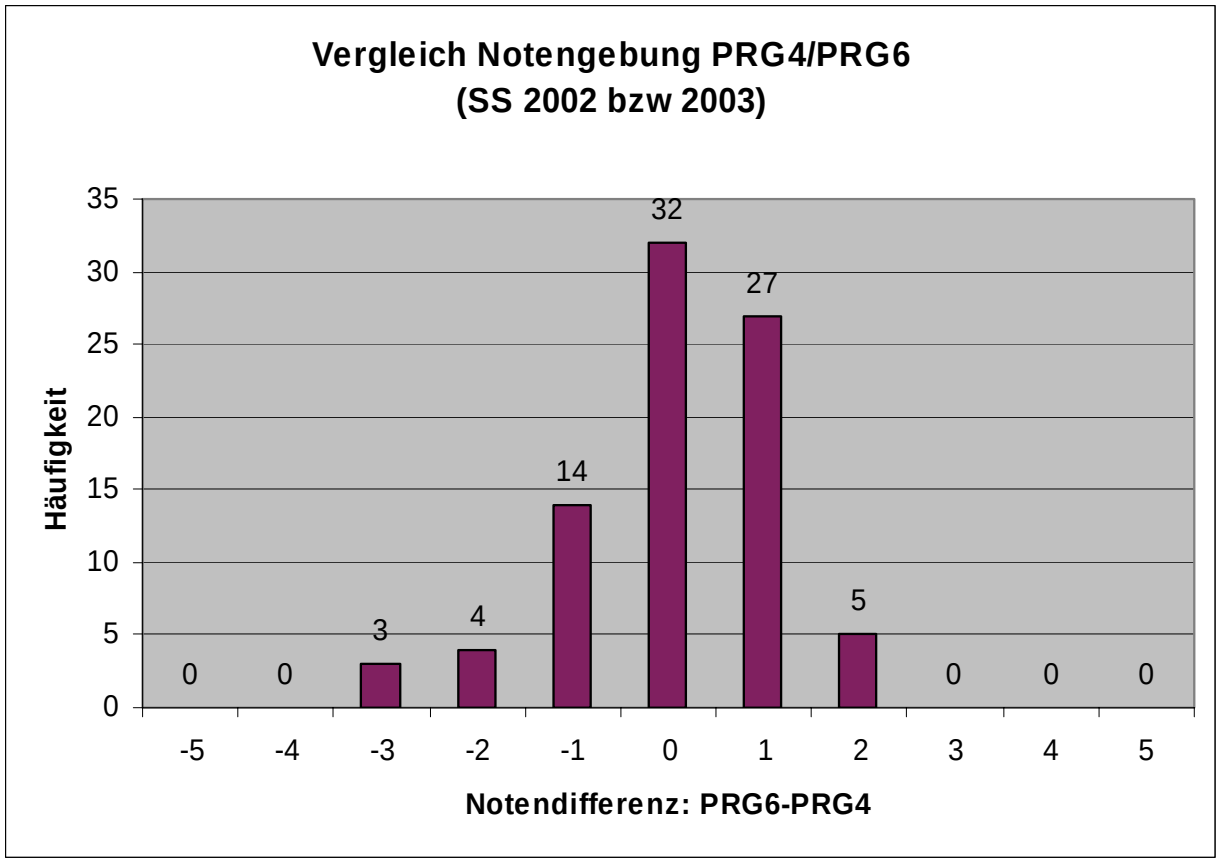
PRG 5
Komponentenorientierte Software-Entwicklung

WS 2004/05

Organisatorisches

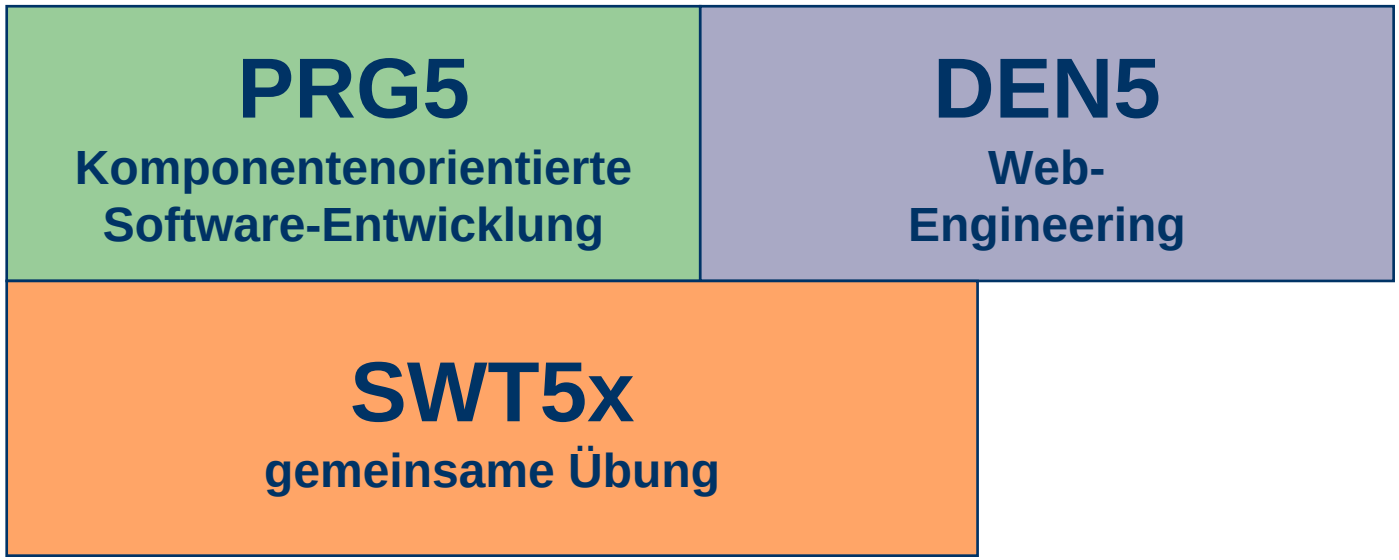
- Beurteilung
  - PRG5 (VL)
    - Schriftliche Klausurarbeit (ohne Unterlagen)
  - SWT5x (ÜB)
    - Keine Klausur
    - Kleinere Projektarbeiten
    - Präsentation der Projektarbeiten
    - Gespräch über Projektarbeiten
- Unterlagen/Literatur
  - http://www.fh-hagenberg.at/staff/jheinzel/PRG5
    - Grundlegende Literatur
    - Weiterführende und ergänzende Literatur

Noten: Statistische Auswertung



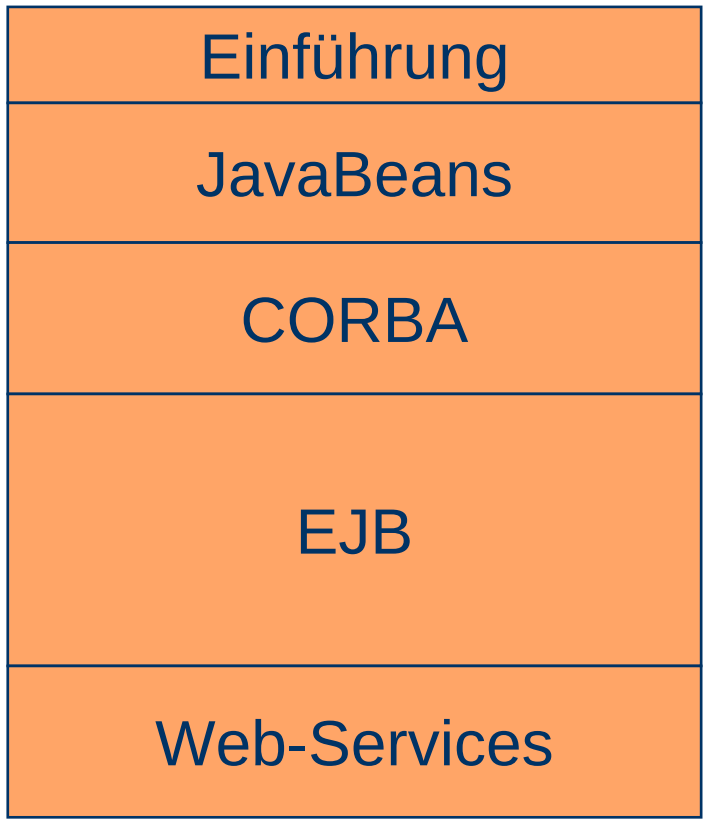
Organisation

- Zusammenspiel mit anderen LVAs

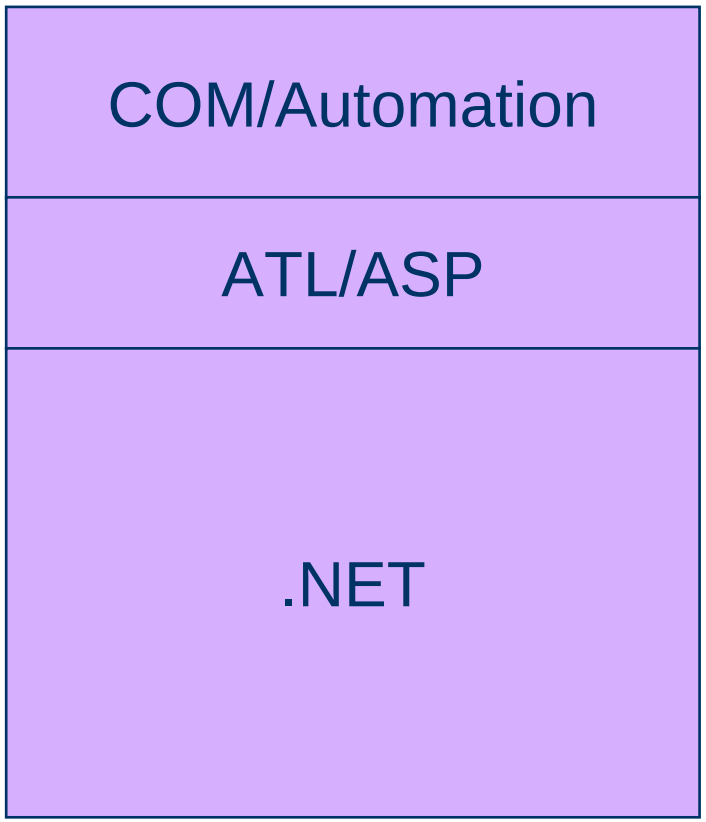


Inhalte - Überblick

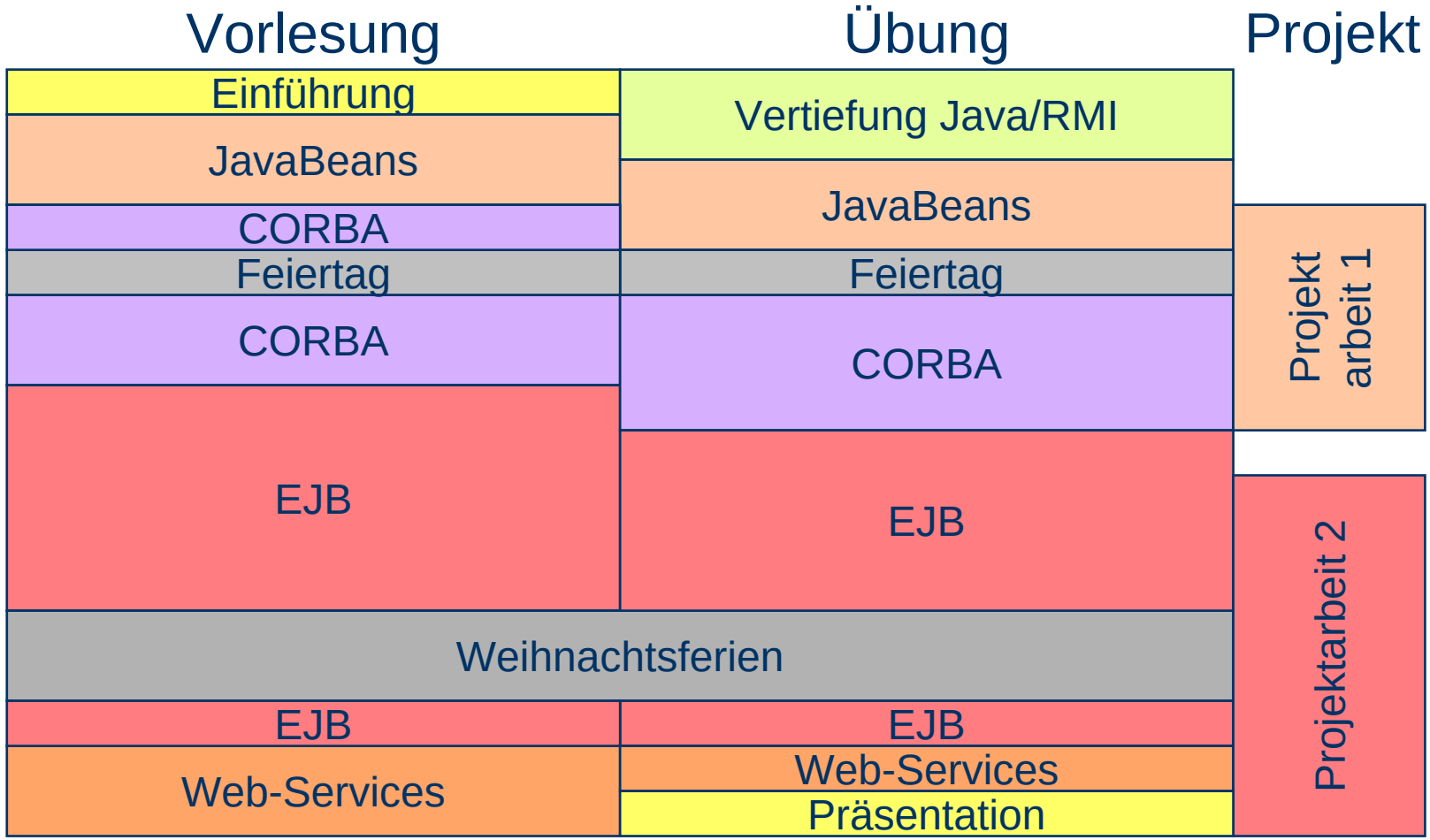
- Wintersemester



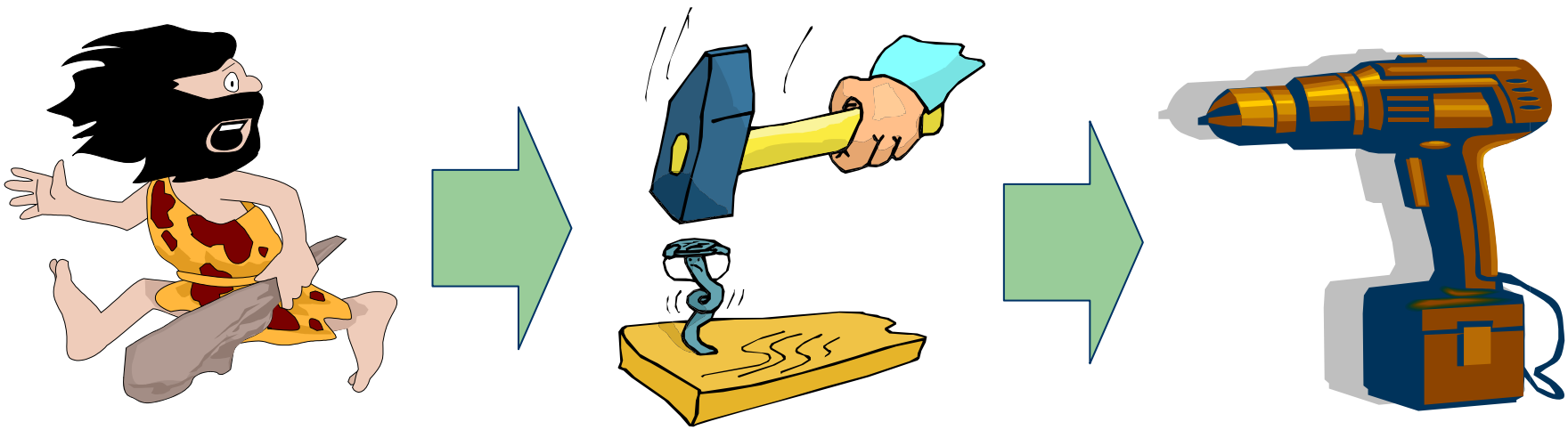
- Sommersemester



Inhalte: Vorläufiger Plan für das WS



Wozu Komponenten?



Traditionelle SW-Entwicklung

Objektorientierte SW-Entwicklung

Komponenten-orientierte SW-Entwicklung

Historische Entwicklung (1)

- 40er-Jahre
  - Maschinensprache.
- 50er-Jahre
  - Teure Computer (~ $ 2.000.000.000),
  - Billige Entwickler,
  - Effizientere Nutzung des Computers:
    - Betriebssystem,
    - Produktivität des Entwicklers war sekundär.
  - Symbolischer Assembler,
  - „Erfindung“ von Unterprogrammen.

Historische Entwicklung (2)

- 60er-Jahre
  - Hochsprachen: FORTRAN, LISP, BASIC, Simula.
  - Entwicklung komplexerer Softwaresysteme.
  - Vergrößerung der Projektteams.
  - Hochsprachen erlaubten keine „Black Boxes“.
- 70er- und 80er-Jahre
  - Software-Entwicklungskosten explodierten.
  - Viele Softwareprojekte scheiterten.
  - → **Softwarekrise.**
- 90er-Jahre:
  - Software-Komponenten.

Komponenten - Definition

- Definition Komponente (1)

  „A small binary object or program that performs a specific function and is designed in such a way to easily operate with other components and applications.“
  (www.webopedia.com)

- Definition Komponente (2)

  „A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to compositions by third parties.“ (C. Szyperski)

Komponenten - Anwendungsgebiete

- Baugewerbe
  - Baustoffe,
  - Ziegel, Fenster,
  - Küchen, Badezimmer.
- Elektronikindustrie
  - Widerstände, Dioden, Transistoren,
  - Integriere Schaltkreise,
  - Mikroprozessoren,
  - DVD-Laufwerke, Motherboards, Karten.
- Die Verwendung von Komponenten ist in diesen Bereichen absolut selbstverständlich!

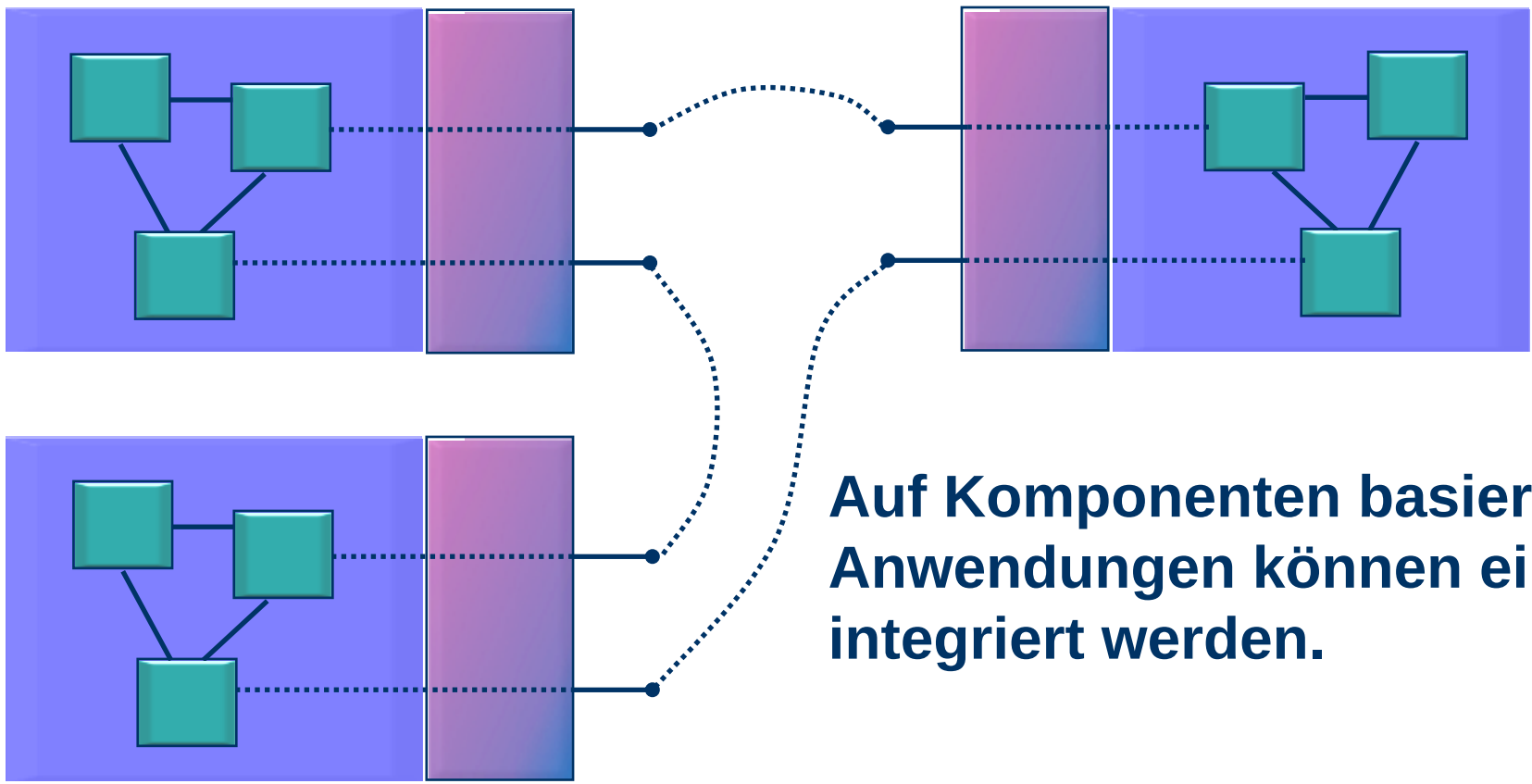
Markt für Komponenten

- Fragestellung in traditionellen Industrien
  - Gibt es bereits eine derartige Komponente?
  - Wo kann ich diese Komponente kaufen?
  - Wo ist die Komponente am billigsten?
  - Wie muss ich Anforderungen ändern, damit eine Standardkomponente eingesetzt werden kann?
- Voraussetzung für Entwicklung eines Marktes: **Standards**
  - Großes Angebot an Standard- und anwendungs-spezifischen Komponenten.
  - Fördern Konkurrenz zwischen Anbietern.

Monolithische Anwendungen

Monolithische Anwendungen sind in sich abgeschlossen und sind kaum ineinander integriert.

Komponenten-basierte Anwendungen



Auf Komponenten basierte Anwendungen können einfach integriert werden.

Abgrenzung zu C++/Java-Klassen

- C++-Klassen
  - C/C++-Laufzeitumgebung ist nicht standardisiert.
  - Beschreibung der Schnittstelle (Header-File) ist sprachabhängig und nicht Teil der binären Repräsentation.
  - Viele Änderungen in der Klasse bedingen Neuübersetzung (Hinzufügen einer Datenkomponente).
  - Komplexe Abhängigkeiten mit anderen Klassen.
- Java-Klassen
  - Entsprechen mehr dem Komponenten-Konzept.
  - Integration in Nicht-Java-Anwendungen ist schwierig.
  - Keine Unterstützung zur Design-Zeit.

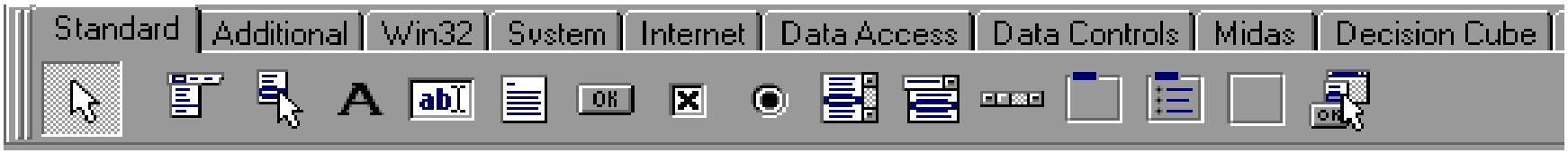
Standards für Software-Komponenten

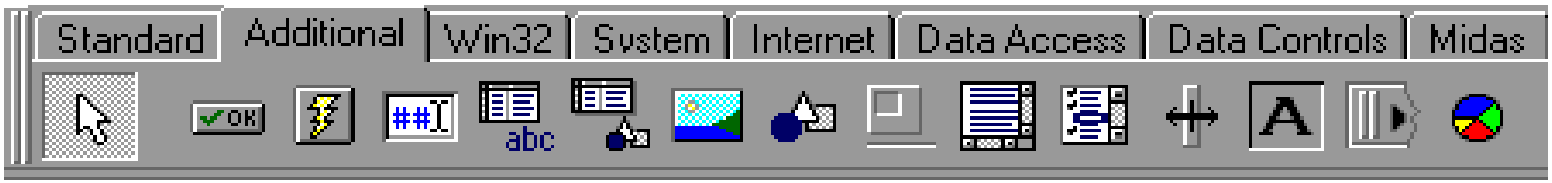
- Microsoft COM
  - *Builder Tool:* Visual Basic, C++, Java, Office Tools
  - *Komponente:* VBX, OCX, ActiveX-Control
- Microsoft .NET
  - *Builder Tool:* Visual Studio .NET
  - *Komponente:* .NET-Komponente (in beliebiger .NET-Sprache)
- Borland/Inprise:
  - *Builder Tool:* Delphi, C++-Builder
  - *Komponente:* Delphi-Komponente
- Sun/JavaSoft
  - *Builder Tool:* BeanBuilder, JBuilder, NetBeans, Sun ONE Studio (Forte), VisualAge, Code Warrior
  - *Komponente:* JavaBean

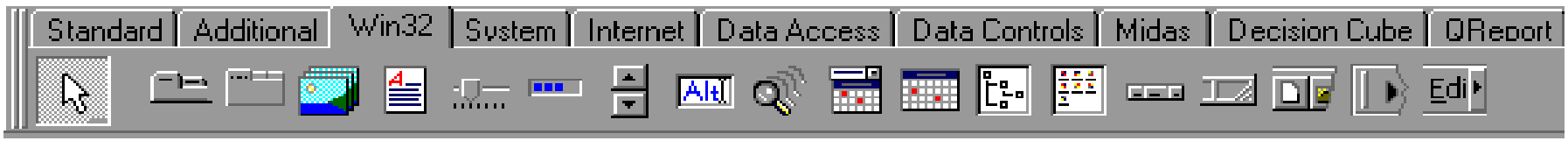
Komponenten-Technologien: Gemeinsamkeiten

- Komponenten liegen in *binärer* Form vor.
- GUI-Design erfolgt grafisch mit *Builder Tool.*
- *Properties* definieren Verhalten und Aussehen.
- *Events* steuern die Interaktion mit anderen Komponenten.
- Properties und Events können bereits zur *Design-Zeit* verändert werden.

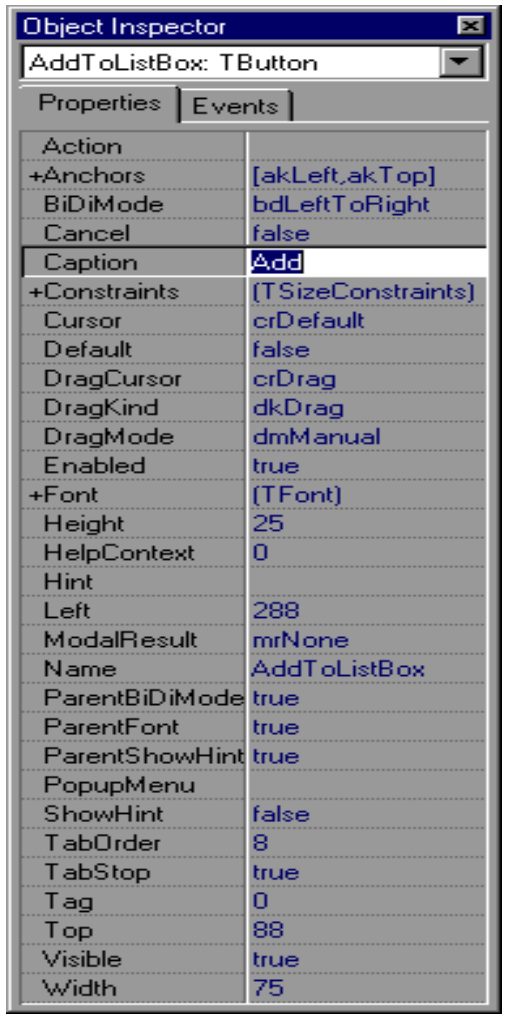
Delphi: Komponenten-Palette

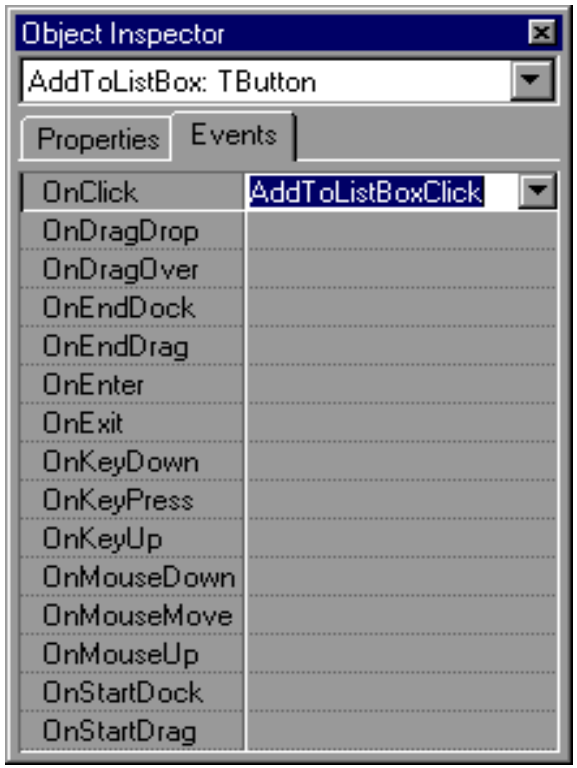




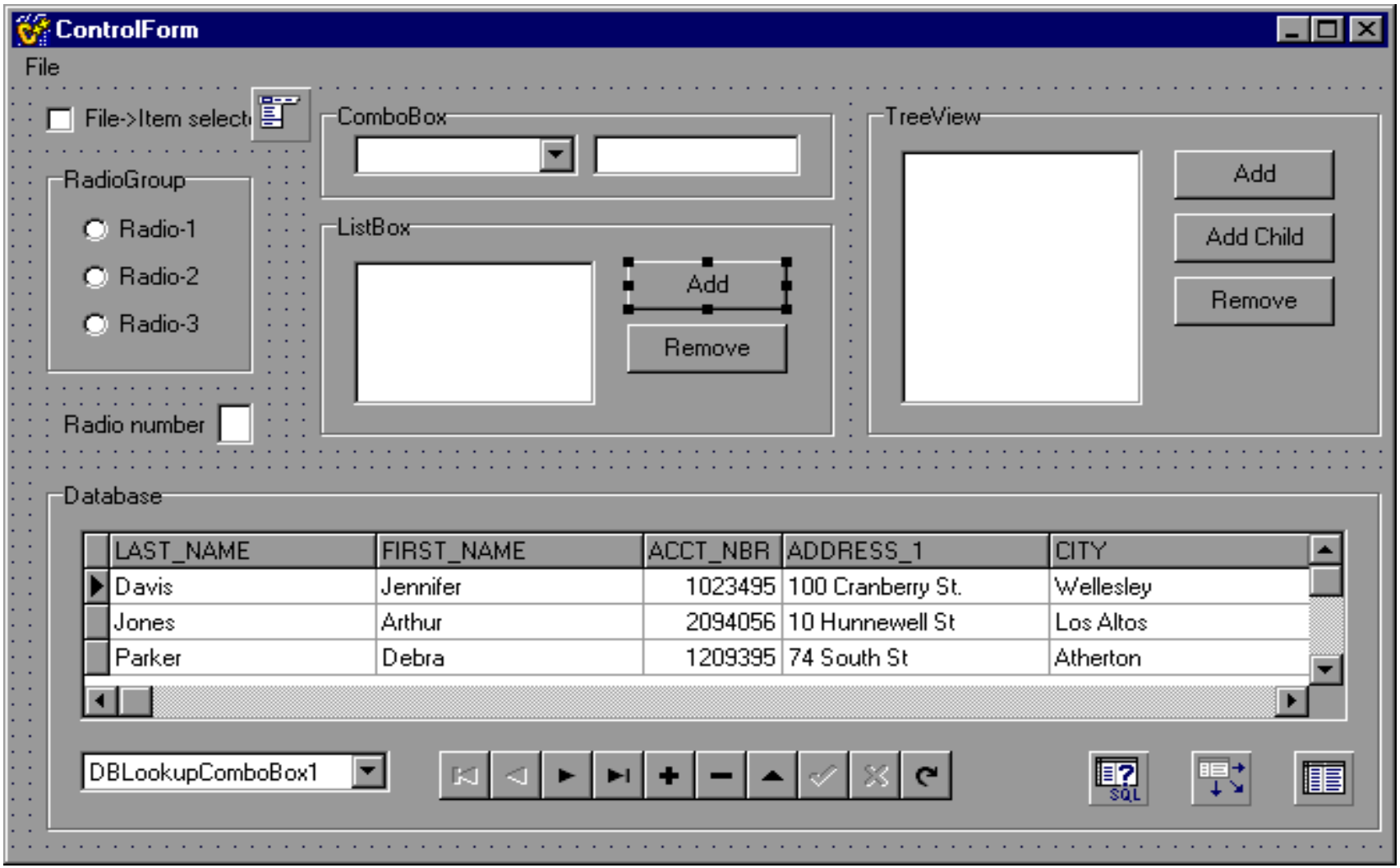


Delphi: Object Inspector

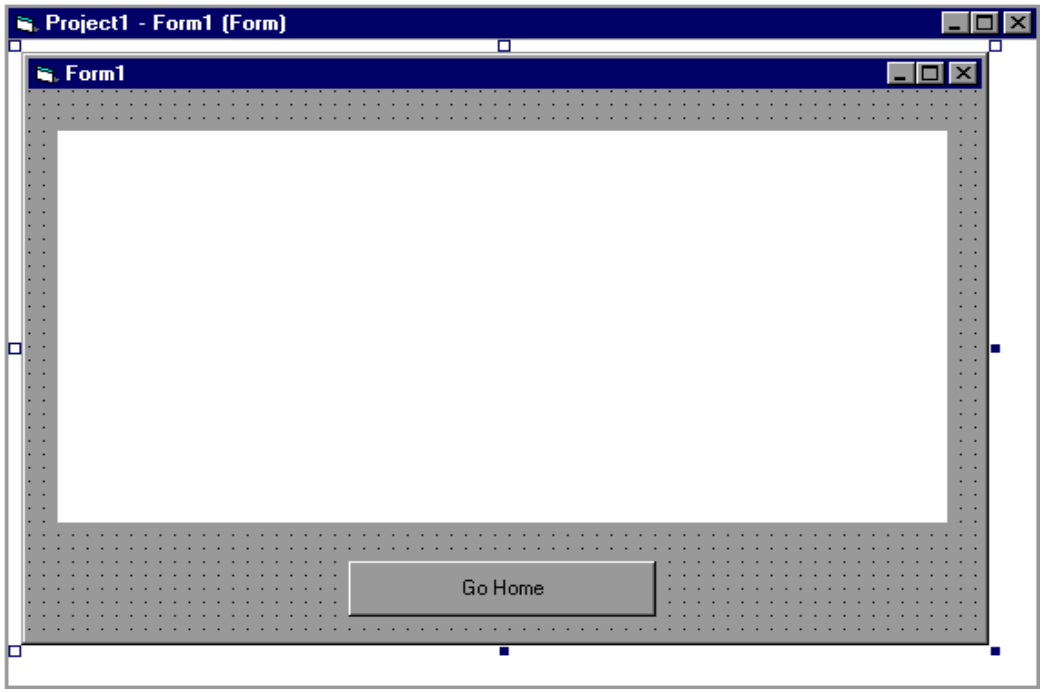


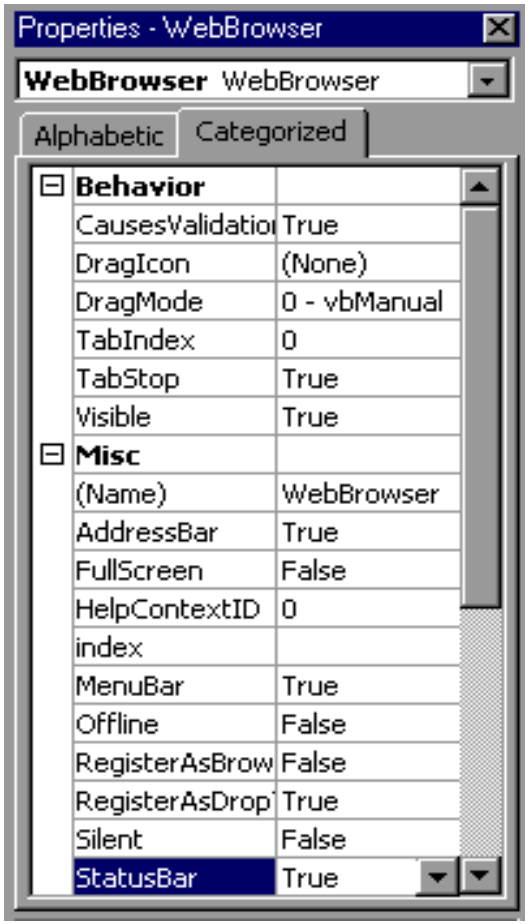


Delphi: GUI-Design

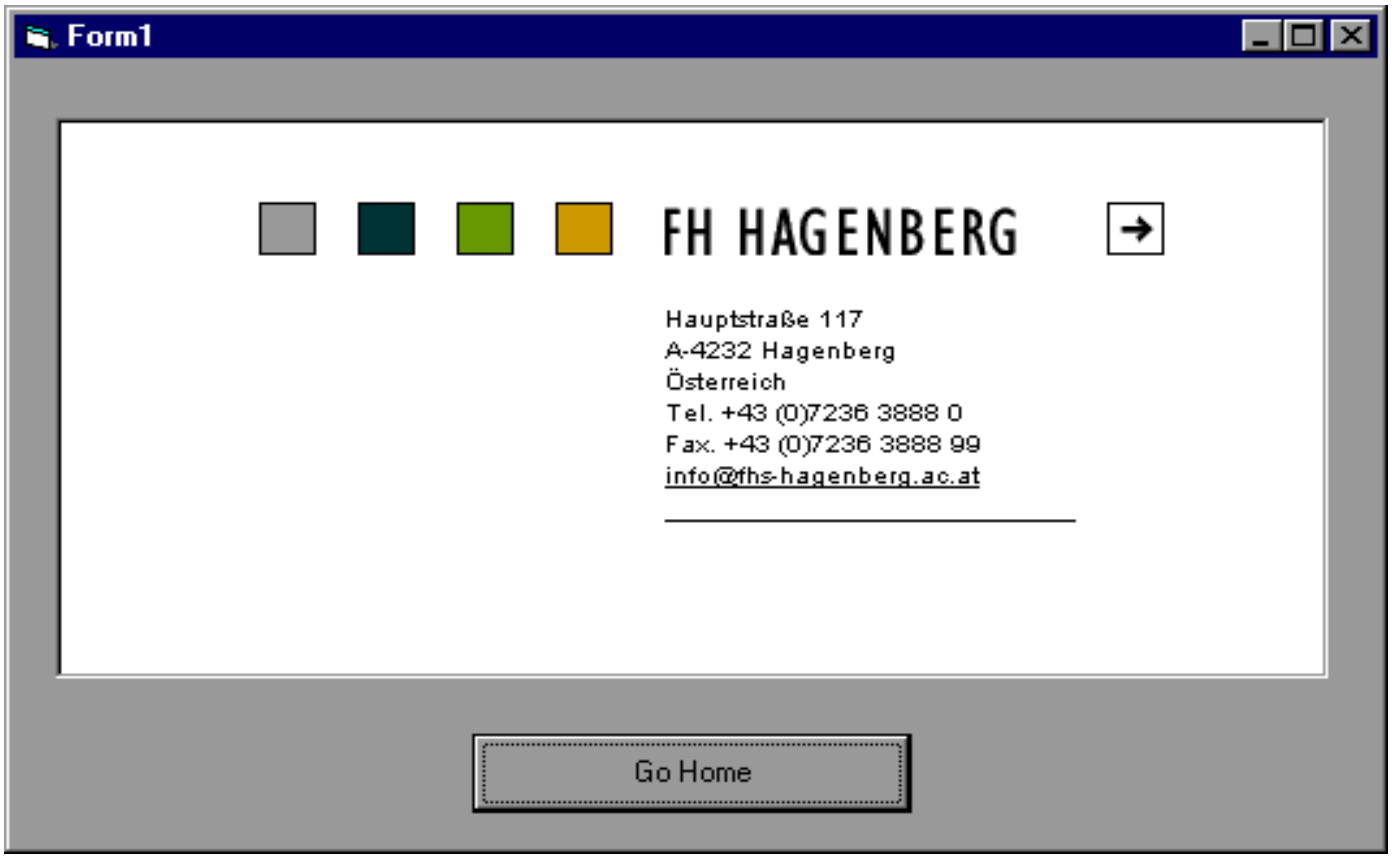


Visual Basic: Design Time

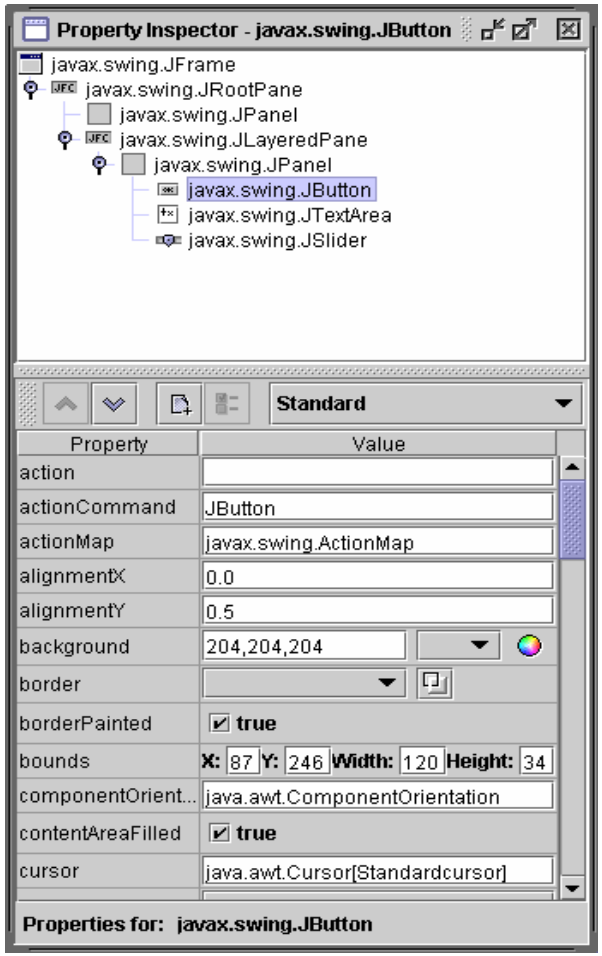


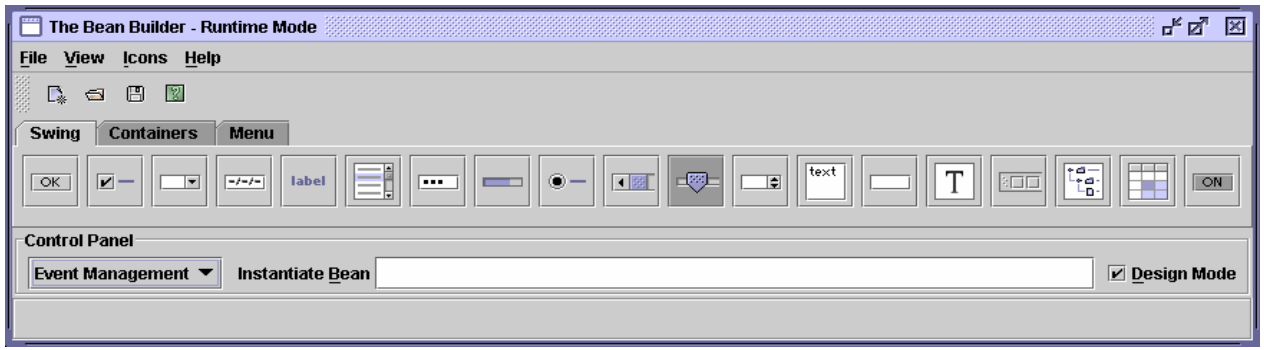


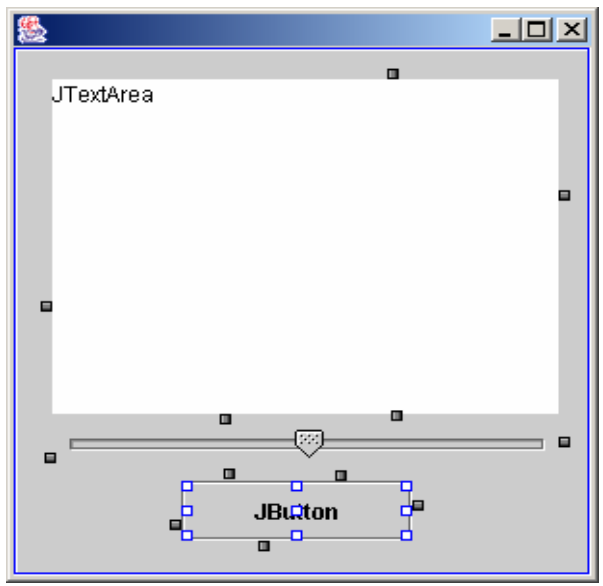
Visual Basic: Run Time



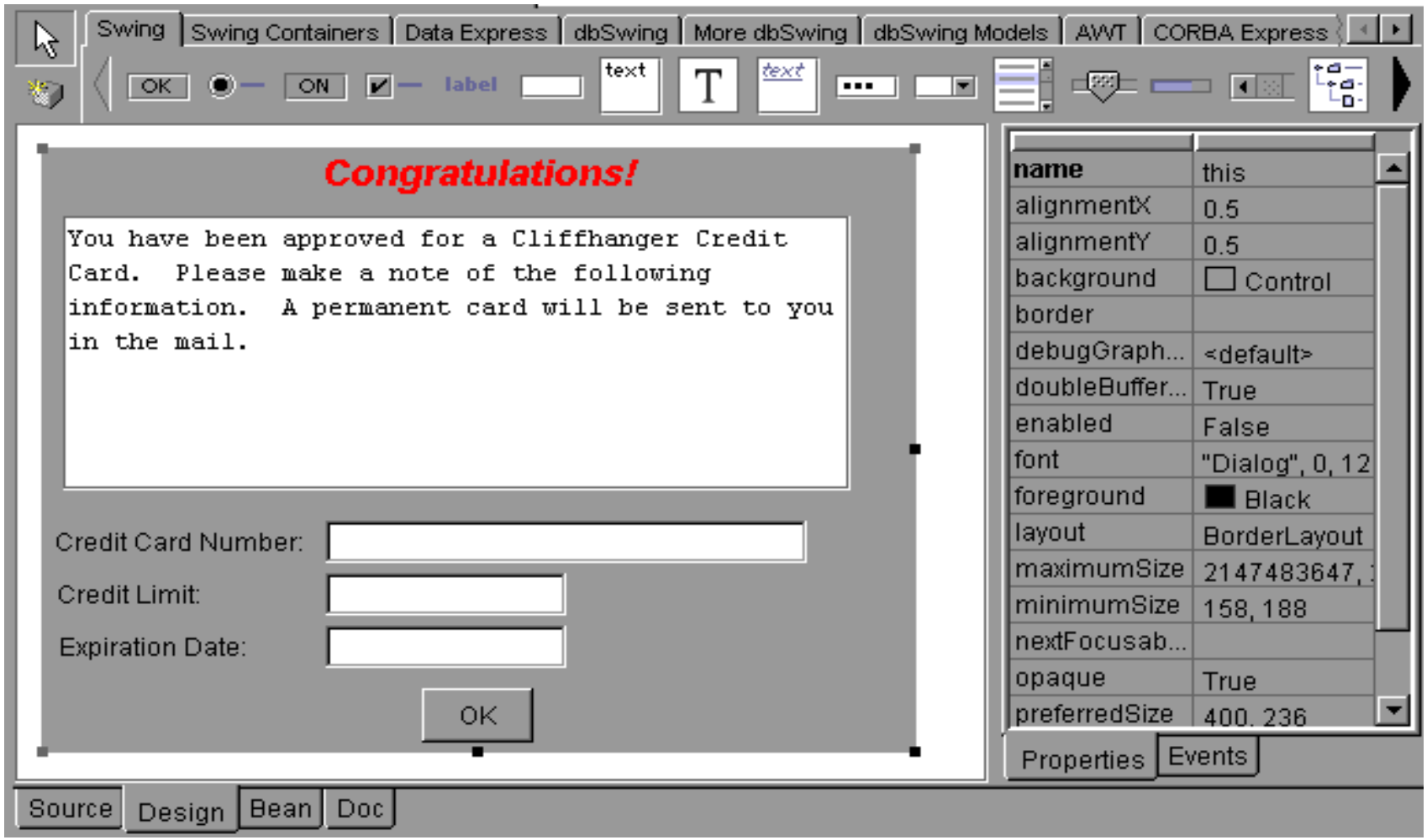
Bean Builder







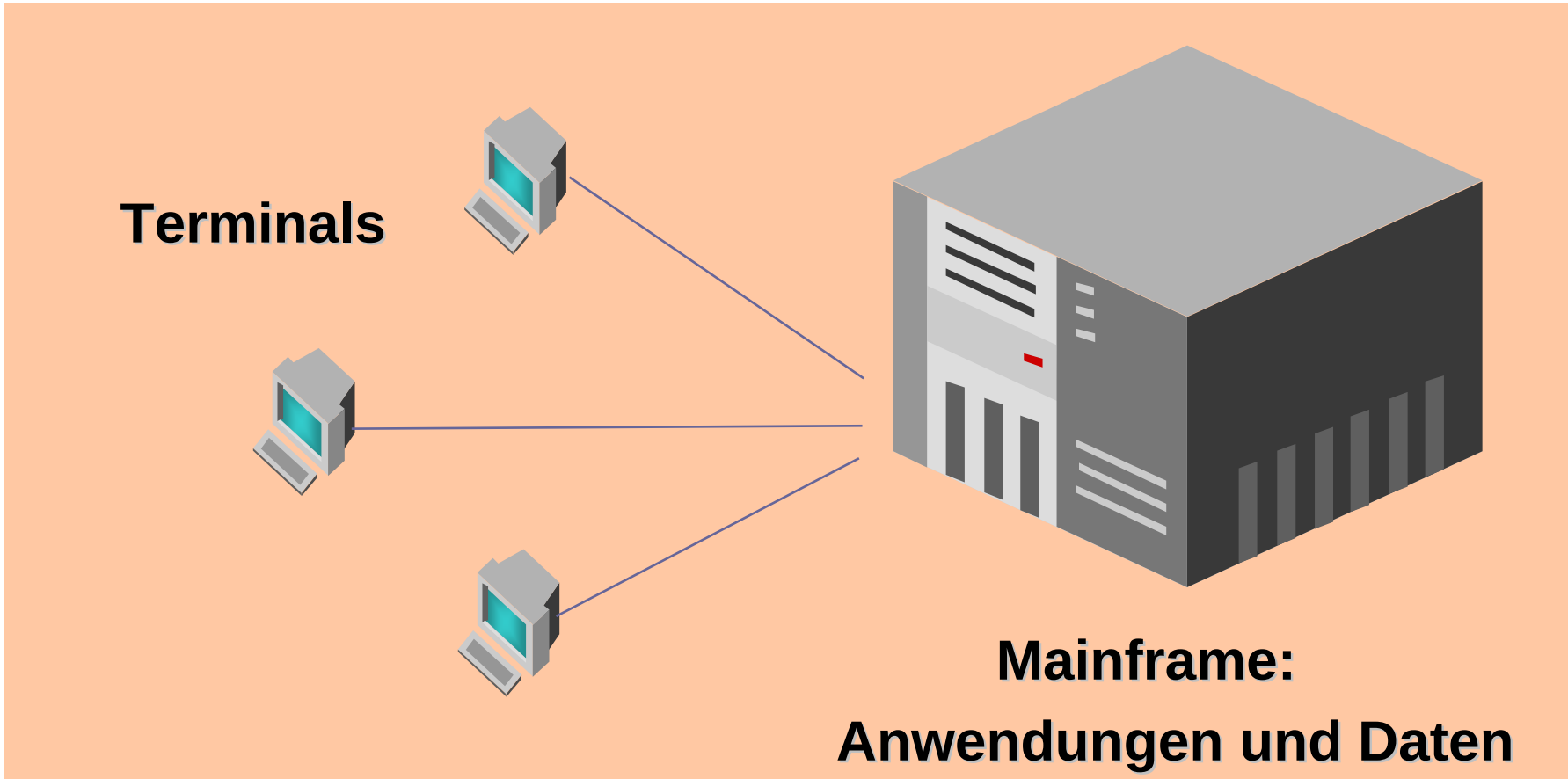
Borland JBuilder



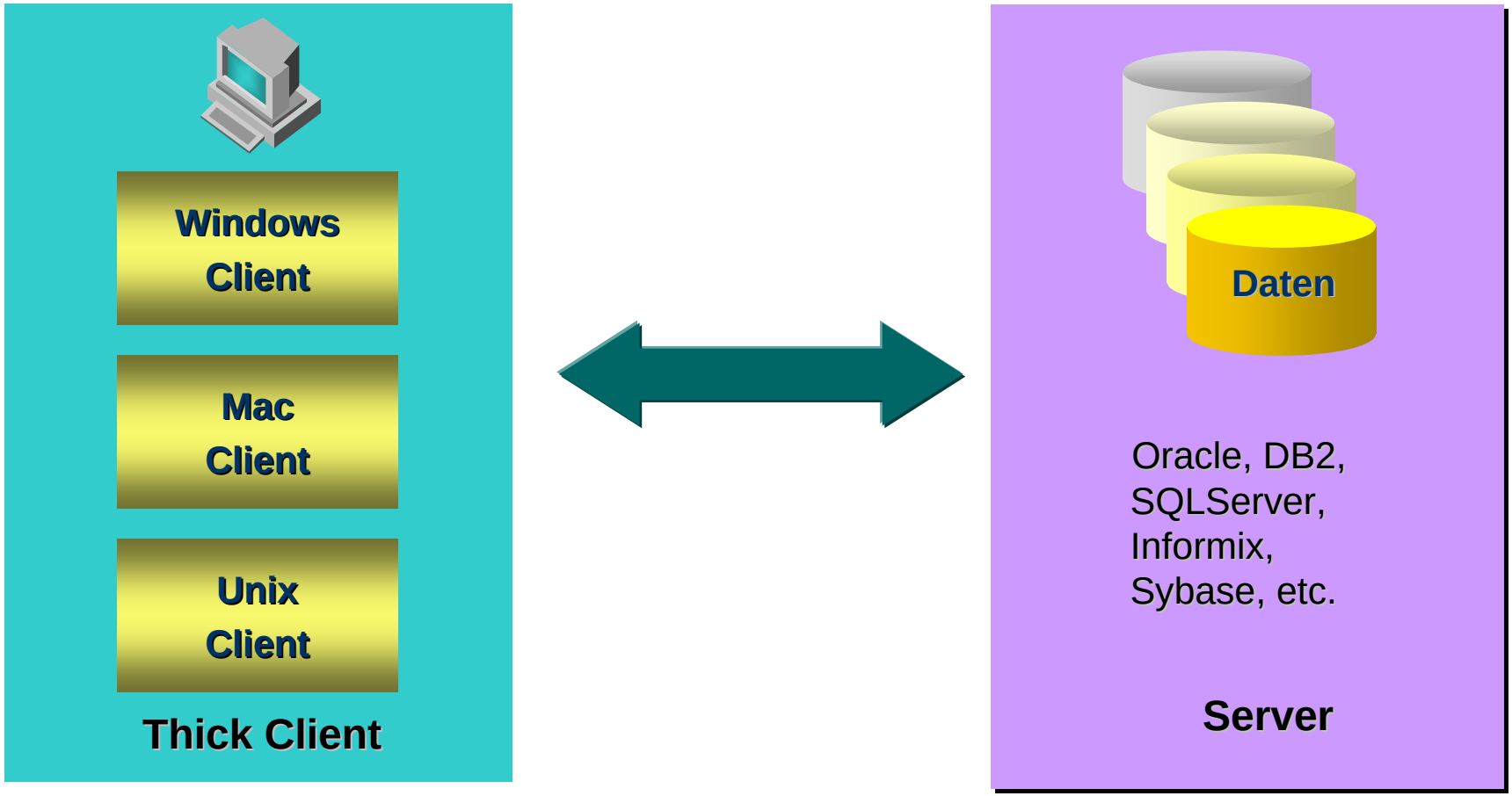
Enterprise-Anwendungen

- Anwendungen und Merkmale
  - komplexe (Business-)Logik,
  - umfangreiche, komplex strukturierten Daten,
  - besondere Sicherheitsanforderungen,
  - Transaktionen stehen im Mittelpunkt,
  - viele Clients auf verschieden Plattformen,
  - Verteilung auf verschiedene Plattformen: UNIX, AS/400, Windows, Macintosh,
  - Heterogene Entwicklungsumgebung: C, C++, Java, COBOL, Basic, Perl, Smalltalk.

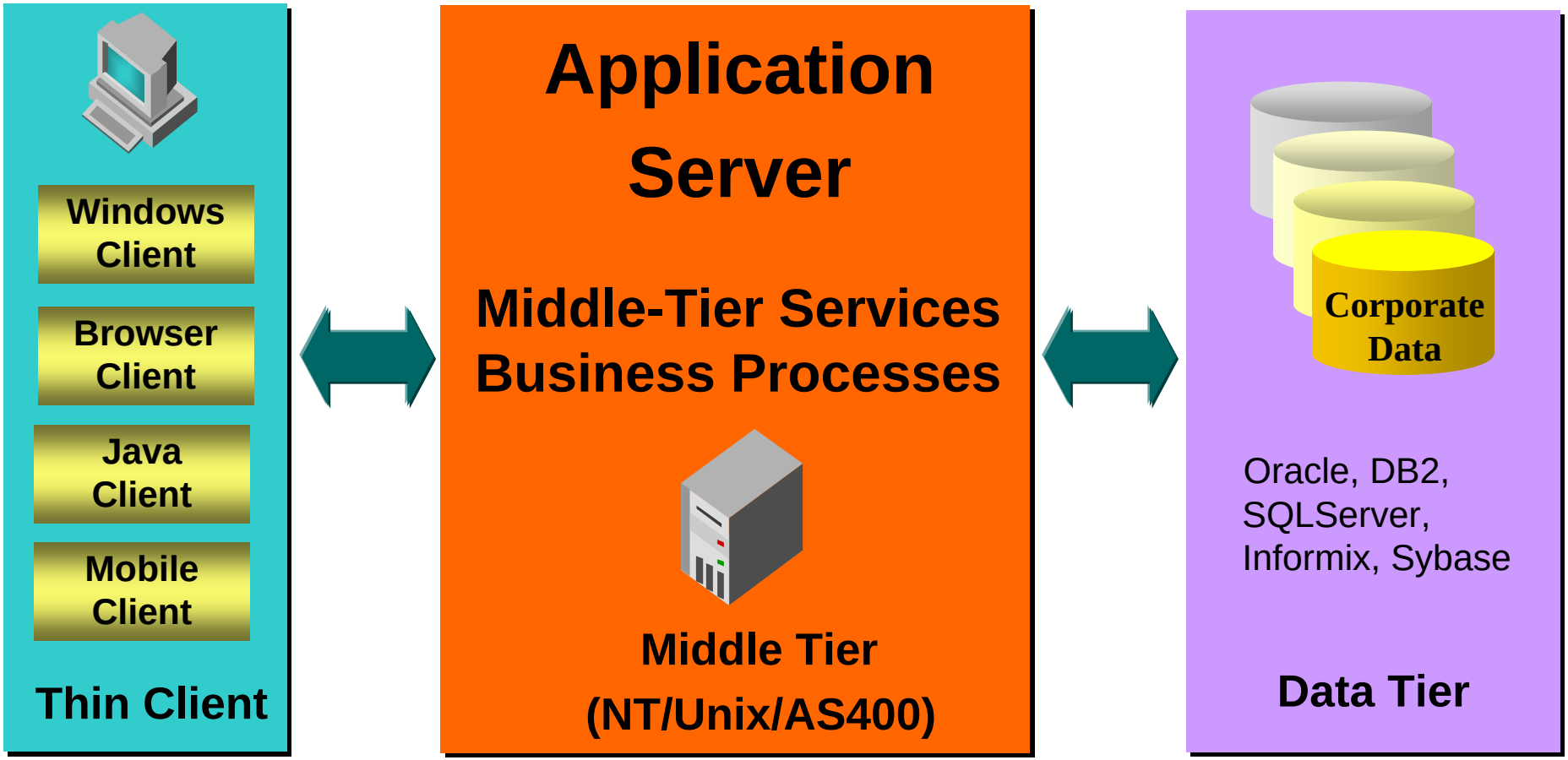
Mainframe-Anwendungen (1)



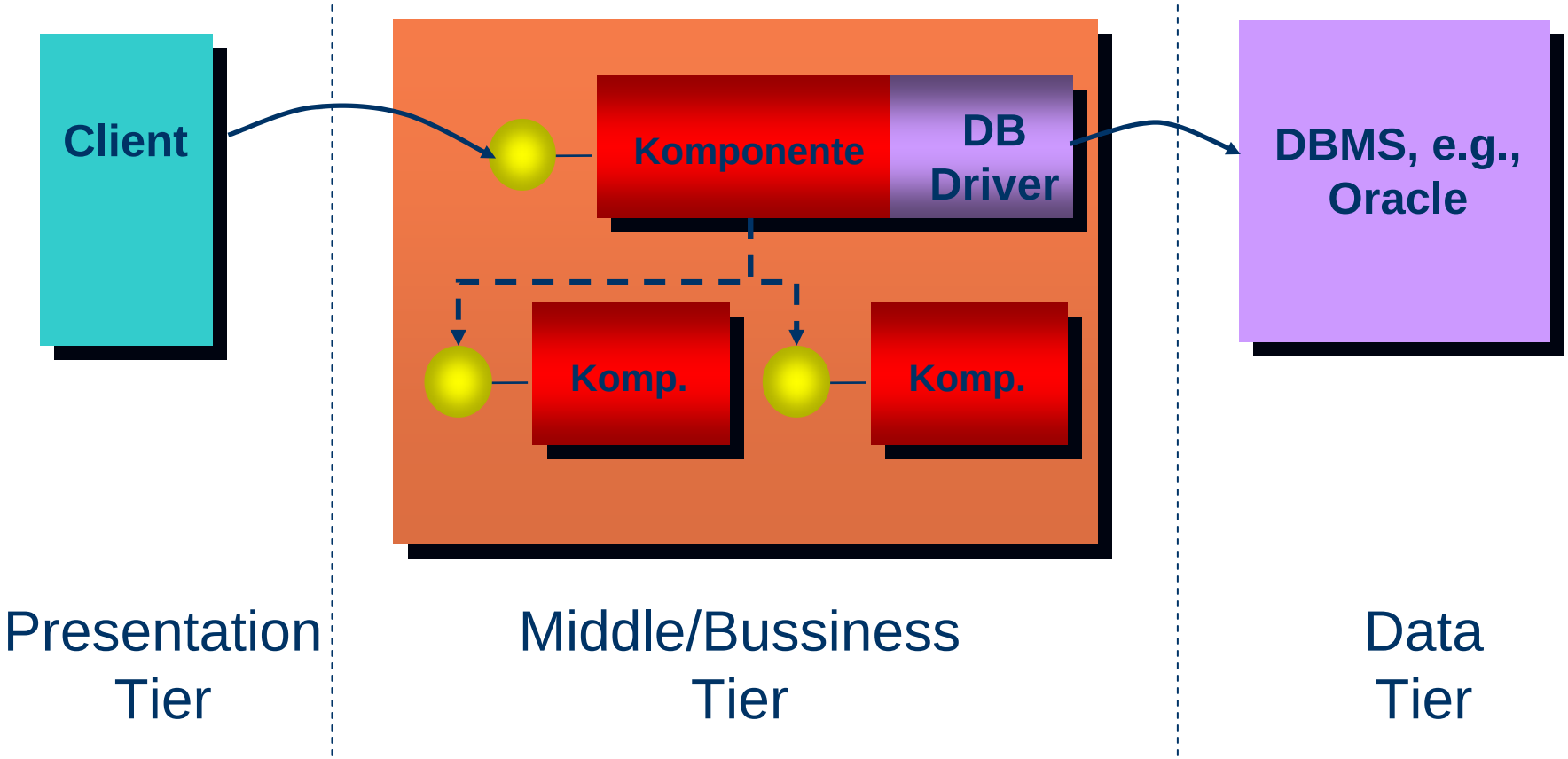
Client/Server-Anwendungen



Mehrschicht-Anwendungen

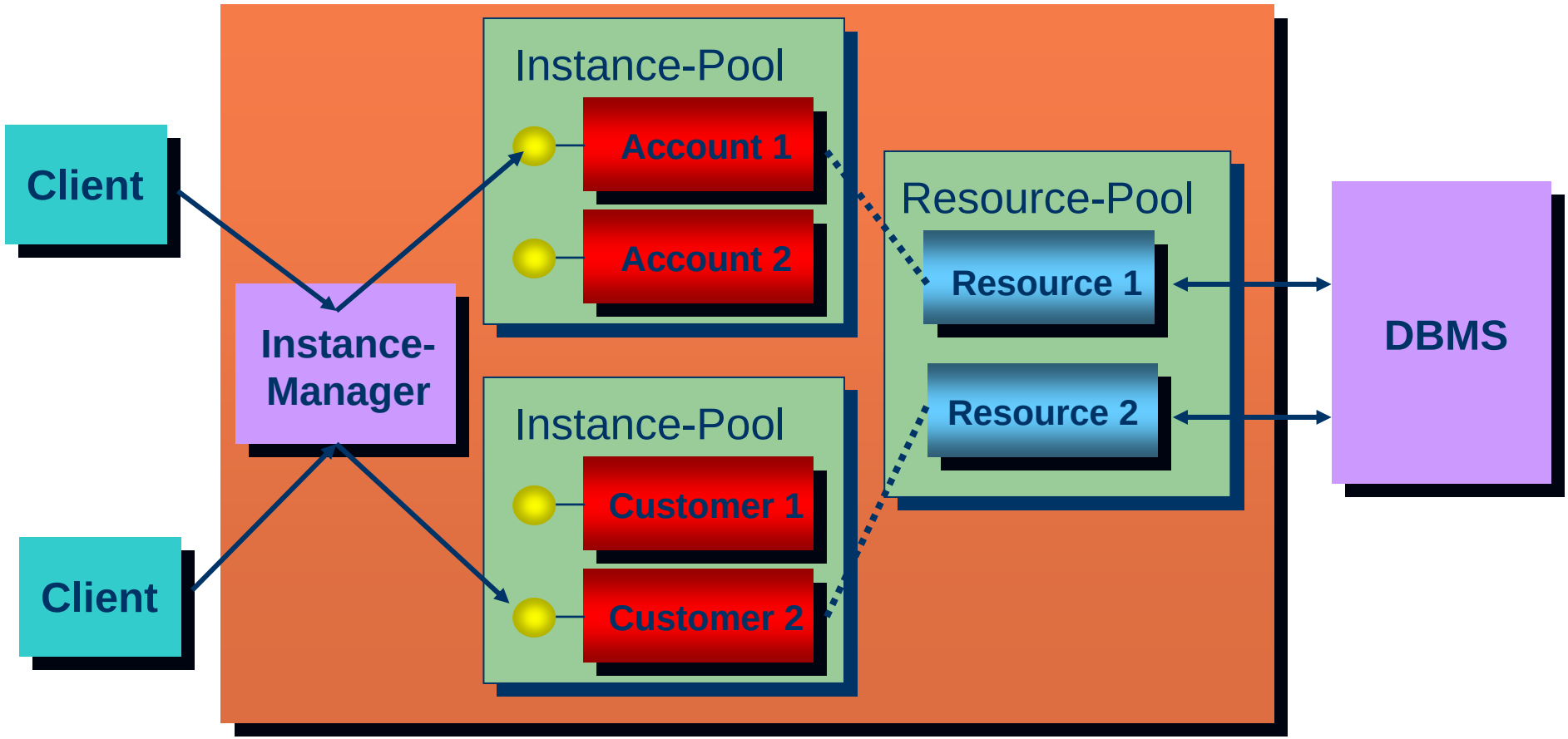


Mehrschicht-Architekturen (1)



Mehrschicht-Architekturen (2)

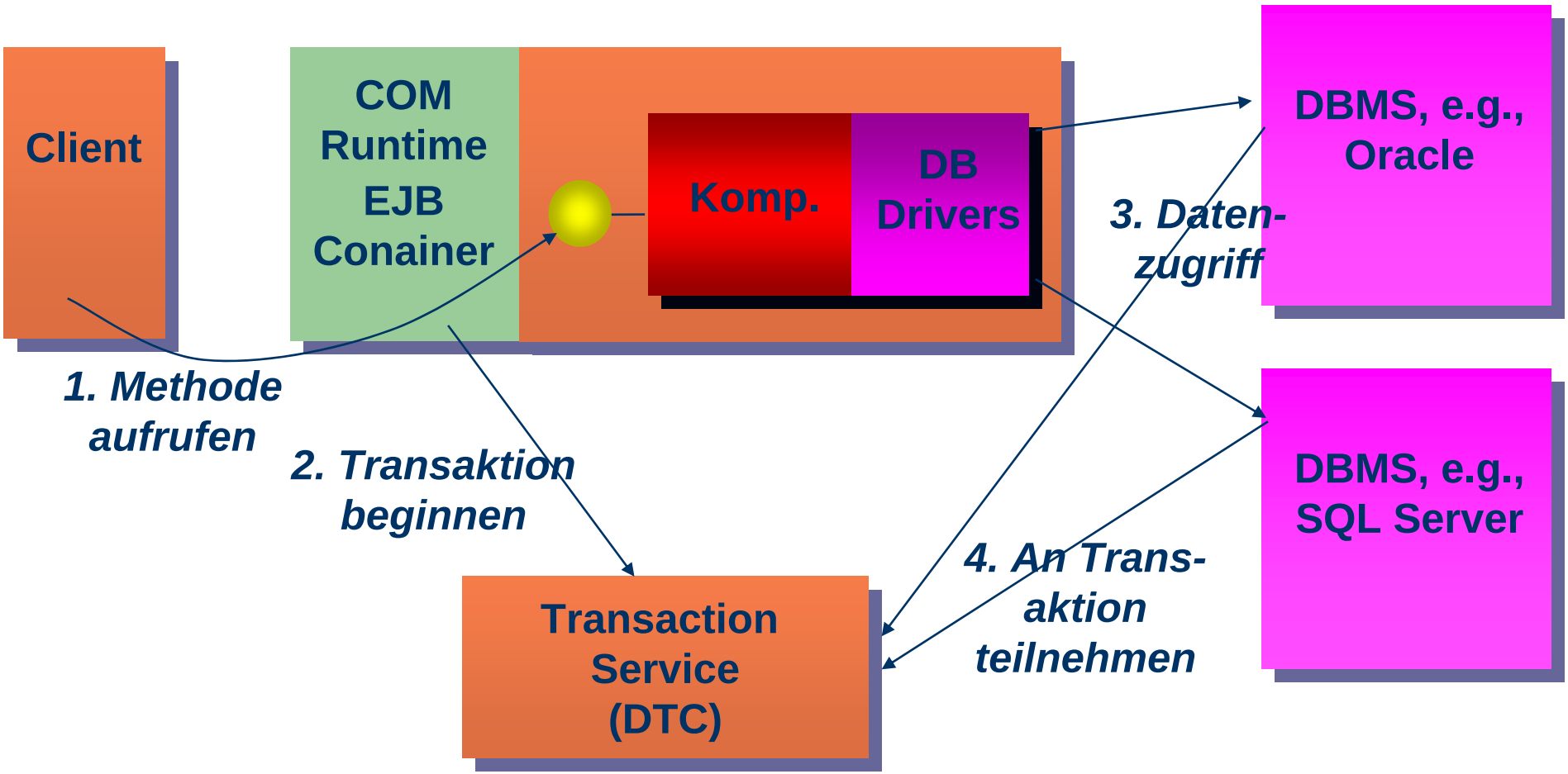
Instance- and Resource-Pooling



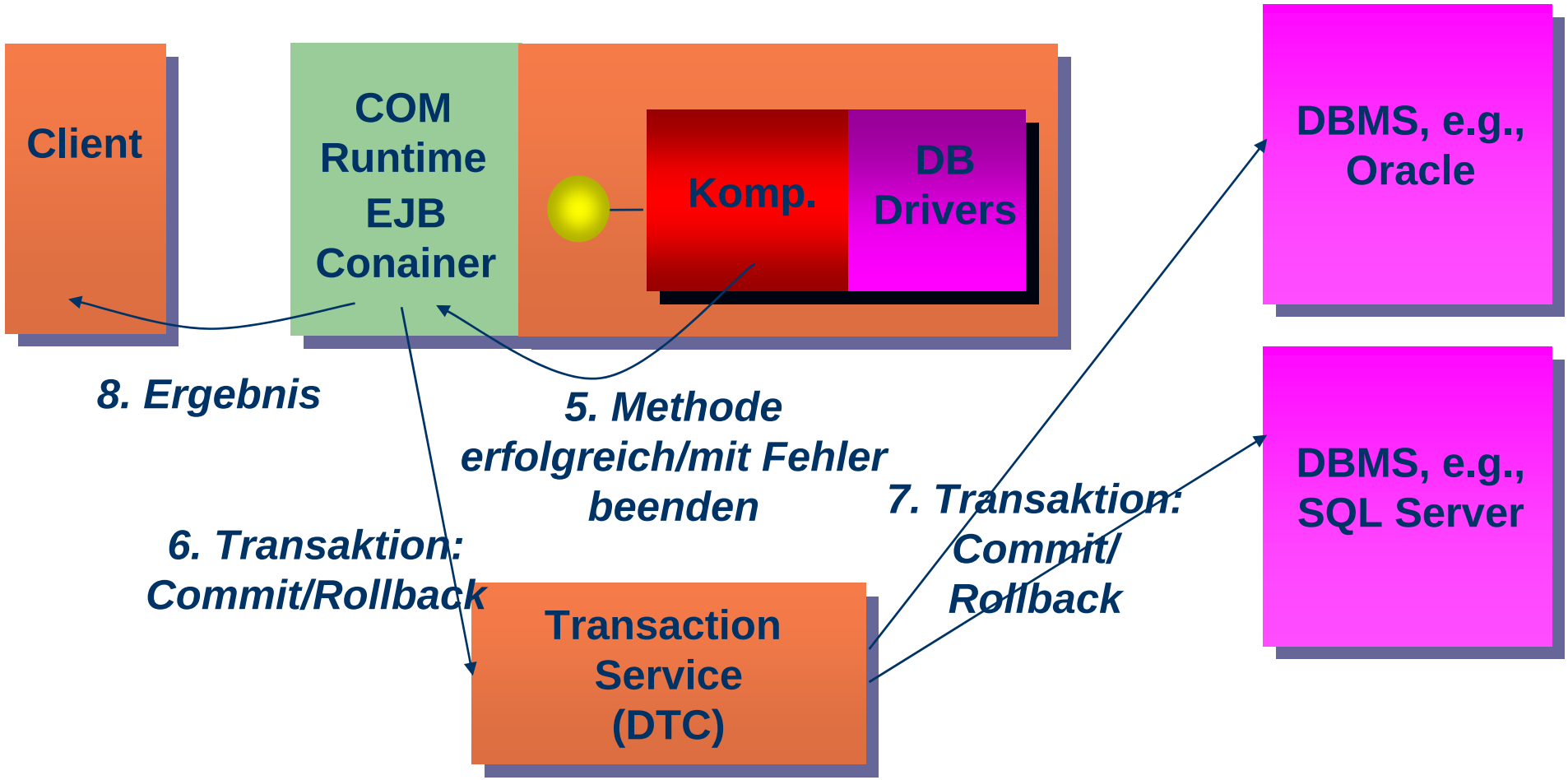
Enterprise-Architekturen

- Big Players
  - Windows DNA (COM+) bzw. .NET Enterprise Services
  - Enterprise Java Beans (EJB)
- Merkmale
  - Viele Clients können bedient werden.
  - Einfache Implementierung von Transaktionen.
  - Verwaltung des Zustands von Objekten.
  - Authentifizierung/Autorisierung ist einfach.
  - Resource Sharing/Pooling.
  - Erleichtert Entwicklung von Anwendungen mit mehreren Threads.

COM+/EJB Architektur (1)



COM+/EJB Architektur (2)



Architekturen für verteilte Anwendungen

- COM/DCOM
  .NET (SOAP)
- CORBA
- Java RMI,
  RMI-IIOP