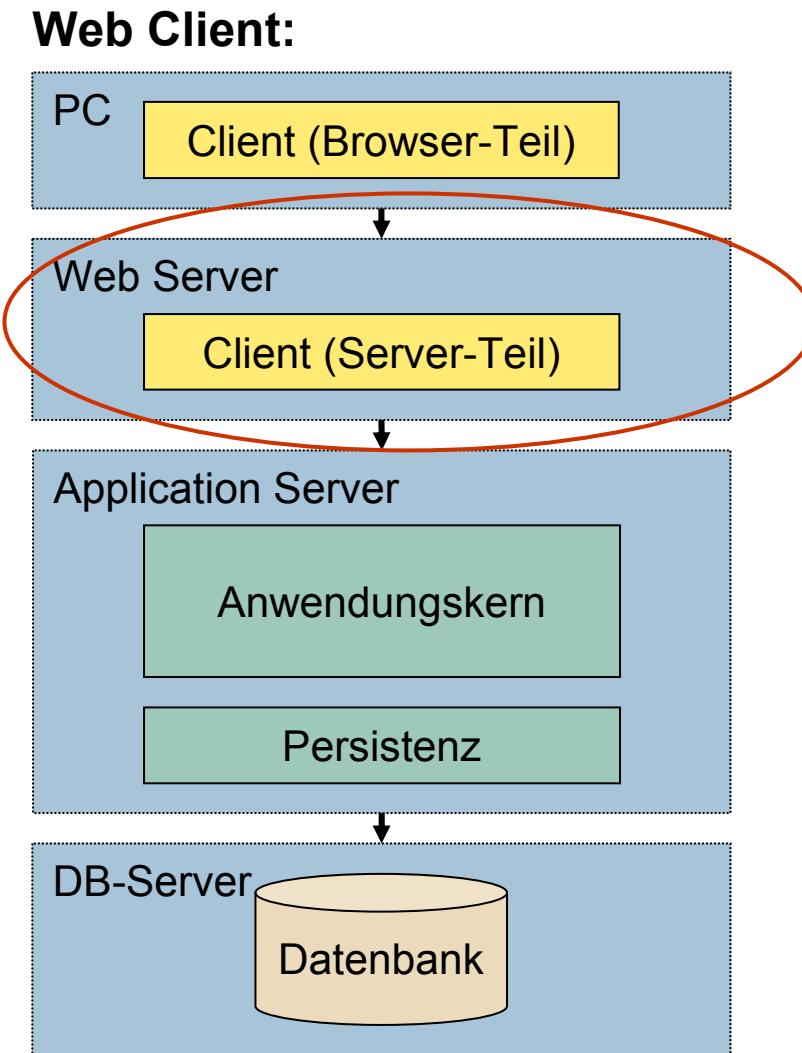
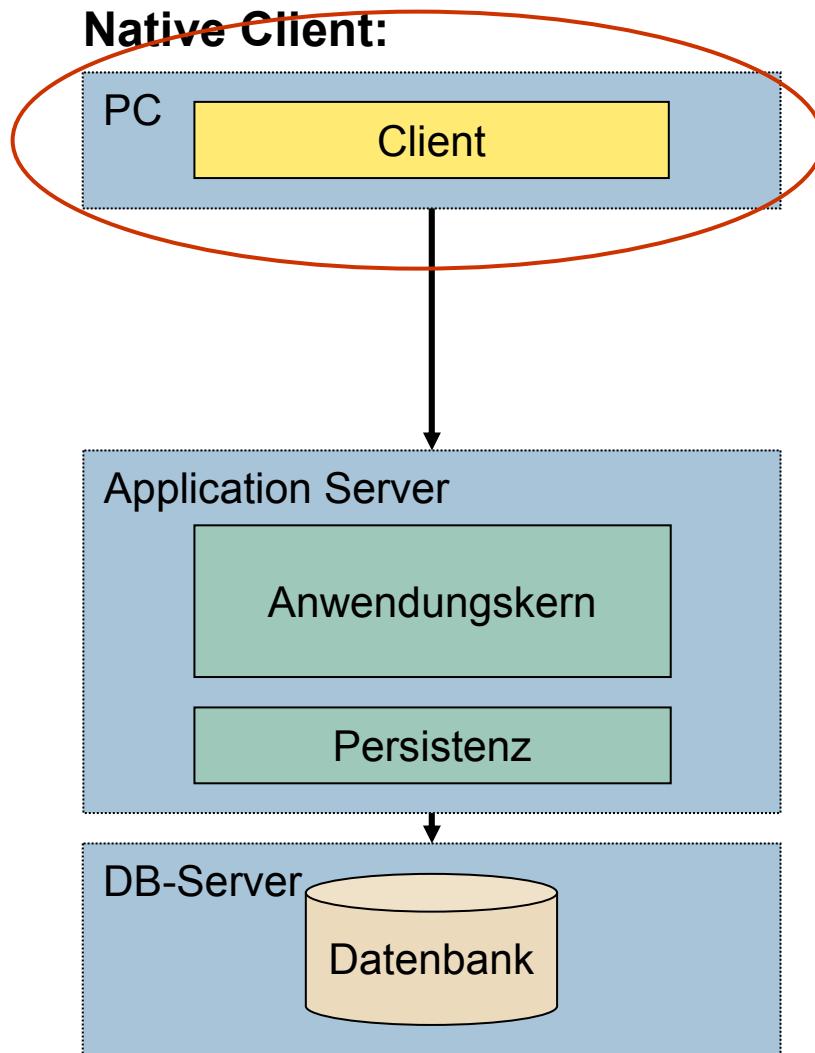
Erfahrungen aus zahlreichen Projekten zeigen die Unterschätzung des Clients

Problem	Details
Schlechte Spezifikation	<ul style="list-style-type: none">• Spezifikation ist nicht vollständig• GUI-Prototyp fehlt• Ergonomie-Regeln werden verletzt
Mangelndes Know-How	Standard Client-Probleme und -Lösungen sind nicht bekannt
Schlechte Testbarkeit	<ul style="list-style-type: none">• Aufwändige Erstellung der Tests (Capture-Replay)• Instabilität der Tests durch fehlenden Refactoring Support
Fehlende Client-Architektur	<ul style="list-style-type: none">• Technischer Chefdesigner für Client wird nicht benannt• Client-Aufgaben werden mehrfach implementiert

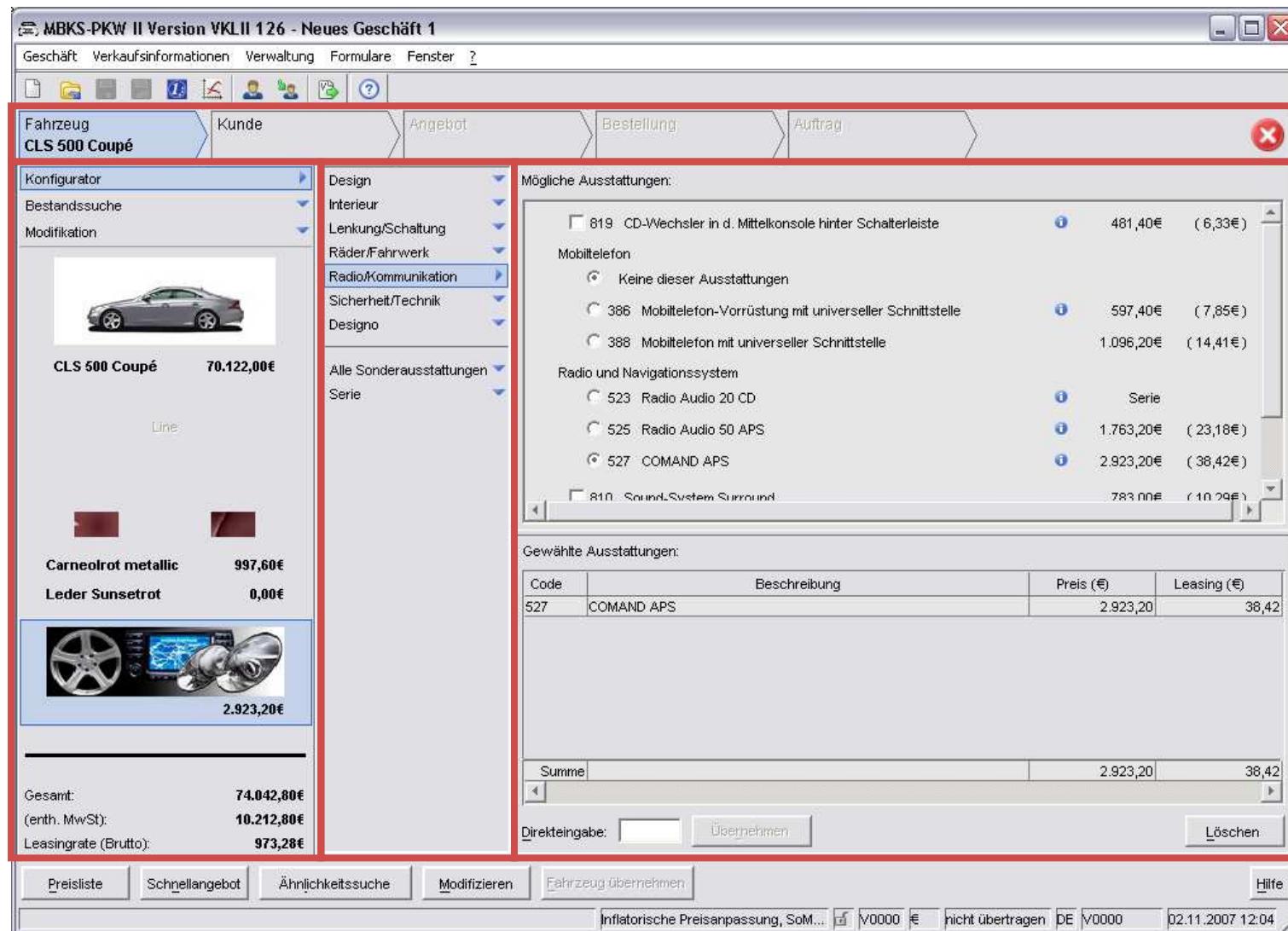
AGENDA

- Motivation: Client-Anforderungen
 - **Quasar Client Architektur**
 - Quasar Client in der Praxis

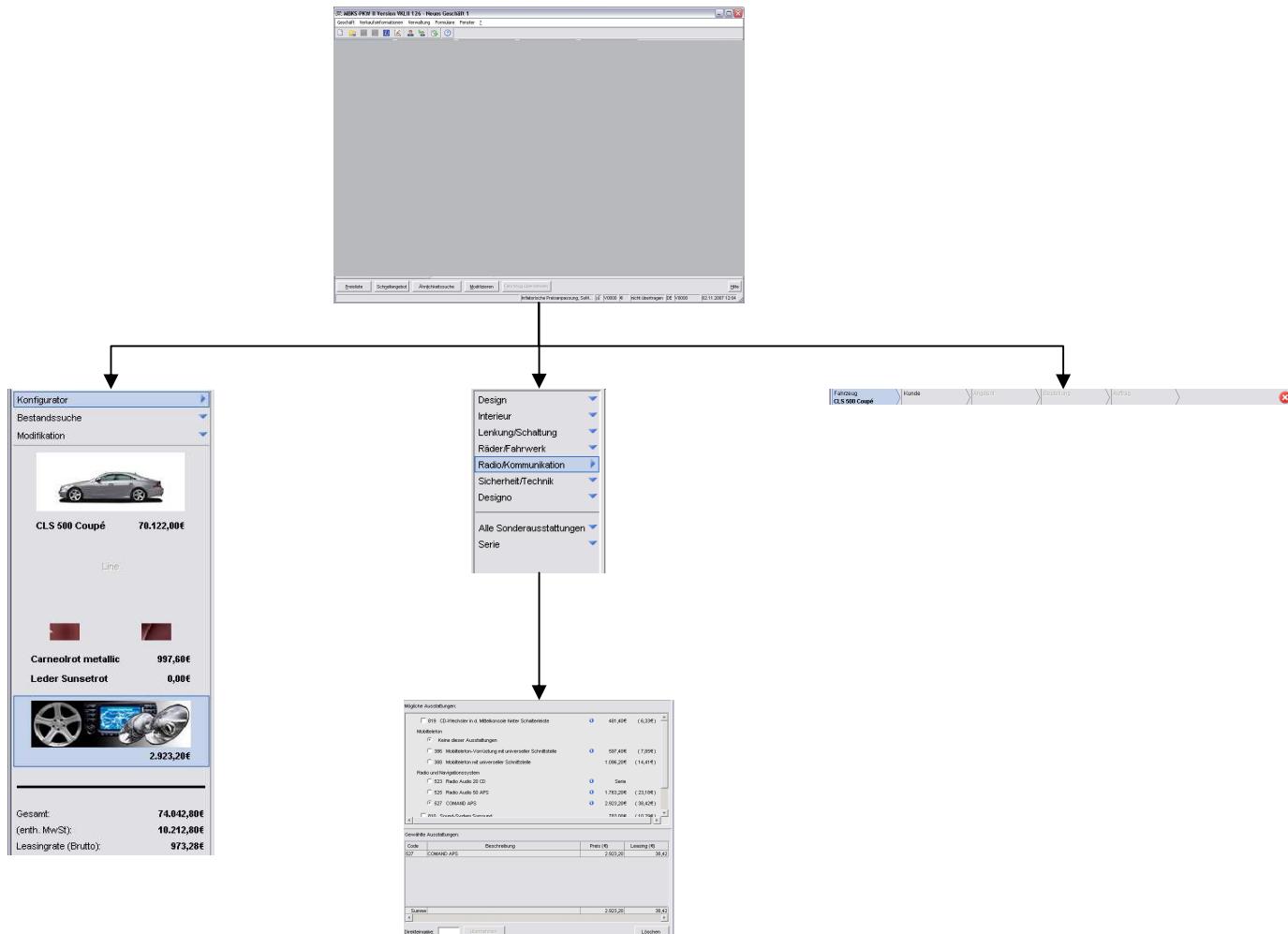
T-Architekturen typischer Capgemini-Anwendungen



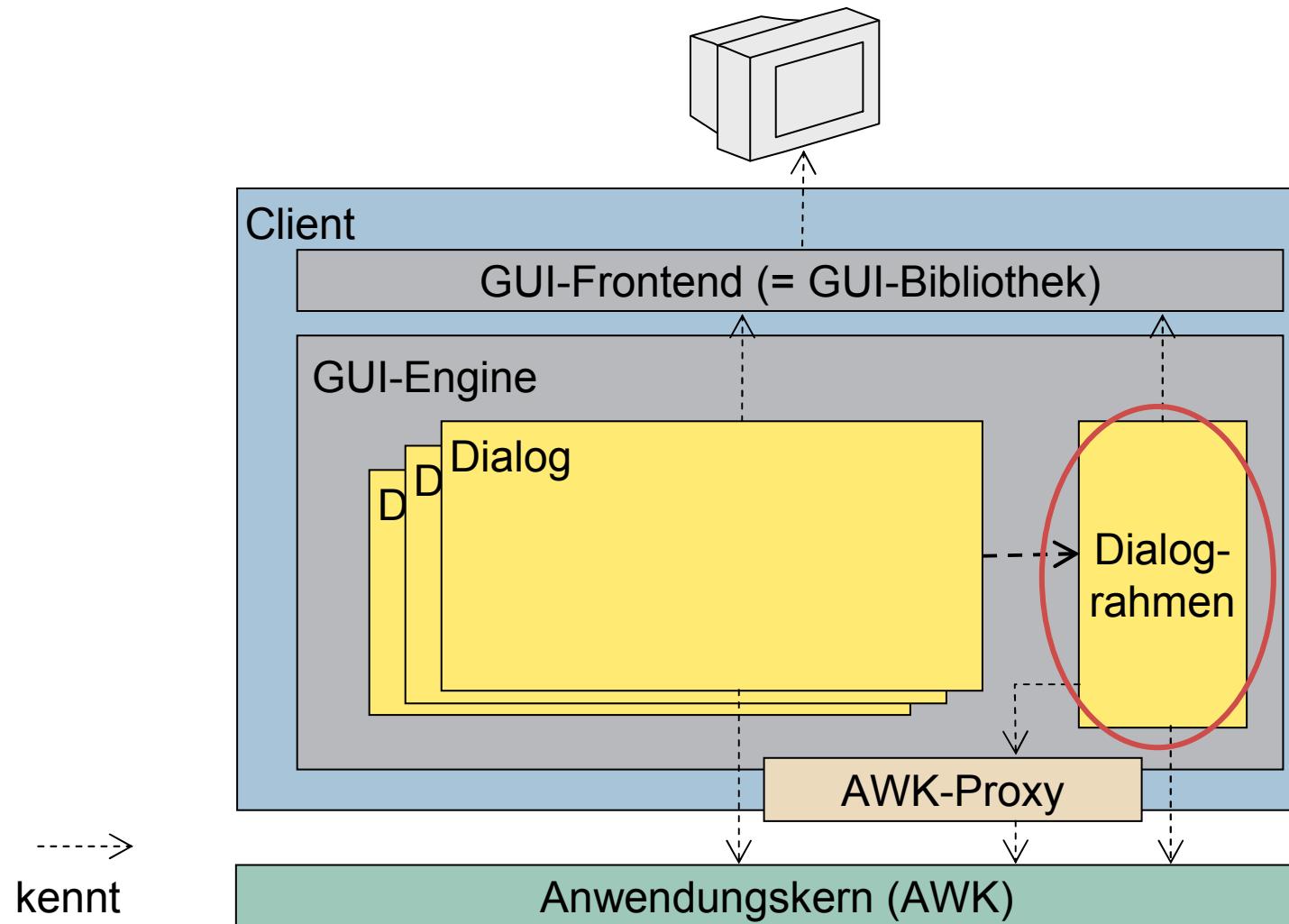
Ziel: Komponentenorientierung im Client ermöglichen



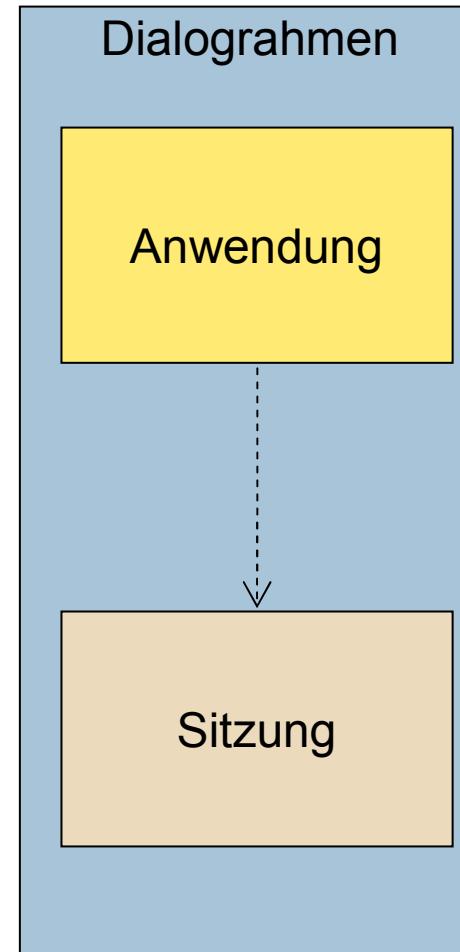
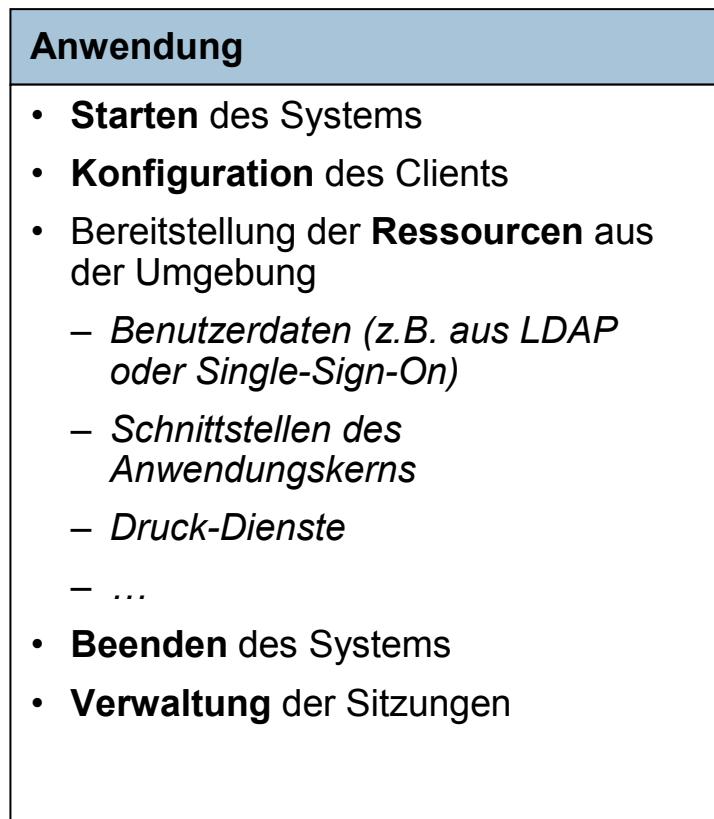
Dialog Komponenten werden gebildet und hierarchisch verbunden



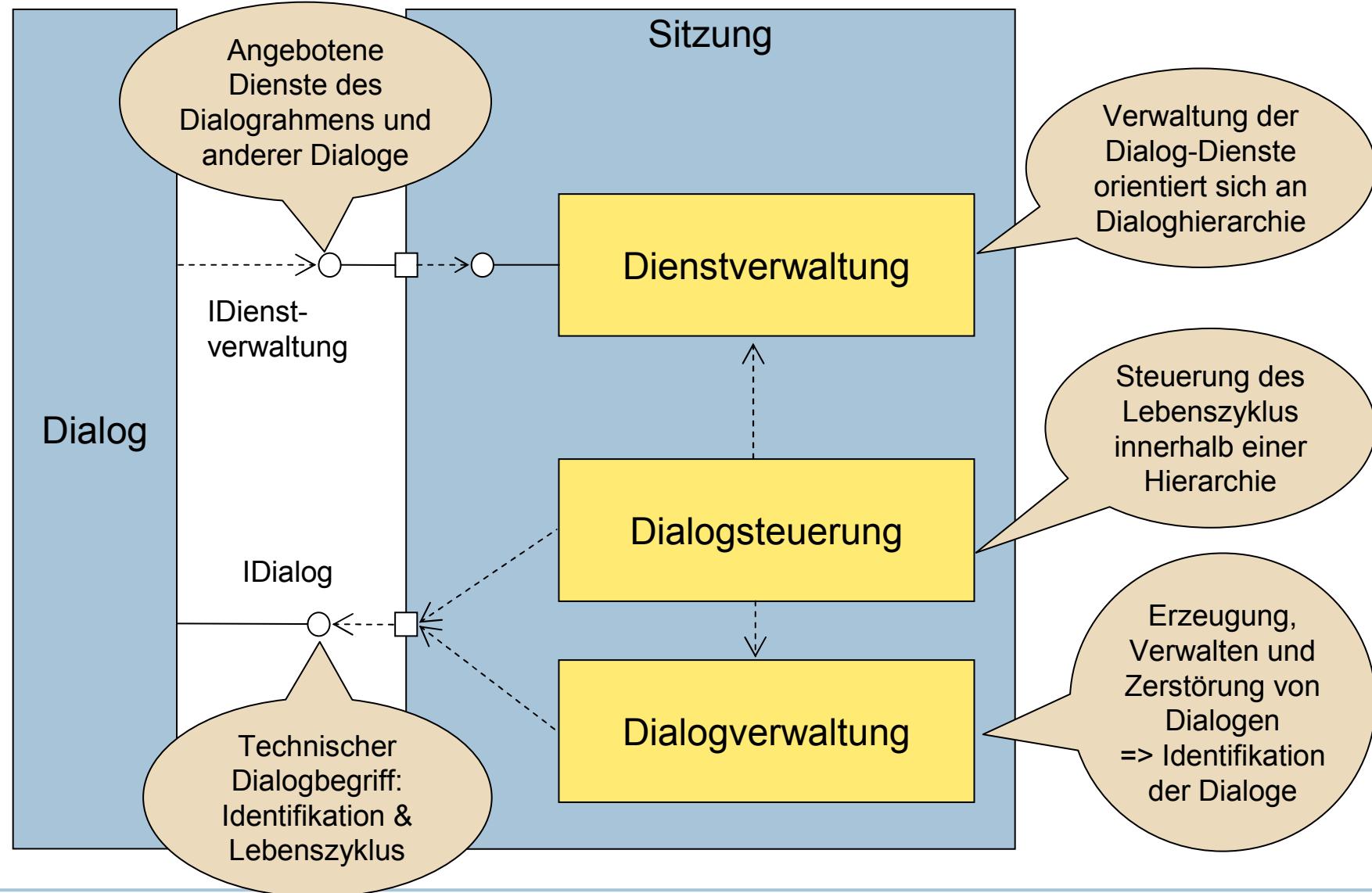
Client-Grob-Architektur



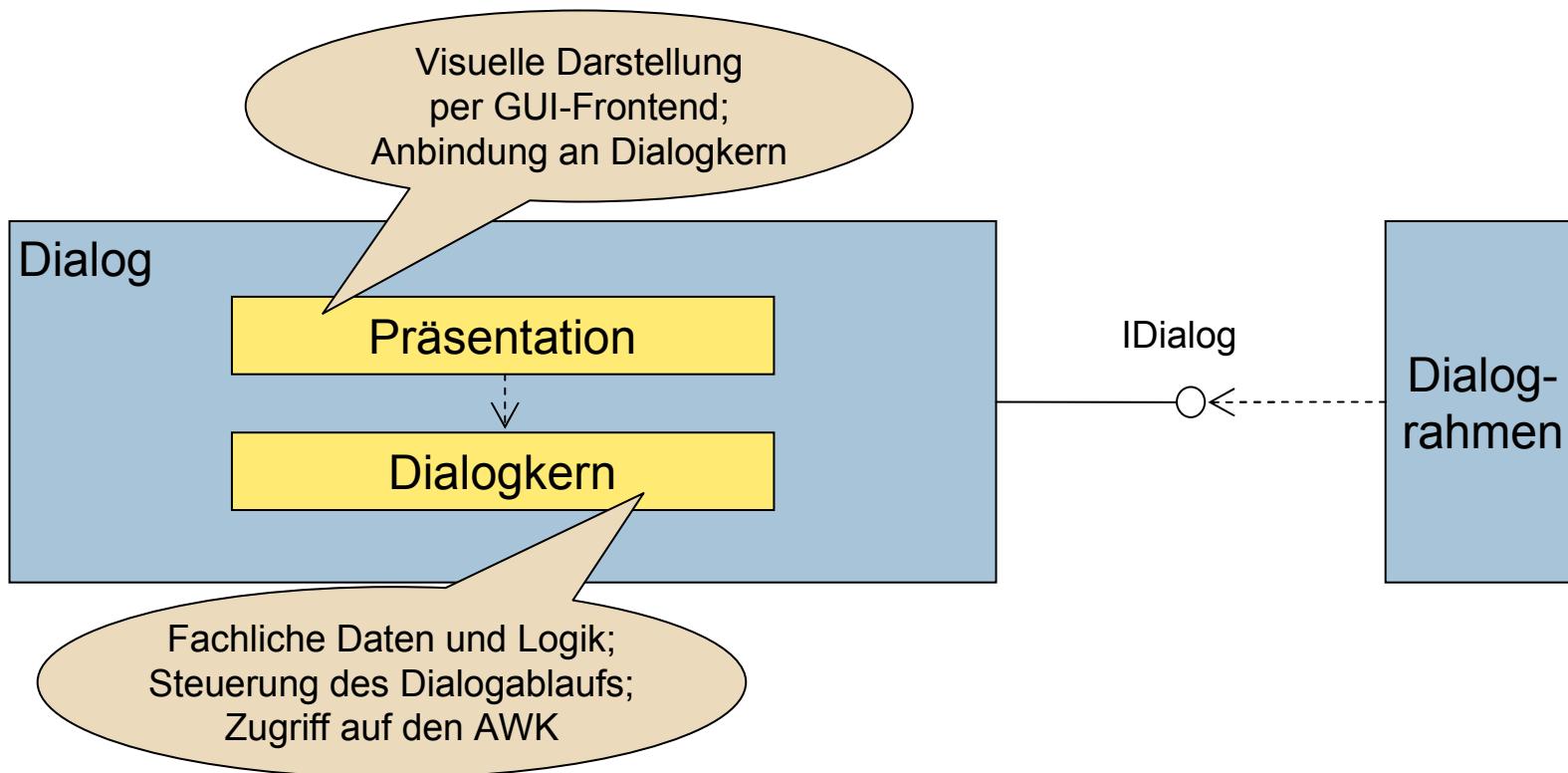
Dialograhmen: Anwendung und Sitzung



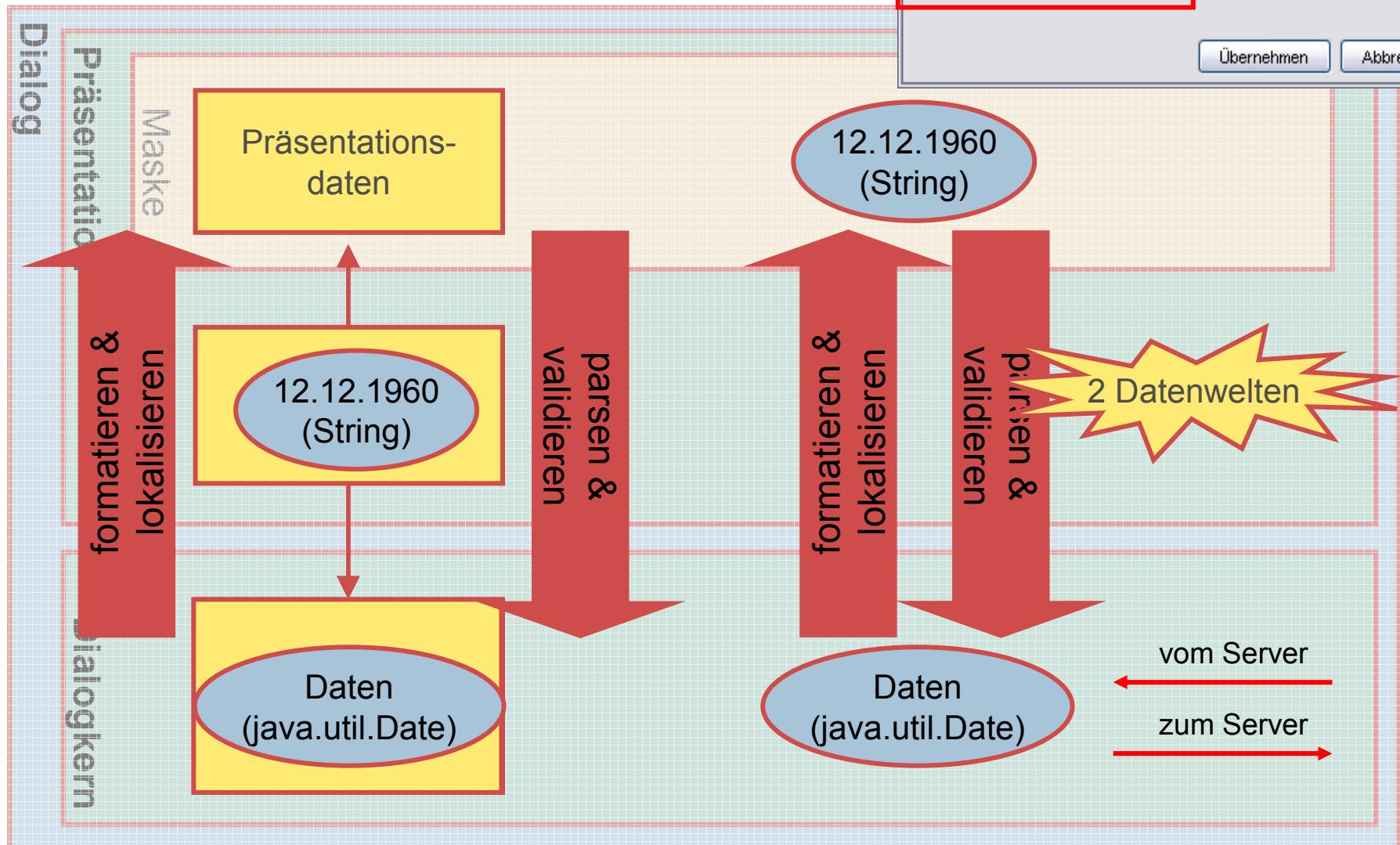
Sitzung: Definiert Dialog + Dienste



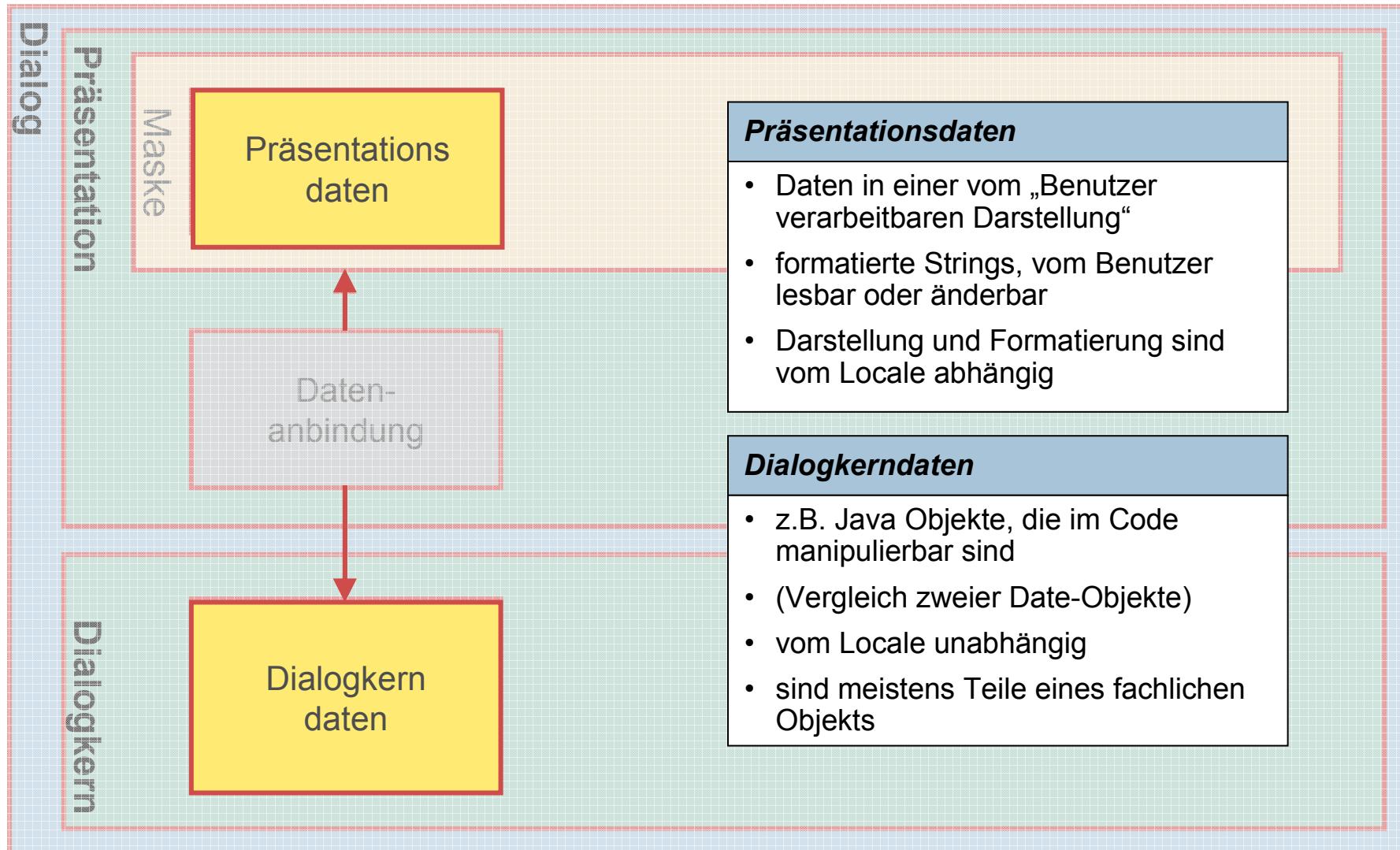
Dialog: Dialogkern und Präsentation



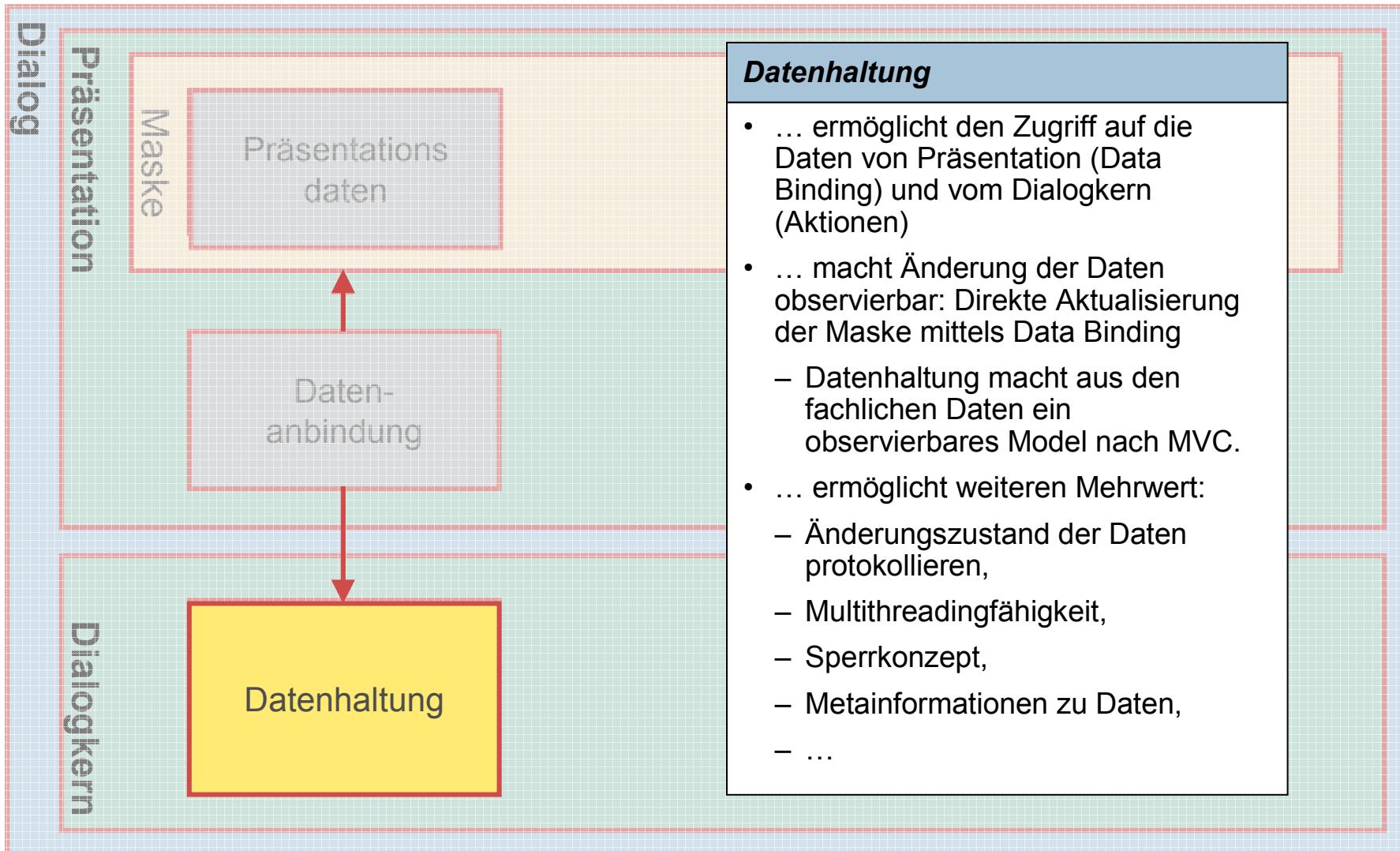
Es gibt immer zwei Datenwelten in einem Dialog



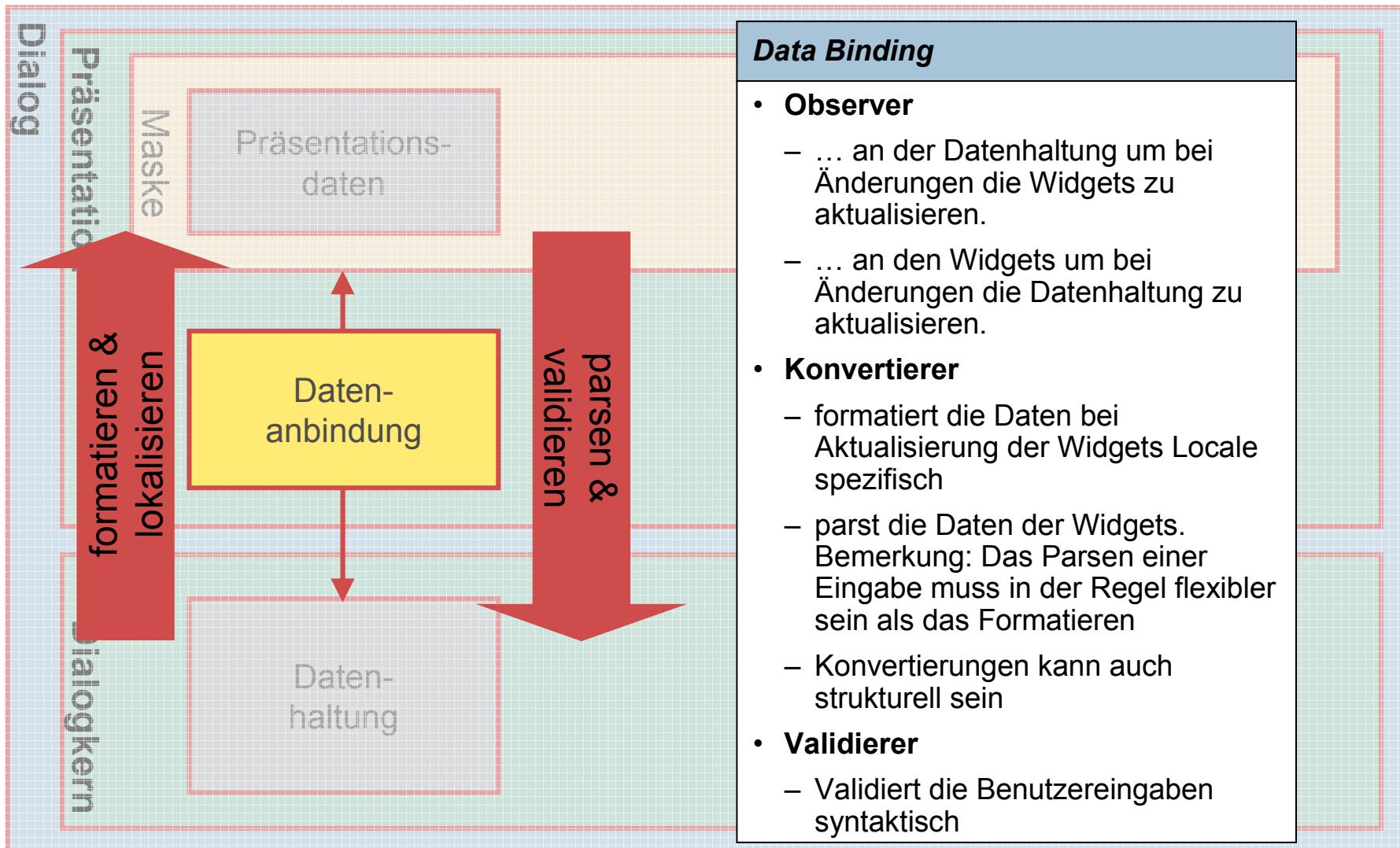
Es gibt immer zwei Datenwelten in einem Dialog



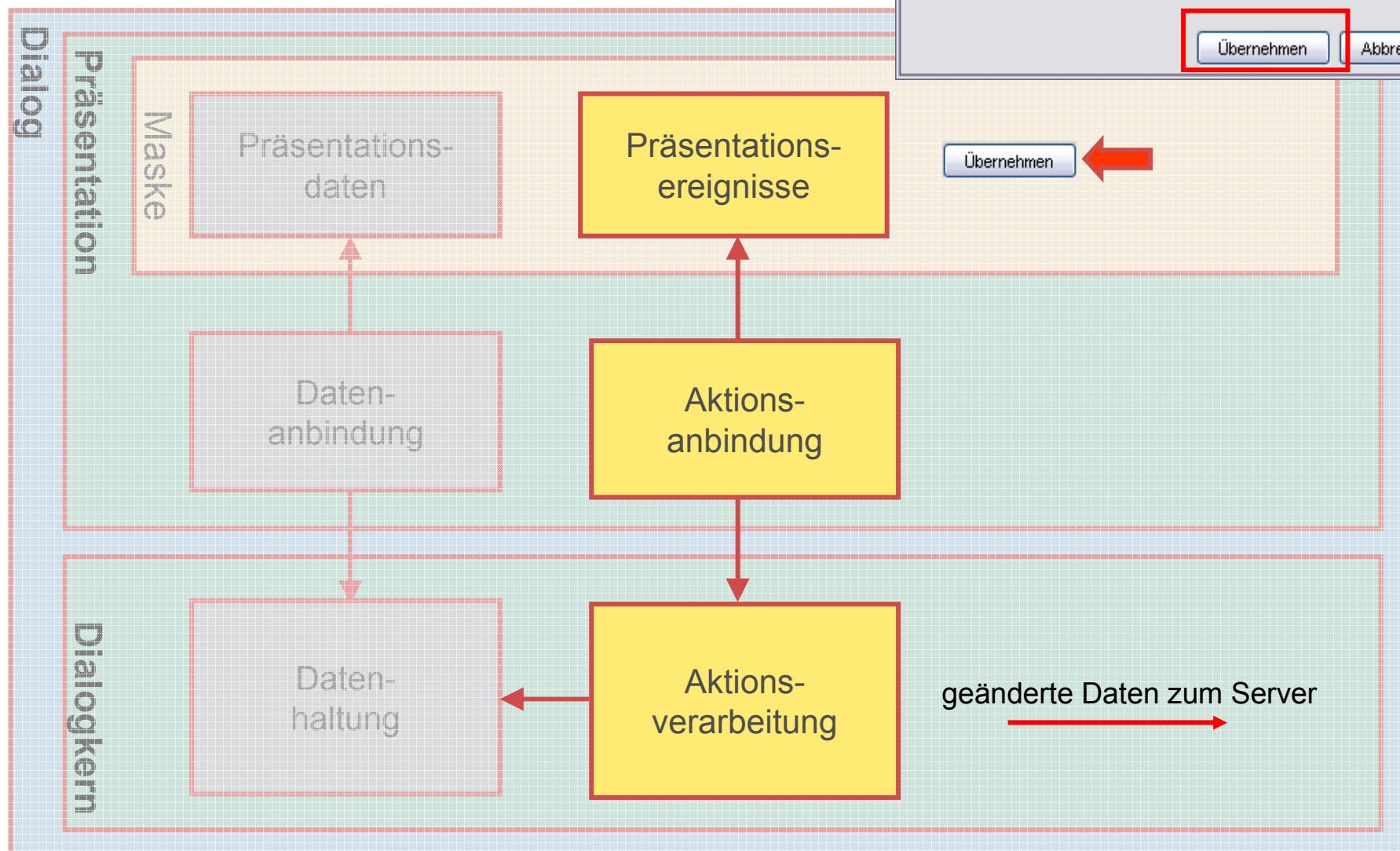
Die Datenhaltung ist ein Wrapper um die fachlichen Daten



Die Datenanbindung (Data Binding) ist ein Adapter zwischen den Widgets und der Datenhaltung



Benutzeraktionen



Ergänzung: Ereignisse vs. Aktionen

Ereignisse sind grob-granularer als Aktionen

Ereignisse	Aktionen	
<ul style="list-style-type: none">• Dialogpräsentation:<ul style="list-style-type: none">– Menü ausgewählt– Toolbar-Button angeklickt– Fenster-Schließen angeklickt– Textfeld-Inhalt geändert– ...• Dialogkern:<ul style="list-style-type: none">– „Benutzer möchte speichern“– „Benutzer möchte den Dialog schließen“– ...		<ul style="list-style-type: none">• Dialogpräsentation:<ul style="list-style-type: none">– Widget-interne Logik• Dialogkern:<ul style="list-style-type: none">– Validieren– Speichern– Dialog schließen

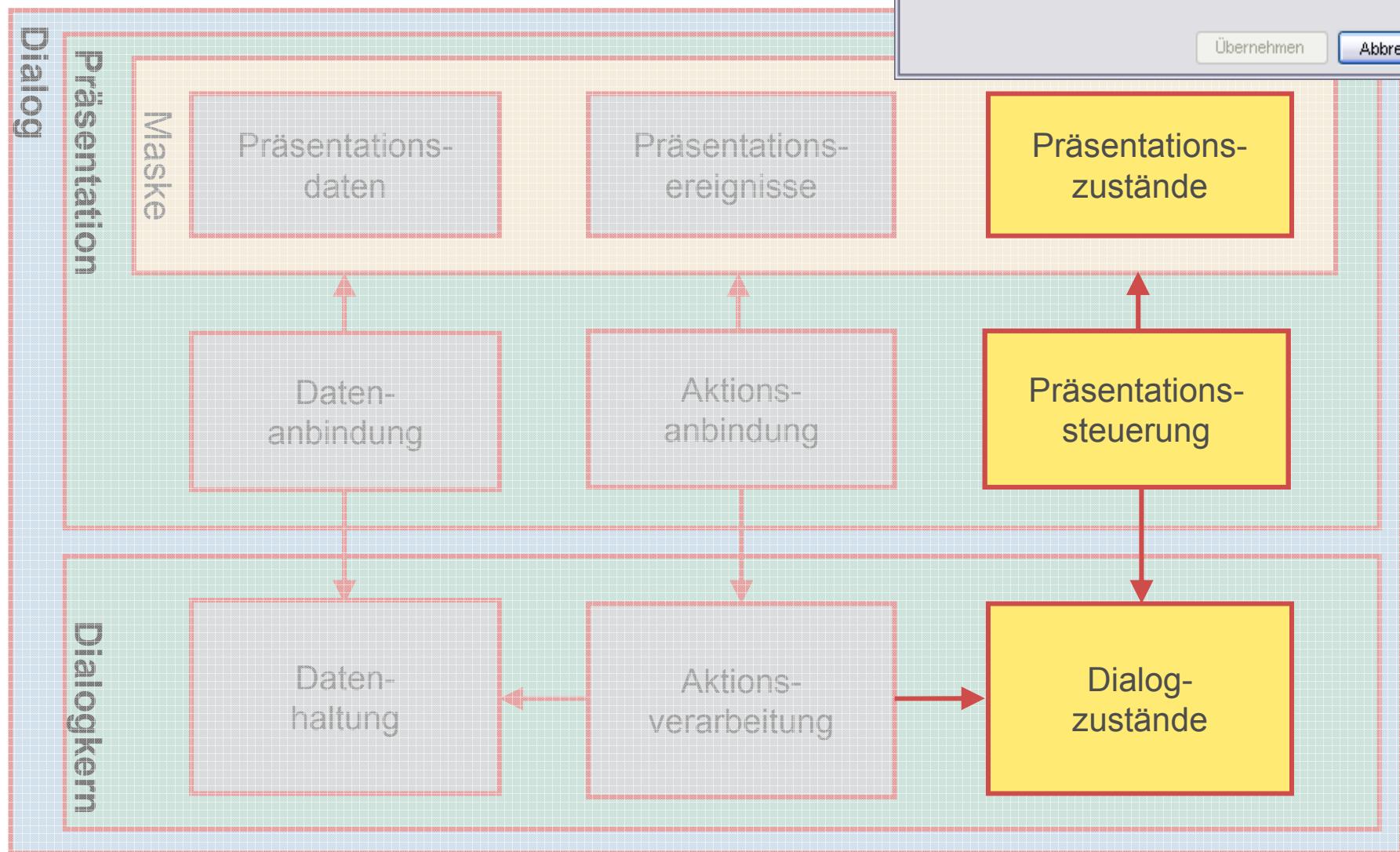
Aktionen des Benutzers lösen Aktionen im Dialogkern aus

- Eine **Benutzeraktion** kann vielfältig ausgelöst werden
 - Drücken eines Buttons
 - Schließen eines Fensters
 - Ändern eines Wertes
 - ...
- Benutzeraktionen werden von Widgets als **Präsentationsereignisse** empfangen (breite Schnittstelle mit feinen Benachrichtigungen)
- Benutzeraktionen, die nur Anpassungen in der Präsentation zur Folge haben, können direkt verarbeitet werden. Alle anderen werden an den Dialogkern weitergeben.
- **Aktionen im Dialogkern** bilden sich auf fachliche Aktionen ab (schmale Schnittstelle mit groben Benachrichtigungen)
- Nicht alle Aktionen sind immer verfügbar (z.B. Speichern bei nicht geänderten Daten). Aktionen können deswegen mit einem **Ausführbar-Zustand** versehen werden.

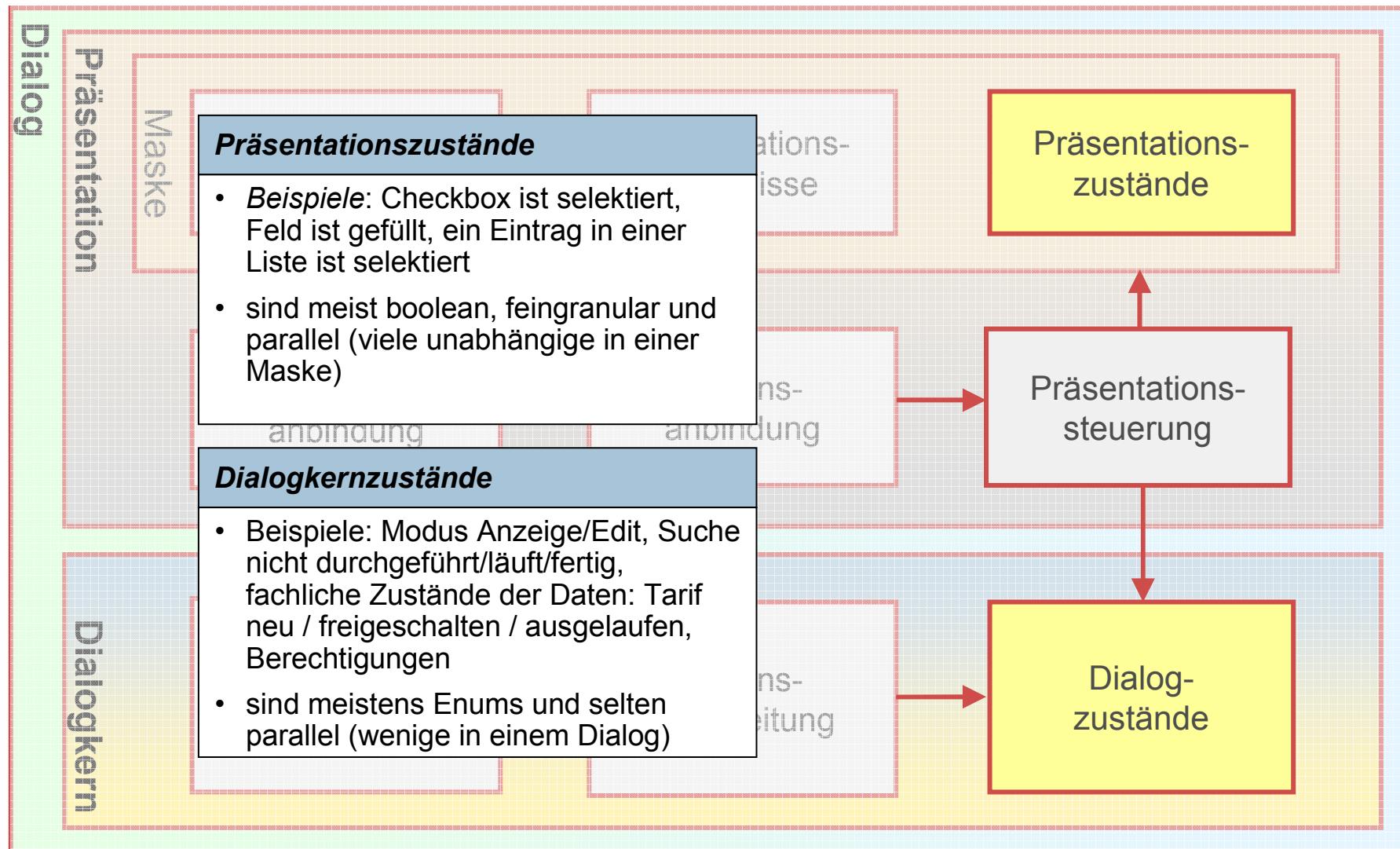
Der Dialogkern kann die Aktionsverarbeitung in verschiedenen Varianten anbieten

Java-Methoden	Kommandos (<i>Command Pattern</i>)
<ul style="list-style-type: none">• Die Präsentation ruft direkt Methoden auf dem Dialogkern auf• Abstraktion durch Interface bietet sich an• Aufruf kann generisch (z.B. über Reflection) oder spezifisch erfolgen	 <ul style="list-style-type: none">• Die Präsentation holt sich ein Kommando vom Dialogkern und ruft es auf• Das Kommando kann einen Zustand zur Ausführbarkeit haben• Ein Kommando kann entweder die fachliche Logik selbst enthalten oder deren Aufruf weiterdelegieren, z.B. an einen Zustandsautomaten

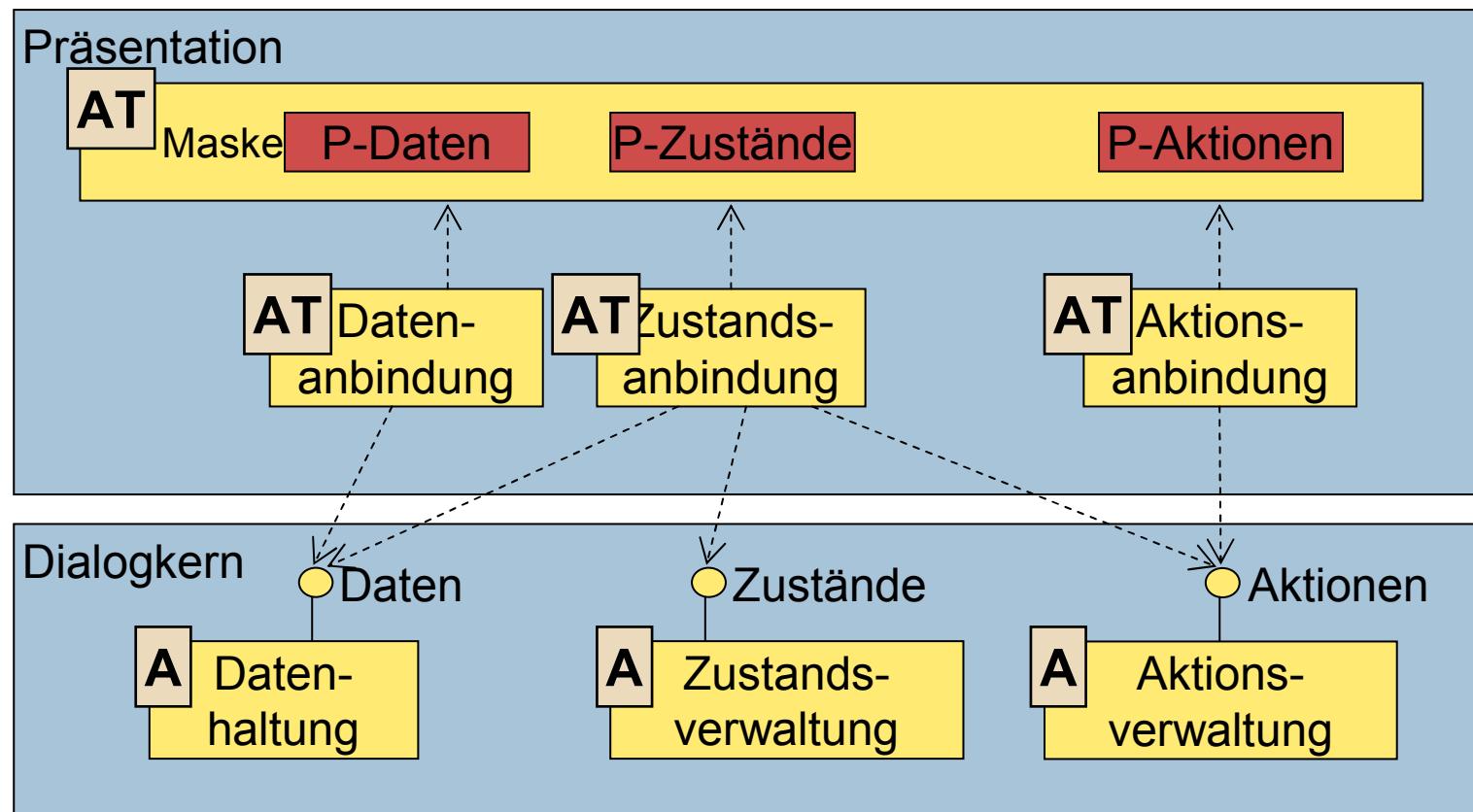
Zustände in der Dialogarchitektur



Zustände in der Dialogarchitektur



Dialog: Dialogkern und Präsentation



Präsentation: A-Code vs. T-Code vs. 0-Code

AT Trennung kann auch in der Präsentation vorgenommen werden



Blutgruppen

A: GUI Layout in Konfiguration (z.B. XML-Datei)

T: Erzeugung des GUI per Generierung/Interpretation

A: Mapping im Code oder per Konfiguration

T: Widget-Adapter (Listener + Updater)

0: Daten-Konverter (z.B. Datum \Leftrightarrow String)

0: Zustands-Regeln (z.B. UND-Verknüpfung)

Dialogkern: A-Code vs. 0-Code

Der Dialogkern besteht nur aus Fachlichkeit



Blutgruppen

A: Dialogdatenstruktur im Code oder in Konfiguration

0: Daten-Pool

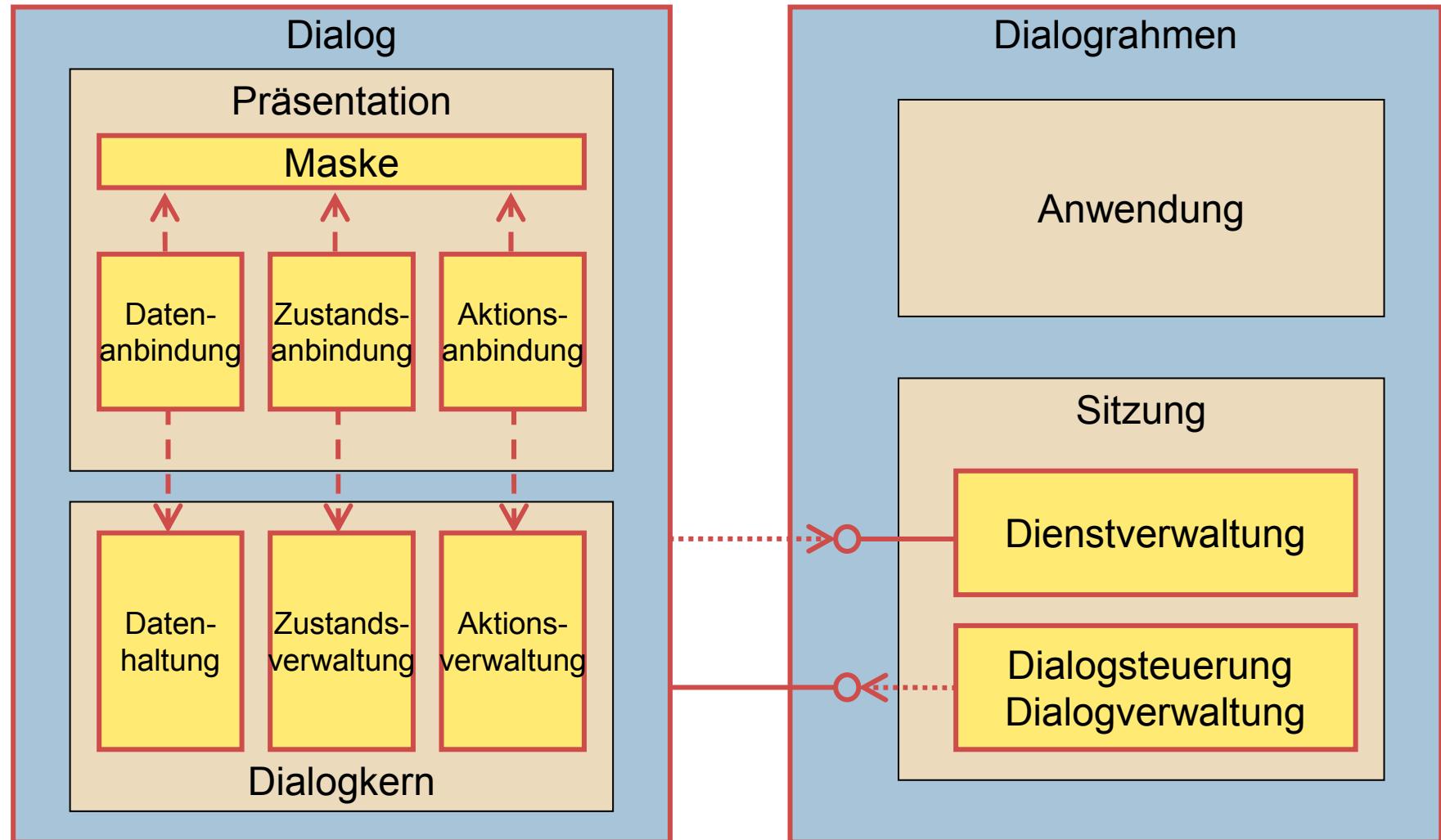
A: Aufzählungstyp für Dialogzustände

0: IAD-Interpreter

A: Konkrete Commands

0: IAD-Interpreter

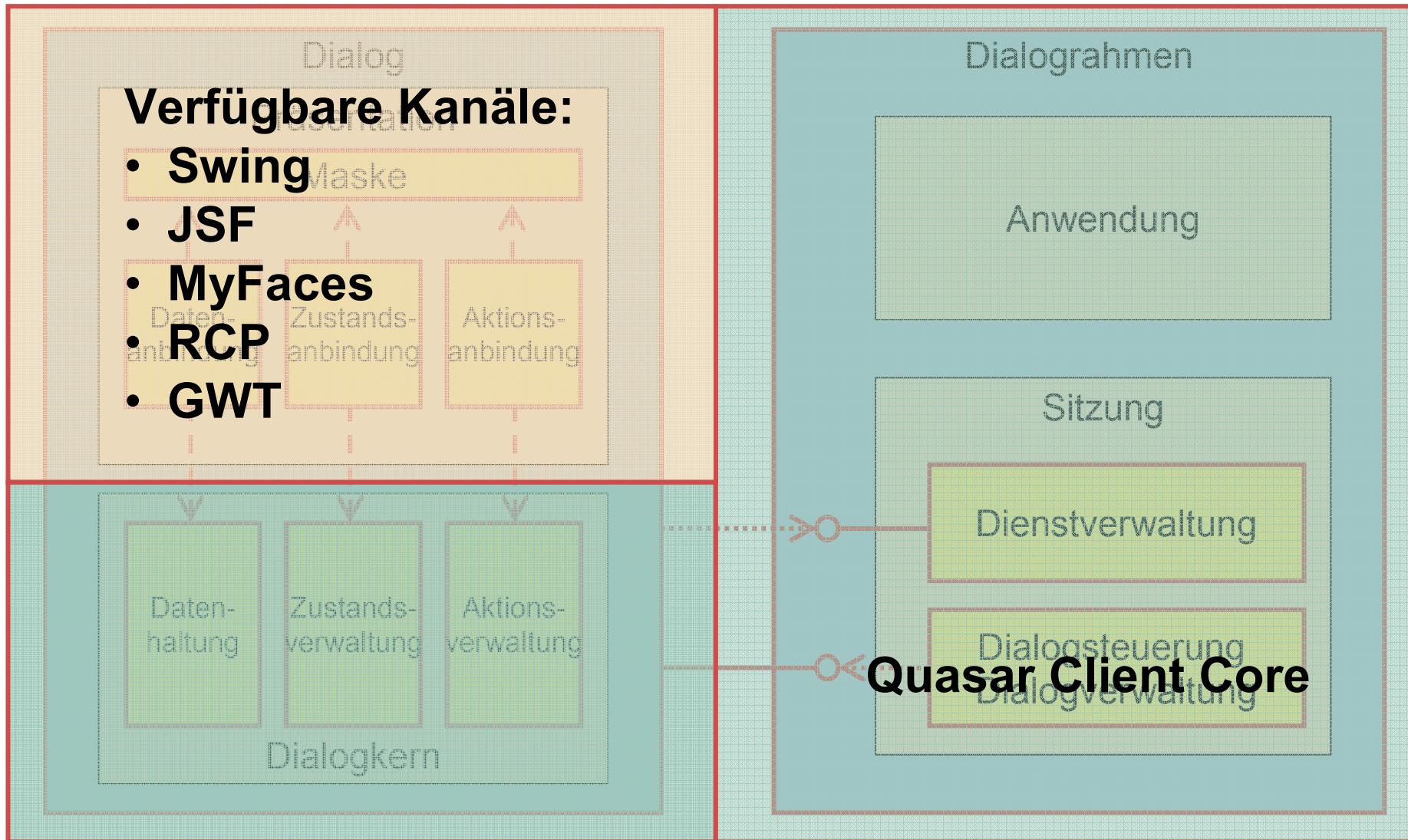
Bausteine in der Quasar-Client-Architektur



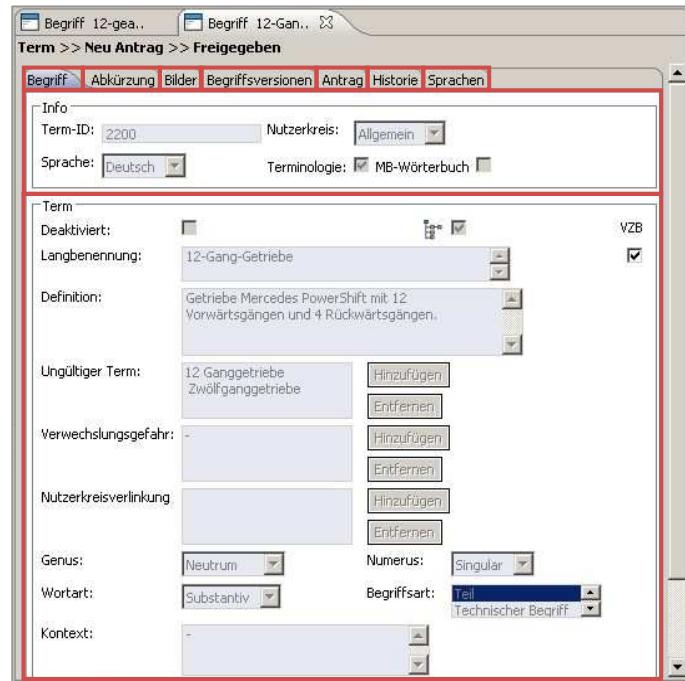
AGENDA

- Motivation: Client-Anforderungen
- Quasar Client Architektur
- **Quasar Client in der Praxis**

Bei Capgemini haben wir mit dem Quasar Client eine konkrete Umsetzung der Architektur in einem Produkt



Die gewählte Technologie kann die genannten Herausforderungen nicht alleine meistern (Beispiel: RCP)



Der Quasar Client erweitert Eclipse RCP:

RCP kennt keine Dialogkomponenten

Quasar Client bietet ein **Komponentenmodell** zur Umsetzung von Dialogkomponenten

RCP unterstützt keine visuelle Hierarchie von Dialogen innerhalb von Views und Editoren

Quasar Client unterstützt **visuelles Embedding** von Dialogkomponenten

RCP besitzt keinen einheitlichen Mechanismus zur Kommunikation innerhalb eines Plugins

Quasar Client bietet **servicebasierte Kommunikation** über Komponentenschnittstellen

RCP besitzt kein einheitliches Programmierparadigma

Quasar Client liefert ein **einheitliches, einfaches Programmierparadigma**

RCP besitzt keine ausreichende Dialog-Datenhaltung (MVC-Modell)

Quasar Client hat eine ausgefeilte und erweiterbare **Datenhaltung** für komplexe Dialoge

RCP bietet keine Modellierung von Präsentations- & Dialogzuständen

Quasar Client bietet Erstellung und Verknüpfung von **Präsentations- & Dialogzuständen**

Die Vorteile der Quasar Client Architektur:

Beherrschung der Komplexität

Durch Aufteilung komplexer Dialoge in einfache Dialogkomponenten

Parallelisierbarkeit der Entwicklung

Klar definierte Aufgabenblöcke, können von verschiedenen Entwicklern umgesetzt werden

Testbarkeit

Durch strikte Kommunikation über Schnittstellen lässt sich die Funktionsweise früh testen

Wartbarkeit & Robustheit

Änderungen in den Dialogkomponenten bleiben lokal – Schnittstellen schützen Dialog voreinander

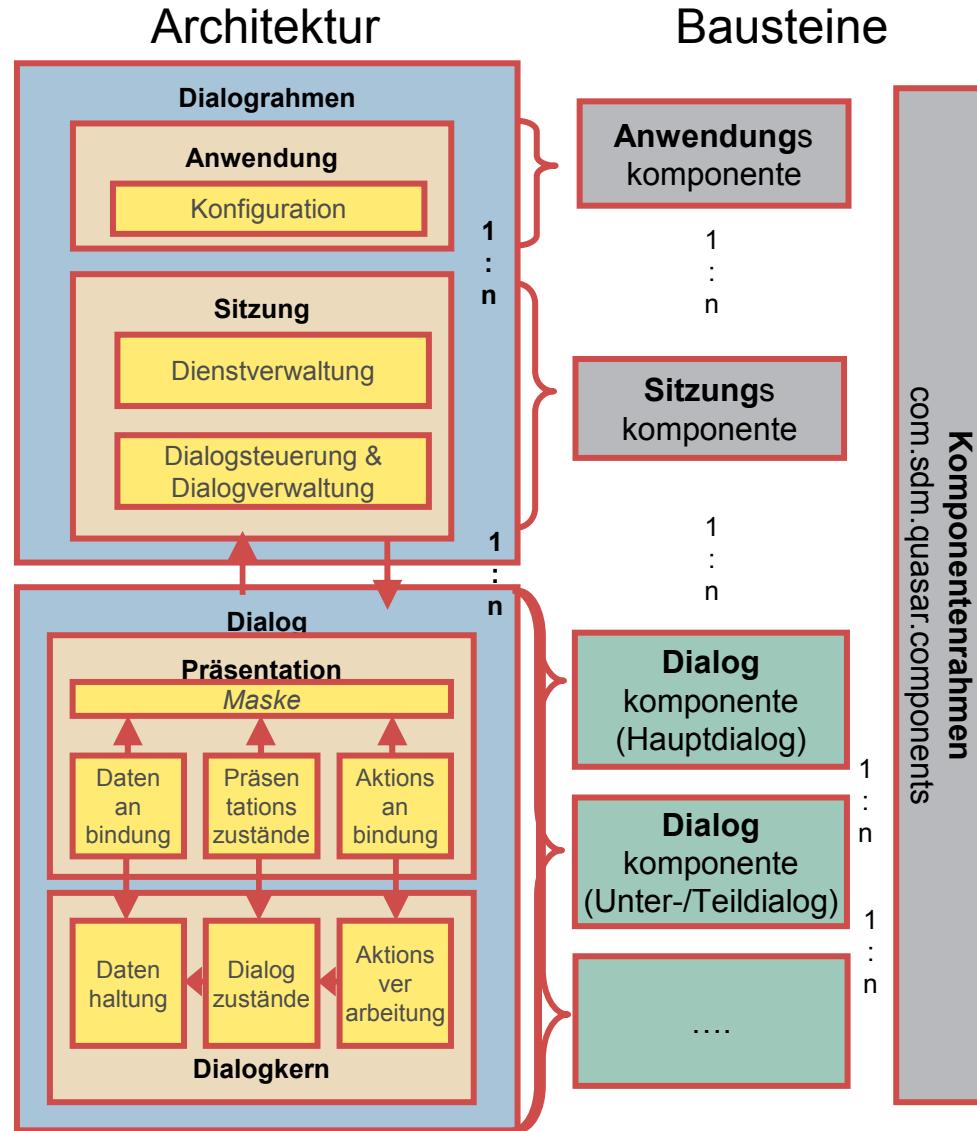
Wiederverwendung

Fachliche oder technische Dialogkomponenten sind in anderen Kontexten wiederverwendbar

Beispiel: Pet Store

Order Header				
Order Date:	<input type="text"/>		Order Time:	<input type="text"/>
Requested Delivery Date:	<input type="text"/>		e.g. "2006-12-31"	
Order Lines				
No.	Product	Price/Item	Amount	Price
1	Bird	10.00 EUR	30	300.00 EUR
2	Cat	30.00 EUR	20	600.00 EUR
3	Dog	100.00 EUR	10	1,000.00 EUR
<input type="button" value="Add"/> <input type="button" value="Remove"/>				
No.:	Product:	Price/Item:	Amount:	Price:
<input type="text"/>	<input style="width: 100px; height: 20px; border: 1px solid #ccc; padding: 2px; margin-right: 5px;" type="text" value="Bird"/> <input type="button" value="▼"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Total: <input type="text"/>				
Delivery Address				
Name:	<input type="text"/>			
Street:	<input type="text"/>			
ZIP Code:	<input type="text"/>	City:	<input type="text"/>	
Billing Address				
<input type="checkbox"/> Same as delivery address				
Name:	<input type="text"/>			
Street:	<input type="text"/>			
ZIP Code:	<input type="text"/>	City:	<input type="text"/>	
<input type="button" value="Save"/> <input type="button" value="Exit"/>				

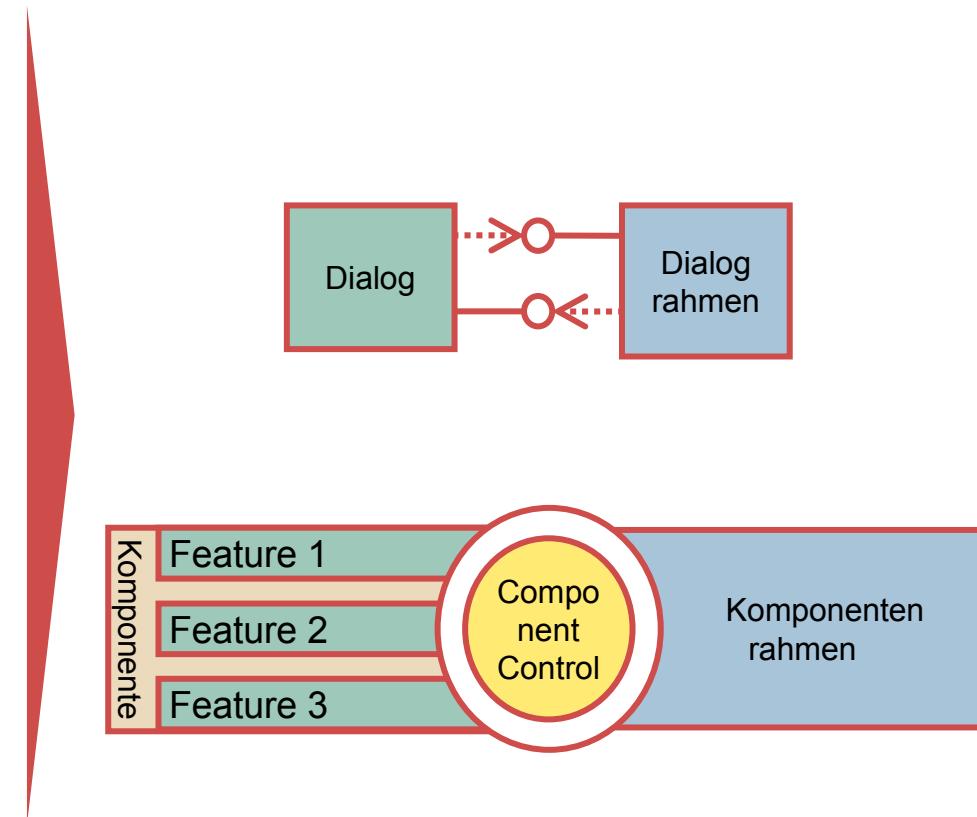
Die Quasar-Client Bausteine definieren ein durchgängiges Komponentenmodell



Überblick: Baustein Komponente

Grundbaustein: Komponente

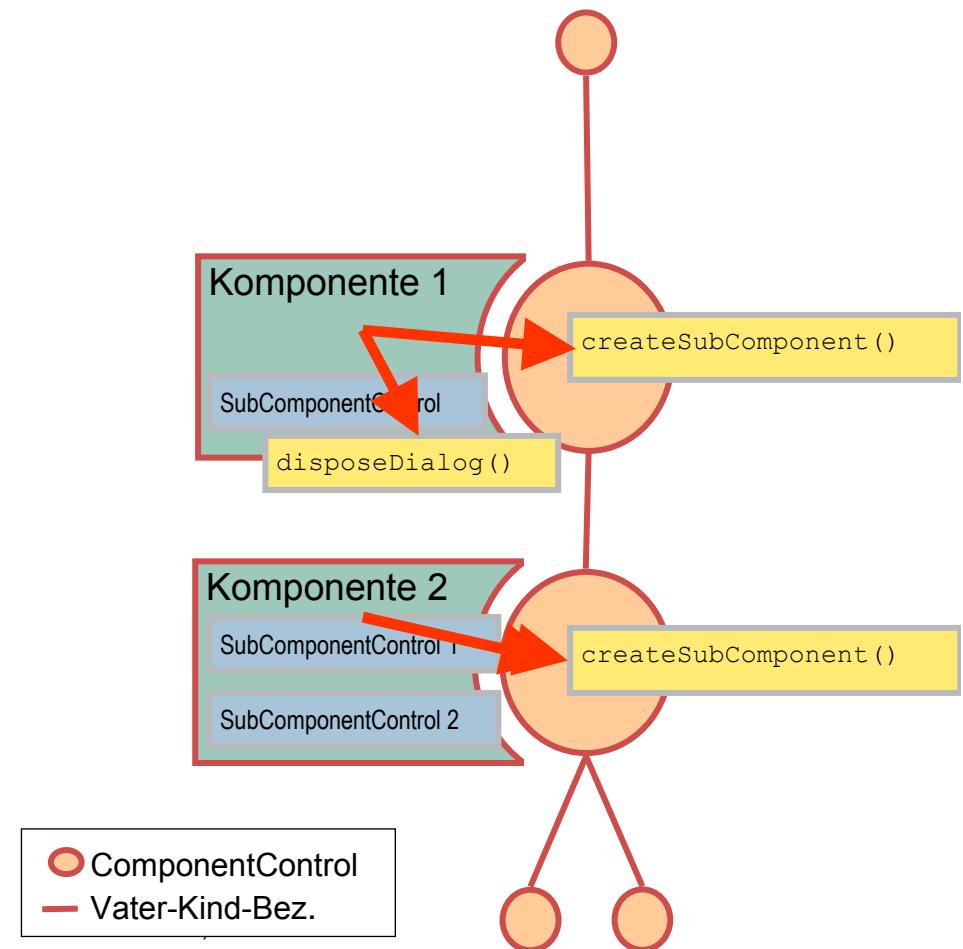
- Eine Komponente besteht aus einem oder mehreren Features
 - Wiederverwendung
 - Strukturierung
- ComponentControl bildet die Schnittstelle zum Komponentenrahmen
- Im Bezug auf die Dialoge:
 - Dialoge sind als Komponenten bzw. Features implementiert
 - Dialograhmen steht als Komponentenrahmen zur Verfügung



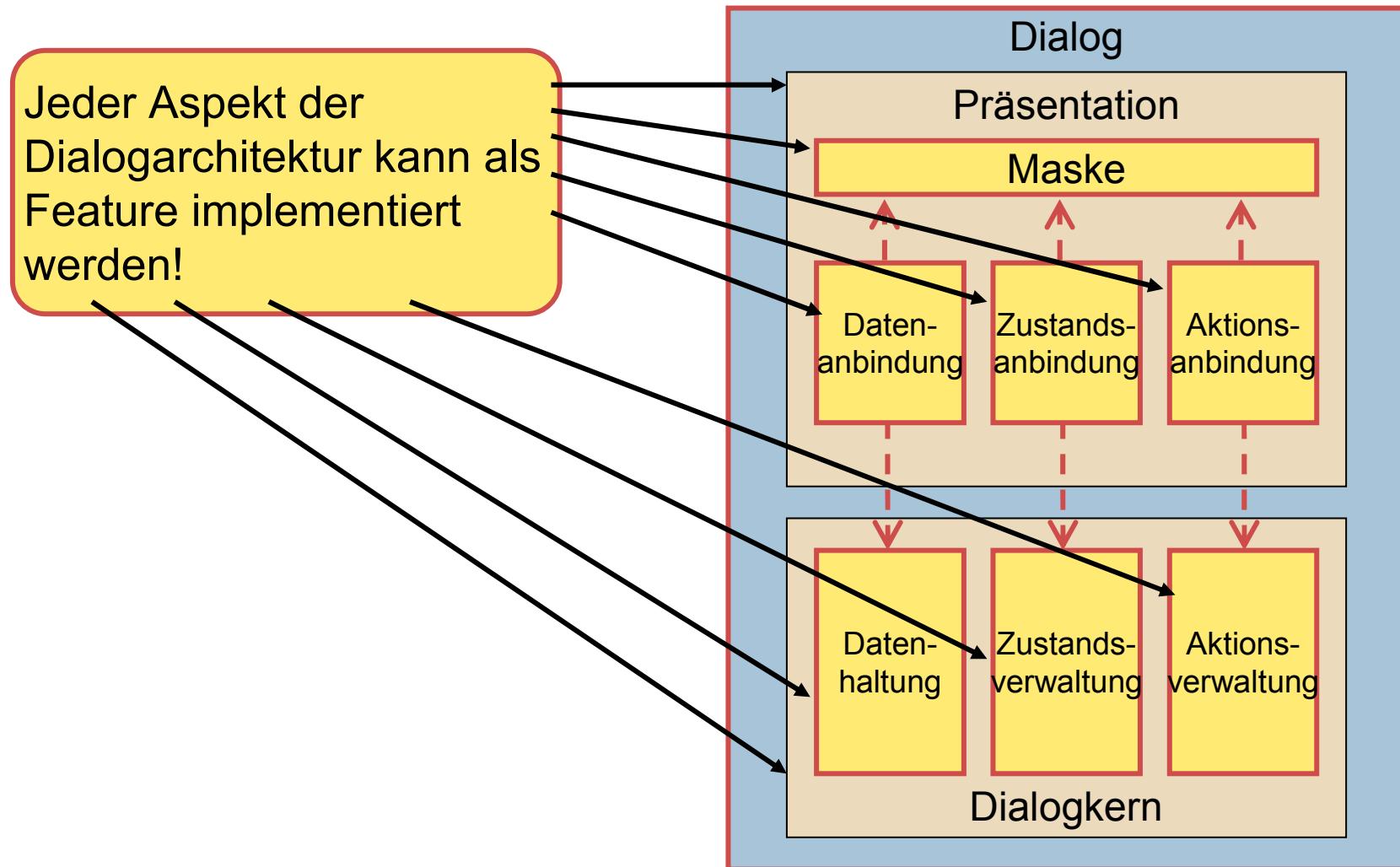
Hierarchische Komponentenverwaltung

Verwaltung entlang der Hierarchie

- Komponenten werden entlang der Hierarchie erstellt
- das ComponentControl enthält die Methode `createSubComponent (...)` zum Erzeugen einer Subkomponente
- zur Kontrolle über die Subkomponente erhält man ein SubComponentControl
- Über die Methode `disposeComponent ()` des SubComponentControls kann die Subkomponente geschlossen werden

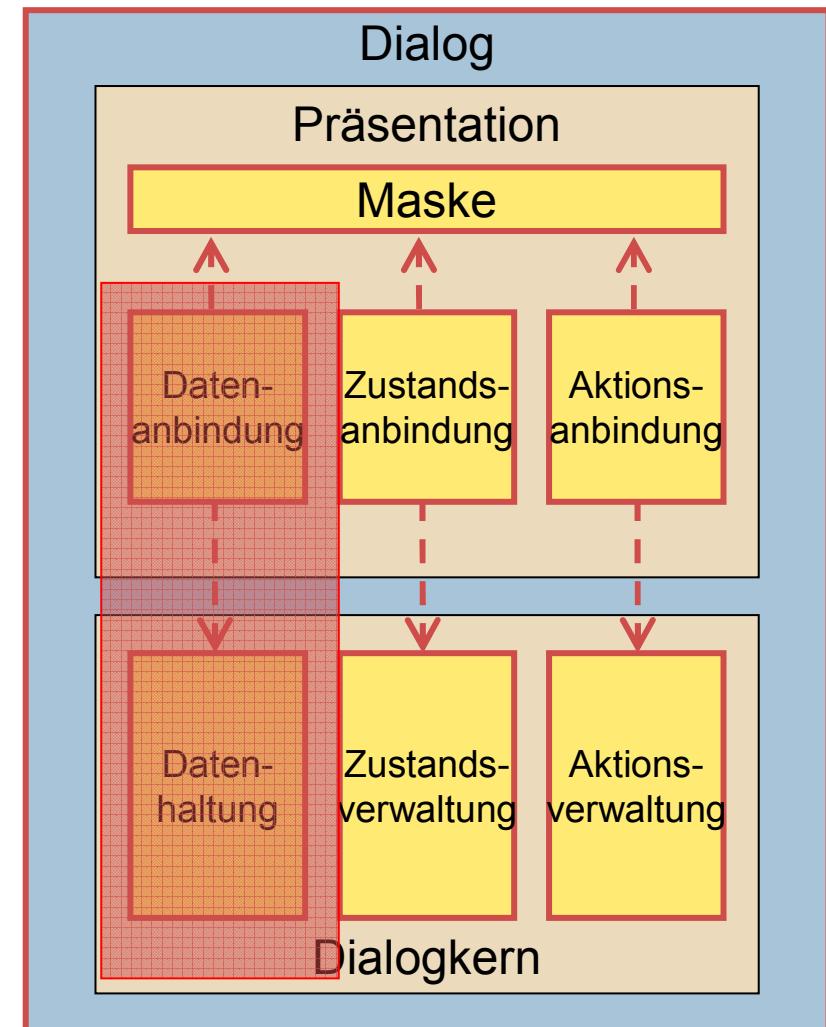
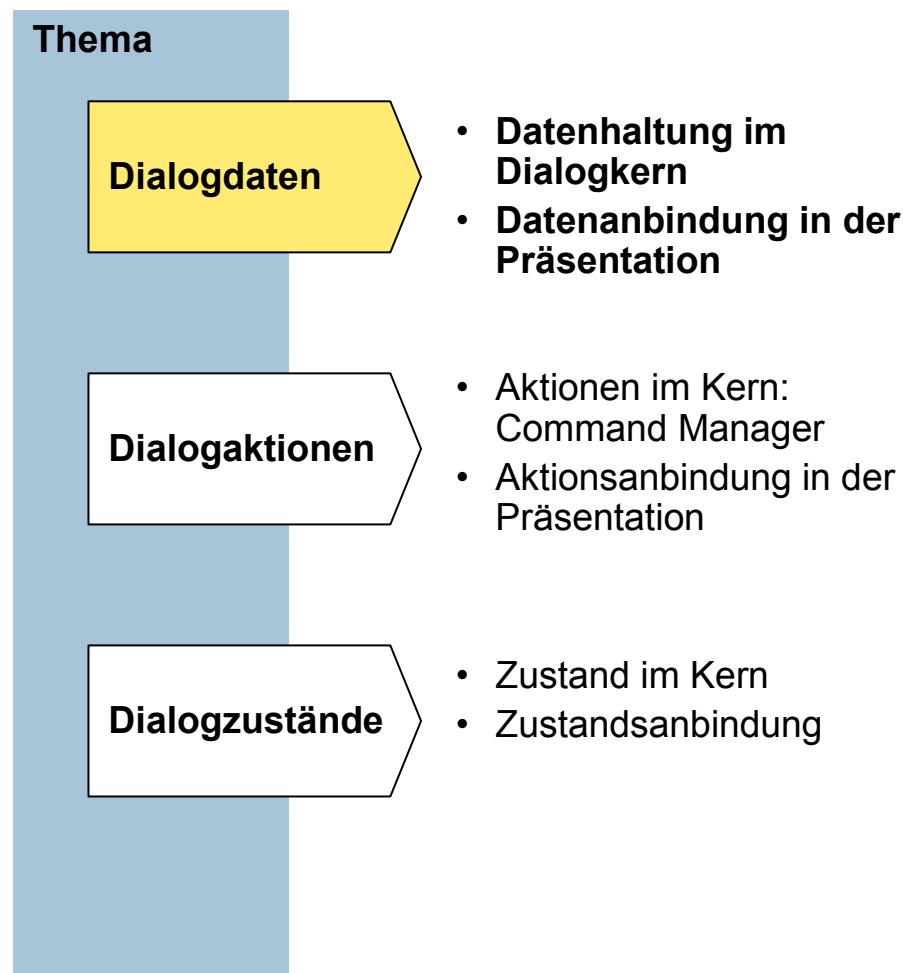


Features in Dialogen

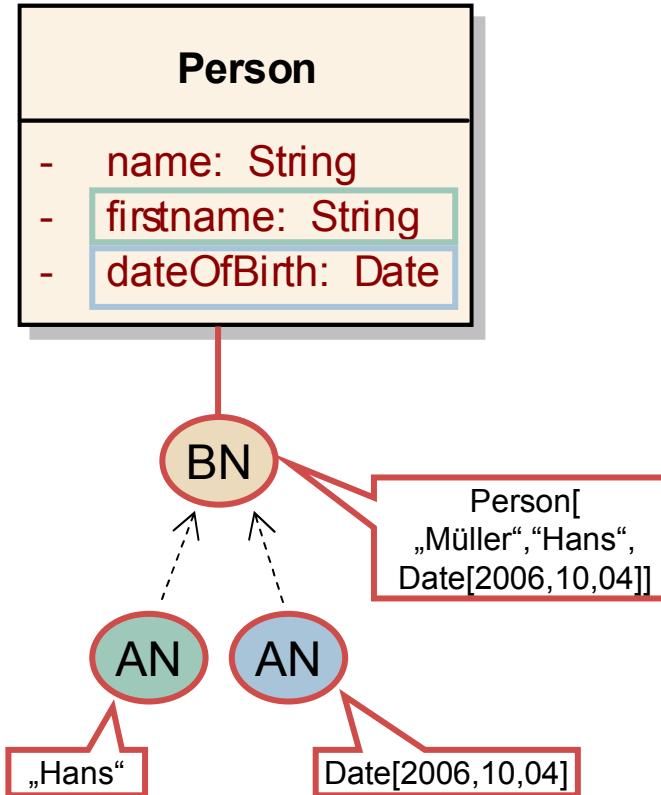


Client-Programmierung – Agenda

Typische Tätigkeiten in der Client Entwicklung



Konfiguration des fachlichen Objectgeflechts



Legende: BN BaseNode, AN AspectNode, IN Indexed Node, CN Computed Node

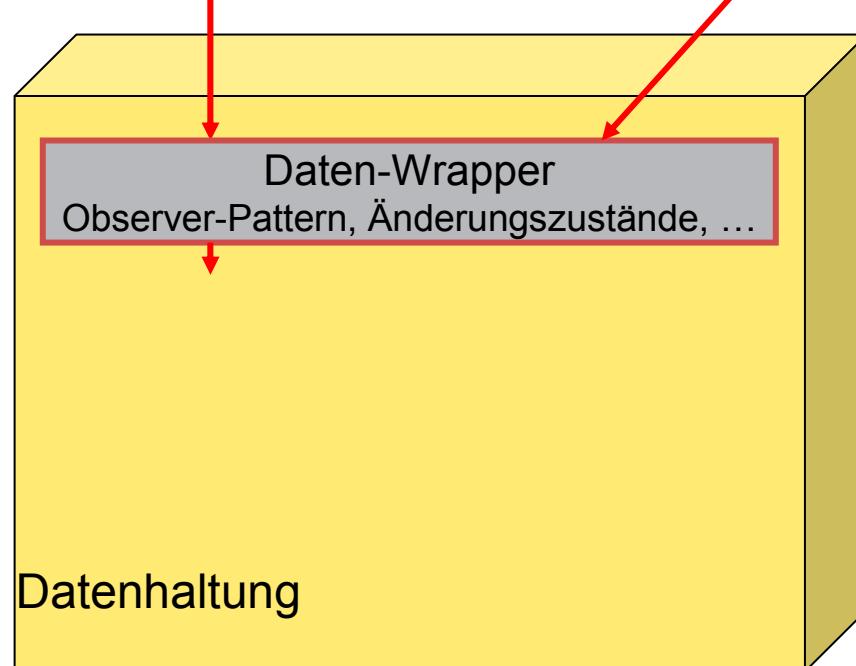
DataManagerFeature zur Datenhaltung: Nutzung, fachliche & technische Konfiguration

Zugriff/Nutzung über Keys:

z.B. Nutzung Binding: `getValue("DTO")`

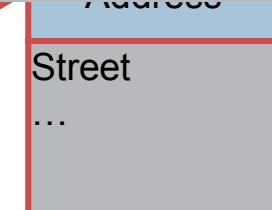
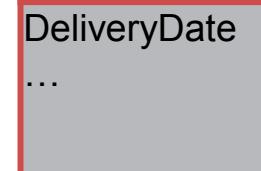
z.B. Nutzung in fachlichen Aktionen:

```
setValue("DTO", new  
PurchaseOrderDto())
```



Fachliche Konfiguration:

```
addBaseNode("DTO",  
PurchaseOrderDto.class);  
  
addAspectNode("DELIVERY_DATE",  
"DTO",  
PurchaseOrderDto.class,  
"deliveryDate", Date.class);  
  
addIndexedNode("ORDER_LINES",  
"DTO",  
PurchaseOrderDto.class,  
"orderLines", List.class);
```



Fachliche Konfiguration des DataManagers mit GenericNodesPlugIn

Beispiel

```
BasicNodesPlugIn basicNodesPlugIn =  
(BasicNodesPlugIn) dataManager.getPlugIn(BasicNodesPlugIn.class);  
  
basicNodesPlugIn.addBaseNode(DTO, PurchaseOrderDts.class);  
basicNodesPlugIn.addAspectNode(DTO, DTS, "purchaseOrder",  
    PurchaseOrderDto.class);  
basicNodesPlugIn.addAspectNode(ORDER_LINES, DTS, "orderLines",  
    List.class);  
// indexed node to support to show the list of employees in a  
table  
basicNodesPlugIn.addIndexedNode(ORDER_LINES_INDEXED, ORDER_LINES,  
    Integer.class, OrderLineDto.class);  
basicNodesPlugIn.addAspectNode(ORDER_LINES_INDEXED_AMOUNT,  
    ORDER_LINES_INDEXED, "amount", Integer.class);
```

Lesen und Schreiben von Daten: Die Nutzschmittstelle des DataManagers

Nutzung

Genereller Zugriff

Zugriff auf Listenelemente

Beispiel

Über den definierten Schlüssel wird der Zugriff gewährleistet:

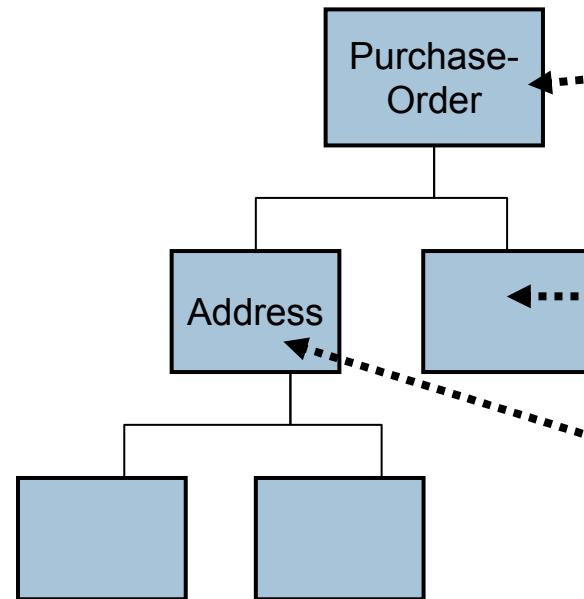
```
Dialogkern  
getDataManagerFeature() .setValue(DTS, orderDts);  
PurchaseOrderDts orderDts =  
    getDataManagerFeature() .getValue(DTS );
```

Mit einem Kontext kann auf Elemente eines Indizierten Knotens zugegriffen werden.

```
Dialogkern  
Object context = new Integer(0);  
Employee employee = getDataManagerFeature()  
    .getValue(ORDER_LINES_INDEXED, context );
```

Der Kontext ist ein Integer bei Listen und Arrays und ein String bei Maps.

Durch Datenanbindung werden können Widgets mit dem Datenmodell verknüpft werden



Order Header			
Order Date: <input type="text"/>			
Requested Delivery Date: <input type="text"/>			
Order Lines			
No.	Product	Price/Item	Amount
1	Bird	10.00 EUR	30
2	Cat	30.00 EUR	20
3	Dog	100.00 EUR	10
No.:	Product:	Price/Item:	Amount
<input type="text"/>	<input type="text" value="Bird"/> <input type="button" value="▼"/>	<input type="text"/>	<input type="text"/>
Delivery Address			
Nahter:	<input type="text"/>		
Street:	<input type="text"/>		
ZIP Code:	<input type="text"/>	City:	<input type="text"/>

Die Verknüpfung bezieht sich zunächst nur auf die Metadaten.
Die eigentlichen Daten werden erst später zugeordnet.

Data Binding mit SwingBindingFeature

Beispiel

```
// Binding eines Attributs auf ein Textfeld  
JTextField txtDelivery = new JTextField();  
getSwingBindingFeature().bindWidget(  
    DATA_DELIVERY, txtDelivery);  
  
// Binding mit Angabe eines Converters  
getSwingBindingFeature().bindWidget(  
    DATA_DELIVERY, txtDelivery,  
    new MediumDateToStringConverter());  
  
// Binding einer Liste auf eine JTable mit Binding des  
selektierten Tabelleneintrags  
getSwingBindingFeature().bindWidget(  
    DATA_ORDER_LINES_INDEXED_SEQ_NO,  
    panel.getSequenceNumberColumn() );
```

Requested Delivery Date:

Presentation

PurchaseOrderDts

1

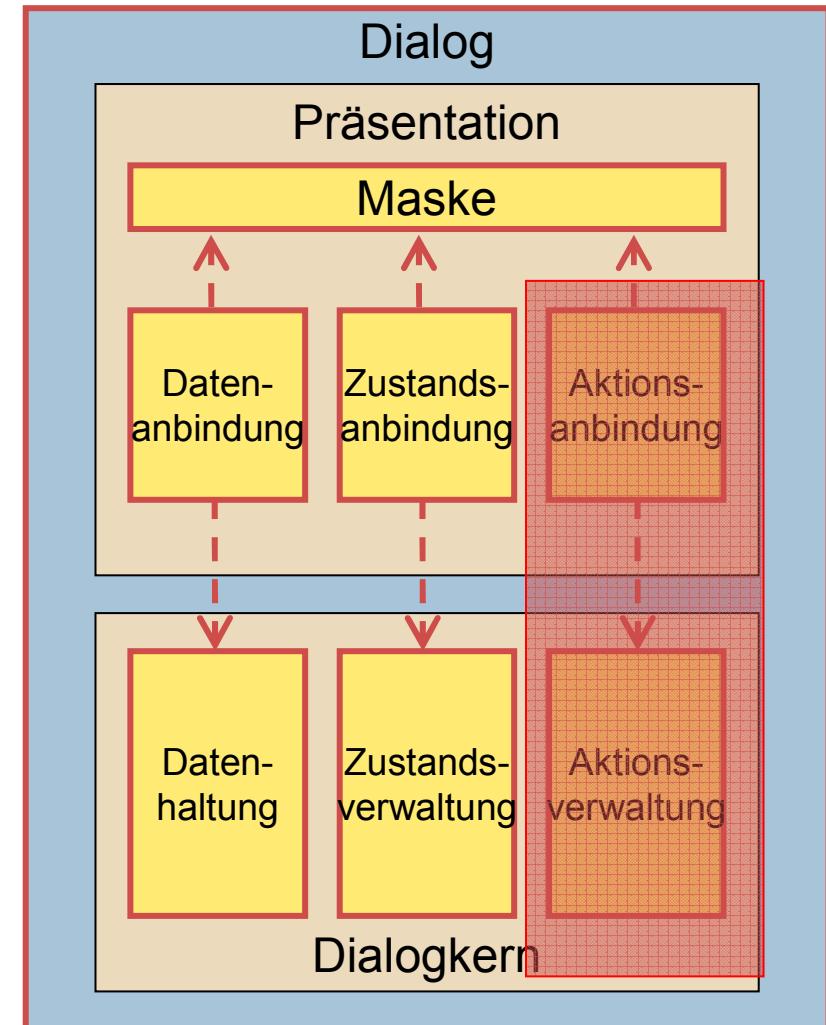
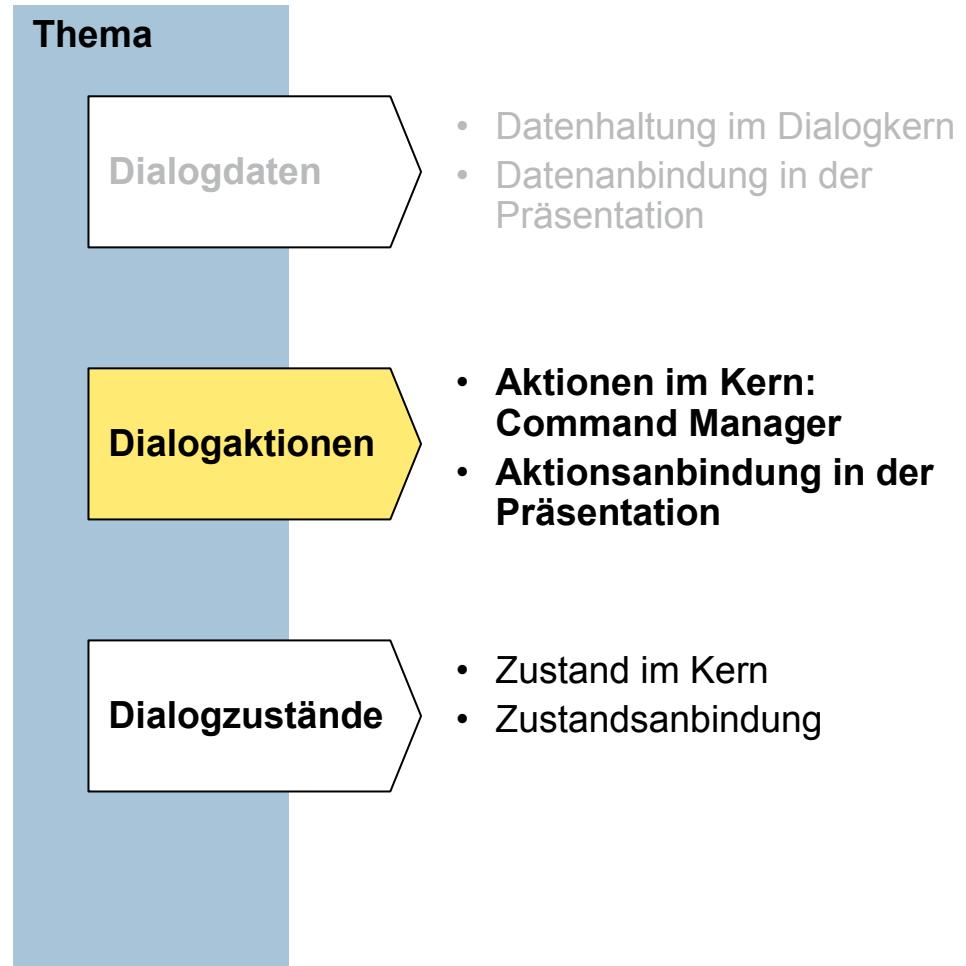
1

PurchaseOrderDto
deliveryDate : String

Order Lines	
No.	Prod
1	Bird
2	Cat
3	Dog

Client-Programmierung – Agenda

Typische Tätigkeiten in der Client Entwicklung



Aktionsverwaltung: der CommandManager

Beispiel

- **Grundsätzliche Implementierung:**
 - direkt als Methoden (Web) -> keine Unterstützung notwendig
 - über das **Kommando-Pattern**
- **Commands werden vom CommandManager verwaltet:**
`CommandManager.registerCommand(...);`
- **Wird vom Dialogkern instanziert und konfiguriert**
- **Verfügbare Implementierungen:**
 - GenericCommandManager: Verknüpft die registrierten Commands mit Methoden eines Zielobjektes mittels Reflection
 - RunnableCommandManager: Commands werden als Runnables implementiert und diese Runnables registriert
- **Der CommandManager wird über das CommandManagerFeature anderen Features zur Verfügung gestellt**

Beispiel: Beenden der Anwendung über 'Exit'

Schritte

Initialisierung im
Dialogkern

Anbindung über
Adapter

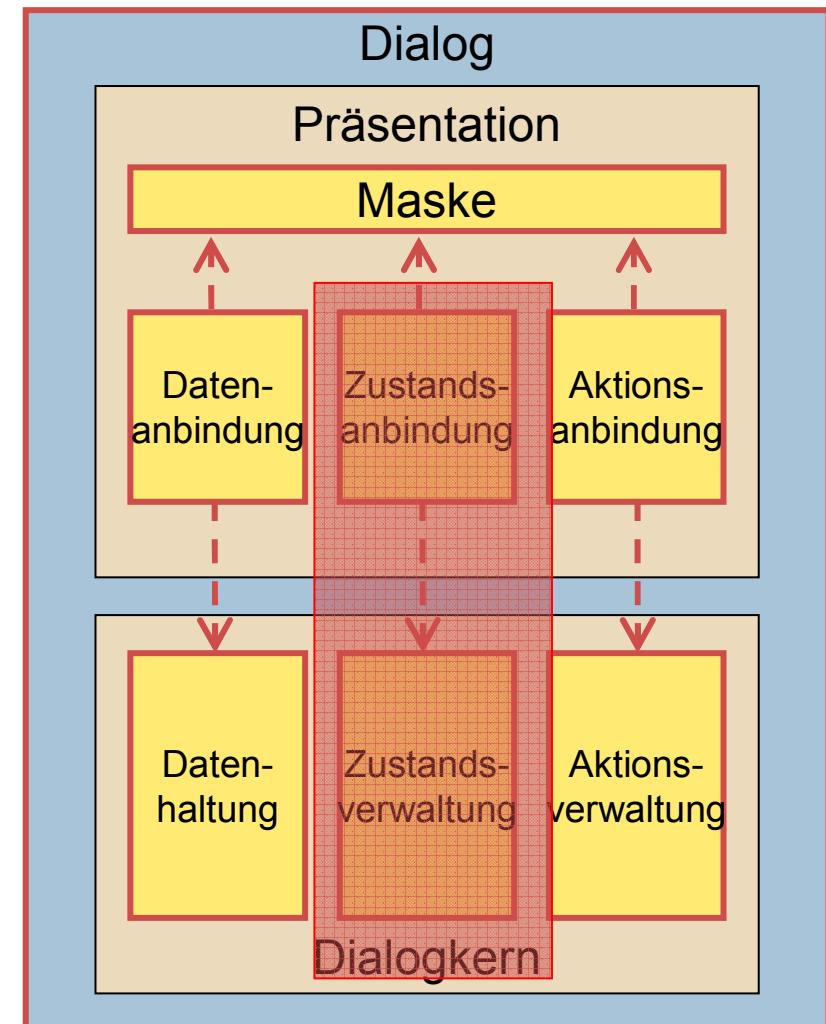
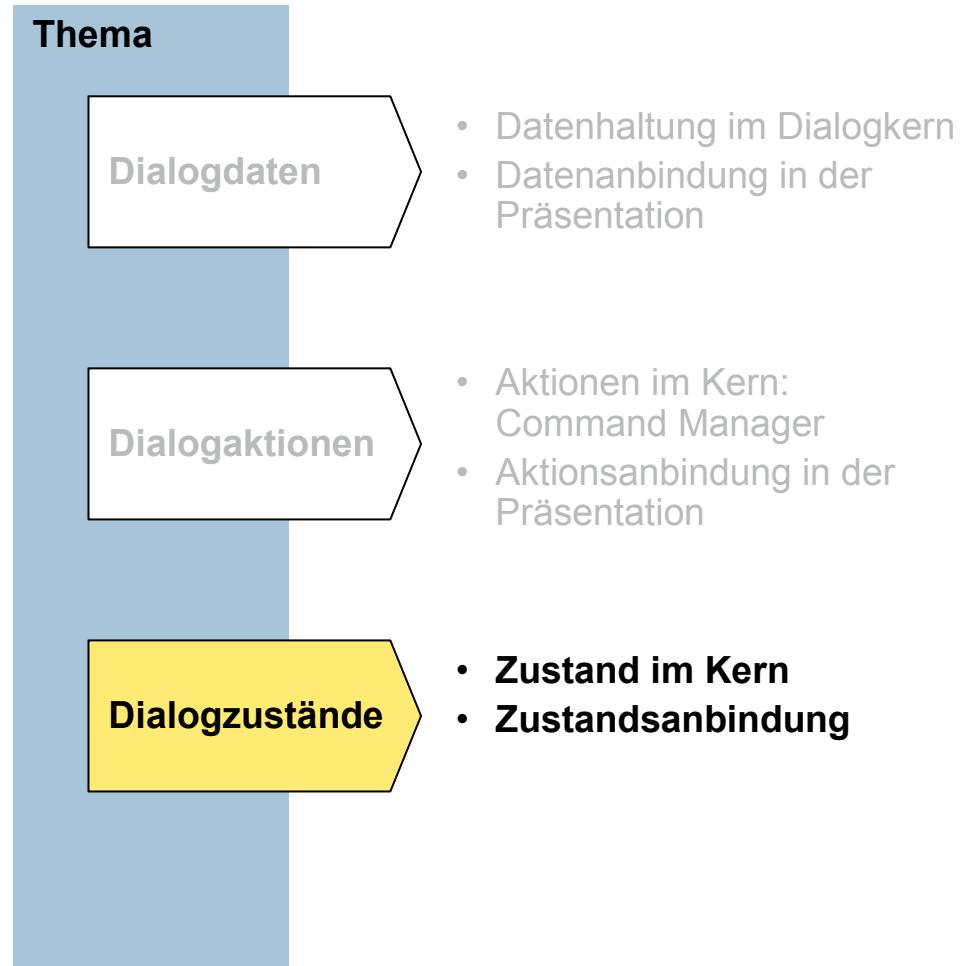
Beispiel

```
GenericCommandManager commandManager =  
    new GenericCommandManager();  
getCommandManagerFeature()  
    .setCommandManager(commandManager);  
commandManager.registerCommand(  
    COMMAND_EXIT, this, "exit", null);
```

```
Command exitCommand = getCommandManagerFtr()  
    .getCommand(COMMAND_EXIT);  
JButton btnExit = new JButton();  
JComponentCommandAdapter exitButtonAdapter =  
    new JComponentCommandAdapter(  
        btnExit, exitCommand);
```

Client-Programmierung – Agenda

Typische Tätigkeiten in der Client Entwicklung



Zustände im Kernel: States und boolesche States



Beispiel

- States sind Zustände, deren Zustandsänderung beobachtbar ist.
- BooleanStates besitzen nur 2 Ausprägungen: true & false.
- BooleanStates können über Boolesche Logik verknüpft werden:

```
BooleanState isEnabledBooleanState =  
    ANDBooleanState.create(  
        "EnabledBooleanState",  
        booleanState1, true,  
        booleanState2, false);
```

- States können auf BooleanStates abgebildet werden:

```
dialogState = new MutableState(DISPLAY);  
isEditState = new StateAcceptedBooleanState(  
    dialogState, EDIT);
```

Beispiel: Minimaler Auftragsbetrag unterschritten

No.:	Product:	Price/Item:	Amount:	Price:	Total:
1	Bird	1.000,00 €	3	3.000,00 €	3.000,00 €

Schritte

Initialisierung

Beispiel

```
acceptor = new Acceptor() {  
    public boolean accept(Object obj) {  
        return (obj != null) &&  
               ((Double) obj) > 10000;  
    } };  
totalAcceptedState = new  
DataManagerValueAcceptedBooleanState(  
    getDataManager(),  
    DATA_ORDER_LINES_TOTAL,  
    acceptor);  
  
new JComponentBorderBooleanStateAdapter(  
    txtTotal, getKernelFeature()  
    .getTotalAcceptedState());
```

Anbindung über
Adapter

Forschung und Lehre leben vom Gedankenaustausch, dabei helfen klare Vereinbarungen:

Die Inhalte dieser Präsentation (u.a. Texte, Grafiken, Fotos, Logos etc.) sowie die Präsentation selbst sind urheberrechtlich geschützt. Alle Rechte stehen, soweit nicht anders gekennzeichnet, Capgemini zu.

Capgemini gestattet ausdrücklich die öffentliche Zugänglichmachung kleiner Teile der Präsentation zu Zwecken der nicht kommerziellen Lehre und Forschung.

Jede darüber hinausgehende Nutzung ist nur mit ausdrücklicher, schriftlicher Einwilligung von Capgemini zulässig.

Haftungsausschluss

Auch wenn diese Präsentation und die damit zusammenhängenden Ergebnisse nach bestem Wissen und sorgfältig erstellt wurden, übernimmt weder Capgemini, noch der/die Autoren, eine Haftung für deren Verwendung.

Bei Fragen wenden Sie sich bitte an:

Capgemini
Tim Lüecke
Lübecker Straße 128
22087 Hamburg
+49 40 254491-314
tim.lueecke@capgemini.com

Vielen Dank für Ihre Aufmerksamkeit!

www.de.capgemini.com