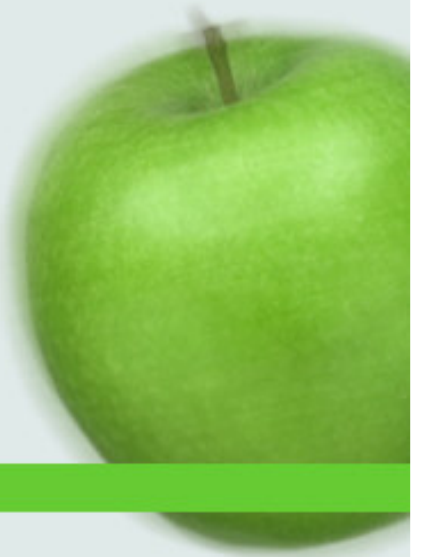




Der Lösungspartner für die mittelständische Fertigungsindustrie



## Software Engineering in der Praxis

Joachim Hertel & Christoph G. Jung



## Etwas Polemik zur Einstimmung

- Theorie ist ...
  - ... wenn alle wissen, wie es geht und es geht nicht.
- Praxis ist ...
  - ... wenn es geht, und keiner weiss, warum.

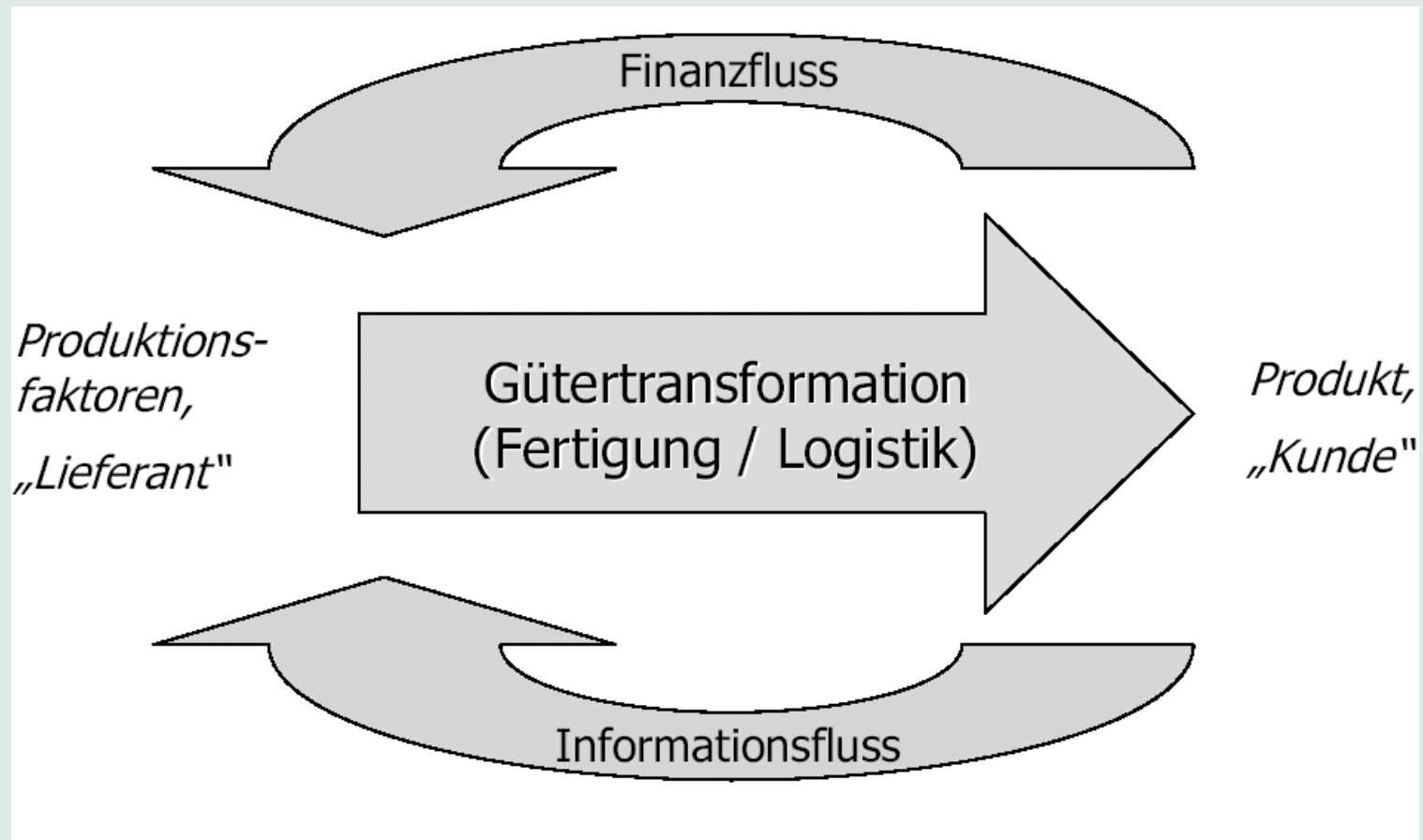


## Übersicht

- *infor AG*
  - Betriebswirtschaftliche Standard-Software mit hoher Anpassungsfähigkeit.
- *Software Reengineering*
  - Moderne Technologien und Entwicklungswerkzeuge in einem hybriden Software-Produkt.
- Java 2 Enterprise Edition (J2EE™)
  - Eine Software-Plattform für Geschäftslogik
- Geschäftskomponenten & Web Services
  - Die vernetzte Zukunft

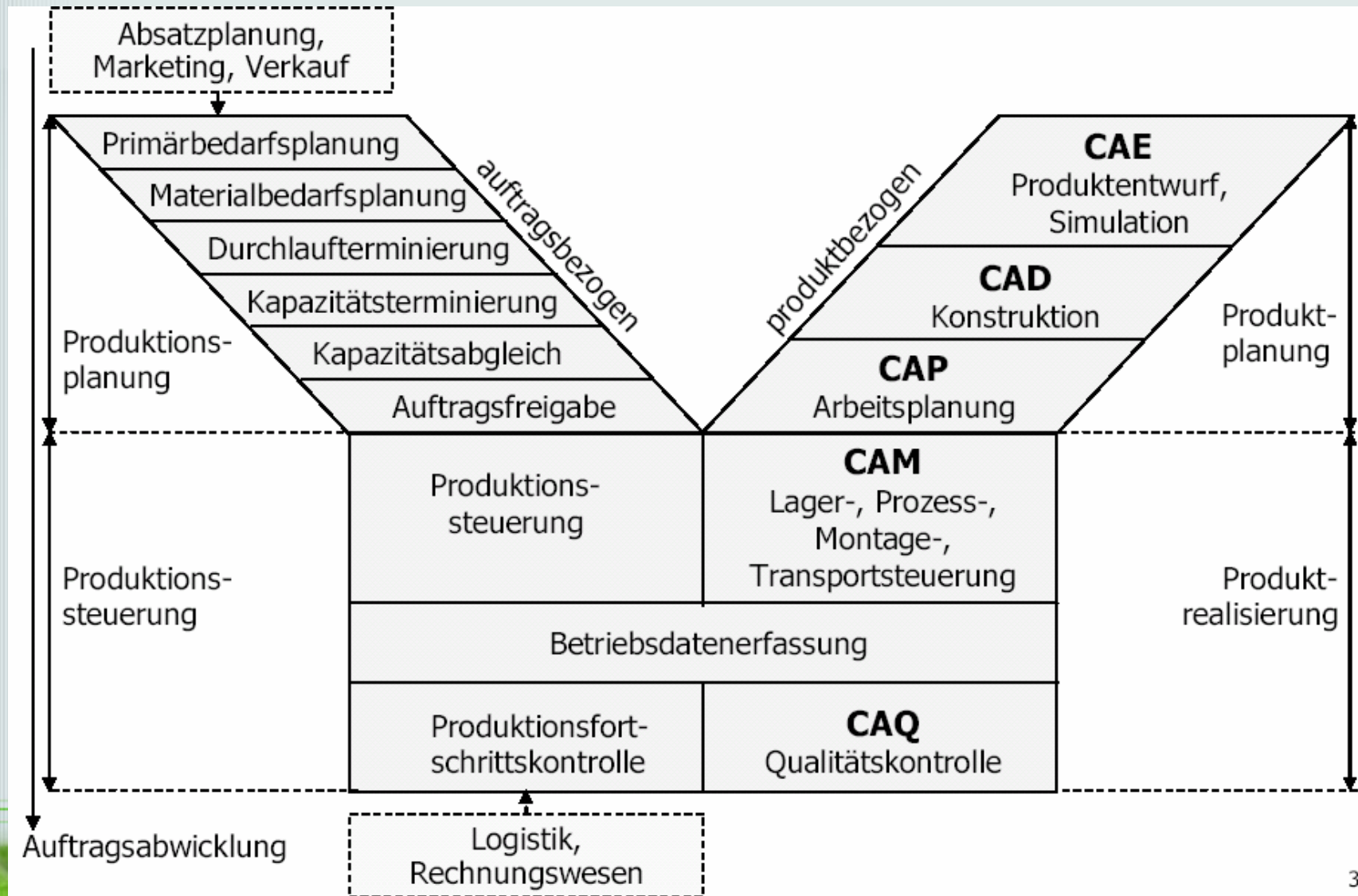


## (K)eine Einführung in die Wirtschaftsinformatik

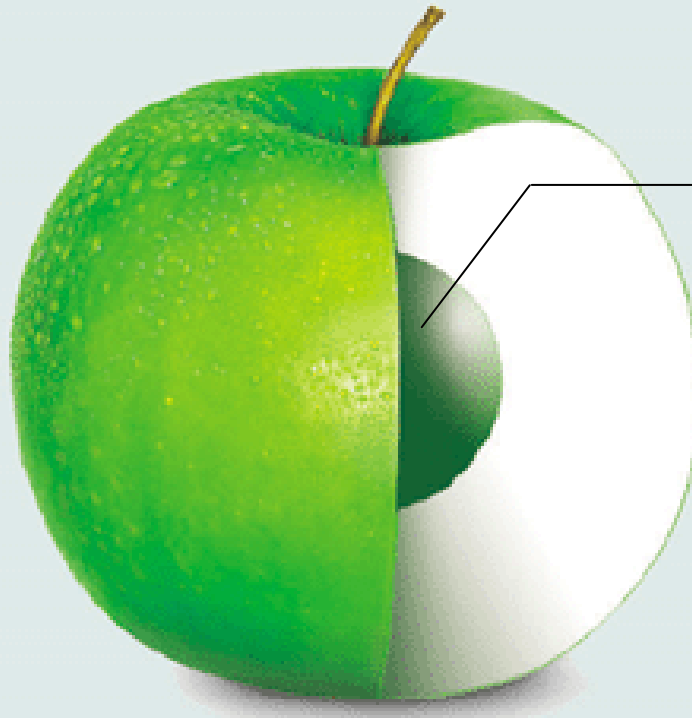


# Computer Integrated Manufacturing

alles läuft leichter!



## Die Business-Software infor:COM ist ...

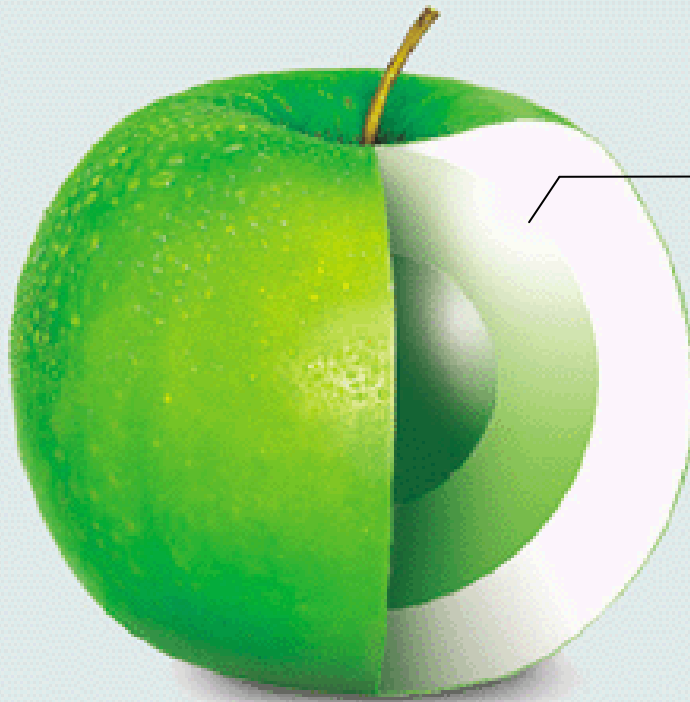


### Produktionsplanung & -steuerung (PPS)

- Fertigungssteuerung
- Material- und Kapazitätsplanung
- Advanced Planning & Scheduling
- BDE / PZE / MDE
- Disposition
- Vertrieb
- Einkauf
- Logistik



## Die Business-Software infor:COM ist ...

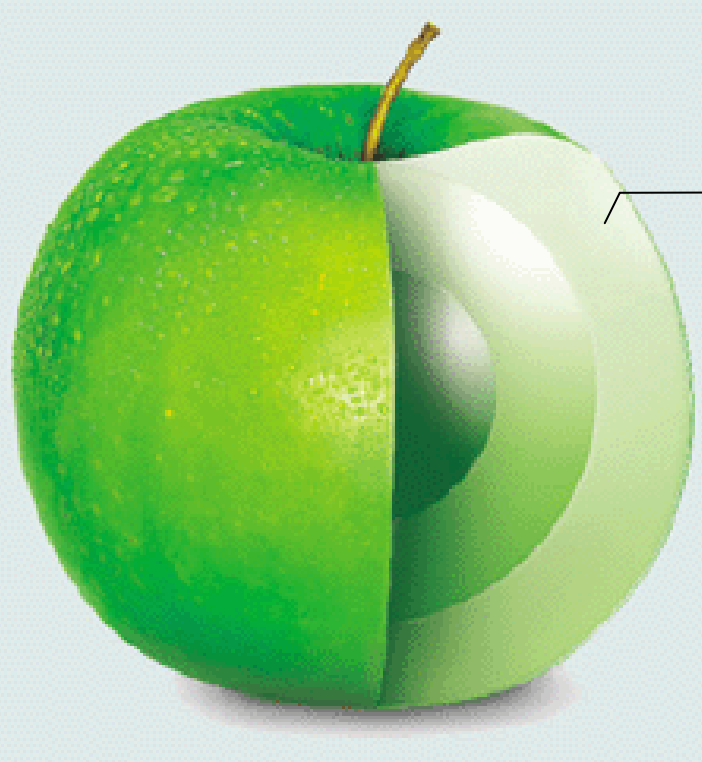


### Enterprise Resource Planning (ERP)

- **PPS +**
- Finanzbuchhaltung
- Anlagenbuchhaltung
- Kostenrechnung
- Personalwesen
- Controlling / MIS
- Projektmanagement
- Dokumentenmanagement

alles läuft leichter!

## Die Business-Software infor:COM ist ...



ERP II

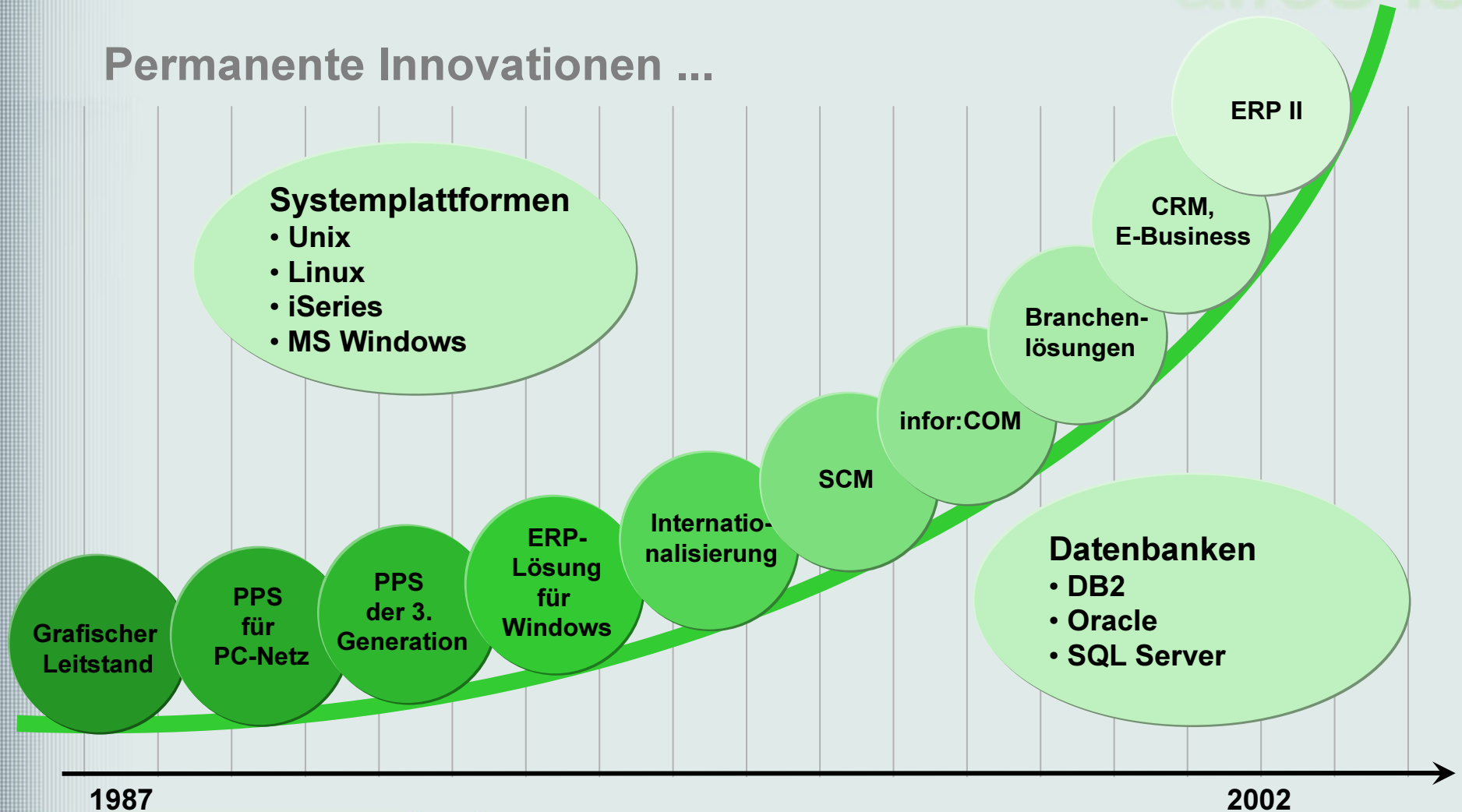
- **ERP +**
- e-Commerce
- e-Procurement
- Supply Chain Management
- Customer Relationship Management

... hochintegriert in einer Gesamtlösung.



alles läuft leichter!

## Permanente Innovationen ...



... zum Nutzen unserer Kunden.

alles läuft leichter!

## Fokus auf unterschiedliche Fertigungstypen und Branchen

- infor:COM für Einzel-, Serien- und Variantenfertiger



u.S.W...

Möbelindustrie

Bauindustrie

Kunststoffindustrie

Anlagenbau

Automotive

**Spezielle Lösungen für  
spezielle Märkte.**

## Über 3500 Kunden in ganz Europa ...



**... bestätigen unsere Erfahrung  
und Kompetenz.**



alles läuft leichter!

## Internationale Präsenz ...

- Deutschland
- Frankreich
- Großbritannien
- Irland
- Italien
- Niederlande
- Polen
- Schweiz
- Spanien
- Ungarn
- Österreich
- Rumänien
- Tschechische Republik
- Rußland
- Australien



## Rahmenbedingungen für praktisches Software Engineering

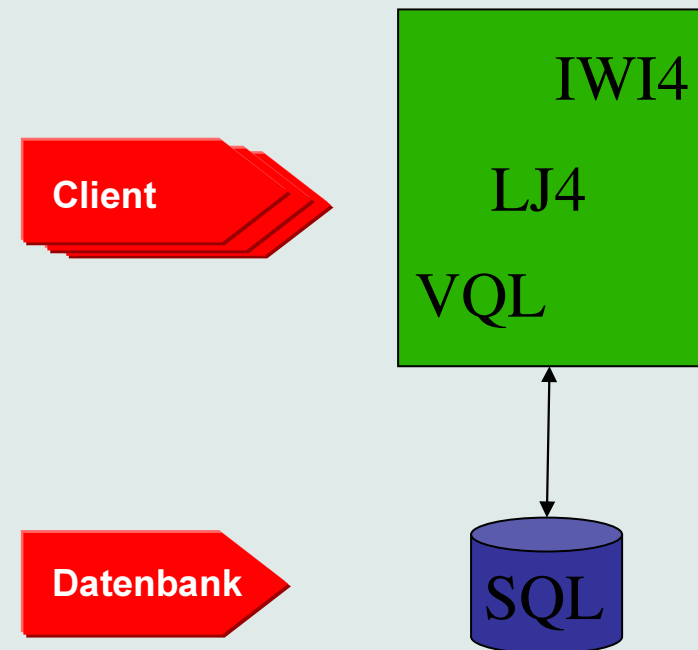
- Kunde steht im Vordergrund des Strebens, nicht die Technik:
  - Big-Bang Methode unpassend.
  - Produkt-Vision vor der Technologie-Vision
- Funktionale Zusicherungen müssen erhalten bleiben:
  - Komplexe Logik.
  - Riesige Datenbanken.
  - Adaptives Verhalten.
- Alle „Umwälzungen“, um neuen Anforderungen gerecht zu werden, haben einen Domino-Effekt:
  - Interne Dokumentation und Roll-Out.
  - Verteilte Standorte.
  - Formalisierte Entwicklung.
- „Historische“ Gegebenheiten müssen bedacht und evtl. migriert werden.



alles läuft leichter!

## infor:COM – Technische Fakten

- Two-Tier Architektur.
- Satz-orientiertes Datenmodell (VQL)
- Proprietäre Programmiersprache (LJ4)
  - Funktionaler Kern.
  - Keine Namensräume.
  - Globale Variablen.
  - Single-Threaded.
  - Ungetypte Handles.
  - Kein Exception-Handling
- Windows-GUI Bibliothek (IWI4).
- Moderner Entwicklungsprozess???
- Modellierung und Dokumentation???
- Skalierbarkeit???
- Internet und Web Clients???
- Plattformunabhängigkeit???
- BAPI???





Das klassische ERP-System ist ein Monolith

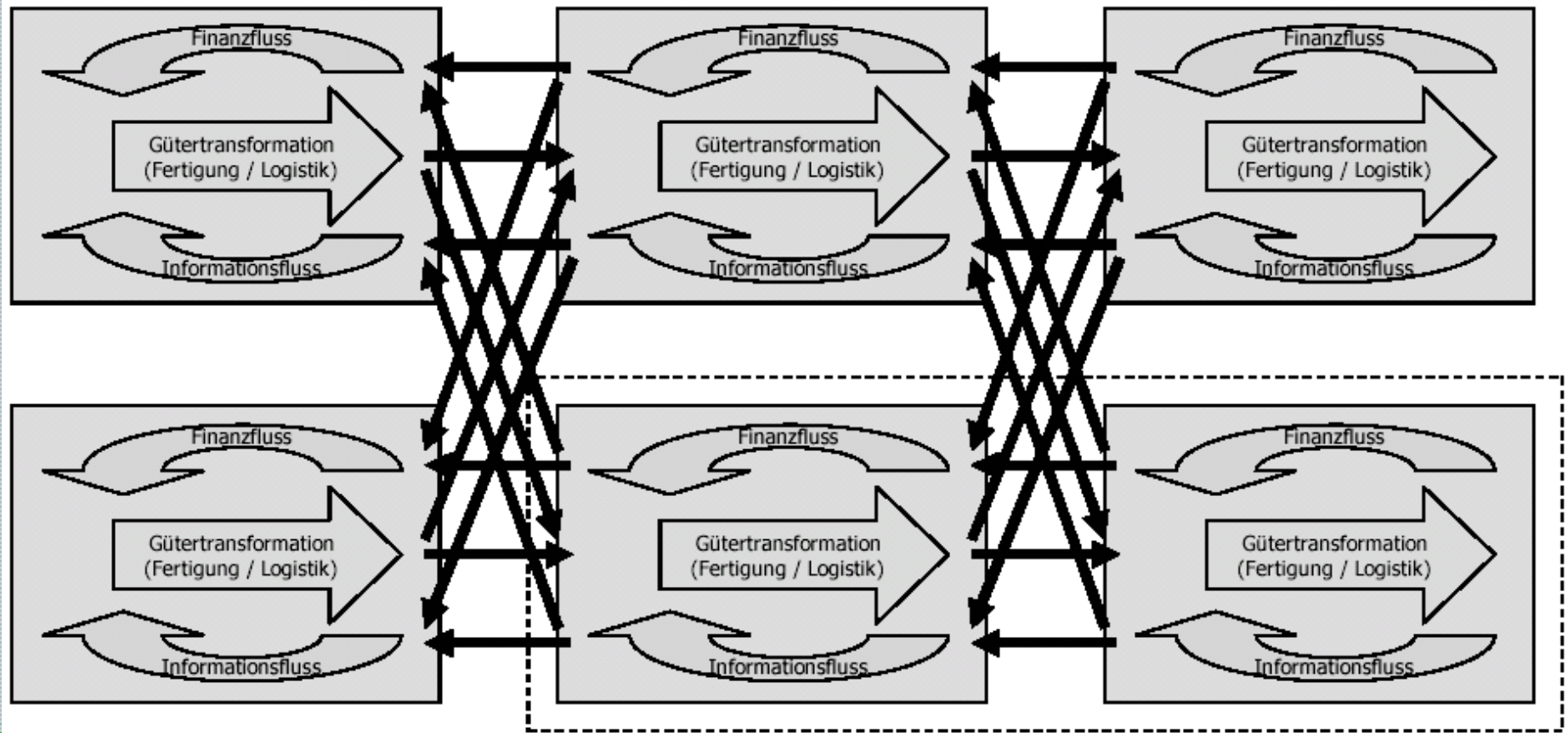
alles läuft leichter!



ERP  
=R.I.P.?

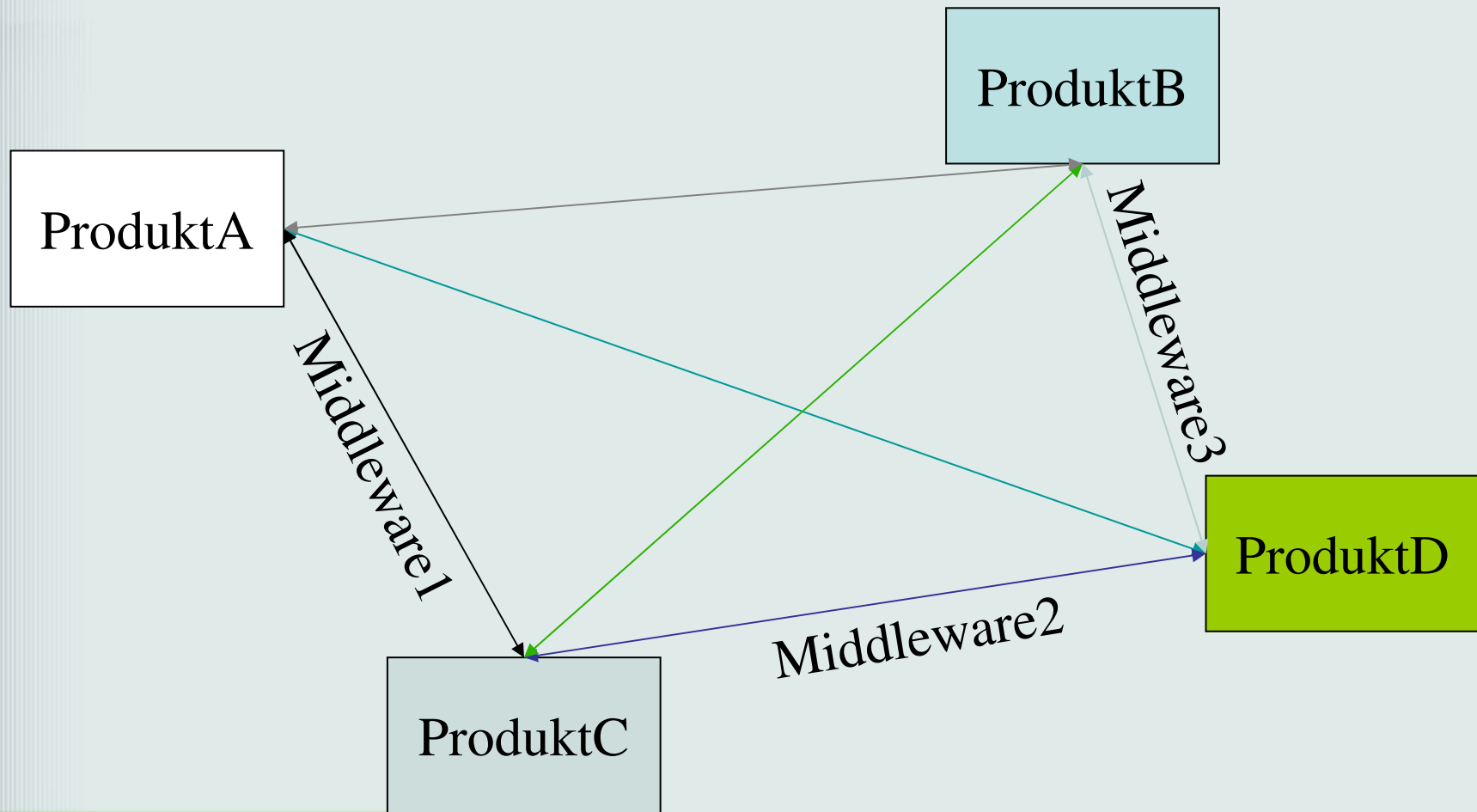
alles läuft leichter!

## „E-Revolution“ von Geschäftsprozessen



# Enterprise Application Integration

alles läuft leichter!

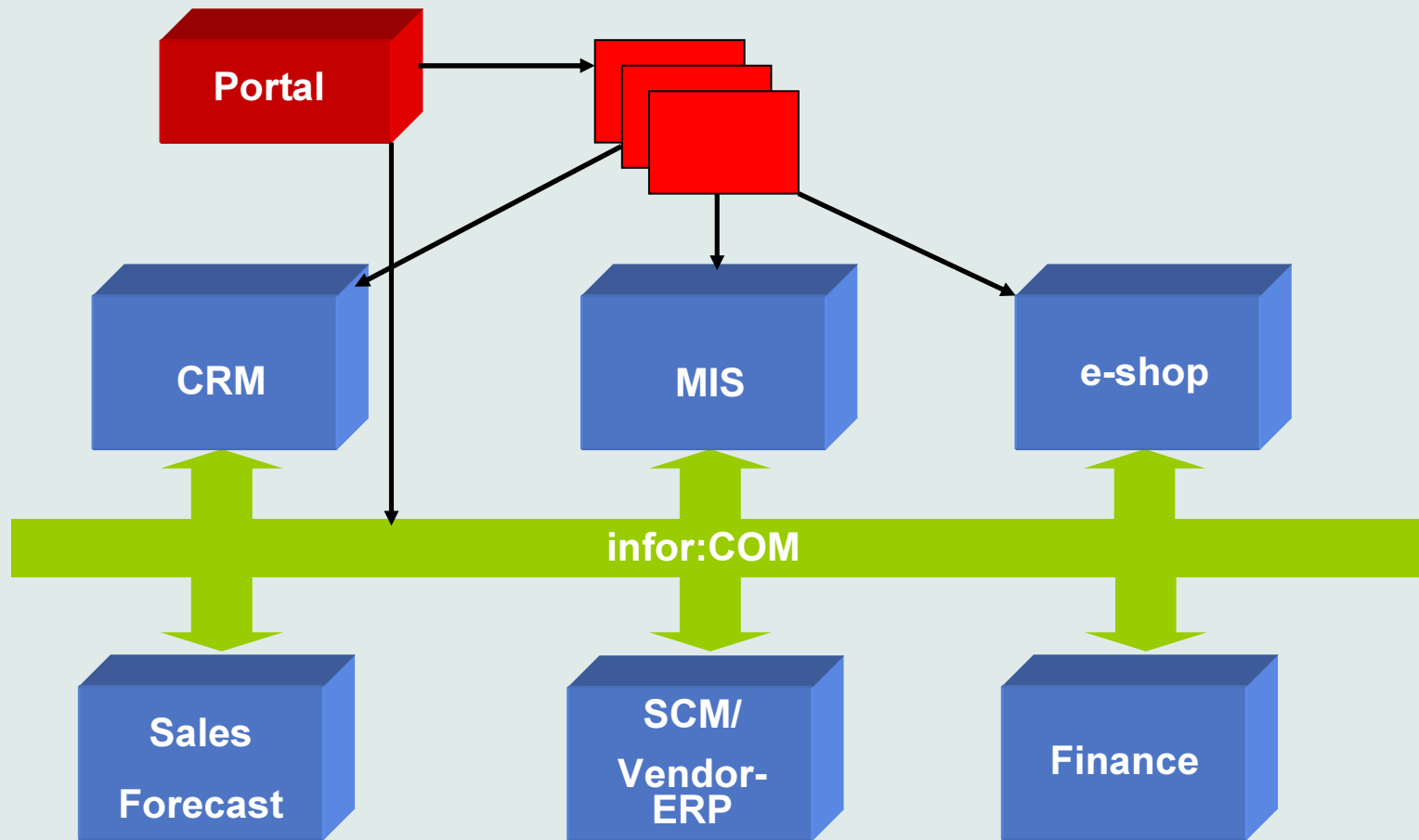


## Quo vadis?

- ***Weder monolithische ERP-Systeme noch heterogene Systemlandschaften sind auf konstanten Wandel ausgelegt.***
- ERP muss sich in Richtung Enterprise Application Integration (EAI) weiterentwickeln.

## ERP-II ist der „Enterprise Backbone“

alles läuft leichter!



## Vorteile eines komponentenbasierten ERP-II

- (Er-)Trägt revolutionäre Geschäftsprozesse.
- Internet ist Basis-Technologie.
- Offene Plattform.
- Speziallösungen können integriert werden.
- Kern-Funktionalität durch ERP-Komponenten.
- Anpassbarkeit und Erweiterbarkeit.
- Modularisierung und Wartbarkeit.
- Skalierbar in zwei Richtungen.





## Anforderungen für ERP-II

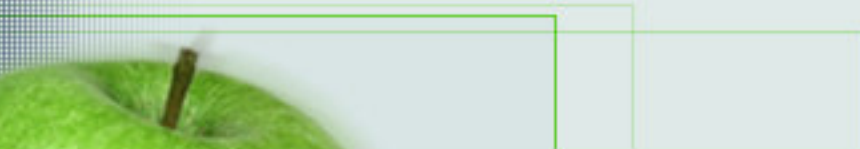
- State-of-the-Art Technologie
  - Plattform- und Datenbankunabhängigkeit
  - Standard-Kommunikationsprotokolle
  - Sicherheit
  - Verteilte Architektur
  - Trennung Geschäftslogik von Basis-Technik
  - Durchgängiger und zyklischer Entwicklungsprozeß
  - Hochentwickeltes Tooling
- 
- ***Sanfte Migrationsstrategie von ERP zu ERP-II***

## Das infor:21 Projekt

- Anfang 2000 bis Ende 2001
- Aufwand ca. 20 Mann-Jahre
- Architektur und Implementierung einer ERP-II Systembasis
  - State-of-the-Art Technologie
  - Plattform- und Datenbankunabhängigkeit
  - Standard-Kommunikationsprotokolle
  - Sicherheit
  - Verteilte Architektur
  - Trennung der Geschäftslogik von der Basis-Technik
  - Durchgängiger und zyklischer Entwicklungsprozeß
  - Hochentwickeltes Tooling
  - ***Sanfte Migrationsstrategie von ERP zu ERP-II***

## 21 Themen

- System Architecture
- Business Process Modelling
- Component-Toolbox
- Base classes
- Technical Architecture
- Persistency
- Transaction handling
- Short Extensions of infor:NT
- Runtime Language
- GUI-Tool
- Internationalisation
- Migration
- Metadata
- Versioning
- Customisation
- Partner-Products
- Installation
- Upgrade
- Testability
- Development Process
- Internal Rollout

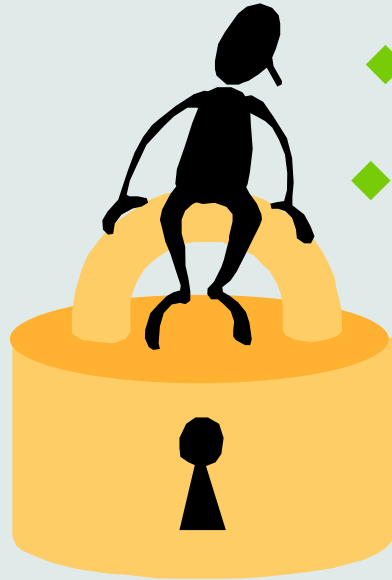


### 3 Phasen

- bis Mitte 2000:        Evaluierung & Design
  - Definition der logischen Architektur
  - Prototypen-Bau für die technische Architektur
- bis Anfang 2001:     Realisierung
  - Implementierung der Systembasis
  - Realisierung einer integrierten Tool-Umgebung
- bis Ende 2001:       Framework & Migration
  - ERP-Grundrepräsentationen
  - Integration in infor:COM
  - Simultan zum Projekt „Absatzplanung:21“

alles läuft leichter!

## Schlüsselkonzepte



◆ „Internet“-Baukasten



◆ Moderne Entwicklung



◆ Skalierbarkeit



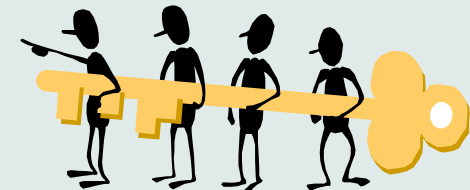
◆ Anpassbarkeit



◆ Wartbarkeit



✓ Objekt-Orientierung

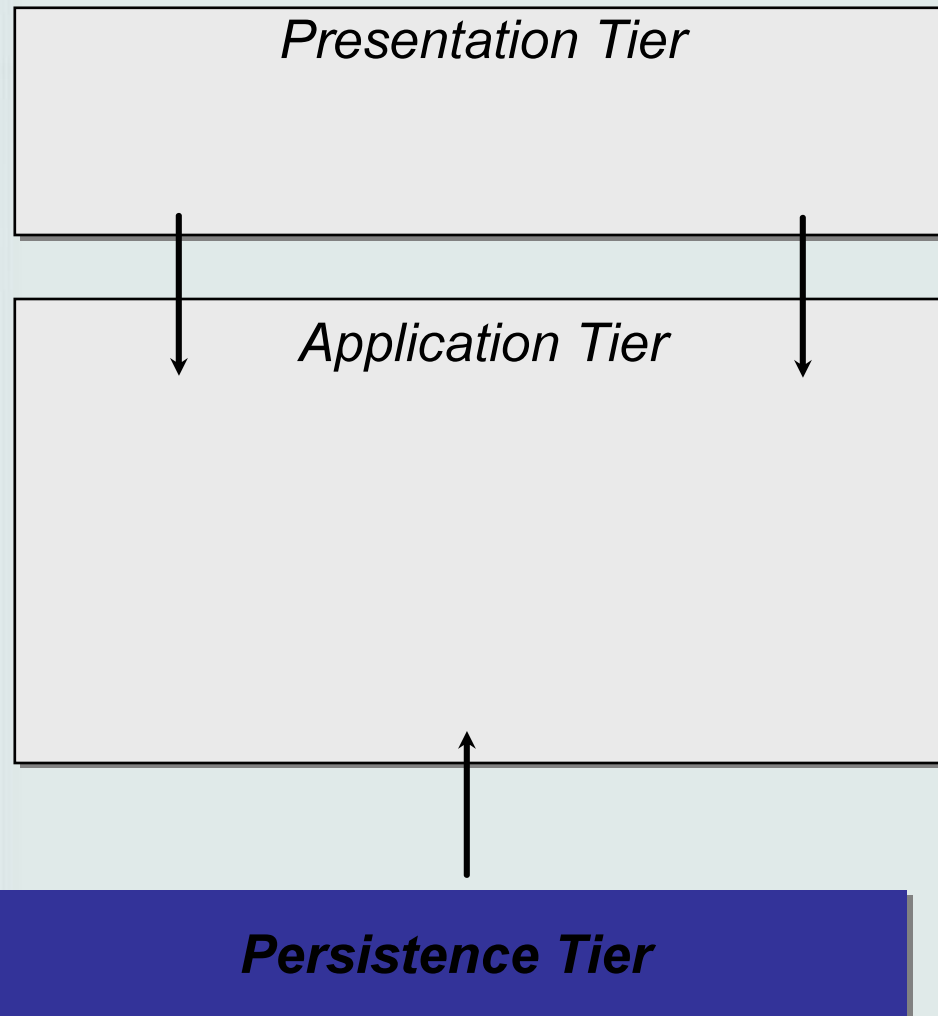


✓ Komponenten-Basierung

✓ Schichten-Architektur

## Logische Architektur: 3-Schicht Modell

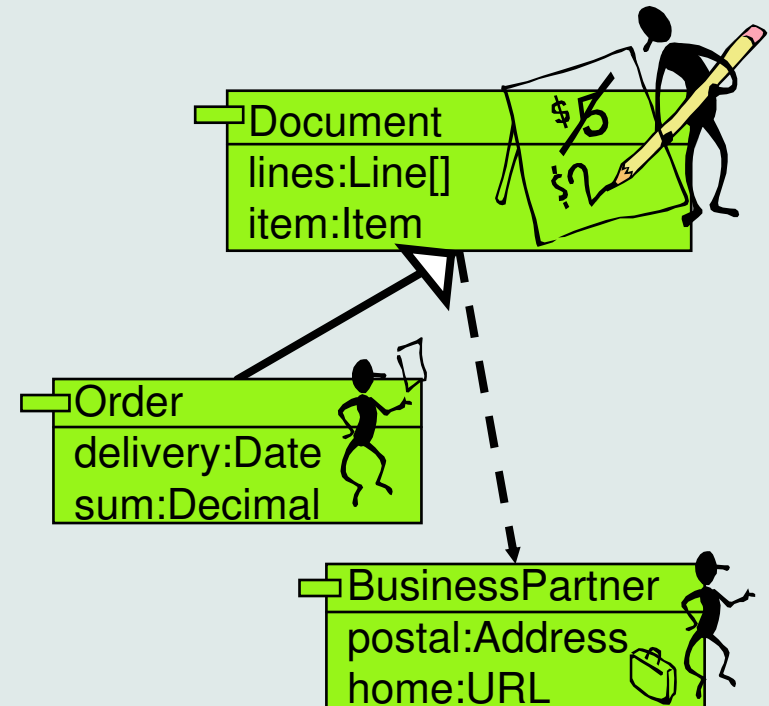
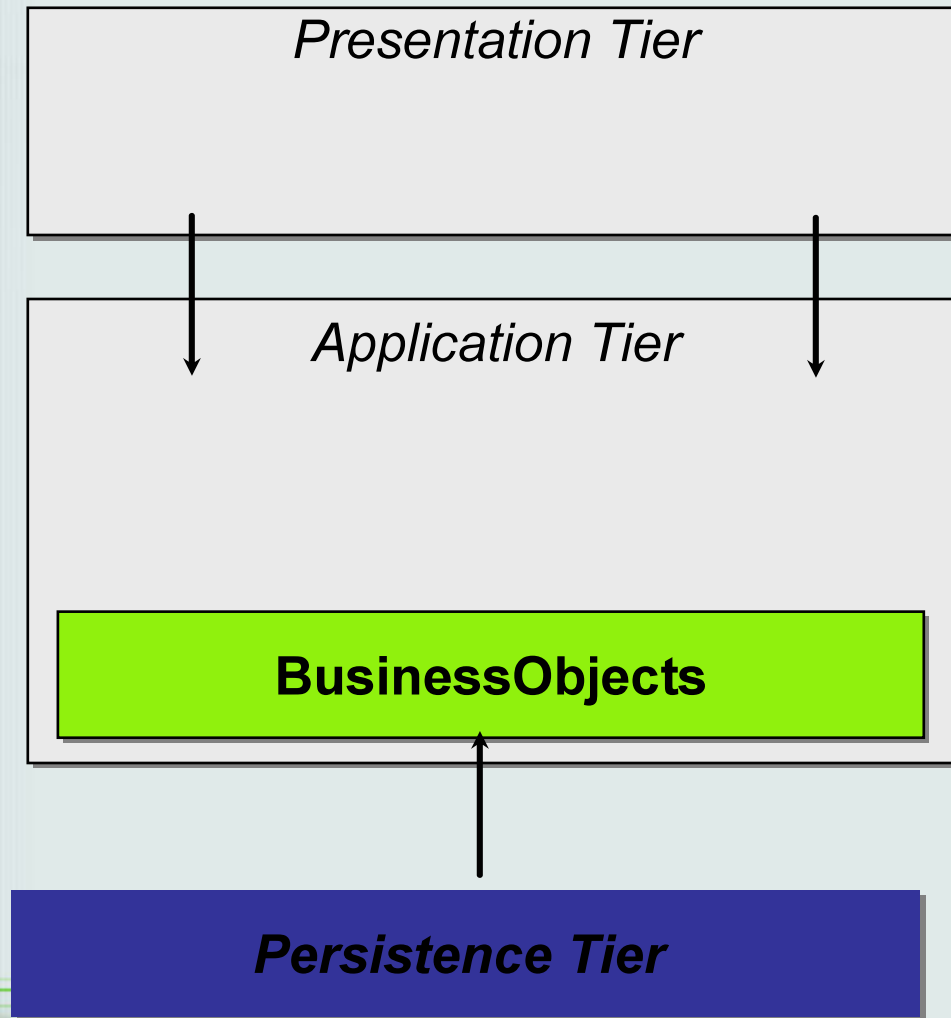
alles läuft leichter!





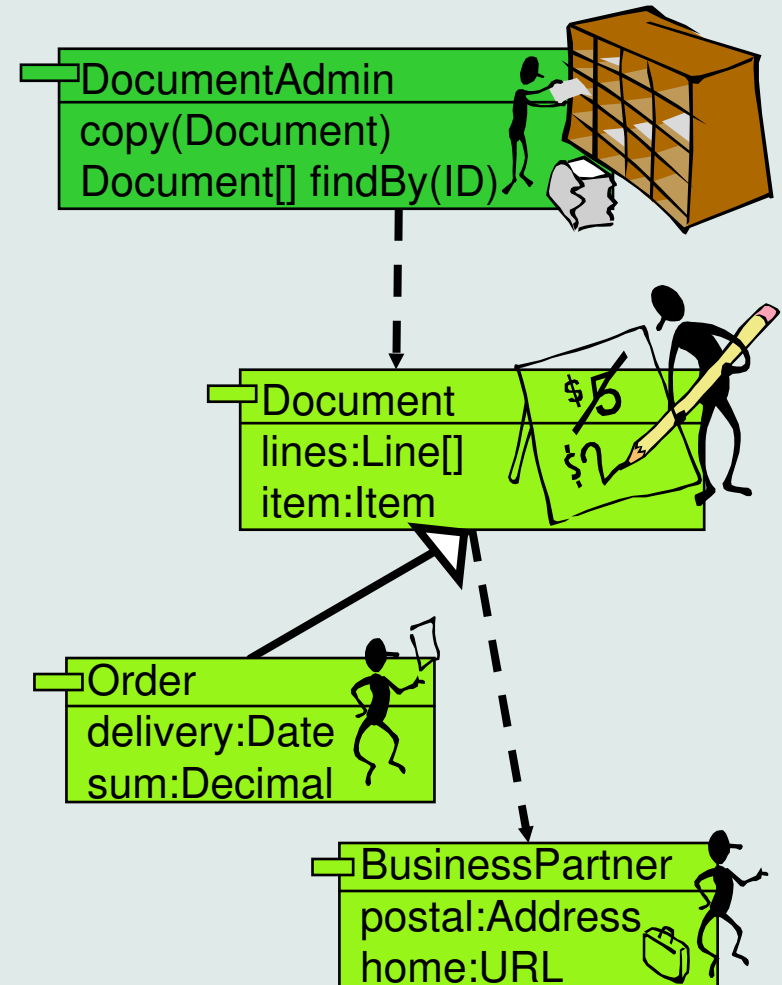
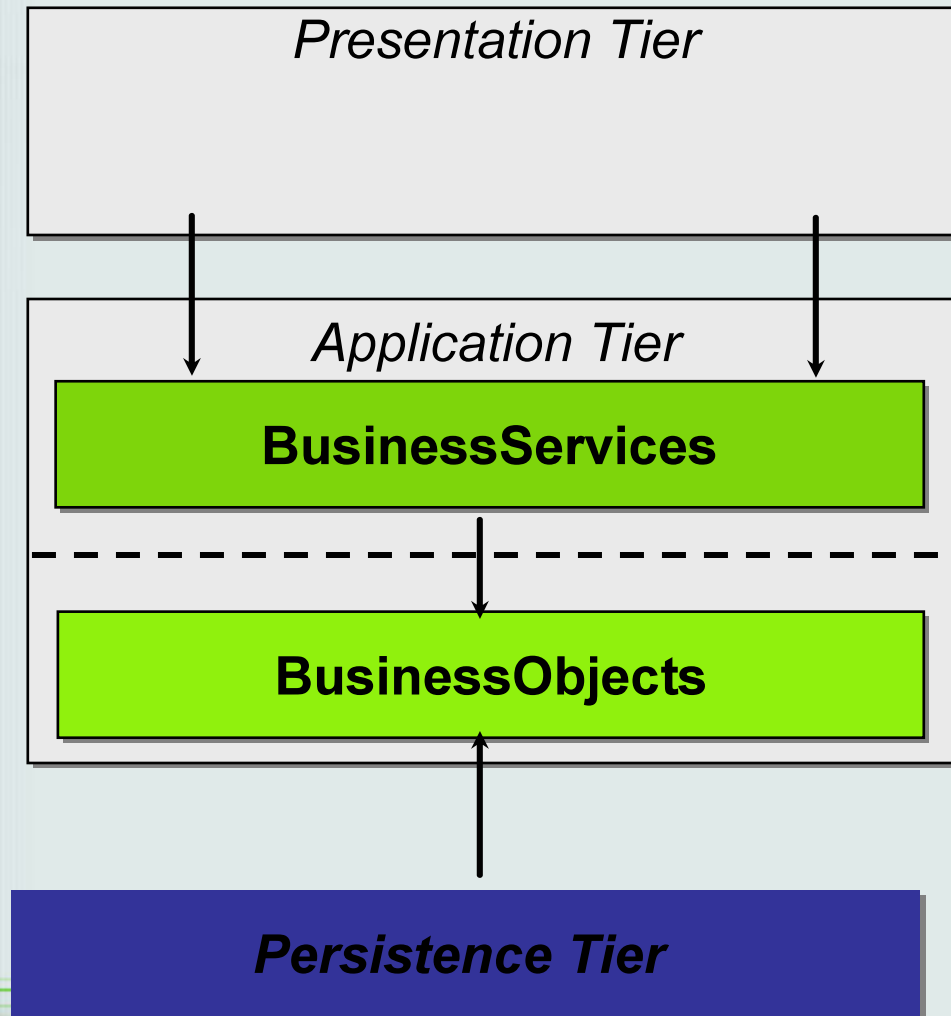
# Logische Architektur: Geschäftsobjekte

alles läuft leichter!

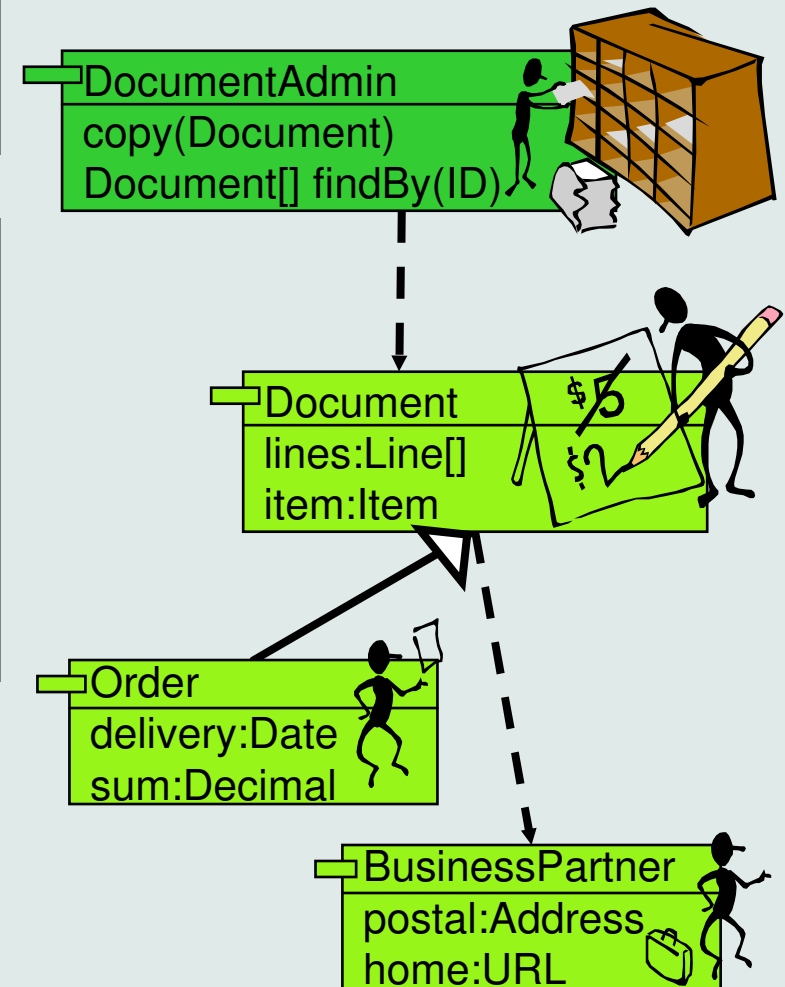
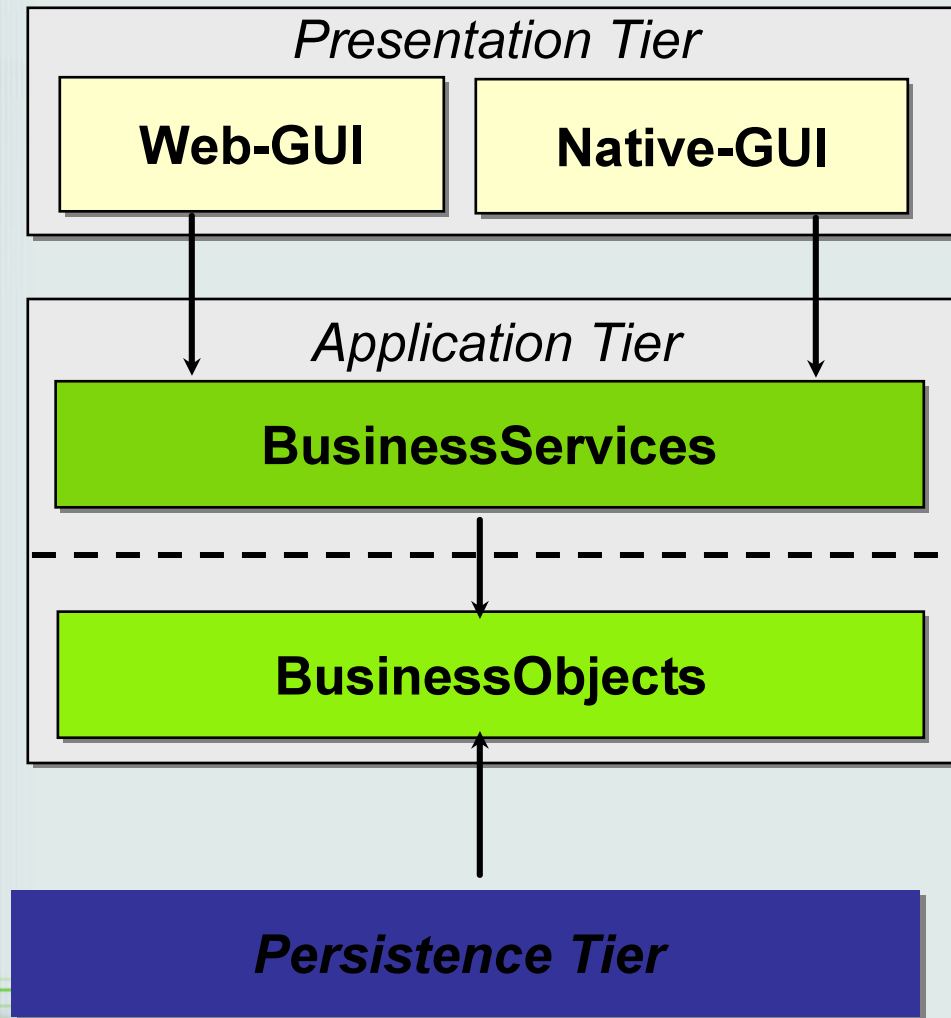


# Logische Architektur: Geschäftsdienste

alles läuft leichter!

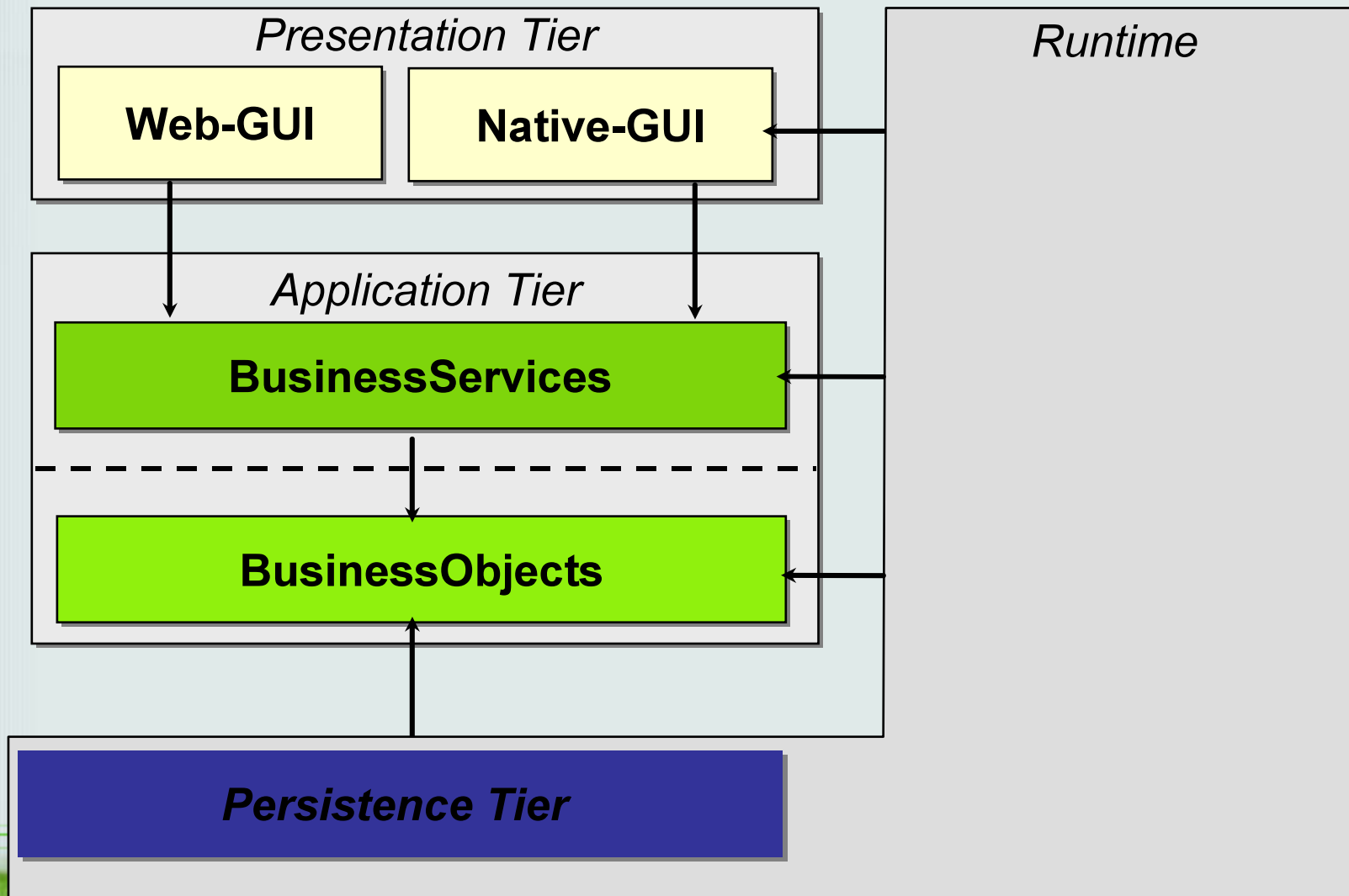


# Logische Architektur: BAPI & Präsentation *alles läuft leichter!*



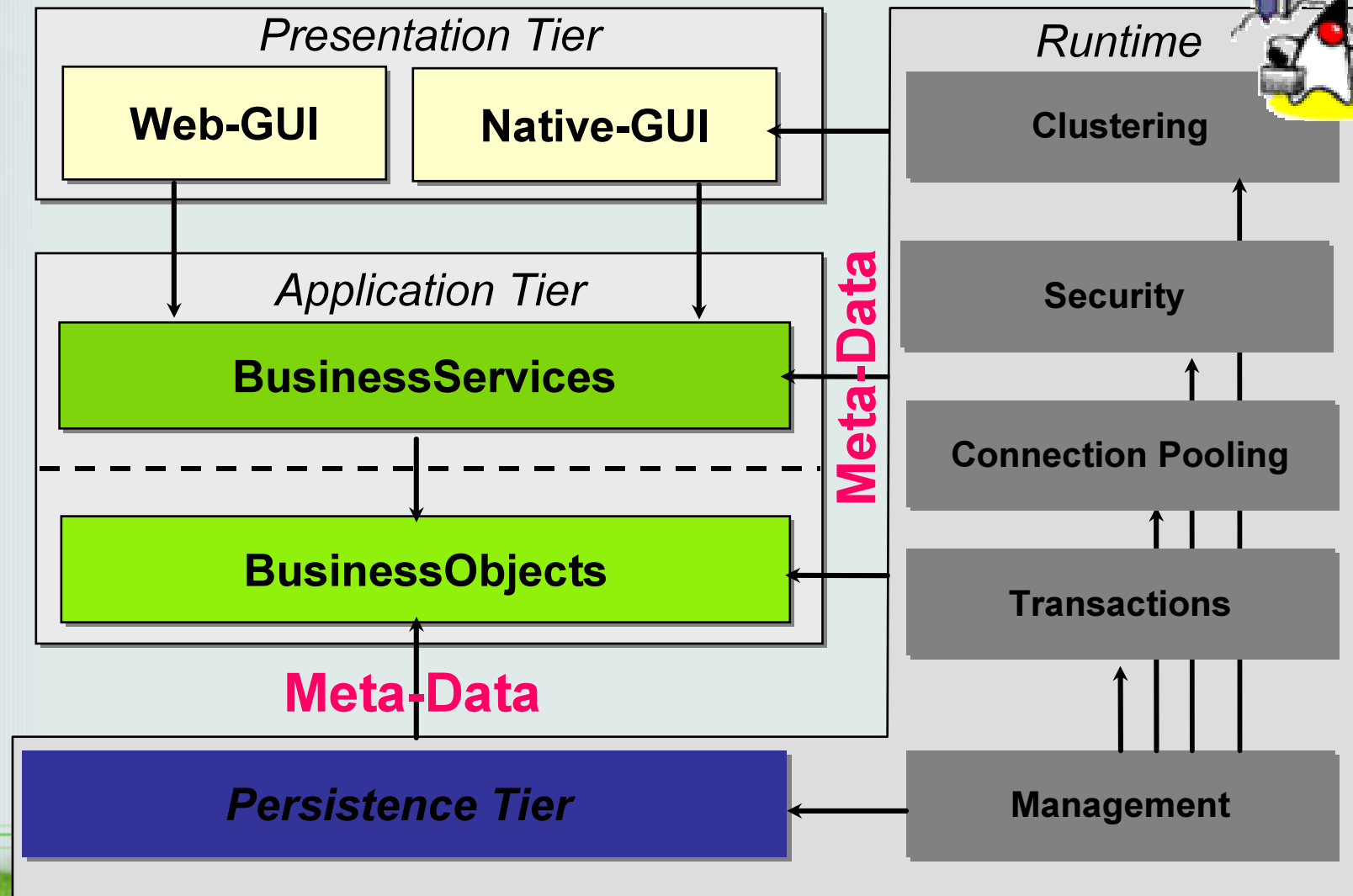
# Logische Architektur

alles läuft leichter!



# Deklarative Programmierung mit J2EE™

alles läuft leichter!ä



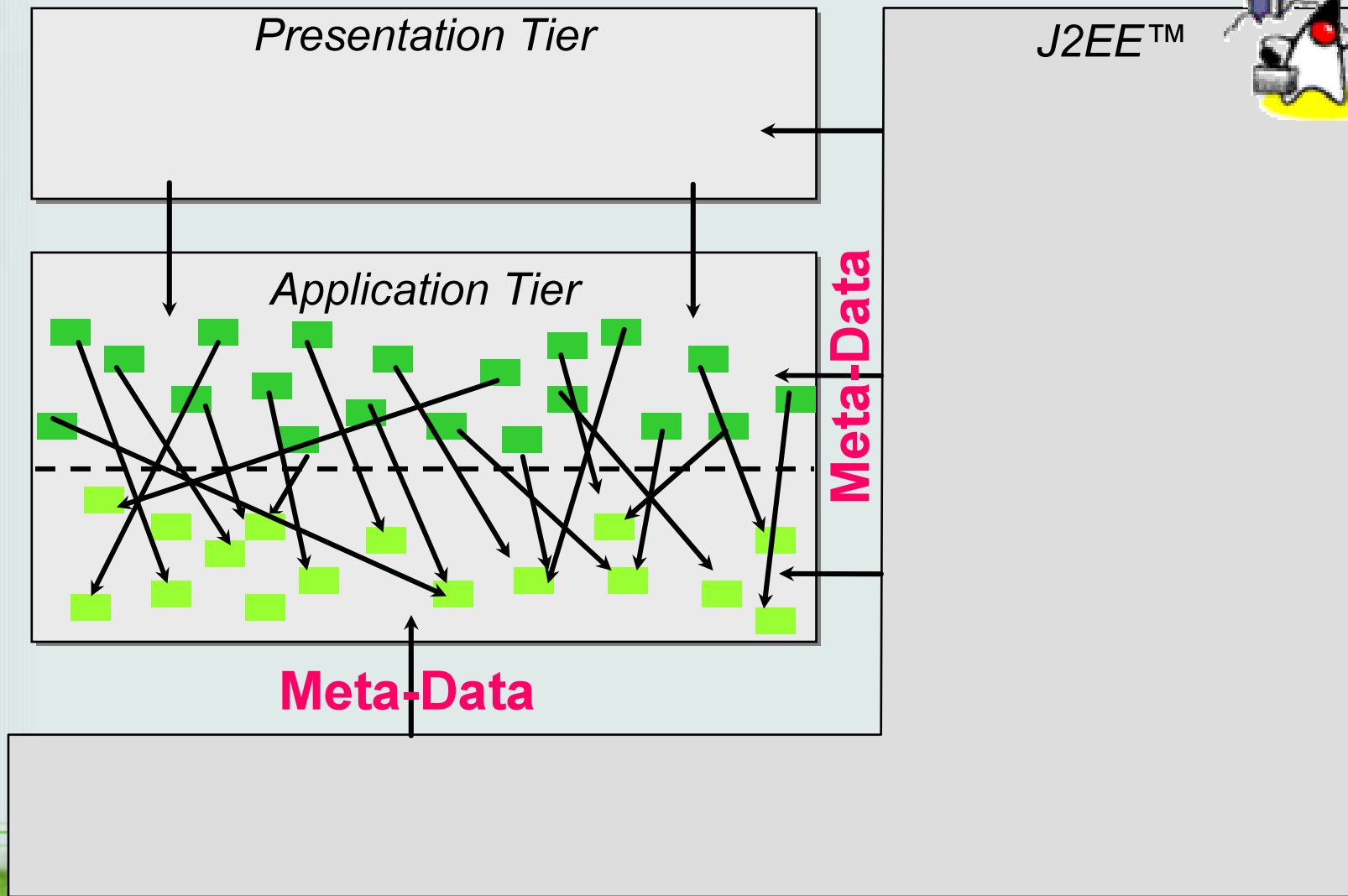
## Third-Party Produkte und Open Source

- infor ist primär ein ERP-Haus, kein Middleware-Hersteller.
- gängige, kommerzielle J2EE™-Produkte sind für ERP ungeeignet.
- hochpreisiger Markt, der auf Laufzeitlizenzen spezialisiert ist.
- Relativ kleine Anbieter.
- Starke Fluktuation.
- Open Source als Alternative:
  - ISV's „outsourcen“ Systembasis.
  - Fokussieren auf Anwendungslogik und evtl domänen-spezifische Erweiterungen.
  - Linux-->GNU-->Apache-->Jonas/JBoss
- Unser Engagement bei [jboss.org](http://jboss.org) ist preisgekrönt:
  - JavaWorld'2002 Best Application Server Award
    - vor IBM Websphere und BEA Weblogic
  - „Das Linux des J2EE™“



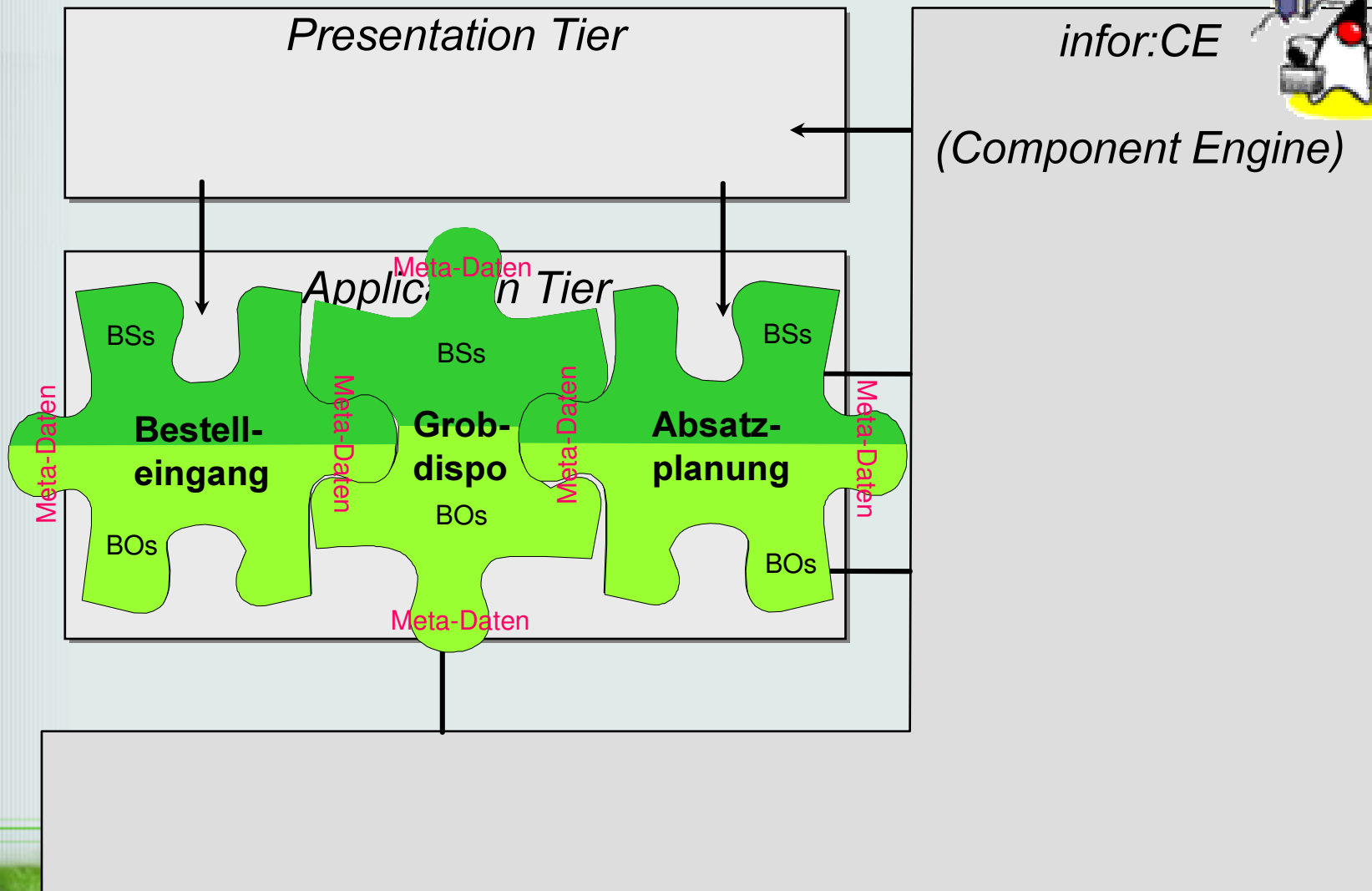
# Objekte sind schlechte Komponenten

alles läuft leichter!



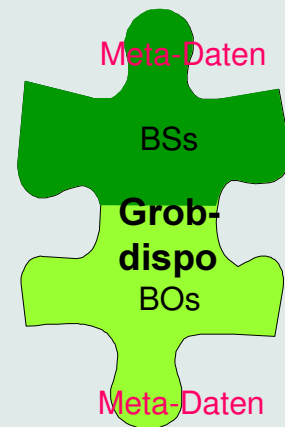
Geschäftskomponenten  
=BOs+BSs+Meta-Daten

alles läuft leichter!



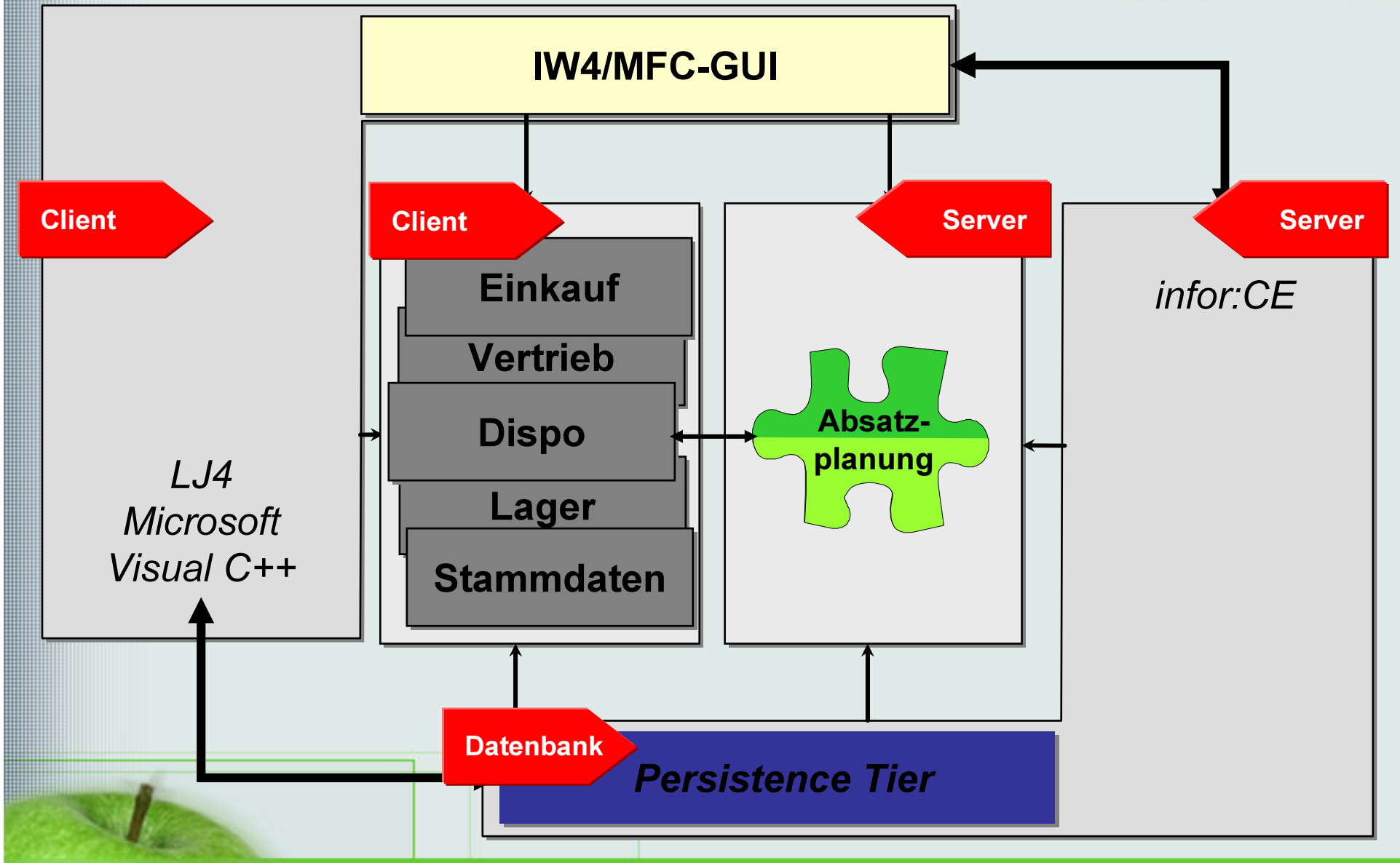
## Geschäftskomponenten

- Komponente = In sich abgeschlossene Logik, die unabhängig entwickelt, getestet, vertrieben, installiert, konfiguriert und von der Laufzeit-Umgebung abgearbeitet werden kann.
- Serverkomponente = Geschäftsdienste+ Geschäftsobjekte+Meta-Daten
- Clientkomponente= Dienstschnittstellen+GUI-Logik
- JAR=Java Archive, ähnlich zu dll mit Zusatzinformationen
- Objekt < Komponente < Modul



# Hybride Architektur

alles läuft leichter!



# Einheitliches Look&Feel

alles läuft leichter!

infor:COM Disposition - [Absatzplanung]

Daten Bearbeiten Ansicht Funktionen Verzweigen Extras Fenster Hilfe

Öffnen Filter Workflow

Artikel 62028  
Kurztext Mutter komplett

Laden

Felder löschen

Beginn Zeitreihen 01.04.2000

Artikel 62028  
Kurztext Mutter komplett

Artikelgruppen 656 656 656 003

ABC-Klasse  
XYZ-Klasse

Verwendung  
☐ Einkaufsteil  
☒ Verkaufsteil  
☒ Eigenfertigung

Planmodus Manuelle Vorgabe

Prognoseprofil 23

Periodizität 45345

Übergabe 211212

Runden auf 1,00 Stück

Auftragsgrenzmenge 99.999.999,00

Zeitreihen Jahresplanmenge Automatische Planung

Jahresplanprofil JPP3

Jahresplanmenge

	Jahr	Planmenge
1	2000	1.000,00
2	2001	1.000,00
3	2002	10.000,00
4		
5		
6		
7		

1 / 3

Vertrieb  
Konstruktion  
Disposition  
Finanzen  
Produktion  
Lager  
Einkauf  
Systemfunktionen

Artikelkonto  
Kapazitätskonto  
Mindestbestandsüberw  
Batchdisposition  
Grobgeplante Aufträge  
Übergabe Feinplanung  
Ausnahmen  
Berechnungen  
Listen  
Absatzplanung  
Artikel  
Prognoseprofile  
Perioden  
Übergabepprofile  
Jahresplanprofile  
Dämpfungsfkt.  
Saison-/Trendprof.  
Abweichungen

Software

Start

Ein... Po... inf... inf... Ein... C:\... inf... inf... inf... scr...

QS-Setup1

18:18



# Integration in 4GL-Tools

alles läuft leichter!

The screenshot displays the Infor Dialog Manager - Developer [DataBase] interface. The main window shows the definition of a database object, 'iceViewItem\_2oskfes', which is a view of 'iceViewBusinessObject\_bp7sls'. The definition includes several fields, each with a type and a handle:

```
DataBaseDef View iceViewItem_2oskfes Like iceViewBusinessObject_bp7sls Except
Field commonItem_RECORDHANDLE As bfldWord
Field replenishmentBreak_RECORDHANDLE As bfldWord
Field dailyIssueForecasts_RECORDHANDLE As bfldWord
Field stockLimit_RECORDHANDLE As bfldWord
Field safetyStockLevel_RECORDHANDLE As bfldWord
Field planMode_RECORDHANDLE As bfldWord
Field orderLimitQuantity_RECORDHANDLE As bfldWord
Field minimumStock_RECORDHANDLE As bfldWord
Field satisfactionOnProduction As bfldReal10d2
Field sourceForTimeSeries As bfldByte
```

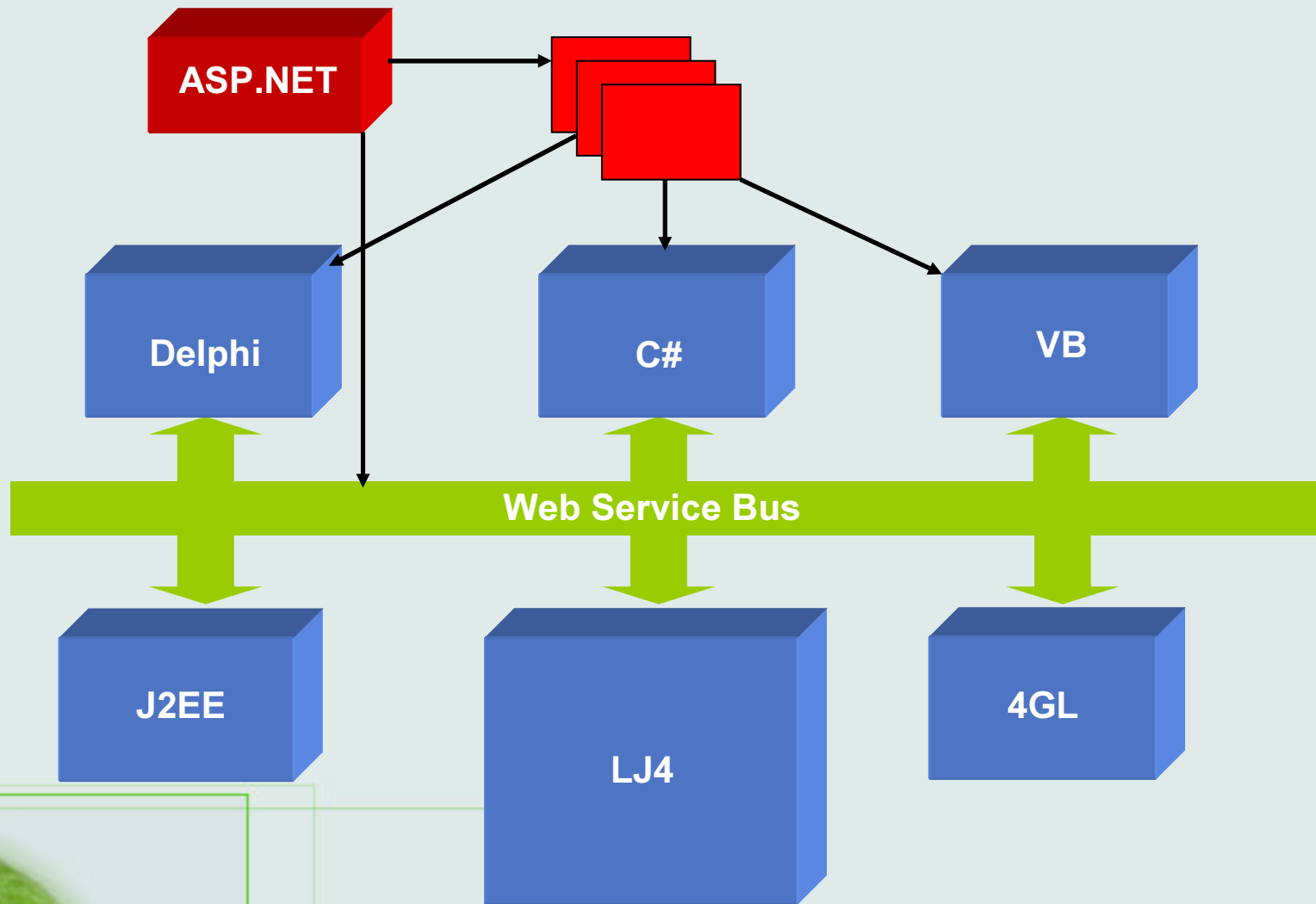
The interface also features a tree view on the left showing the project structure, including 'Root', 'Index', 'Table', 'RecordCheck', 'Field', 'Struct', 'View', and 'iceViewUniqueObject\_1kk87n0'. A 'Properties' window is open on the right, showing the 'Eigenschaften' (Properties) tab for the object 'dlgInfor21QueryFilterEdit'. The properties table lists various attributes and their values:

Eigenschaft	Lokal	Geerbt von	Wert
1 Alignment	<input type="checkbox"/>	Root	AlignLeft
2 AutoOk	<input type="checkbox"/>	Root	False
3 BackColor	<input type="checkbox"/>	Root	&H00BDB38D&
4 BorderStyle	<input type="checkbox"/>	Root	LineNull
5 Caption	<input checked="" type="checkbox"/>	Root	steFilter
6 ContextObjectLink	<input type="checkbox"/>	Root	0
7 DataBaseLink	<input checked="" type="checkbox"/>	Root	iceViewFilter14qyy2k
8 DefaultTextBlockLin	<input type="checkbox"/>	Root	
9 Description	<input checked="" type="checkbox"/>	Root	Dialog: Filter bearbeiten
10 DialogEntryLink	<input type="checkbox"/>	Form	dlgDummyFace
11 Enabled	<input type="checkbox"/>	Root	True
12 EnforceMenus	<input type="checkbox"/>	Root	False
13 FontBold	<input type="checkbox"/>	Root	False

The status bar at the bottom indicates the current operation: 'Parasen 'LocaleDef StringTableEntry steWert [5300.0002.0]:'.

# Technologische Vision infor:COM

alles läuft leichter!

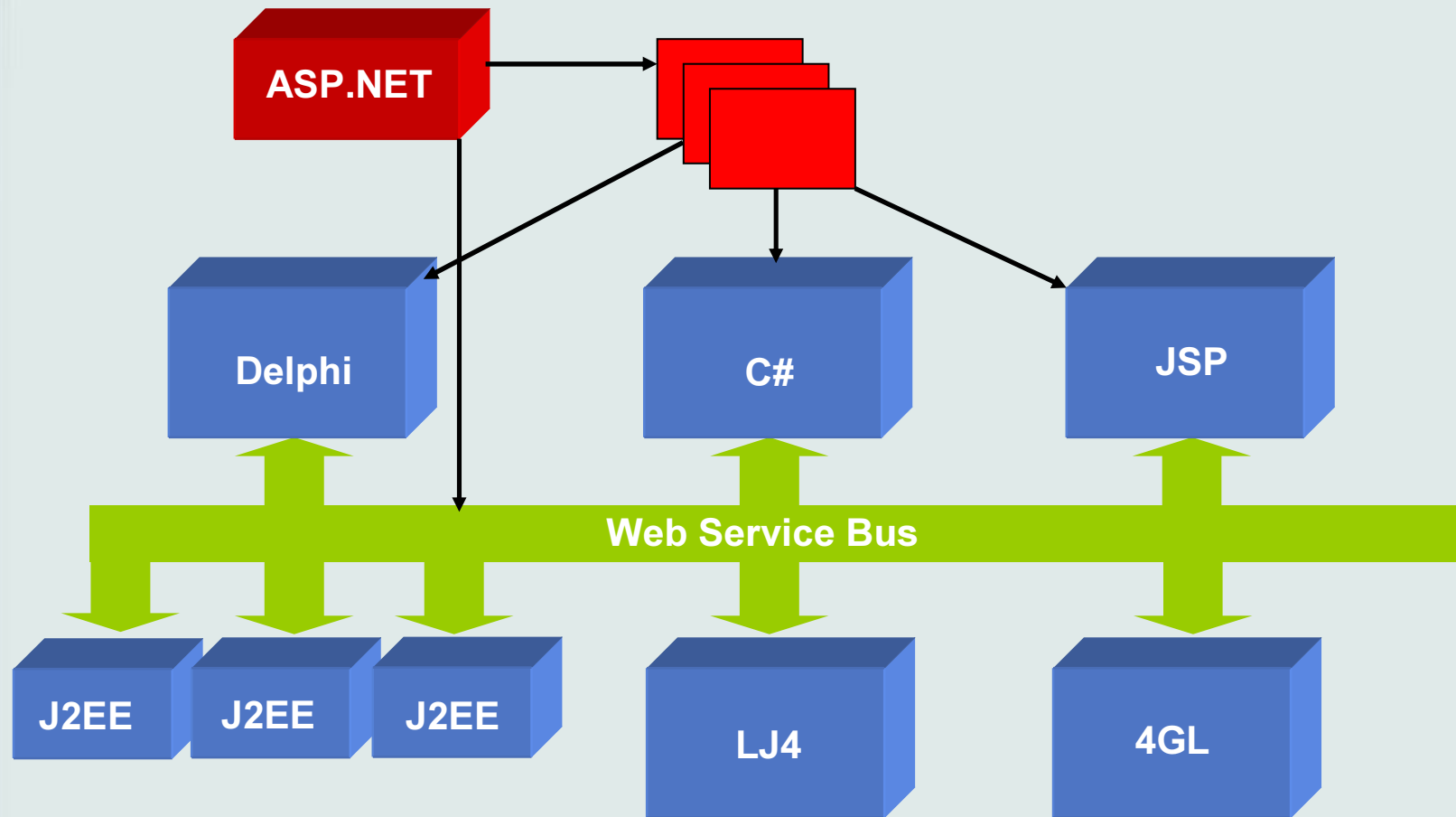


→ J2EE, .NET, Web Services



alles läuft leichter!

## Weiche Migration



## Entwicklungsprozeß

- Für ein Software-Unternehmen ist eine Formalisierung des Entwicklungs-Prozesses lebenswichtig.
- Unterschiedliche Spezialisten
  - Domänen-Spezialist/Produkt-Manager
  - Architekt
  - Entwickler: Basis, Anwendung, GUI-Designer
  - Berater, Service-Mitarbeiter
- Eine gemeinsame Sprache zur Dokumentation und Kommunikation zwischen den verschiedenen Spezialisten ist notwendig
  - kein Source-Code!
- Tool-Support ist notwendig.

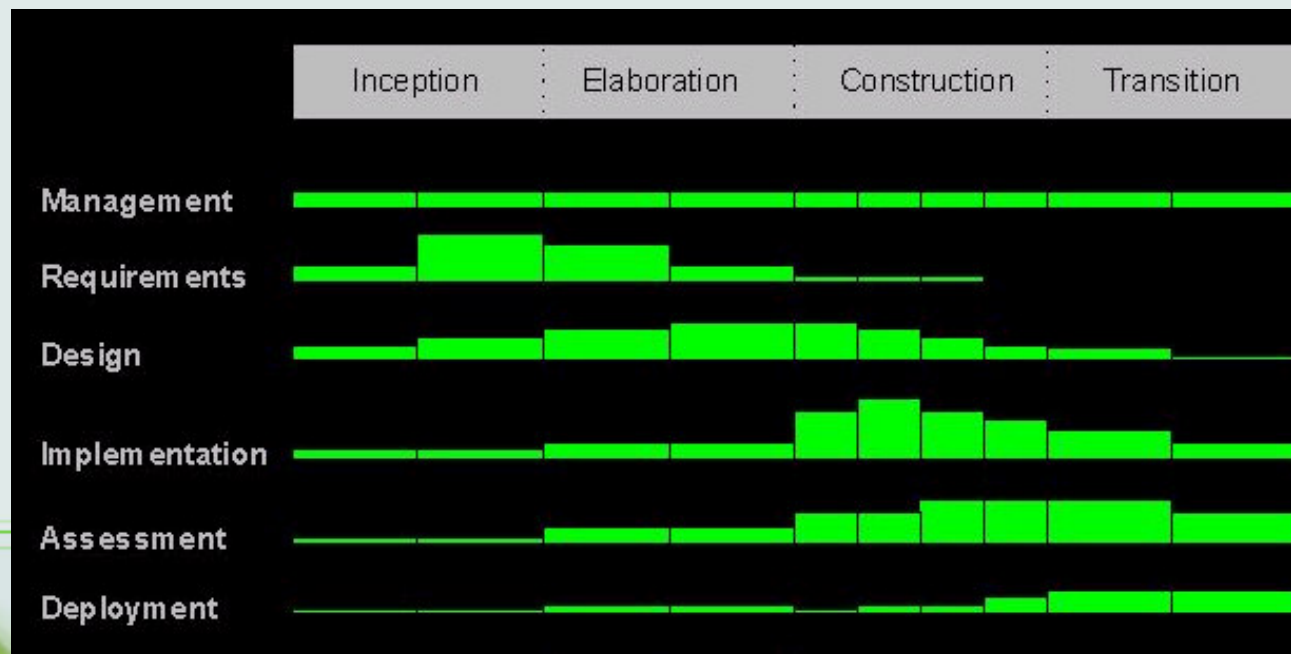
## Fehlerbehebung in der Entwicklung

Entwicklungs-phase	Anteil	Fehlfunktionen	aufgedeckt	Kosten der Behebung
Anforderung / Analyse	5%	40%	18%	1.0
Design	10%	30%	10%	1.0- 1.5
Implementierung	60%	25%	50%	10-15
Einsatz/Wartung	25%	5%	22%	10-100



## Iterativer Entwicklungsprozess

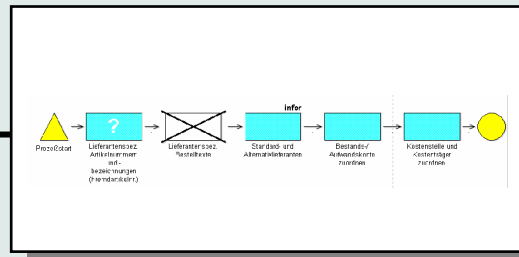
- ermöglicht ein frühes und kostengünstiges Erkennen von Fehlern:
  - Rapid-Prototyping
  - Spiral-Modelle
  - Feature-Based Programming
  - eXtreme Programming
  - Rational Unified Process



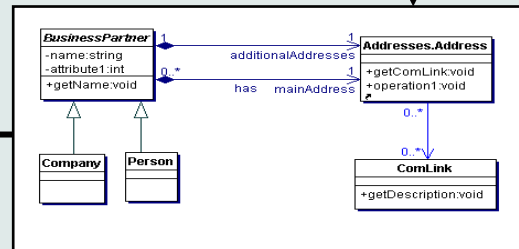
# Toolgestützter Entwicklungsprozess

alles läuft leichter!

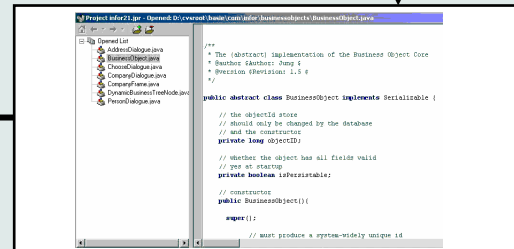
**PERFORCE**  
Repository (Version Control,  
Document Management)



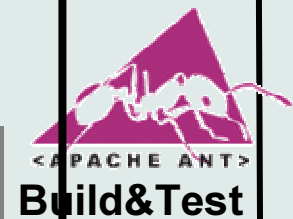
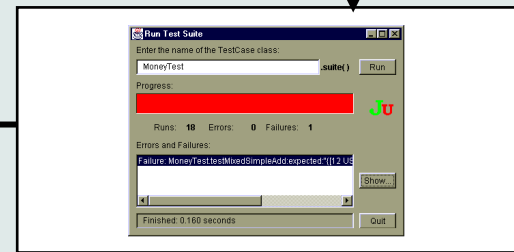
**Business Process Modelling**



**Design**



**Implementation**



## UML – Unified Modeling Language

- graphische Sprache zur Modellierung (objekt-orientierter) Systeme
- Spezifikation, Konstruktion, Visualisierung und Dokumentation
- Industriestandard
- von den meisten Tools und Programmiersprachen unterstützt
  - Rational Rose
  - Together/J

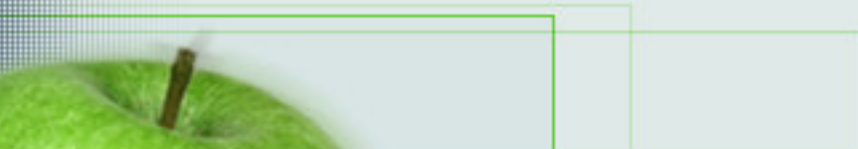




## Modellelemente der UML

Diagramme lassen sich auf verschiedenen Abstraktionsebenen einsetzen:

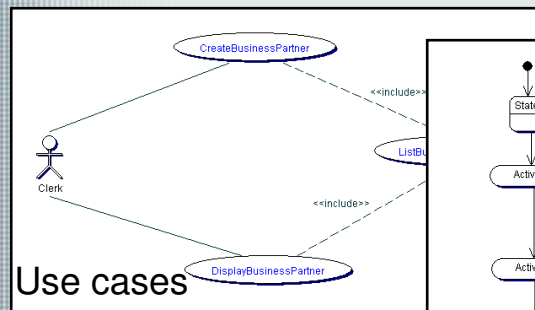
- **Anwendungsfalldiagramm** (use case diagram)
  - zeigt Akteure, Systemgrenzen und Anwendungsfälle
- **Aktivitäts- und Zustandsdiagramme**
  - Zeigen Systemzustände und ereignisgesteuerte Aktivitäts-/Prozessketten
- **Sequenz- und Kollaborationsdiagramme**
  - zeigen den Nachrichtenaustausch von Systembestandteilen
- **Klassendiagramm**
  - Namensräume, Klassen und ihre Beziehungen
- **Komponenten und Deploymentdiagramme**
  - Physikalische Realisierung, Abhängigkeiten und Installation des Systems



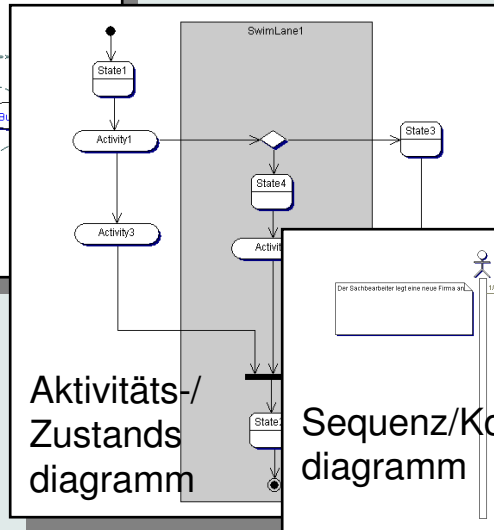
# Modellierung mit UML

alles läuft leichter!

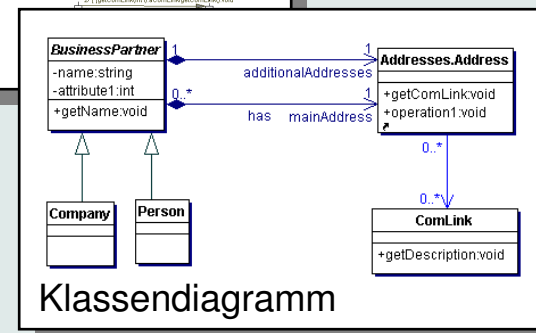
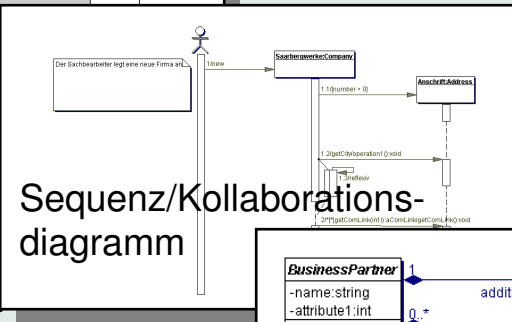
## Anforderungsaufnahme



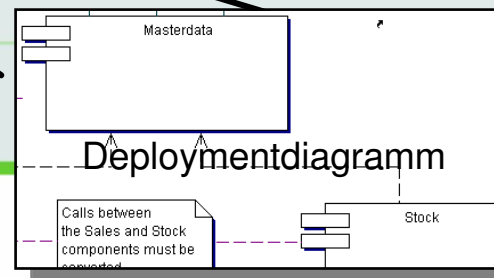
## Analyse



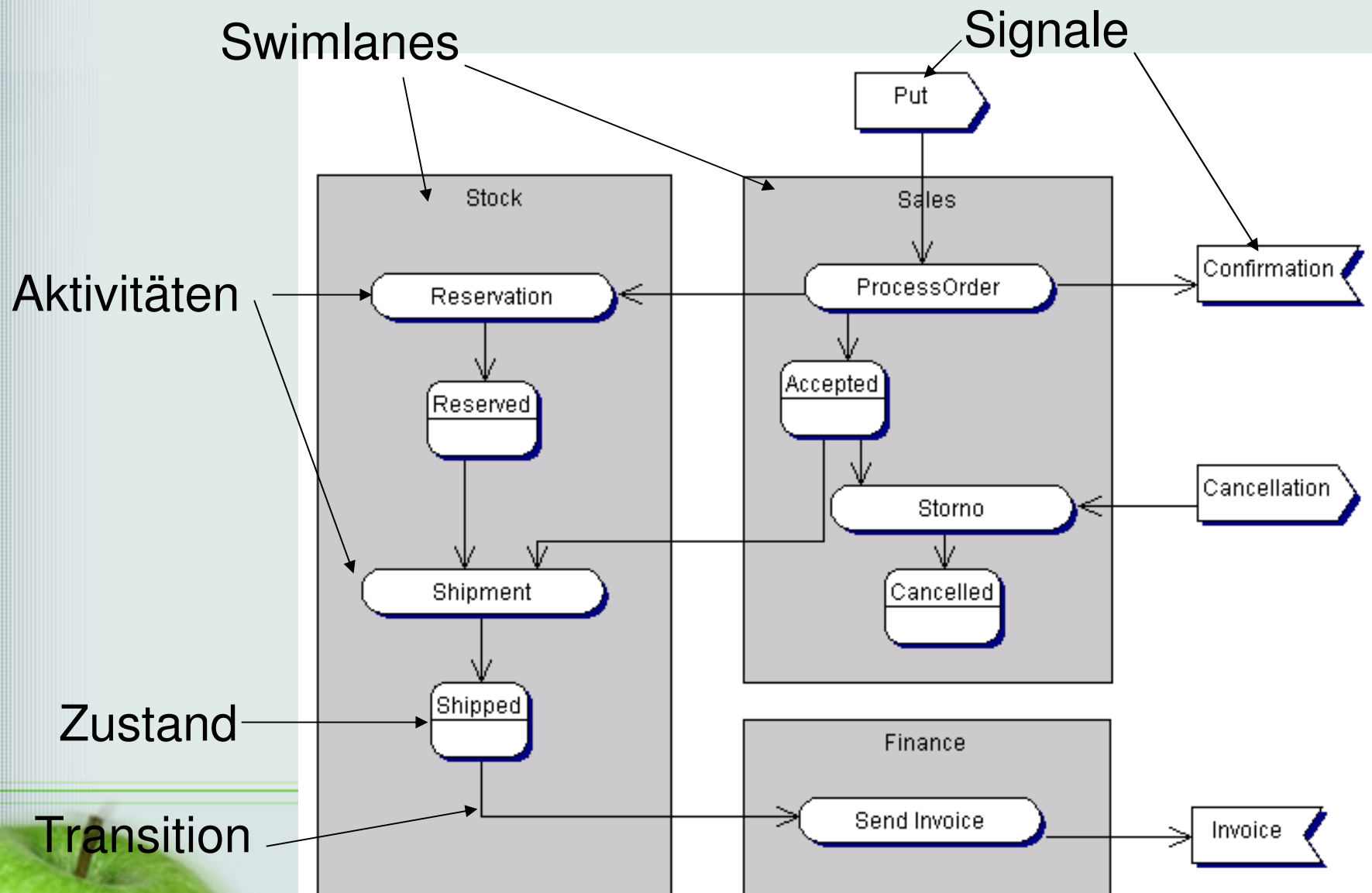
## Design



## Implementierung

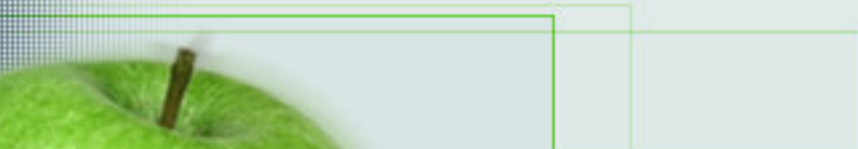


# Aktivitäts- und Zustandsdiagramme für GPM alles läuft leichter!



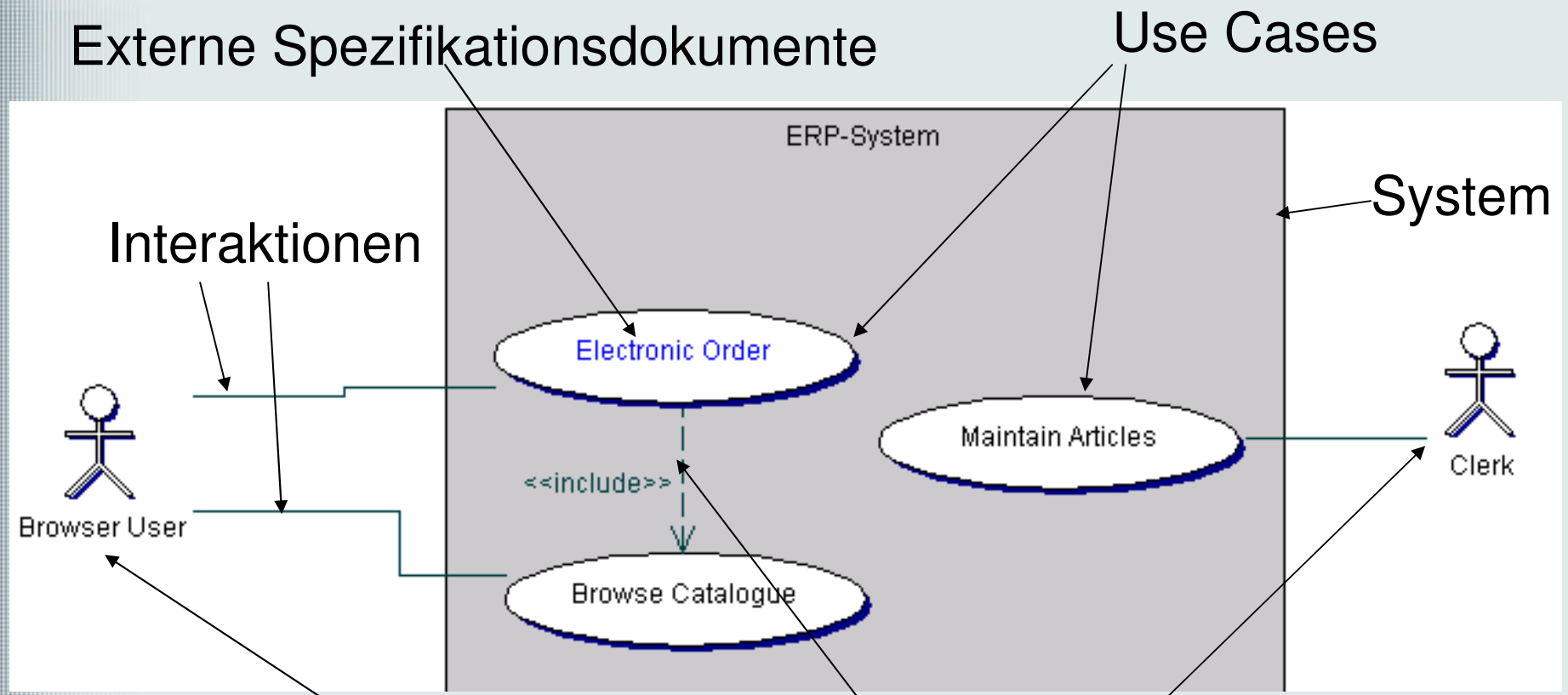
## Anwendungsfälle zur Anforderungsaufnahme

- Festlegen der Systemgrenzen und Akteure/Rollen
- Funktionale Anforderungen an das System durch beispielhafte Fallbeschreibungen („Use Cases“)
- Medium zwischen Produktmanagement/Beratung und Architekten
- Hilft dem Architekten, sich auf die Sichtweise des Anwenders zu konzentrieren und nicht auf interne Details
  - weniger Missverständnisse zwischen Anwender und Entwicklung.
- Nachträgliche Ideen bedeuten neue / geänderte Anwendungsfälle:
  - Änderungen im Anwendungsfall werden transparent:
  - Gründe für Kostenerhöhung oder Verzögerungen werden transparent.



# UseCase-Diagramm

alles läuft leichter!



# Externe Spezifikation

alles läuft leichter!

UseCase Electronic Order.doc - Microsoft Word

Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?

60% 12 F K U

20 40 60 80 100 120 140 160 180

<b>USE CASE</b>	PlaceElectronicOrder.	
<b>Author</b>	Wolber, Prasse, Jung	
<b>Goal in Context</b>	Shopping-Cart Functionality for Placing a Single Order, e.g., via an E-Shop	
<b>Actors</b>	Individual External User/E-Shop	
<b>Preconditions</b>	Up-To-Date Item-Catalogue and Warehouse, Client Already Known as a Business Partner	
<b>Success End Condition</b>	A New Order has been Accepted.	
<b>Failed End Condition</b>	No New Order has been placed.	
<b>Trigger (opt.)</b>	User has Successfully Logged In and Accessed the Shopping Area.	
<b>Description</b>	<b>Step</b>	<b>Action</b>
	1	Client is Assigned a Customer Object/Id.
	2	Client Creates a New Shopping Cart (or Request For Quote) and Assigns Its Customer Object/ID to It.
	3	Server is Asked to Deliver the List of Available Items (UseCase: <a href="#">BrowseItems</a> ).
	4	Client Adds a New Line to the Shopping Cart for a Particular Item in a Particular Quantity.
	5	If User Not Satisfied, Steps 2-3 Are Repeated.
	6	Client Requests a Quote for the Shopping Cart.
	7	Server Creates a Quote and Computes a Total using either Default Prices or Customer-Specific Prices (UseCase: <a href="#">CreateQuoteFromRequest</a> ).
	8	Client creates an Order by Transferring And Manipulating Some Lines of the Quote.
	9	Server Processes the Order by Calculating a Possible Due Date (UseCase: <a href="#">ProcessOrder</a> ).
<b>Variations</b>	<b>Step</b>	<b>(optional) Condition: Branching Action, e.g.</b>
	1a	If the Customer belonging to that Business Partner is already registered: return.
	1b	Otherwise: a new Customer with a New Id is generated and returned.
	1c	The Business Partner is not Known in Which Case Error 1 occurs.
	1d	The Request For Quote contains an invalid line in which case Error 2 occurs.

Zeichnen AutoFormen

Seite 1 Ab 1 1/2 Bei 114 mm Ze 17 Sp 1 MAK ÄND ERW ÜB Englisch (G)



## Sequenz- und Kollaborationsdiagramme

- **Übergang vom Anwendungsfall auf das konkrete Systemverhalten**
- **Identifikation von Entitäten**
  - **Struktureigenschaften**
- **Identifikation von Diensten**
  - **Logikeigenschaften**
- **Medium zwischen Architekt und Entwickler**

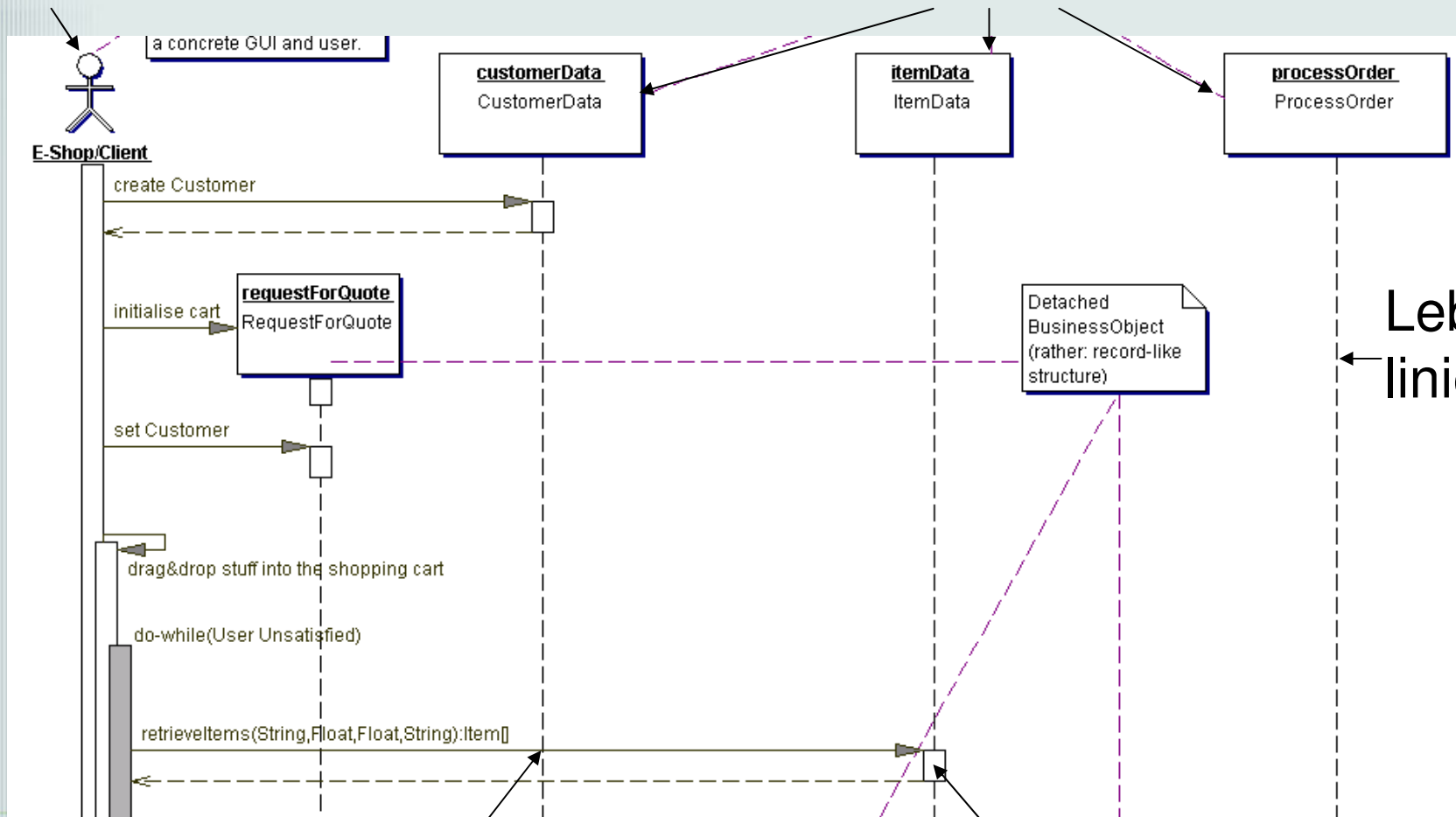


# Sequenzdiagramm

alles läuft leichter!

Akteur

(Objekt-)Instanzen



Lebens-  
linie

Nachricht

Methode

# Klassendiagramme

- **Logische Struktur des Codes**
- **Pakete, Sichtbarkeiten**
- **Klassen**
- **Attribute**
- **Methoden**
- **Beziehungen zwischen Klassen**
- **Übergang zum Code und dessen Dokumentation**



# Klassendiagramme

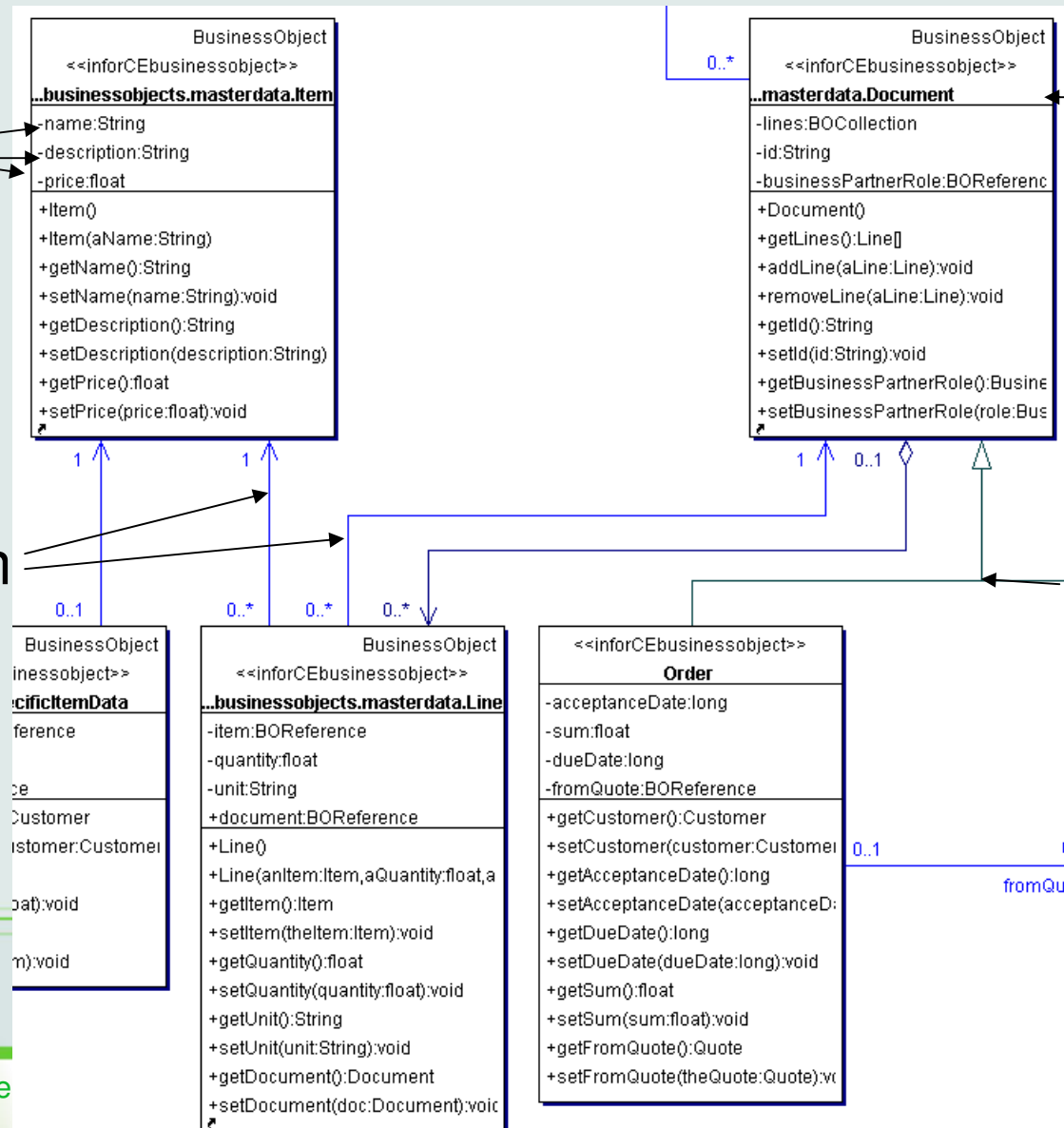
alles läuft leichter!

Attribute

Klasse

Relationen

Vererbung



## Komponenten- und Deploymentdiagramme

- **Physikalische Verpackung und Verteilung des Codes**
- **Exportierte Schnittstellen**
- **Laufzeitabhängigkeiten**
- **Medium zwischen Entwickler und Deployer**



# Komponentendiagramme

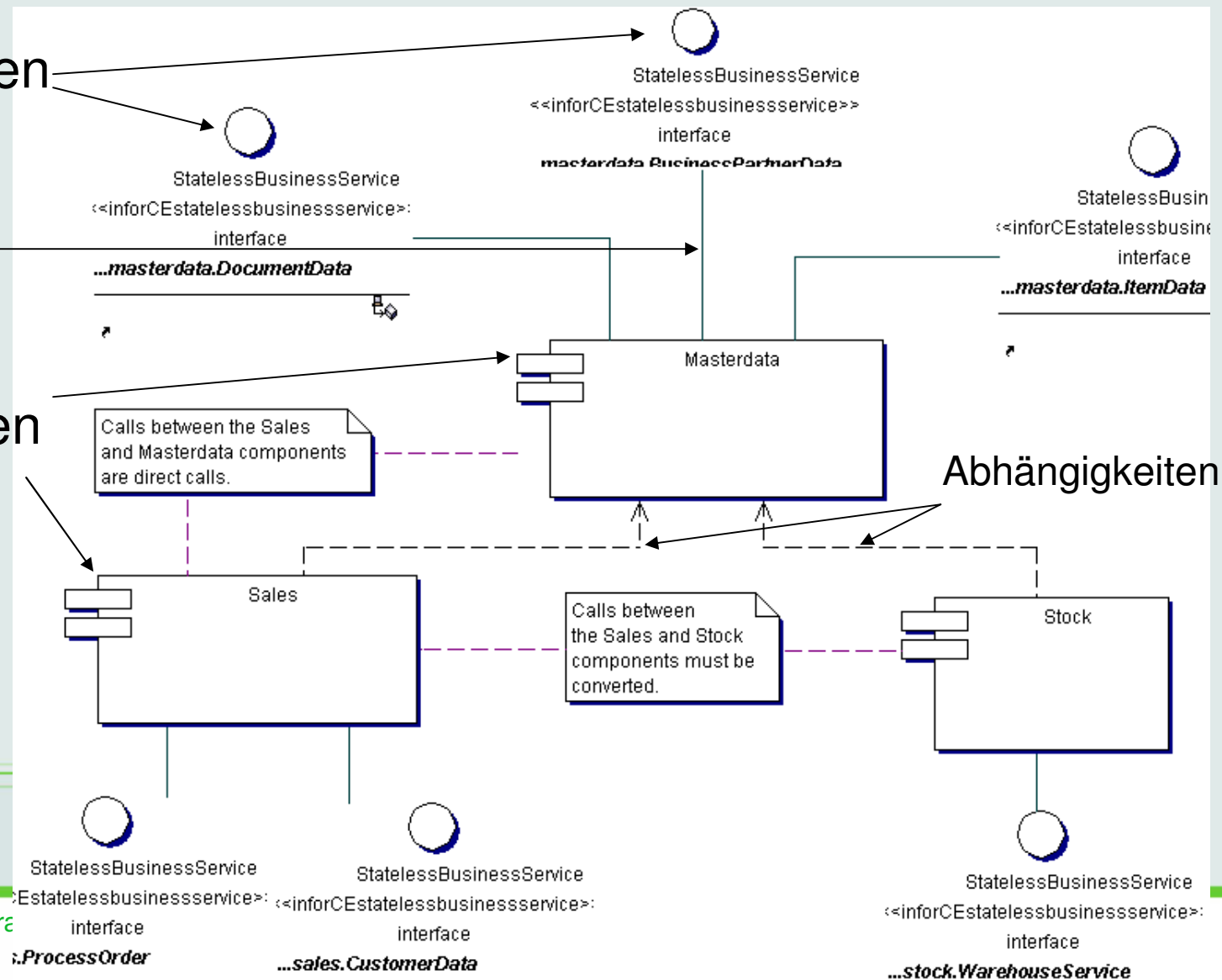
alles läuft leichter!

Schnittstellen

Support





Komponenten

Abhängigkeiten





## Bsp: Anlegen eines Geschäftsobjekts in Together/J

 Class	Ctrl+L	New	▶
 Interface	Ctrl+Shift+L	Properties...	Alt+Enter
 Class by Pattern		Add Shortcut...	Ctrl+Shift+A
 Package	Ctrl+E	Zoom	▶

**Choose Pattern - BusinessObject - Pattern properties**

Pattern

- Patterns
  - Coad Components
  - HP E Speak
  - inforce
    - BusinessObject**
    - SerializableObject
  - Coad Classes

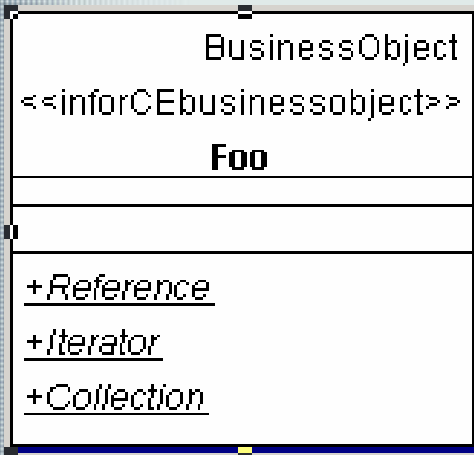
Name	Value
Business Object Class	de.infor.businessobjects.tutorial.Foo
Business Object Super Class	de.infor.businessobjects.BusinessObject
OR-Mapping Type	horizontal

Description

< Previous    Next >    Finish    Cancel    Help









alles läuft leichter!

## Deklarative Annotierungen



Properties [Foo]

infor:CE Properties Hyperlink View Description Javadoc Htmldoc Req Beans

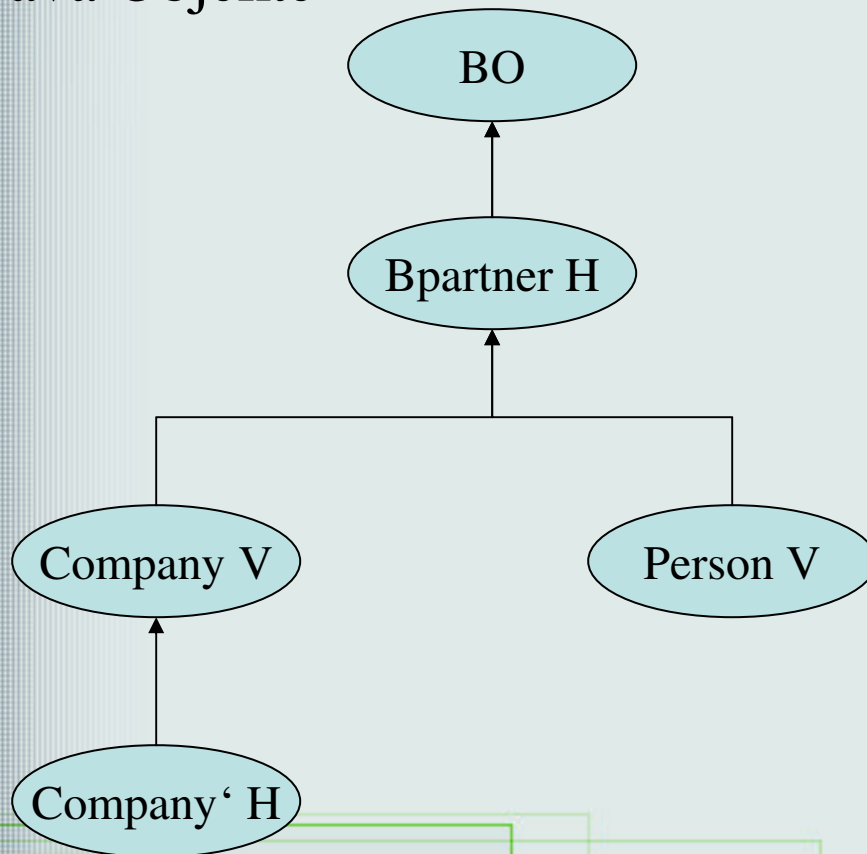
Name	Value
XML Namespace	
XML Type/SOAP Binding	
SOAP Error Code	
Roles Permitted to Read fr... 	
Roles Denied to Read fro... 	
Roles Permitted to Write in... 	
Roles Denied to Write into ... 	
OR-Mapping Type	horizontal ▼
Table in Database	

Ctrl+Enter applies changes and closes the Inspector

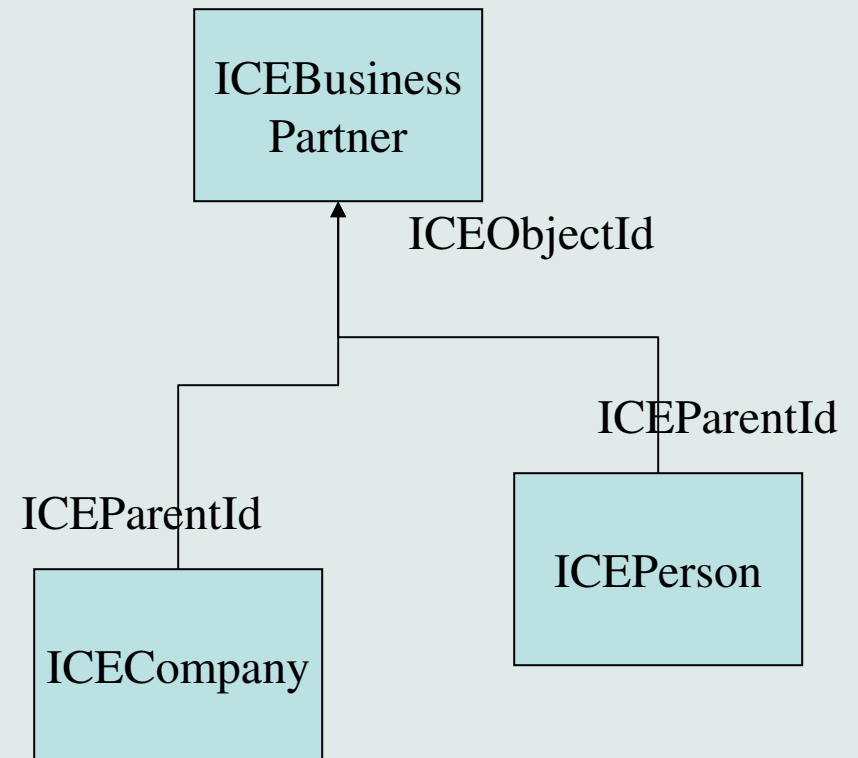
# Meta-Daten gesteuertes OR-Mapping

alles läuft leichter!

## Java Objekte



## SQL-Tabellen



## Persistente Geschäftsobjekte

- Geschäftsobjekte sind das OO-Pendant zu 4GL Views/Records
- erben von `class BusinessObject`
- tragen persistenten Zustand in Form von
  - primitiven Attributen
  - komplexen getypten Assoziationen (Relationen) zu anderen Geschäftsobjekten:
    - `interface BusinessObject.Reference`
    - `interface BusinessObject.Collection`  
(Pendant zu 4GL RecordSets)
    - `interface BusinessObject.Iterator`
  - komplexen Datenelementen (Maße, Intervalle, etc.)

# Primitive Attribute

alles läuft leichter!

The screenshot illustrates the process of adding a primitive attribute to a class in a software development tool. The main window shows a class diagram with two classes: **BusinessObject** (labeled `<<inforCEbusinessobject>>`) and **Foo**. A context menu is open over the **Foo** class, with the **New** option selected. This opens a sub-menu where **Property** (with the shortcut **Ctrl+B**) is highlighted. The **Primitive Attribute** inspector window is then displayed, showing various configuration options for the new attribute.

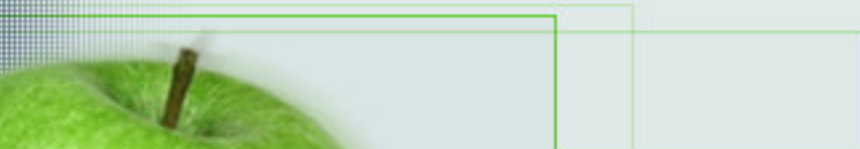
Property	Value
Disable serialization	<input type="checkbox"/>
Disable persistence	<input type="checkbox"/>
May not be null	<input type="checkbox"/>
Column in Database Table	
Is Part of a Primary Key	<input type="checkbox"/>
Is Part of Object Versioning	<input type="checkbox"/>
String Width	30
Roles Permitted to Read ...	
Roles Denied to Read fr ...	
Roles Permitted to Write ...	
Roles Denied to Write int...	

Ctrl+Enter applies changes and closes the Inspector

Software Engineering i

## Assoziationen zwischen Geschäftsobjekten

- Werden über getypte Schnittstellen repräsentiert: `Collection`, `Reference`
- Unterstützte Methoden: `set`, `get`, `add`, `remove`, `iterator`, `toArray`, `filter`
- Kardinalitäten
  - `*:1`, `1:*`, `*:*`
  - `0..1`, `1`, `*`, `1..*`
- Abhängigkeit („Ownership“)
  - Simple Assoziation
  - Aggregation
  - Komposition





## Anlage und Pflege von Assoziationen

**Choose Pattern - OneToMany Reference - Symmetric Attribute at Client**

Pattern

- Patterns
  - link
    - inforce
      - OneToMany Reference
      - ToMany Reference
      - ToOne Reference
    - JGL 3.1 containers

Name

Name	Value
Supplier Attribute Name	bars
LinkType	aggregation
Supplier Cardinality > 0	<input type="checkbox"/>
Expose Reference Add Method	<input type="checkbox"/>
Expose Reference Remove Method	<input type="checkbox"/>
Expose Collection	<input type="checkbox"/>

Description

Next > Finish Cancel Help

**Properties [bars]**

infor:CE Properties Hyperlink Description Javadoc Htmldoc Req

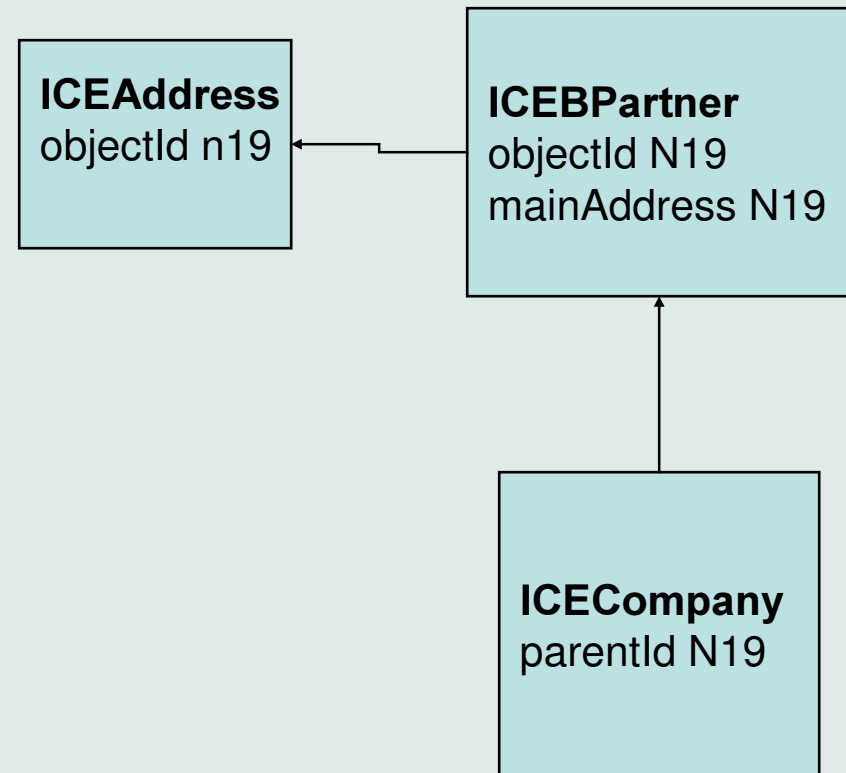
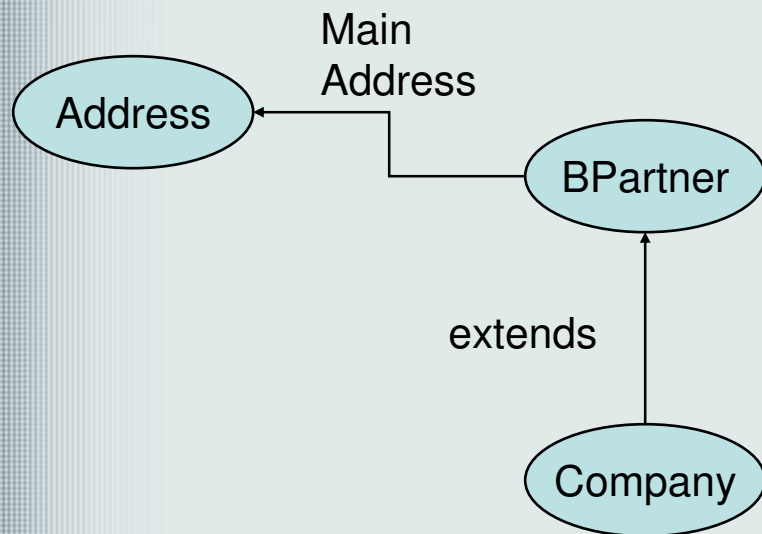
Name	Value
Disable serialization	<input type="checkbox"/>
Disable persistence	<input type="checkbox"/>
May be null	<input type="checkbox"/>
Column in Database Table	
No Shallow Retrieval	<input type="checkbox"/>
Backreferring Attribute in Target	refBackToFooX

Ctrl+Enter applies changes and closes the Inspector

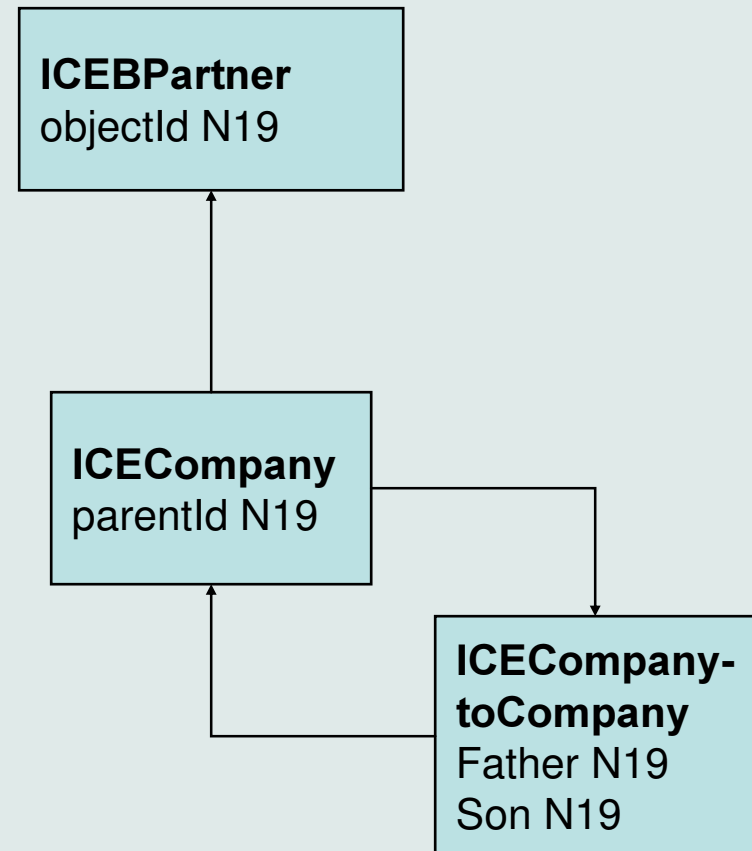
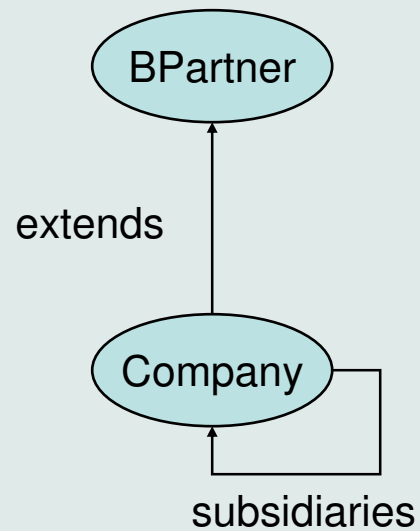
Sc

alles läuft leichter!

## \*:1 Assoziationen

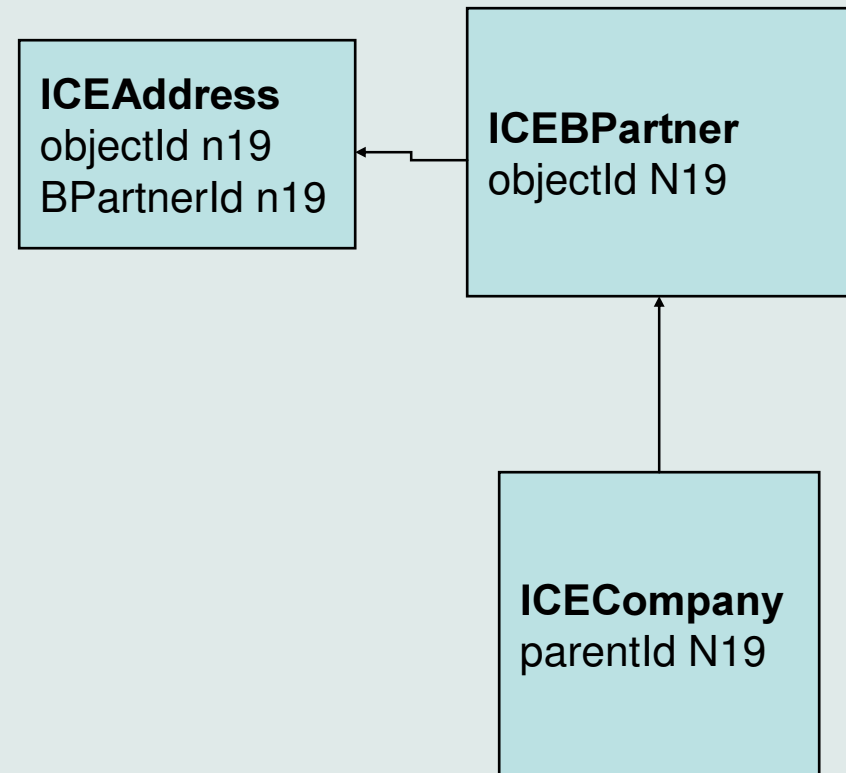
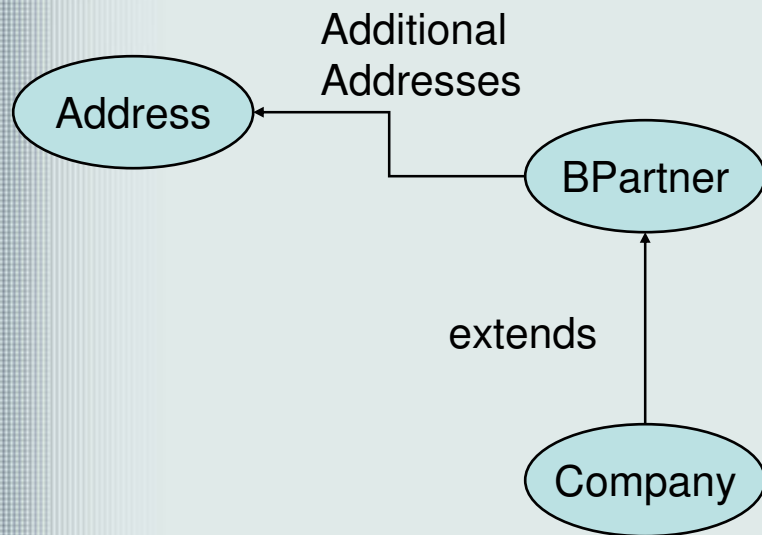


## \*:\* Assoziationen



alles läuft leichter!

## 1:\* Assoziationen



## Einschränkungen für Geschäftsobjekte

- Zugriff über Methoden (Strukturänderungen)
- keine Floats; Doubles zugunsten von BigDecimal(28,10)
- Strings haben maximale Länge
- keine nicht-serialisierbaren Attributtypen
- Tragen aus Wiederverwendbarkeitsgesichtspunkten keinerlei oder nur minimale Logik (keine DB-Zugriffe, etc.)
  - Werden nicht durch `new BusinessObject()` erzeugt!
  - Default (Protected) Konstruktor
  - Unterstützten erweitertes Callback/Visitor-Pattern:
    - ```
final public boolean visit(BusinessVisitor visitor)  
    throws VisitationException;
```
- → Code-Analyse, Audit-Tools



## Die Aufgabe von Geschäftsdiensten

- Die Methoden von `BusinessServiceBeans` erzeugen, manipulieren, persistieren, filtern und löschen Geschäftsobjekte.
- `BusinessService`-Schnittstellen exportieren einen Teil dieser Methoden zu anderen Services und zum Client.
- `BusinessService.Homes` sind Factory-Schnittstellen zum Erstellen/Freigeben von Geschäftsdiensten.
- *Connection-Handling (DB, IP, Queuing), Transaktionalität, Sicherheitsverhalten, etc. werden implizit von der Laufzeit übernommen*





## Anlegen von Diensten in Together/J

Choose Pattern - StatelessBusinessService - Pattern properties

Pattern

- Patterns
  - Coad Components
  - HP E Speak
  - inforce
    - BusinessObject
    - SerializableObject
    - StatefulBusinessService
    - StatelessBusinessService**
  - Coad Classes
  - GoF
  - EJB Client

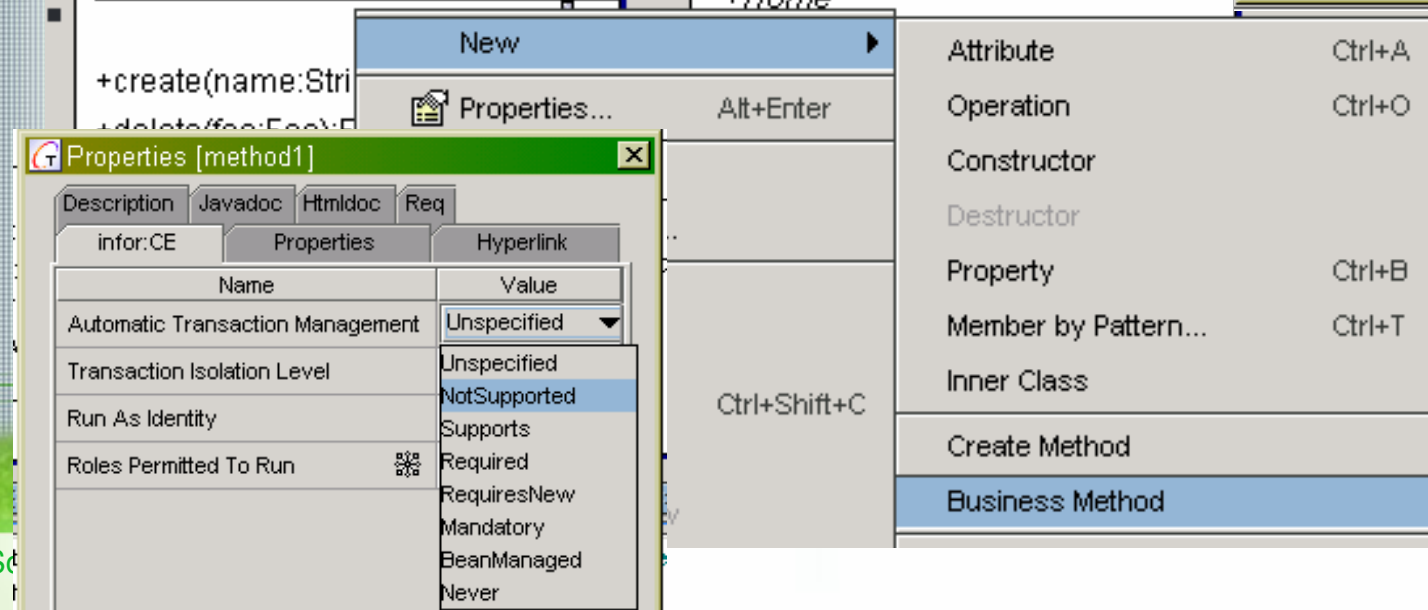
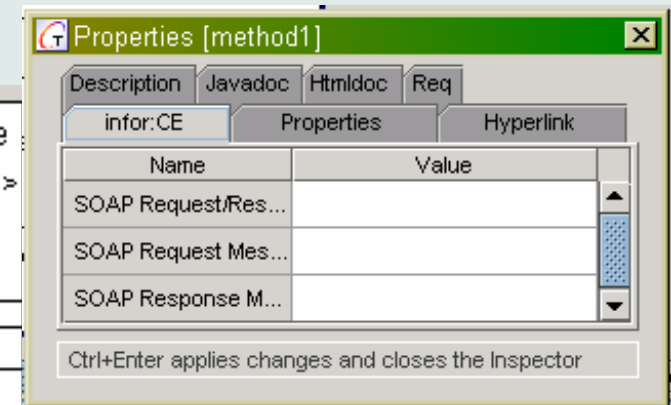
| Name                            | Value                                                   |
|---------------------------------|---------------------------------------------------------|
| Business Service Bean           | de.infor.businessservices.tutorial.FooAdministratorBean |
| Business Service (Remote Int... | de.infor.businessservices.tutorial.FooAdministrator     |
| Super Service                   | de.infor.businessservices.BusinessServiceBean           |
| Abstract                        | <input type="checkbox"/>                                |
| Automatic Transaction Manag...  | Required                                                |
| Transaction Isolation Level     | read committed                                          |
| Run As Identity                 | unknown                                                 |
| Roles Permitted To Run          |                                                         |

Description

< Previous   Next >   Finish   Cancel   Help

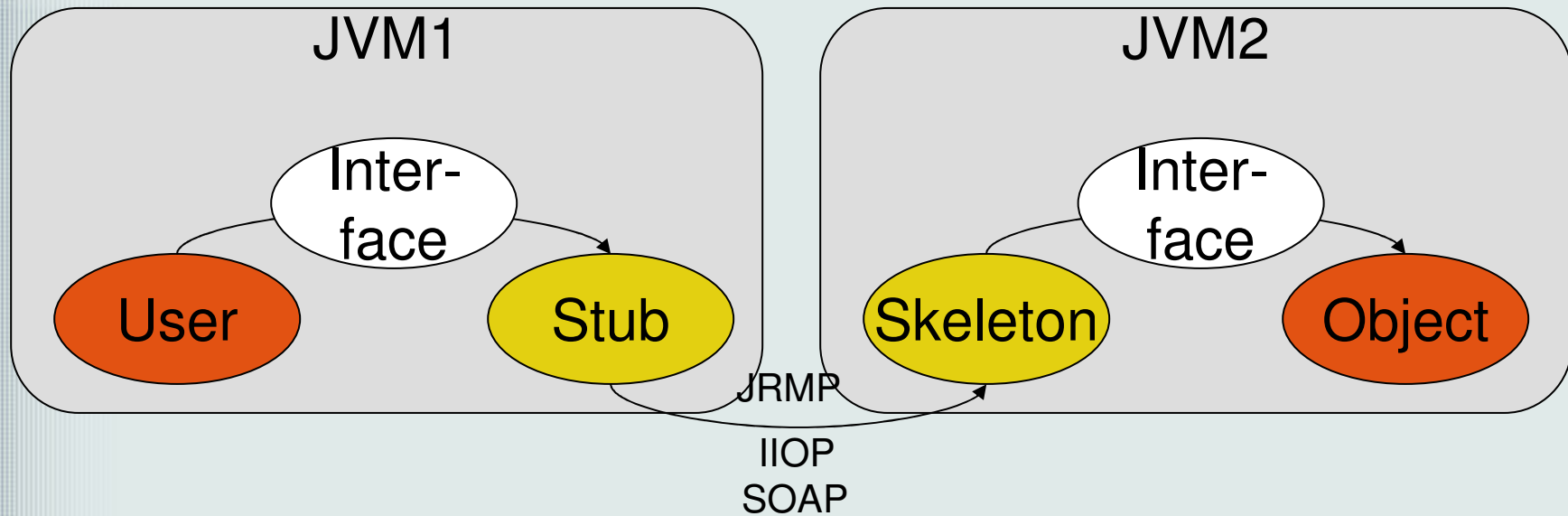
## Bean-Methoden und Annotationen

- Gemeinsame Pflege von Bean und Remote-Interface



# RMI-Stubs = Proxy-Pattern

alles läuft leichter!



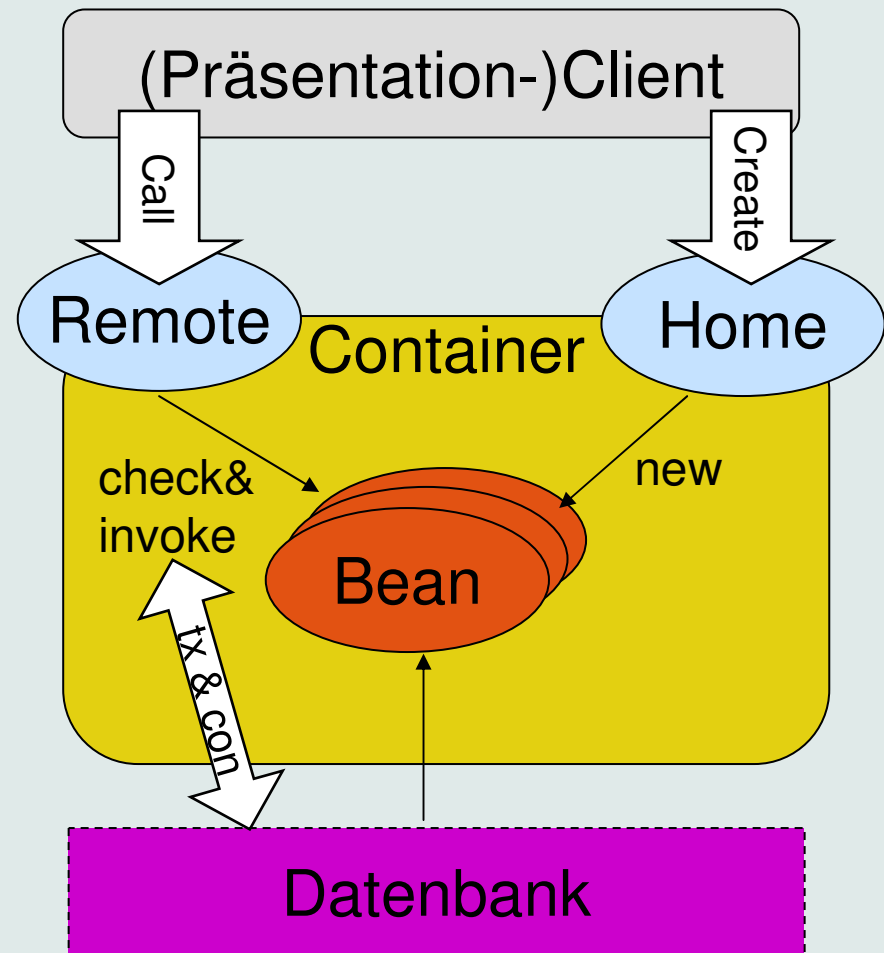
```
public interface Hello extends java.rmi.Remote {  
    public String sayHello() throws java.rmi.RemoteException;  
}
```

```
class HelloImpl extends java.rmi.server.UnicastRemoteObject implements Hello {  
    public String sayHello() { return "Hello World"; }  
}
```

# EJB „erweitert“ RMI

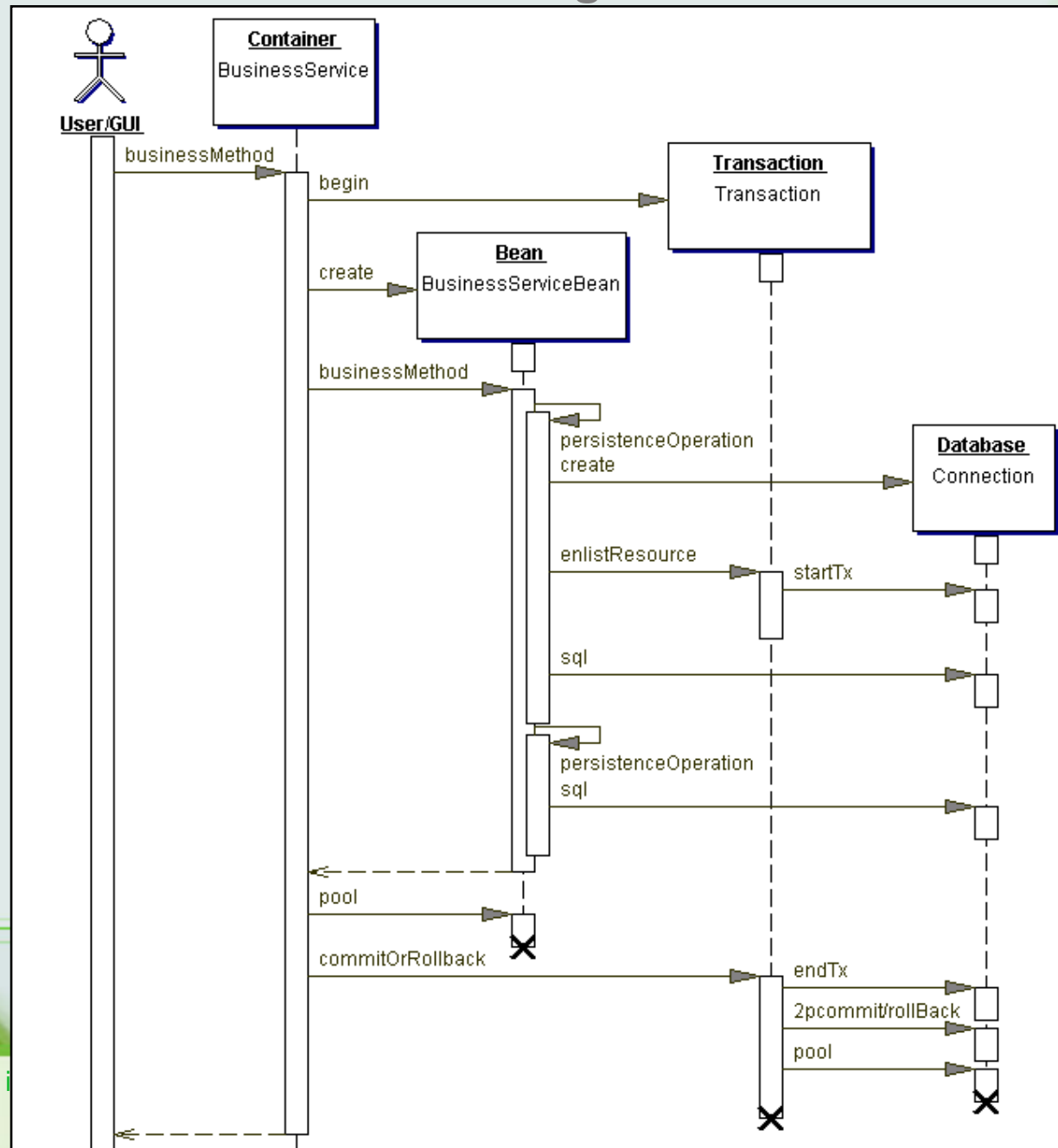
alles läuft leichter!

- Bean=Middle-Tier-Objekt
  - Methoden=Geschäftslogik
  - Zustand=Geschäftsdaten
  - Remote-Schnittstelle=BAPI
- Container=erweitertes Skeleton mit technischen Diensten
  - Naming
  - Lifecycle
  - ...
- Home
  - exportierte Factory-Schnittstelle



# Deklaratives Transaktionshandling

alles läuft leichter!



alles läuft leichter!

## Deployment=Installation

**ejb-jar.xml - Editor**

Datei Bearbeiten Format ?

**Meta-Daten**

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar>
  <enterprise-beans>
    <session>
      <description>Hello world Example</description>
      <ejb-name>session/Hello</ejb-name>
      <home>session.HelloHome</home>
      <remote>session.Hello</remote>
      <ejb-class>session.HelloBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Bean  
Byte-Code

**WinZip - session.jar**

File Actions Options Help

New Open Favorites Add

Name	Type	Path
ejb-jar.xml	XML Document	meta-inf\
Manifest.mf	MF File	meta-inf\
Hello.class	CLASS File	session\
HelloBean.class	CLASS File	session\
HelloHome.class	CLASS File	session\
User.class	CLASS File	session\

Selected 1 file, 2KB Total 6 files, 4KB

JAR-File

Container



alles läuft leichter!

# Erstellen und Deployen von Geschäftskomponenten

infor:CE Tool Suite

infor:CE (c) 2000-2001 infor business solutions AG

**infor**  
component engine

ModellImport SchemaGenerator XmlGenerator EjbGenerator Packager Deployer

Input Specification Global Configuration Compiler DependencyFinder XmlExport

sourceFiles de.infor.businessobjects.tutorial Remove

borders Remove

schemaFiles Add Remove

dependingComponents Add Remove

componentFile W:\inforCE\dist\lib\components\tutorial.jar ...

dependencies Remove

xmiFile ...

clientComponentFile W:\inforCE\dist\lib\client\components\tutorial.jar ...

contractFile ...

dataaccessSchema ...

description

createSQLCommand ...

EJBJarFile ...

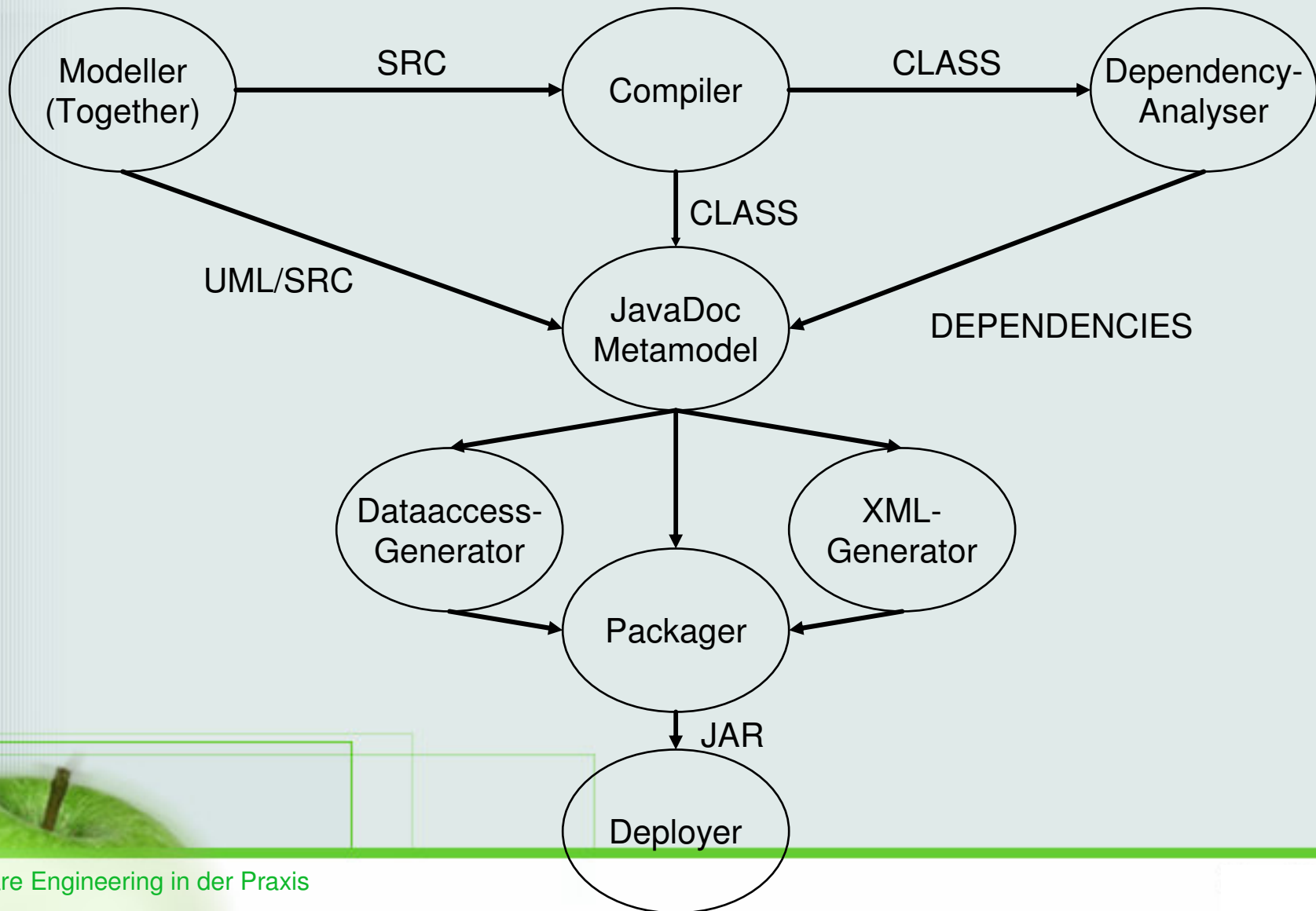
Logging messages will appear here.

0%

OK Cancel

# Aufbau des Composer

alles läuft leichter!



## Projektverwaltung mit Source Code Management

- Verteilte Teamarbeit über ein Filesystem undenkbar.
- Simultane Arbeiten an einem einzigen Systemstand unmöglich.
- Globales Repository zur konsistenten Speicherung und Synchronisation von lokalen Zwischenständen von:
  - Source Code
  - Dokumentation
  - Modellierung
  - Byte-Code
- Nützliche Tools: WinCVS, P4
- Wichtigste Operationen: Diff, Merge, Resolve, Lock
- Parallele Entwicklung:
  - Labels
  - Branches

# Ein toolgestützter Entwicklungszyklus

alles läuft leichter!

- Einzelne Projekte verzweigen als Branches aus der Mainline.
- Regelmässige Integration der Mainline in den Branch.
- Entwickler bekommt globale Dokumentation (UML/HTML)
- Entwickler bekommt alle notwendigen Sourcen
  - der eigenen Komponenten
  - der öffentlichen Schnittstellen von externen Komponenten
  - Änderungsrecht nur auf den für ihn relevanten Dateien.
- Compile → Test|Doc → Link
- Byte-Code und Doku werden nur bei erfolgreichem Compile und Ablauf der Unit-Tests gebaut.
- Im Projekt-Branch wird nach eigenem Gutdünken eingchecked.
- Integration: Branch → Mainline nur Source und nach Absprache mit QS.
- Binärdateien der Mainline werden von einem Masterbuild-Prozess eingchecked
  - Checkout → Compile → Test|Doc → Link → Integration-Tests → Checkin

## Das Make-Tool Ant (<http://jakarta.apache.org>)

- Java-basiertes Make
- Viele, für Java typische Tasks eingebaut
- Externe batches
- Erweiterbar um eigene tasks, Logger, EventHandler (z.B.: Mail-Verschicken durch Masterbuild) mittels Java Programmierung
- Make-Dateien sind XML-basiert



## Ant-Makefiles

```
<project name=„ module“ default=„dist">
  <target name=„compile">
    <javac srcdir=„module/src/" destdir=„build/module/lib/classes/" >
      <include name="**/*.java"/>
    </javac>
  </target>

  <target name=„jar“ depends=„compile">
    <jar jarfile="build/lib/module.jar"
      basedir=„build/module/lib/classes" includes="**"/>
  </target>

  <target name=„dist" depends=„jar">
    <copy toDir=„dist/lib/">
      <fileset dir=„build/lib/module.jar"/>
    </copy>
  </target>

</project>
```



# Build & Test

alles läuft leichter!

```
<target name="jar" depends="runtests">
  <jar jarfile="build/lib/module.jar"
      basedir="build/module/lib/classes" includes="**"
      excludes="**/test/*.class"/>
</target>

<target name="runtests" depends="compile" if="junit.present">
  <java fork="yes" classname="junit.textui.TestRunner"
      taskname="junit" failonerror="true">
    <arg value="module.test.ModuleTest"/>
    <classpath>
      <pathelement location="build/modules/lib/classes/" />
      <pathelement path="${java.class.path}" />
    </classpath>
  </java>
</target>

<target name="scm-checkout"/>
<target name="scm-checkin"/>
```



## Automatisches Testen

- Möglich geworden durch komponenten-basierte 3-Schicht Architektur.
  - Regressionsstrategie für verschiedene Schichten
  - Tests für separierte Teilfunktionalitäten
- Wichtiger Teil von inkrementellem Vorgehen, bzw. eXtreme Programming.
- Verfassen von Testcode vor, bzw. beim Entwickeln.
- Regelmässiger Ablauf des Testcodes stellt Funktionalität und Stabilität sicher.



# Trade-offs beim Automatischen Testen

alles läuft leichter!

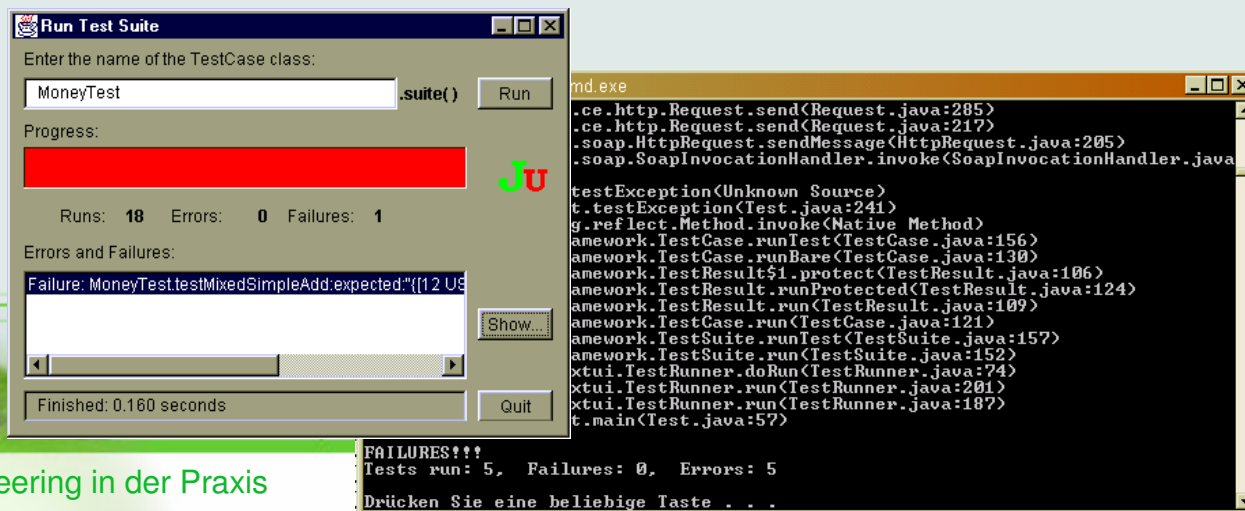
- Vorteile:
  - Sicherheit für den Entwickler
  - Team-Arbeit entlastet
  - Spezifische Hinweise auf Fehlerquellen, Fehlspezifikationen
- Nachteile:
  - Anforderungen an Code-Disziplin
  - Erfordert genaue Spezifikation!
  - Für GUIs problematisch ...
- Abwägen der Kosten von Testerstellung versus Fehlersuche
- Auch Tests müssen designed werden!
- Recorder/Robots für GUI



JUnit (<http://www.junit.org>)

alles läuft leichter!

- Ein „Minimal“-Framework, um Testcode in Java zu schreiben und ablaufen zu lassen.
  - Assertions: Erwartungen an Variablenbelegung
  - Test-Methode: Eine abgeschlossene Logik wird aufgerufen; Assertions werden sichergestellt und geloggt.
  - Test-Fall: Menge von Test-Methoden mit gleicher Initialisierung und gleichem Clean-Up.
  - Test-Suite: Zusammenstellung von Test-Methoden sowie untergeordneten Test-Suites.
- Interaktiver versus Batch-Modus



# Verschiedene Arten von Tests

alles läuft leichter!

- Blackbox-Tests sind auf der Präsentationsschicht angesiedelt und können die spezifischsten Use-Cases unmittelbar verifizieren.
  - Web-Tests: Servlet-basierte TestCases
- Whitebox-Tests sind auf der Anwendungsschicht angesiedelt, haben Zugriff auf innere Zustände der Logik und können so die Integrität sichern.
- Performance- und Lasttests
  - Erweiterung der Blackbox-Tests
- Tests können mit der eigentlichen Logik zusammen auch beim Kunden installiert und zur Analyse von Fehlkonfigurationen herangezogen werden.
- Tests können als Grundlage zum Aufspüren von Fehlern durch Debuggen oder zum Messen von Performance-Engpässen durch Profiling dienen.

# Remote Debugging

alles läuft leichter!

- Ausschriebe sind kein effektives Mittel zur Fehlerbehebung.
- moderne virtuelle Maschinen bieten spezielle Remote-Schnittstelle zum Debugging an (auch übers Internet/Firewall zum Kunden).
- externe IDE wird „attached“.
- Selten: Debugging zwischen den Schichten.
- Debuggen von aus Templates erzeugtem Code:
  - JSP-Debugger z.B. in JBuilder
- Good Practice: Source als „Doku“ mit in den Byte-Code stecken.

- Funktionalität vor Performanz
  - Bottlenecks nicht vorhersehbar
  - Performanz-Tests mit geeignete Setup/Teardown Methoden zur Messung
  - Nebenläufige Stress-Tests
- Tool-Support um Bottlenecks zu finden
  - Memory-Leaks auch in modernen Sprachen mit Garbage-Collection möglich.
- Ähnliche Schnittstellen zu Debugging.
- Memory- versus CPU Profiling
- Manueller oder Programmatischer Betrieb
- On-line versus Off-line Profiling

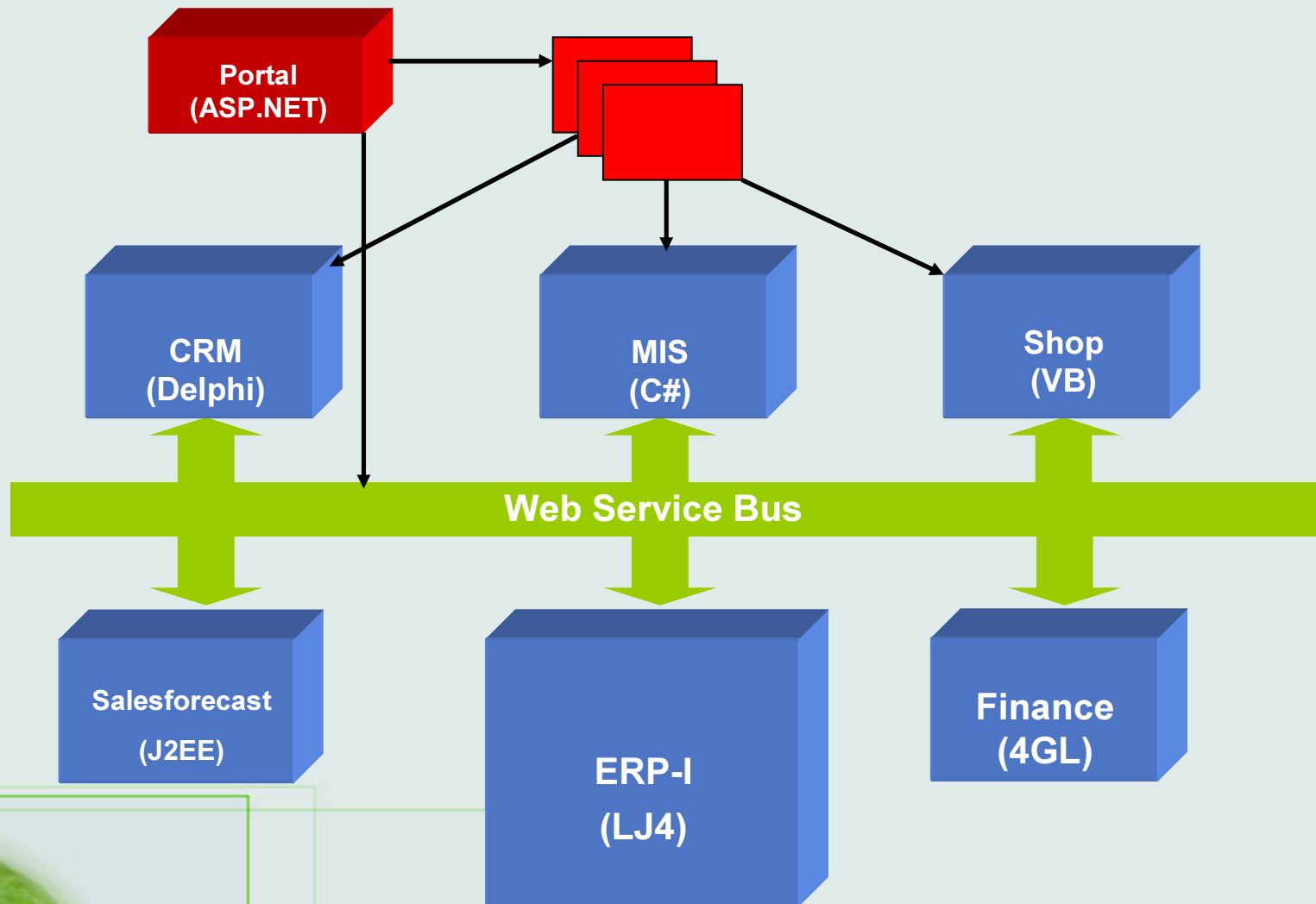


## Good Practices

- Alle Entwickler verwenden zum Build denselben Compiler
- Inkrementelles Compilieren nur zum Syntax-Check.
- gemeinsame IDE mit Erweiterungen vorhanden, Benutzung freigestellt
- (Java-)doc Kommentare!
- Namens- und Strukturkonventionen für Pakete, Klassen, Schnittstellen und Methoden
  - Code-Guide
  - Templates
  - Design Patterns
- Keine \* imports.
- println() vermeiden.
  - Für Traces ist der Debugger da.
  - flexibles Logging-Paket (z.B. log4j)
  - Internationalisierung
- Inspektion der existierenden Quellen von grossen Systemen (Apache, JBoss, etc)

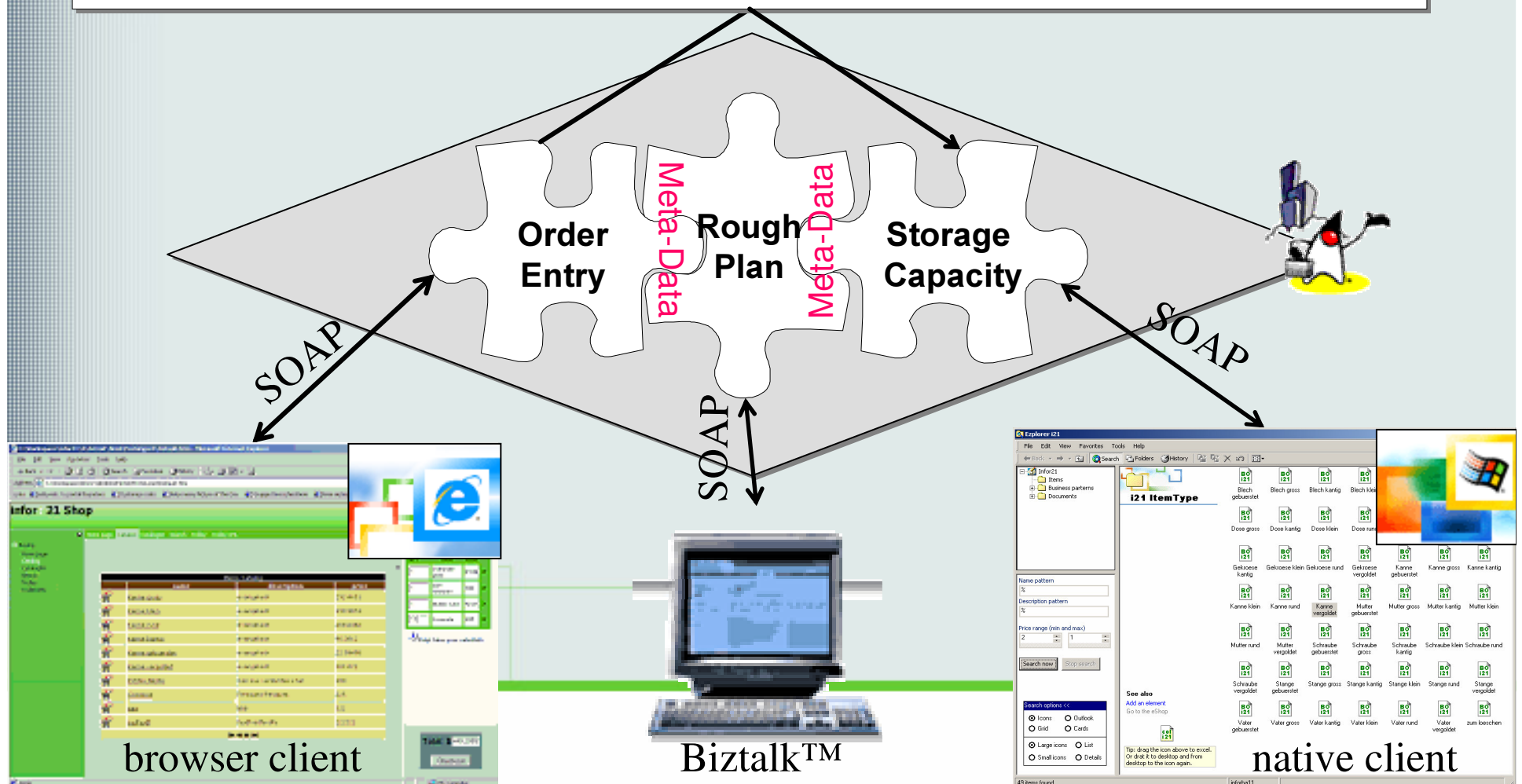
## Recap: Technologische Vision infor:COM

alles läuft leichter!



```
<?xml version="1.0"?>
<SOAP:Envelope>
  <SOAP:Body>
    <store:reserveItemRequest xmlns:store="http://www.infor.de/store">
      <item md:id="2121" md:desc="screwdriver" xmlns:md="http://www.infor.de/masterData"/>
      <quantity>20</quantity>
    </store:reserveItemRequest>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP = XML over HTTP/SMTP



```

<schema
  targetNamespace=
    „http://www.infor.de/masterData“>
  <complexType name=„Item“>
    <attribute name=„id“
      type=„int“/>
    <attribute name=„desc“
      type=„string“/>
  </complexType>
</schema>

```

Item.xsd

```

<definitions
  targetNamespace=„http://www.infor.de/store“
  xmlns:md=„http://www.infor.de/masterData“>
  <types>
    <include schemaLocation=„Item.xsd“/>
  </types>
  <message name=„reserveItemsRequest“>
    <part name=„item“ type=„md:Item“ />
    <part name=„quantity“ type=„xsd:int“/>
  </message>
  <portType name=„EmployeeService“>
    <operation name=„reserveItems“>
      <input message=„reserveItemsRequest“/>
      <fault message=„StoreException“ />
    </operation>
  </portType>
</definitions>

```

Warehouse.wsdl

*Application Tier*

**BusinessServices**

**BusinessObjects**

## Praktisches Software-Engineering

- Nah an den theoretischen Konzepten.
- Nicht ganz so „sauber“, wie man es sich in der Theorie vorstellt.
- Umso anspruchsvoller, mit den Randbedingungen zu leben:
  - Kundenanspruch.
  - Existierende Code- und Entwicklerbasis.
  - Externe Tools und Produkte sinnvoll einbinden.
- Technologie ist kein Selbstzweck!
  - Produktvision vor Technologievision.
  - Geeignete Standards müssen identifiziert und integriert werden.
  - Aktuelle Trends müssen auf Substanz und Relevanz geprüft werden.



alles läuft leichter!

**Guten Appetit!**