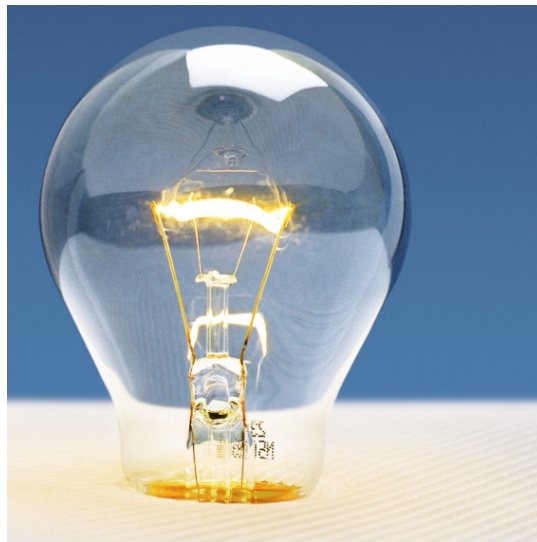
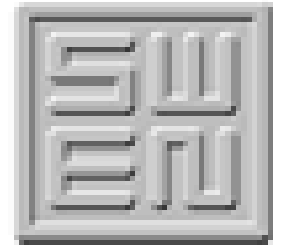

Persistenz mit Hibernate

MDSD im Einsatz

SWEN-Talk, 18. März 2008, FH Brugg-Windisch

Ulrich Brawand, Zühlke Engineering AG



- Einführung / Grundlagen
- Hibernate → wo liegen die Probleme
- Prinzipien von MDSD
- Metamodelle und Modell
- Generierbare Architektur
- Generieren mit oAW
 - Modell überprüfen
 - POJO's erzeugen
 - Hibernate-Konfigurationsdatei erzeugen
 - DB Schema erstellen (inkrementell)
- Testen
- Zusammenfassung



MDSD mit OpenSource

Slide 2
18. März 2008 18. März 2008

Einführung / Grundlagen

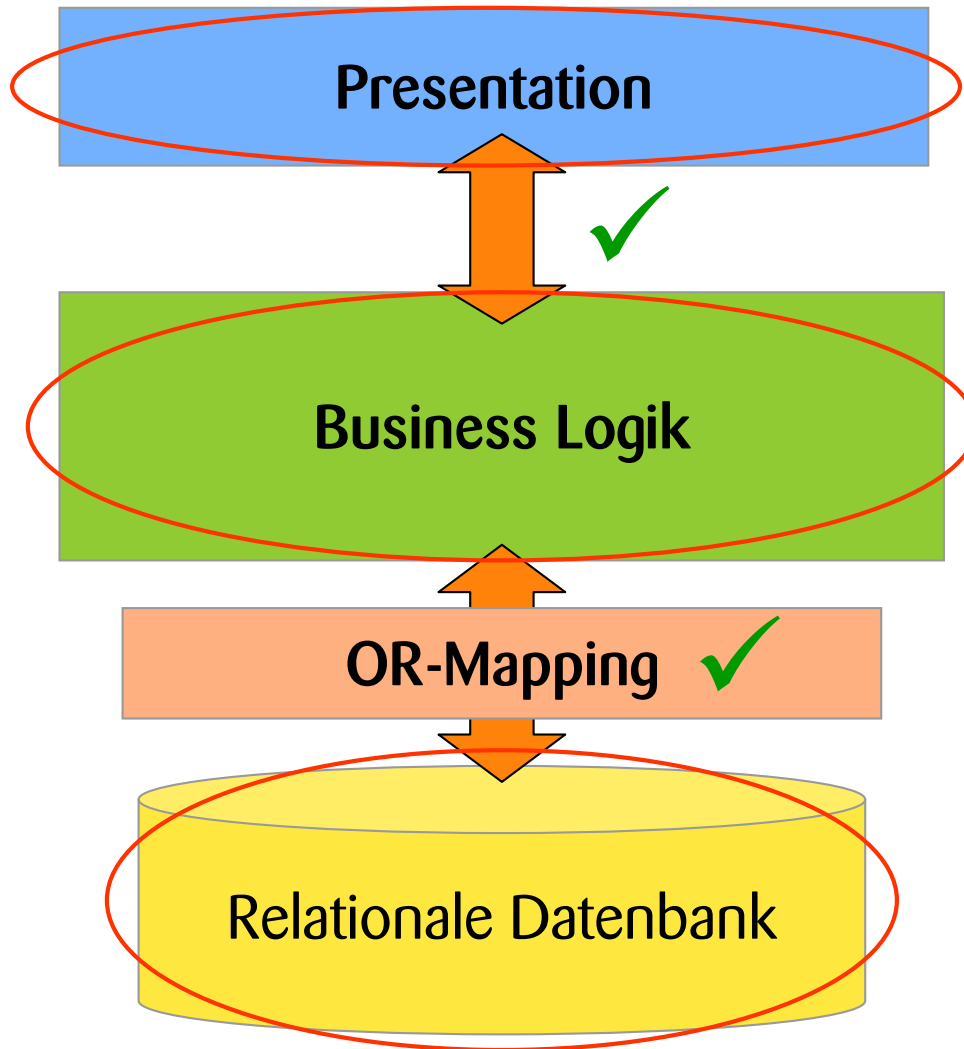
Was muss ich wissen?



MDS mit OpenSource

Slide 3
18. März 2008

Objektorientierung in Applikationen



MDS mit OpenSource

Slide 4
18. März 2008 18. März 2008

Wie können Objekte länger leben als die Applikation?

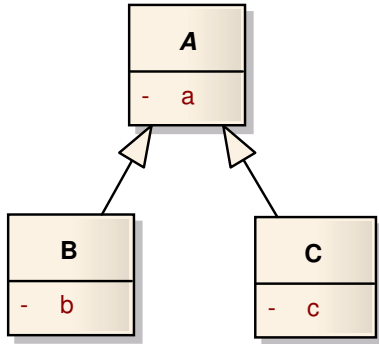
- **Binäre Serialisierung in eine Datei**
 - „Alles oder nichts“
 - Datenverlust bei Strukturänderung
- **XML-Serialisierung**
 - Keine generische Auswahl möglich
 - Keine Transaktionen
- **Mit DAO auf RDB zugreifen**
 - Schnell und leistungsfähig
 - Funktionalität des DBMS einsetzen
 - Lesen und speichern individuell pro Klasse
- **OR-Mapping Frameworks und RDB**
 - Konfiguration notwendig
 - Spezialfälle nur schwer abbildbar



MDS mit OpenSource

Slide 5
18. März 2008 18. März 2008

Abbildung von Vererbung



Eine Tabelle

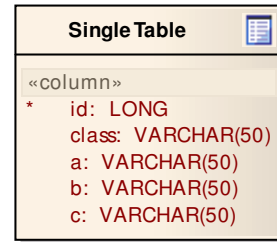


Tabelle pro konkrete Klasse

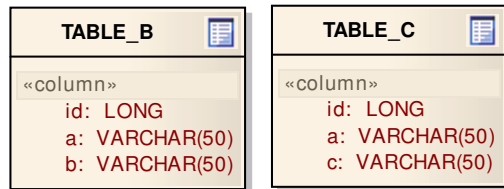
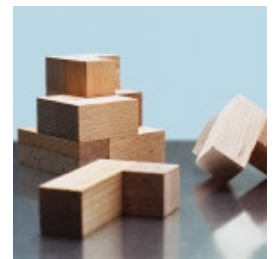
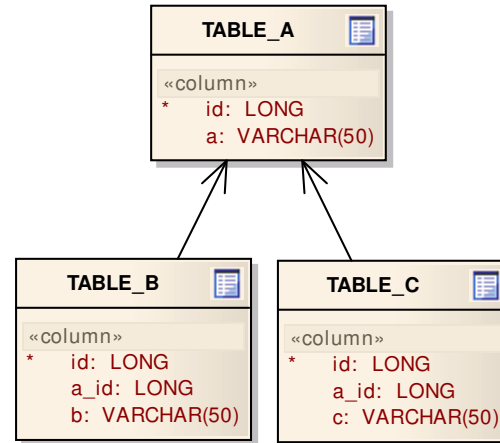


Tabelle pro Klasse



MDSO mit OpenSource

Slide 6
18. März 2008 18. März 2008

Ulrich Brawand
© Zühlke 2007

- **Ein Fetch pro Objekt**
 - Naheliegender und einfach
 - Bei vielen Objekten schlechte Performance
- **Ein Fetch pro Tabelle**
 - Einfach, bei Tabelle pro konkretes Objekt
 - Schwieriger bei SingleTable oder Full Table
- **Joined Fetch (alle Tabellen pro Klasse)**
 - Effizient
 - Teilobjekte auch laden oder eigener Fetch?
- **Caching**
 - Management der Objekte notwendig
 - Wann nachladen von DB
- **Locking**
 - Welche Strategie?



MDS mit OpenSource

Slide 7
18. März 2008 18. März 2008

Wo wird ein Attribut eines Geschäftsobjekts verwendet?

- Datenbank-Tabelle
- Konfiguration OR-Mapping
- Business-Object
- Data Transfer Object (Value Object)
- Search Object
- UI-Element

Ev. auch noch

- Business Service
- Web Service
- UI Backing Beans (JSF)
- Export File



MDSO mit OpenSource

Slide 8
18. März 2008 18. März 2008

Hibernate

Wo liegen die Probleme?



MDS mit OpenSource

Slide 9
18. März 2008

■ Allgemein

- Lösung für OR-Mapping
- Neue Probleme / Komplexität in Projekt

■ Framework

- Unterschiedliche Parametrierungsmöglichkeiten
- HQL oder SQL
- Spezialprobleme nicht lösbar?

■ Konfigurationsdatei vs. Annotations im Code

- „Verschmutzen“ des Codes
- Nachträgliche Änderungen ohne Kompilation
- Alle Informationen an einem Punkt

■ Wird verändert/verbessert

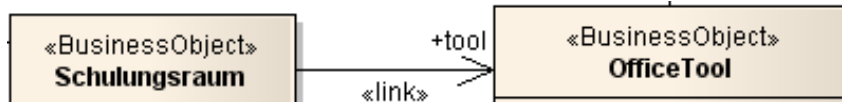
- Neue Releases → neue Möglichkeiten



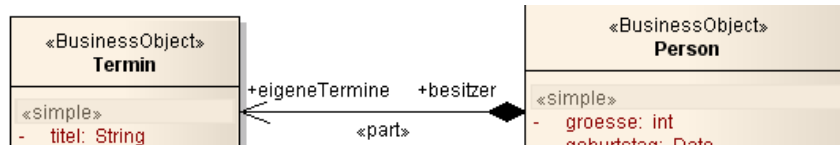
MDSD mit OpenSource

Slide 10
18. März 2008 18. März 2008

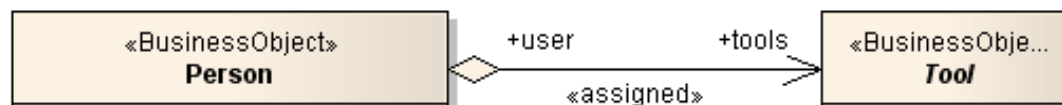
■ Look-up Beziehung (N:1)



■ Ganzes-Teil-Beziehung (1:N)



■ Zuteilungs-Beziehung (M:N)



■ Uni- oder Bi-Direktional?



MDS mit OpenSource

Slide 11
18. März 2008 18. März 2008

Mapping: Lookup (N:1)

Schulungsraum → OfficeTool



■ Uni-Direktional

– Konfiguration

```
<many-to-one name="tool" column="tool_id" />
```

– Bean-Code

```
OfficeTool tool;
```

– SQL-Code

```
tool_id integer,
```

■ Bi-Direktional (zusätzlich bei OfficeTool)

– Konfiguration

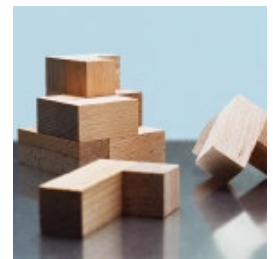
```
<bag inverse="true" name="rooms">  
  <key column="tool_id"/>  
  <one-to-many class="Schulungsraum"/>  
</bag>
```

– Bean-Code

```
List<Schulungsraum> rooms = new ArrayList<..>();
```

– SQL-Code

nicht benötigt.



MDSD mit OpenSource

Slide 12
18. März 2008 18. März 2008

Mapping: Ganzes-Teil-Beziehungen (1:N)

Person → Termin

■ Uni-Direktional

– Konfiguration

```
<set name="eigeneTermine" table="TERMINE">  
  <key column="besitzer"/>  
  <one-to-many class="Termin"/>  
</set>
```

– Bean-Code

```
Set<Termin> eigeneTermine = new HashSet<Termin>();
```

– SQL-Code

```
besitzer integer
```

■ Bi-Direktional (zusätzlich bei Termin)

– Konfiguration

```
<many-to-one column="besitzer" name="owner"/>
```

– Bean-Code

```
Person owner;
```

– SQL-Code

nicht notwendig.

und bei Person:

```
<set inverse="true" name ...
```



MDSD mit OpenSource

Slide 13
18. März 2008 18. März 2008

Mapping: Zuteilungs-Beziehungen (M:N)

Person → Tool Teil 1



■ Uni-Direktional

– Konfiguration

```
<list name="tools" table="TOOL_USERS">
  <key column="person_id"/>
  <list-index column="ind"/>
  <many-to-many class="Tool" column="tool_id"/>
</list>
```

– Bean-Code

```
List<Tool> tools = new ArrayList<Tool>();
```

– SQL-Code

```
create table TOOL_USERS (
  ID integer primary key,
  person_id integer,
  tool_id integer,
  ind integer);
```



MDS mit OpenSource

Slide 14
18. März 2008 18. März 2008

Mapping: Zuteilungs-Beziehungen (M:N)

Person → Tool Teil 2



■ Bi-Direktional (zusätzlich bei Tool)

– Konfiguration

```
<set inverse="true" name="user" table="TOOL_USERS">  
  <key column="tool_id"/>  
  <many-to-many class="Person" column="person_id"/>  
</list>
```

– Bean-Code

```
Set<Person> user = new HashSet<Person>;
```

– SQL-Code nicht notwendig.



MDS mit OpenSource

Slide 15
18. März 2008 18. März 2008

Vererbung: Single Table Ressource → Beamer



■ Vererbungsbeziehung

– Konfiguration

```
<subclass discriminator="Beamer" extends="Ressource"
            name="Beamer">
  <property column="..." />
</subclass>
```

– Bean-Code

```
class Beamer extends Ressource;
```

– SQL-Code

```
RESSOURCE_TYPE varchar(255),
```



MDS mit OpenSource

Slide 16
18. März 2008 18. März 2008

Vererbung: Leaf Table Alert → RemoteAlert



■ Vererbungsbeziehung

– Konfiguration

```
<union-subclass extends="Alert" name="RemoteAlert"
    table="REMOTE_ALERT">
  <property column="..." />
</union-subclass>
```

– Bean-Code

```
class RemoteAlert extends Alert;
```

– SQL-Code

```
create table REMOTE_ALERT (
  ID integer primary key,
  lockid integer,
  meldung varchar(250),
  zeitpunkt Date,
  email varchar(250) );
```



MDSD mit OpenSource

Slide 17
18. März 2008 18. März 2008

Vererbung: Full Table Tool → Software

■ Vererbungsbeziehung

– Konfiguration

```
<joined-subclass extends="Tool" name="Software"
    table="SOFTWARE">
  <key column="tool_id"/>
  <property column="..."/>
</joined-subclass>
```

– Bean-Code

```
class Software extends Tool;
```

– SQL-Code

```
create table SOFTWARE (
  ID integer primary key,
  tool_id integer,
  version varchar(250) );
```



MDSD mit OpenSource

Slide 18
18. März 2008 18. März 2008

■ Komplexität des Business-Modell

- Wie soll Business abgebildet werden?
- Wie führe ich Änderungen in den Attributen nach?
- Wie kann ich nachträglich Beziehungen einfügen oder löschen?
- Wie kann ich Vererbung in Business-Objekten abbilden?

■ Einsatz von Hibernate

- Wie soll Hibernate eingesetzt werden?
- Wie halte ich Hierarchien konsistent (Mapping Type)?
- Wie soll ich Beziehungen abbilden?
- Wie sollen Objekte bei Beziehungen geladen werden?
- Wie kann ich Mappingdefinition mit Modell synchron halten?
- Wie soll Locking implementiert werden?

■ Definition der Datenbank

- Welche Tabellen sollen angelegt werden?
- Wie sollen die Attribute benannt werden?
- Wie halte ich das Mapping mit den Tabellen synchron?
- Wie kann ich Änderungen in der DB-Struktur nachführen?



MDS mit OpenSource

Slide 19
18. März 2008 18. März 2008

■ Komplexität des Business-Modell

- Wie soll Business abgebildet werden?
- Wie führe ich Änderungen in den Attributen nach?
- Wie kann ich nachträglich Beziehungen einfügen oder löschen?
- Wie kann ich Vererbung in Business-Objekten abbilden?

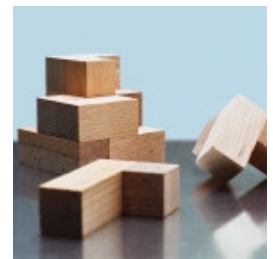
■ Einsatz von Hibernate

- Wie soll Hibernate eingesetzt werden?
- Wie kann ich Hierarchien konsistent (Mapping-Tools)?
- Wie soll ich Beziehungen abbilden?
- Wie sollen Objekte bei Lazy Loading geladen werden?
- Wie kann ich Mapping-Tools mit Modell synchron halten?
- Wie kann ich Mapping-Tools komplementiert werden?

■ Definition der Datenbank

- Welche Tabellen sollen angelegt werden?
- Wie sollen die Attribute benannt werden?
- Wie halte ich das Mapping mit den Tabellen synchron?
- Wie kann ich Änderungen in der DB-Struktur nachführen?

Wie können wir Komplexität reduzieren?



MDS mit OpenSource

Slide 20
18. März 2008 18. März 2008

Prinzipien von MDSD

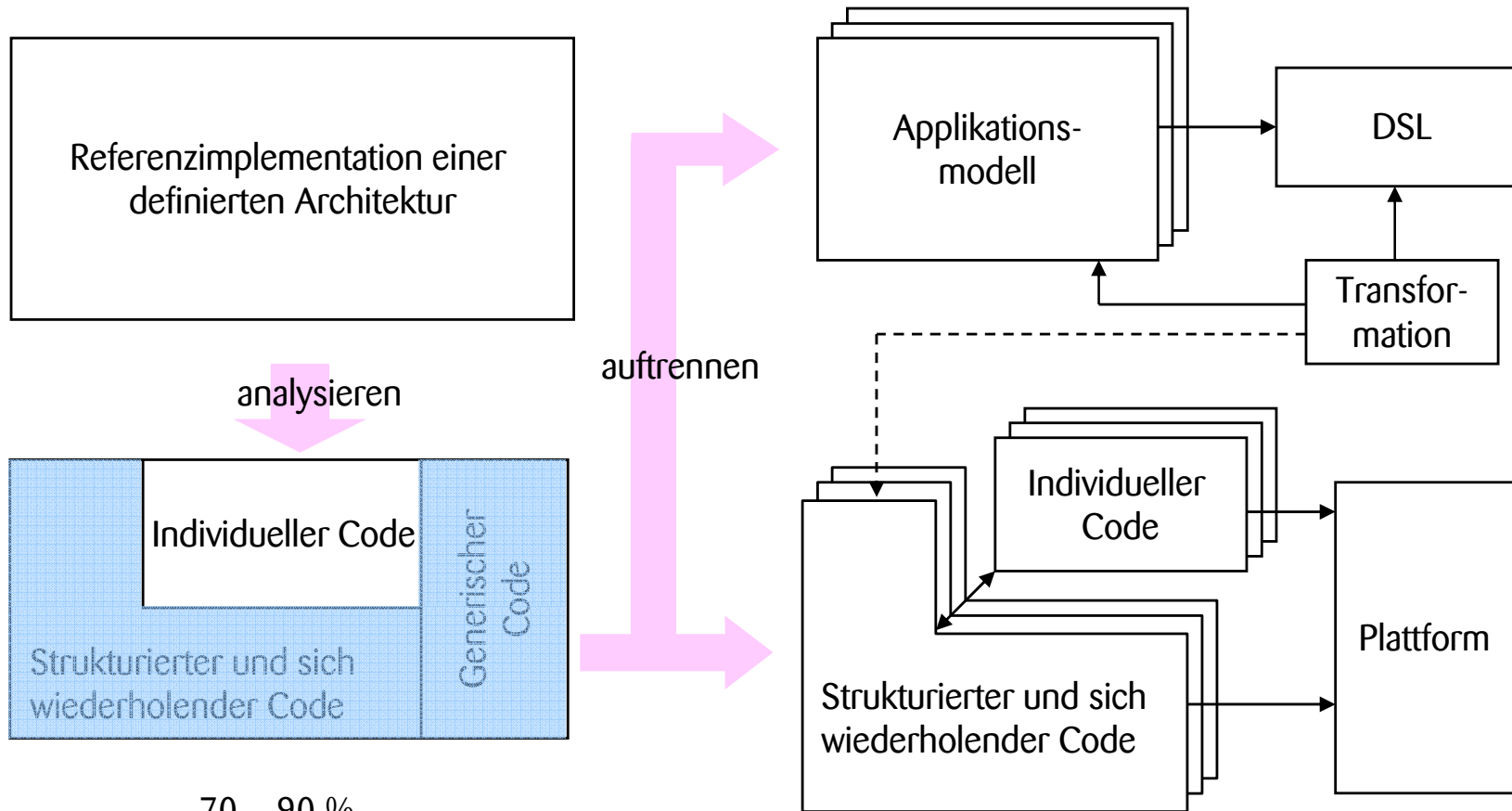
Wie kann ich Arbeit erleichtern?



MDSD mit OpenSource

Slide 21
18. März 2008

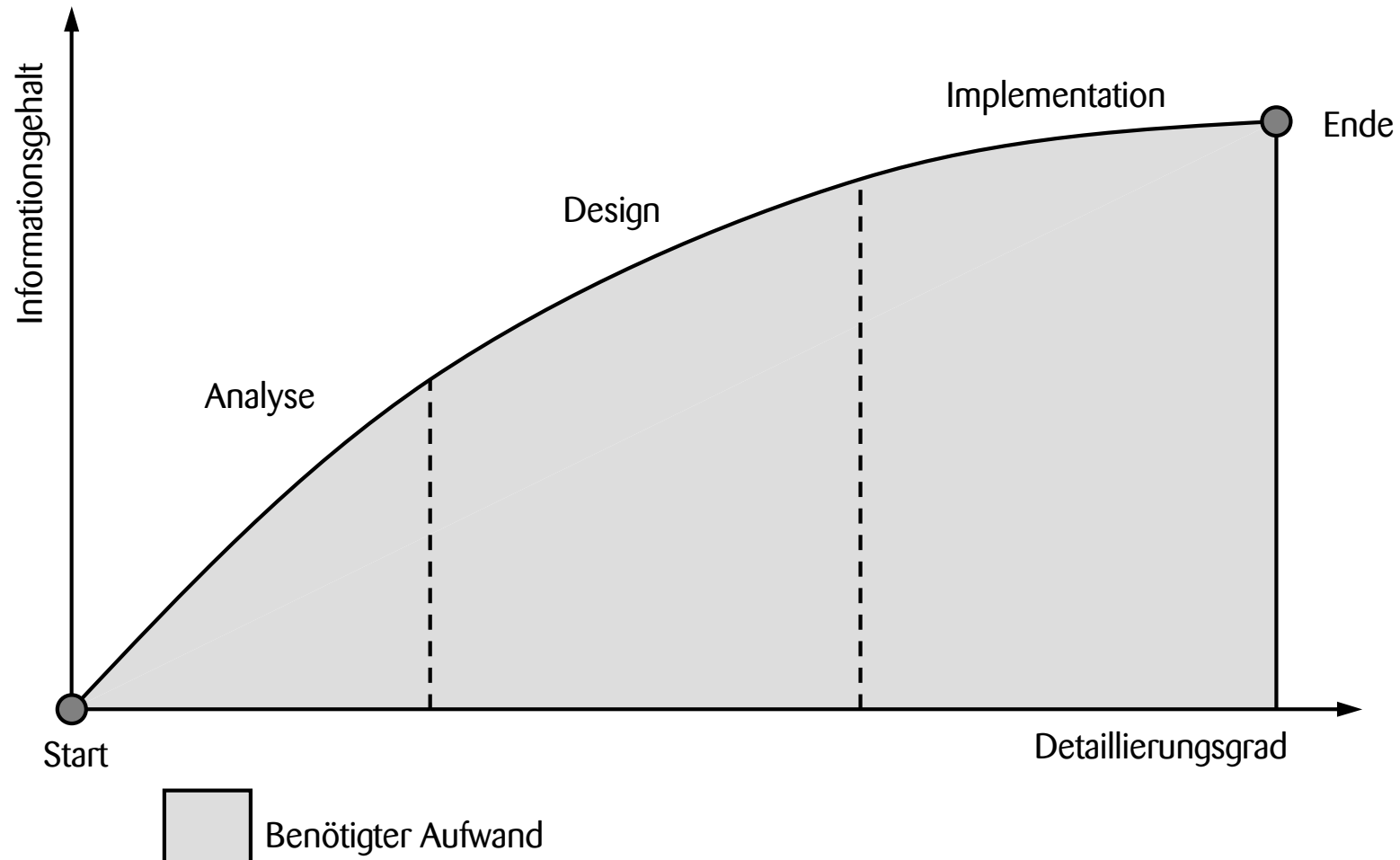
Grundidee von MDSD



MDSD mit OpenSource

Slide 22
18. März 2008 18. März 2008

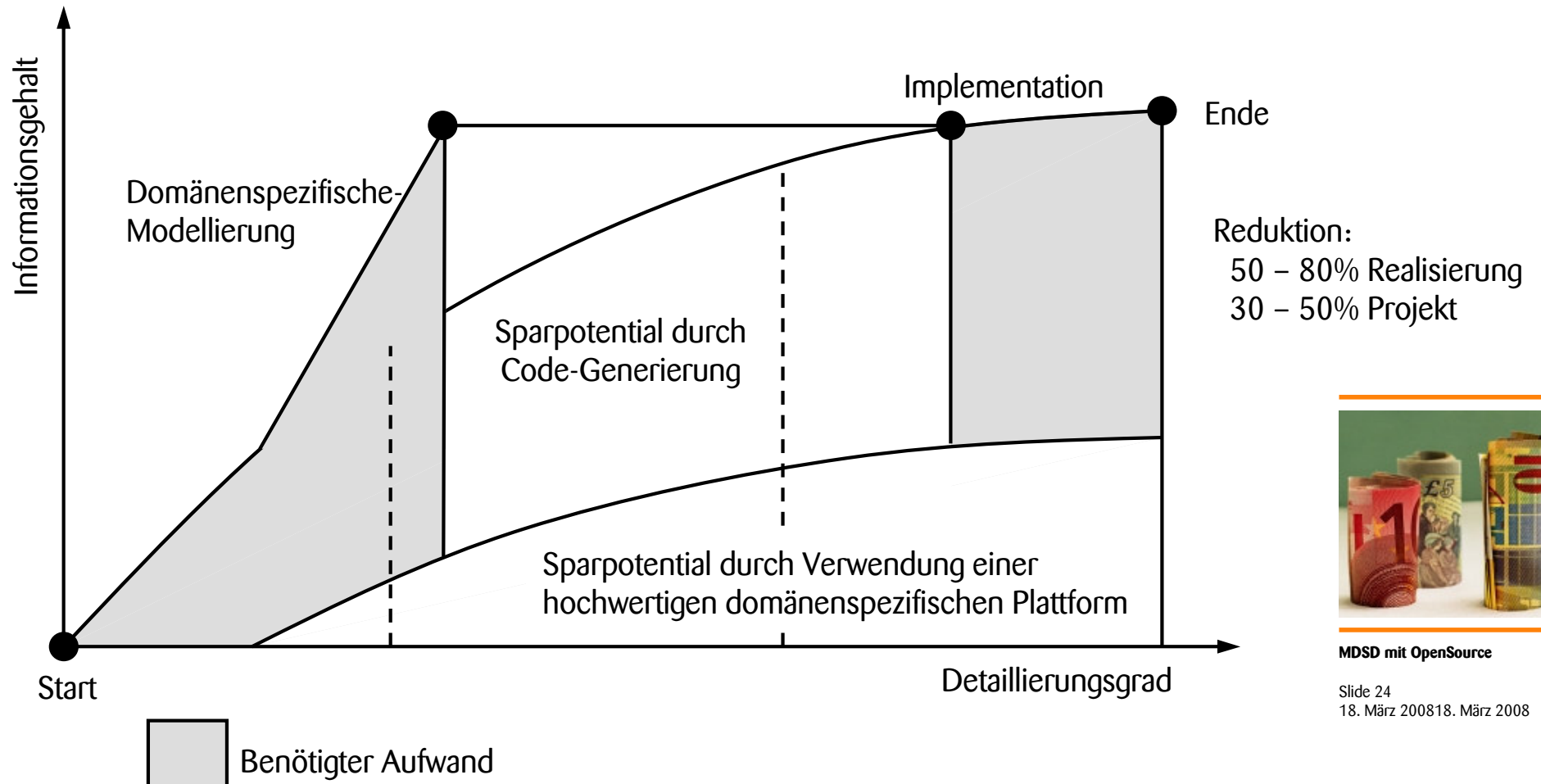
Aufwand bei traditioneller Entwicklung



MDS mit OpenSource

Slide 23
18. März 2008 18. März 2008

Zeitersparnis bei MDSD



MDSD mit OpenSource

Slide 24
18. März 2008 18. März 2008

Hauptziele

- Effizienz der Entwicklung verbessern
- Qualität und Wiederverwendung von SW verbessern
- Erstellung von Infrastruktur-Code vereinfachen
- Redundanz des technischen Codes vermeiden
- Einhaltung der Architektur sicherstellen

Architektur ist der Schlüsselpunkt

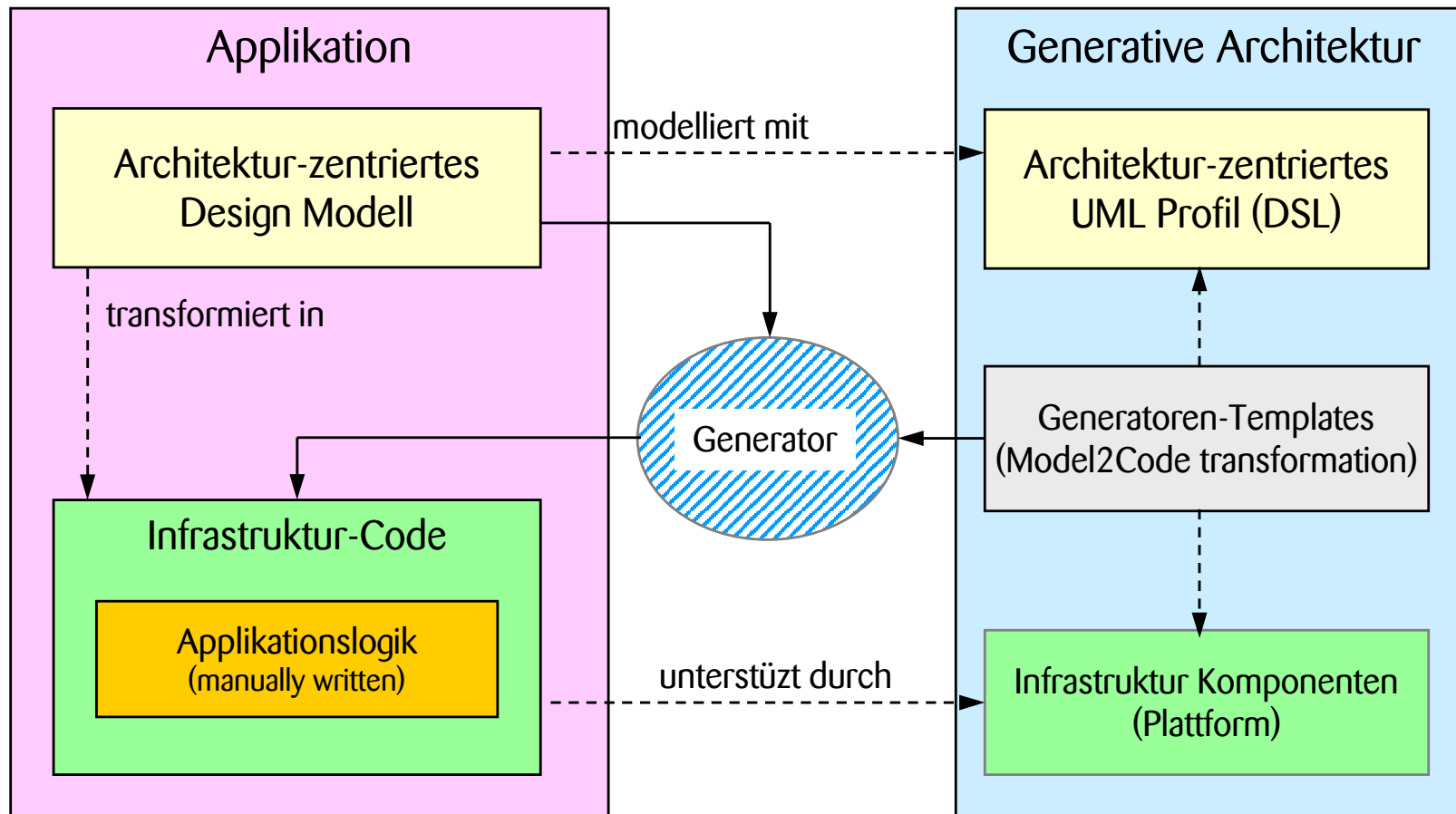
Verwenden von Architektur-zentrierten UML Profilen (DSL)



MDSD mit OpenSource

Slide 25
18. März 2008 18. März 2008

Prinzip des AZ-MDSD



MDSD mit OpenSource

Slide 26
18. März 2008 18. März 2008

Auftrennung der Entwicklung von Architektur und Applikation

Entwicklung der Architektur (1)

- Design und Implementation einer Referenzapplikation
- Definition und Verwaltung der DSL
- Design and Implementation der Transformations-Scripts

Entwicklung der Applikation (N)

- Entwerfen der Applikation mit der DSL (modellieren)
- Infrastruktur-Code generieren
- Applikationsspezifischen Code implementieren
- Rückmeldung an Architektur-Entwicklungs-Team



MDSD mit OpenSource

Slide 27
18. März 2008 18. März 2008

Modelle, Metamodell, DSL und UML-Profile

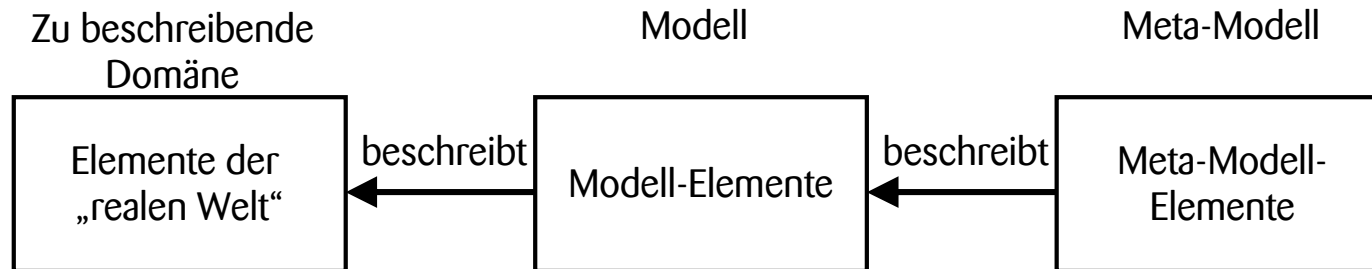
Eine eigene Begriffswelt schaffen.



MDSO mit OpenSource

Slide 28
18. März 2008

Realität, Modell und Metamodell



Beispiele:

Applikation

Tabellen in DB

Applikation

Architektur-Elemente

SourceCode

DDL-Code

UML Modell

Domänen-Modell

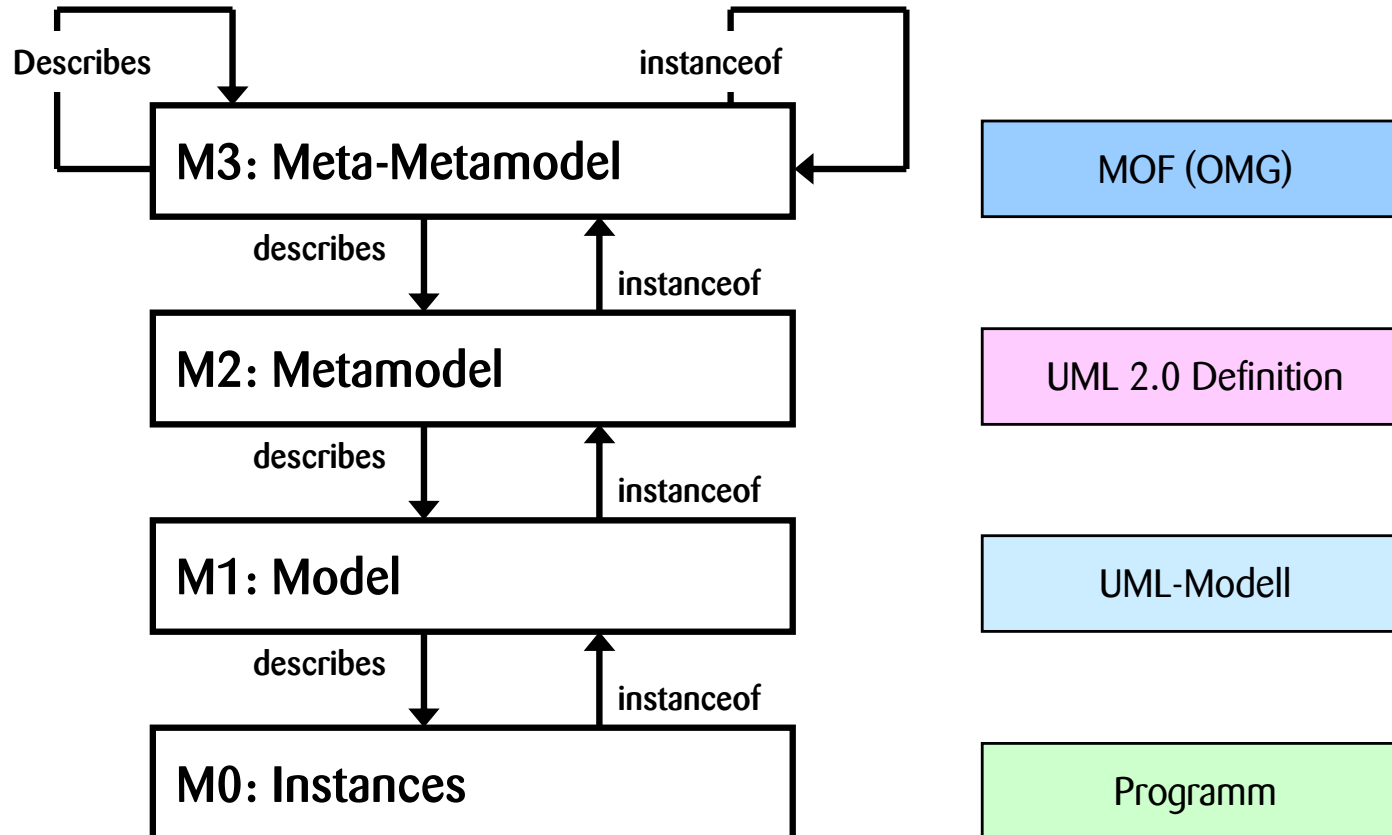
Java Syntax

SQL Syntax

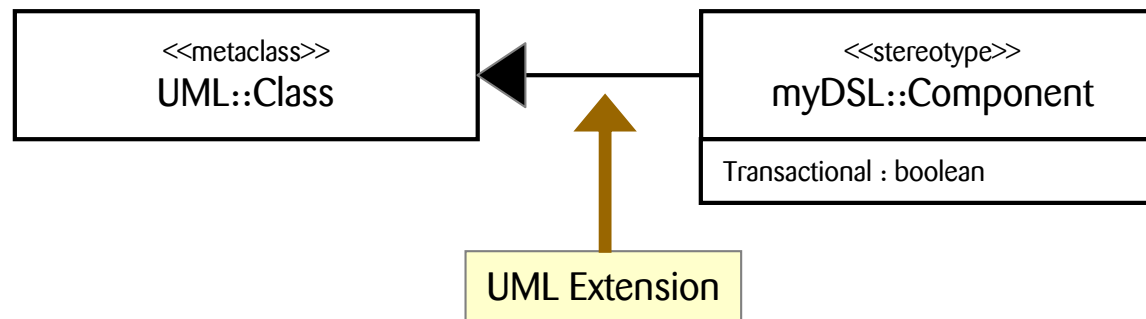
UML-Metamodell

UML-Profil

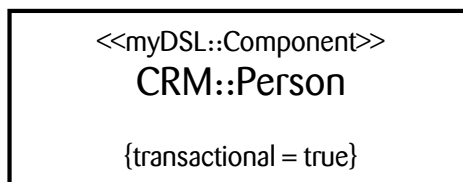
Meta-Layer der OMG



Definition:



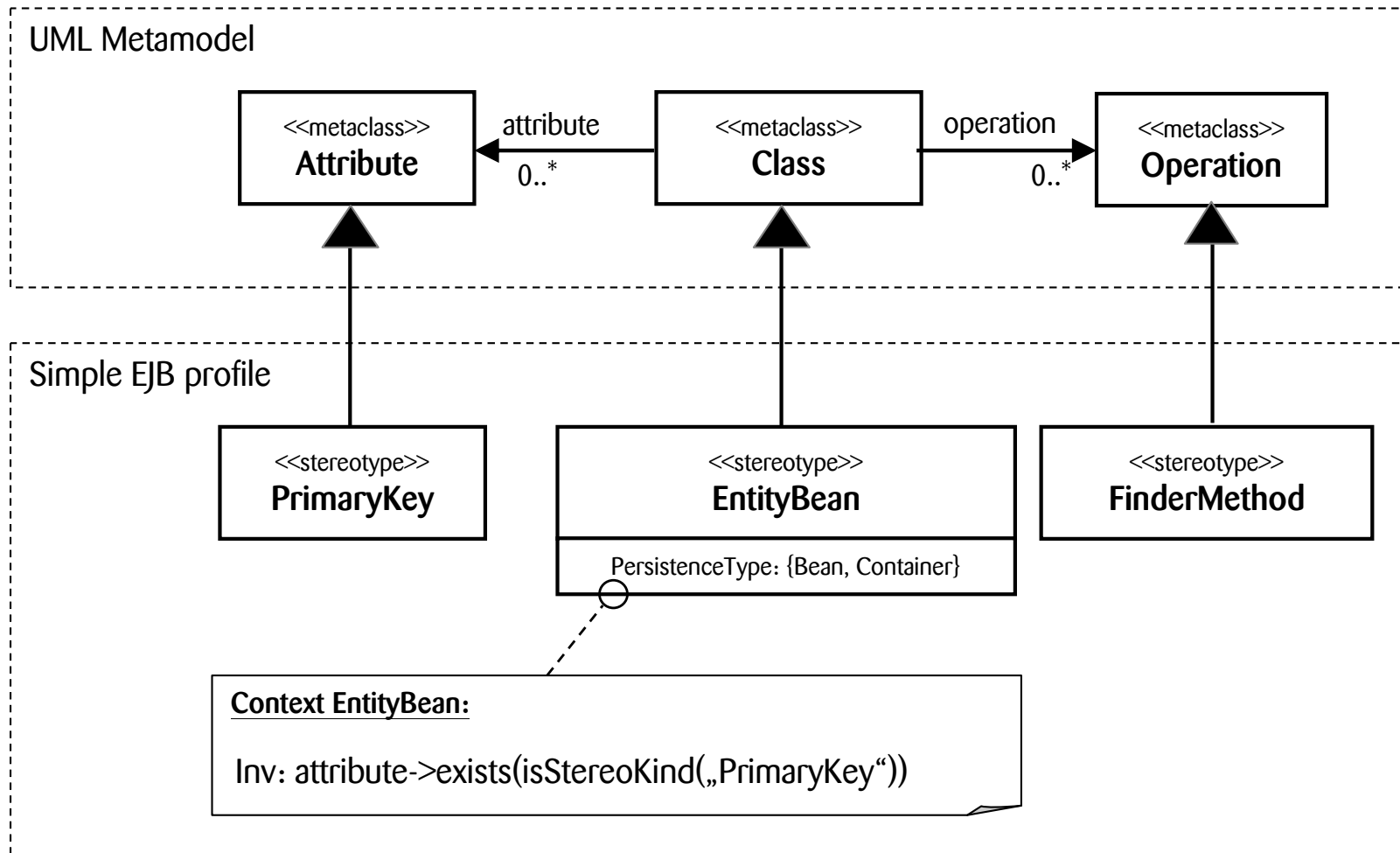
Verwendung:



Vorteil:

- Eigene Modellierungselemente
- Typisierte Tagged Values
- Unterstützt durch EA 😊

UML-Profil Beispiel



Generierbare Architektur

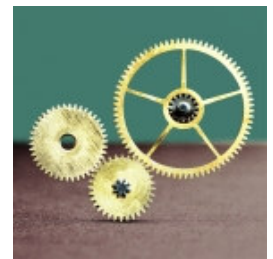
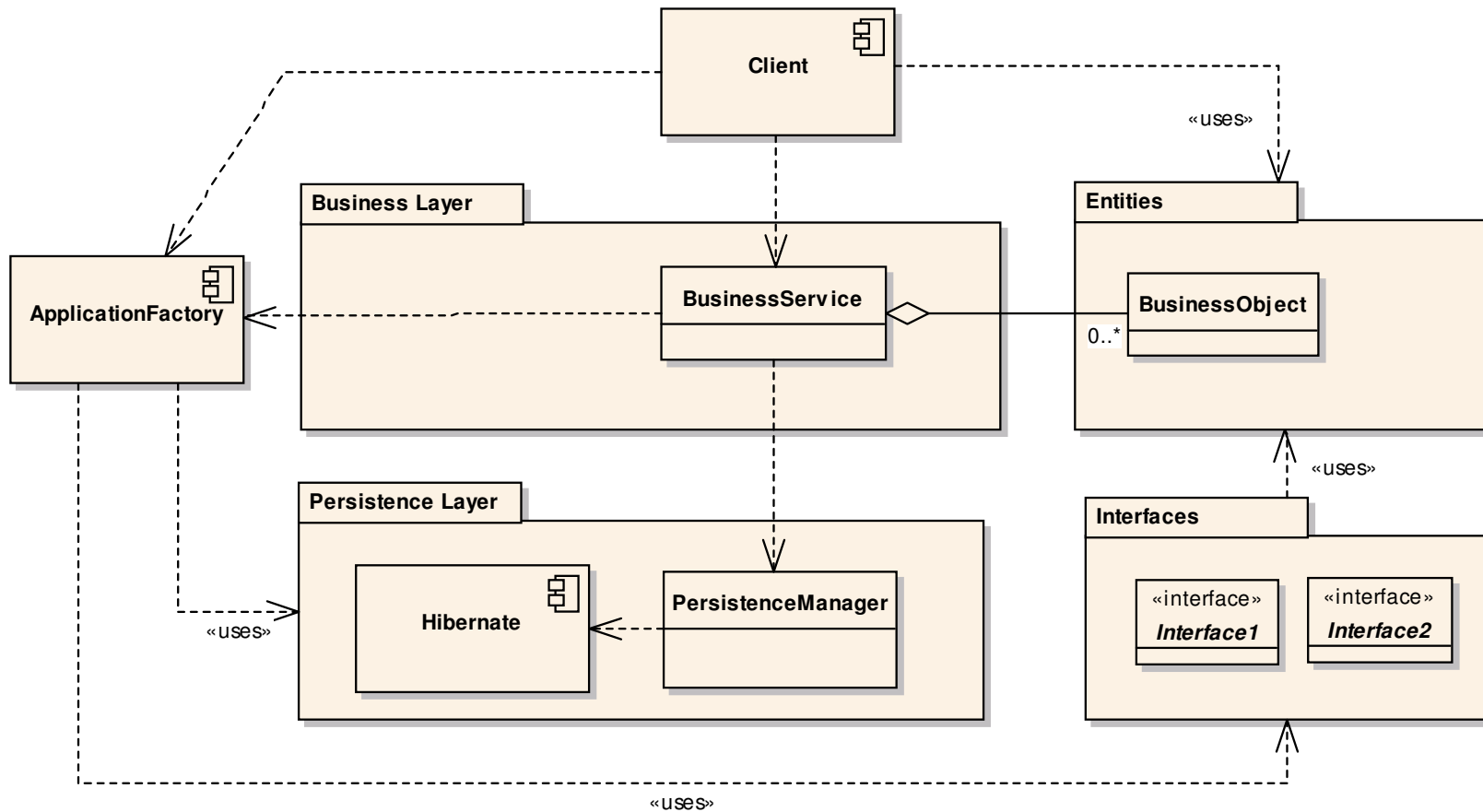
Wie sehen generierte Lösungen aus?



MDSO mit OpenSource

Slide 33
18. März 2008

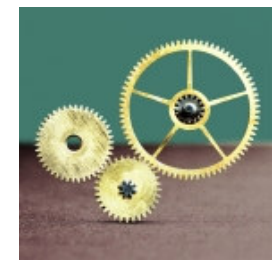
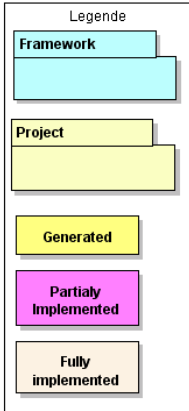
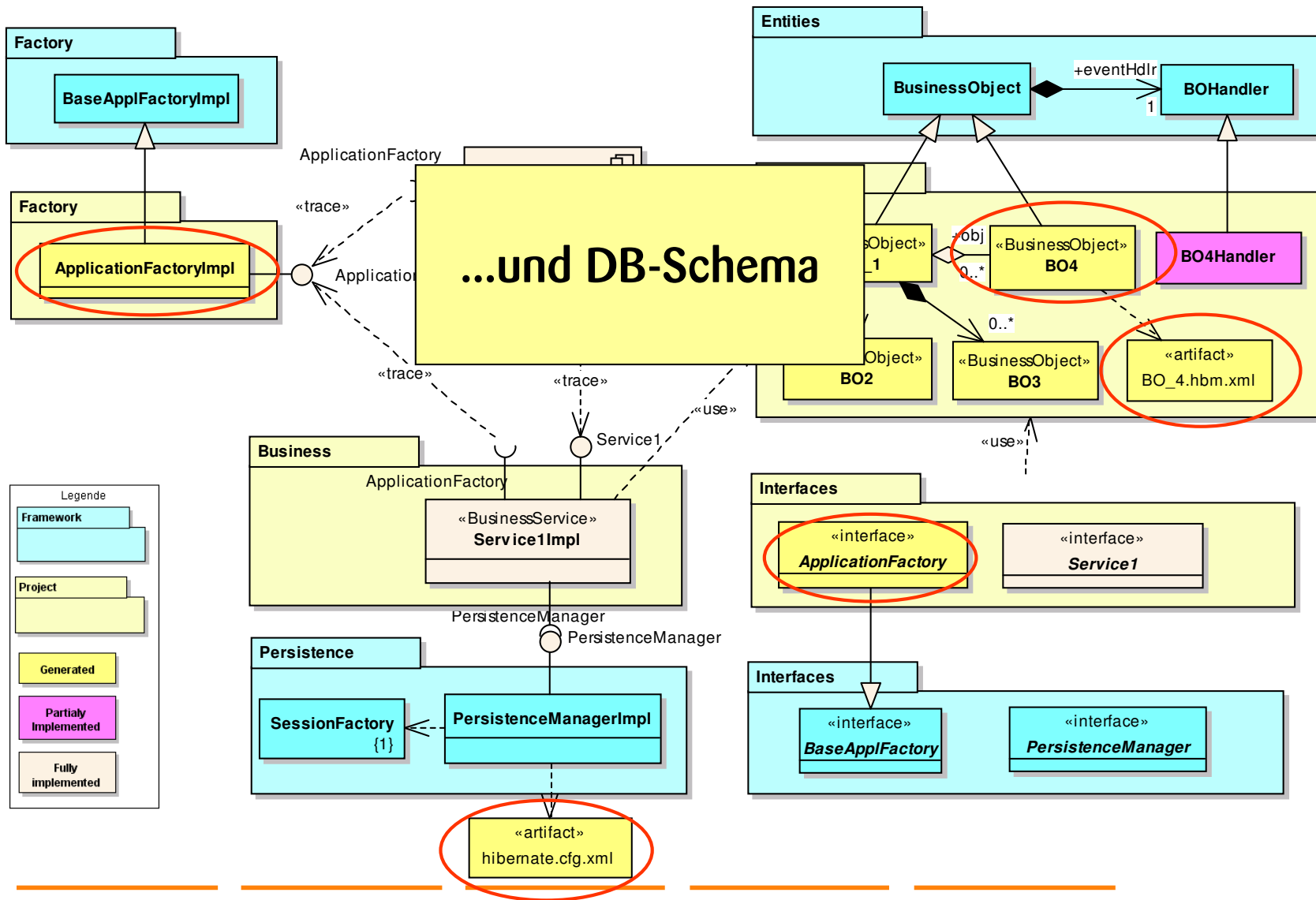
Beispiel-Architektur



MDSO mit OpenSource

Slide 34
18. März 2008 18. März 2008

Generierbare Architektur definieren

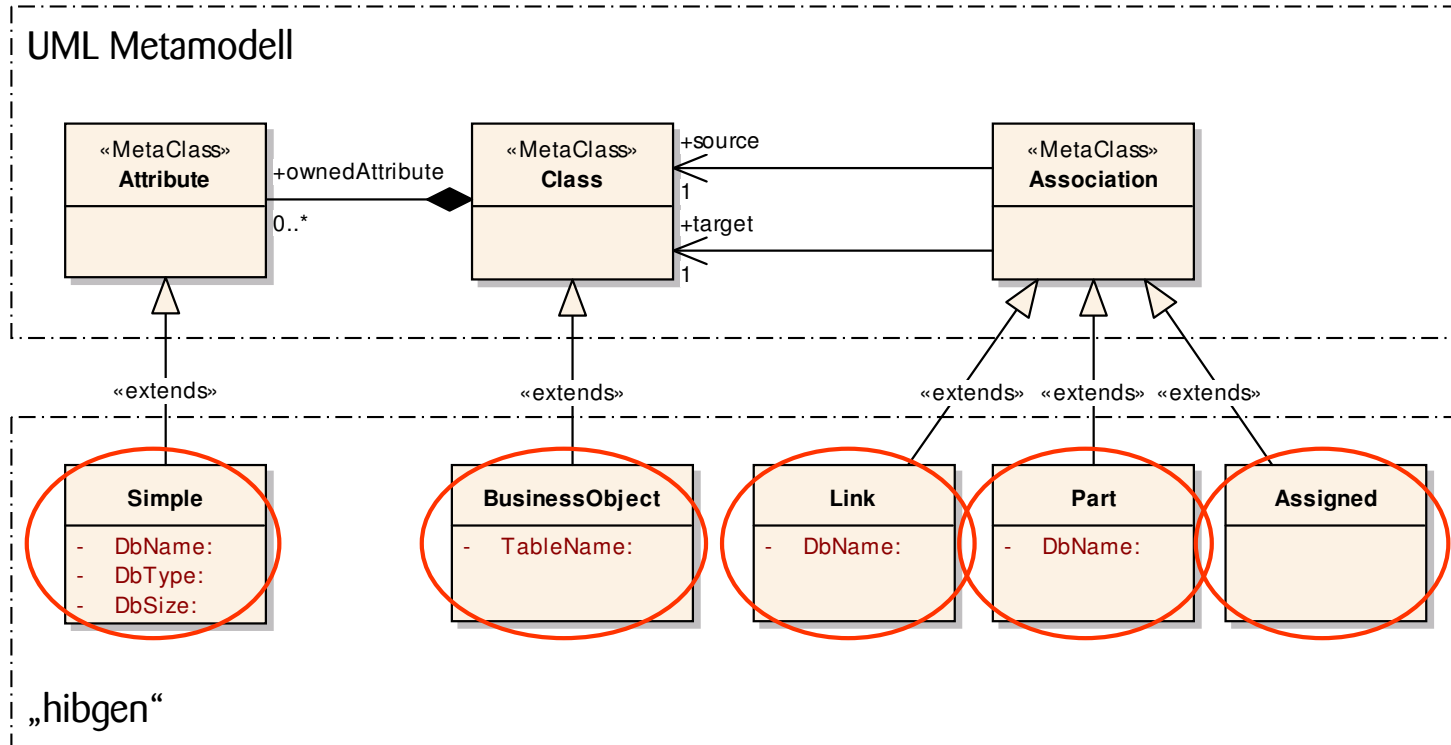


MDSO mit OpenSource

Slide 35
18. März 2008 18. März 2008

DSL definieren

UML Profil „hibgen“ (Hibernate Generator)



oAW im Einsatz

Zeit sparen, Flexibilität gewinnen, Qualität sicherstellen

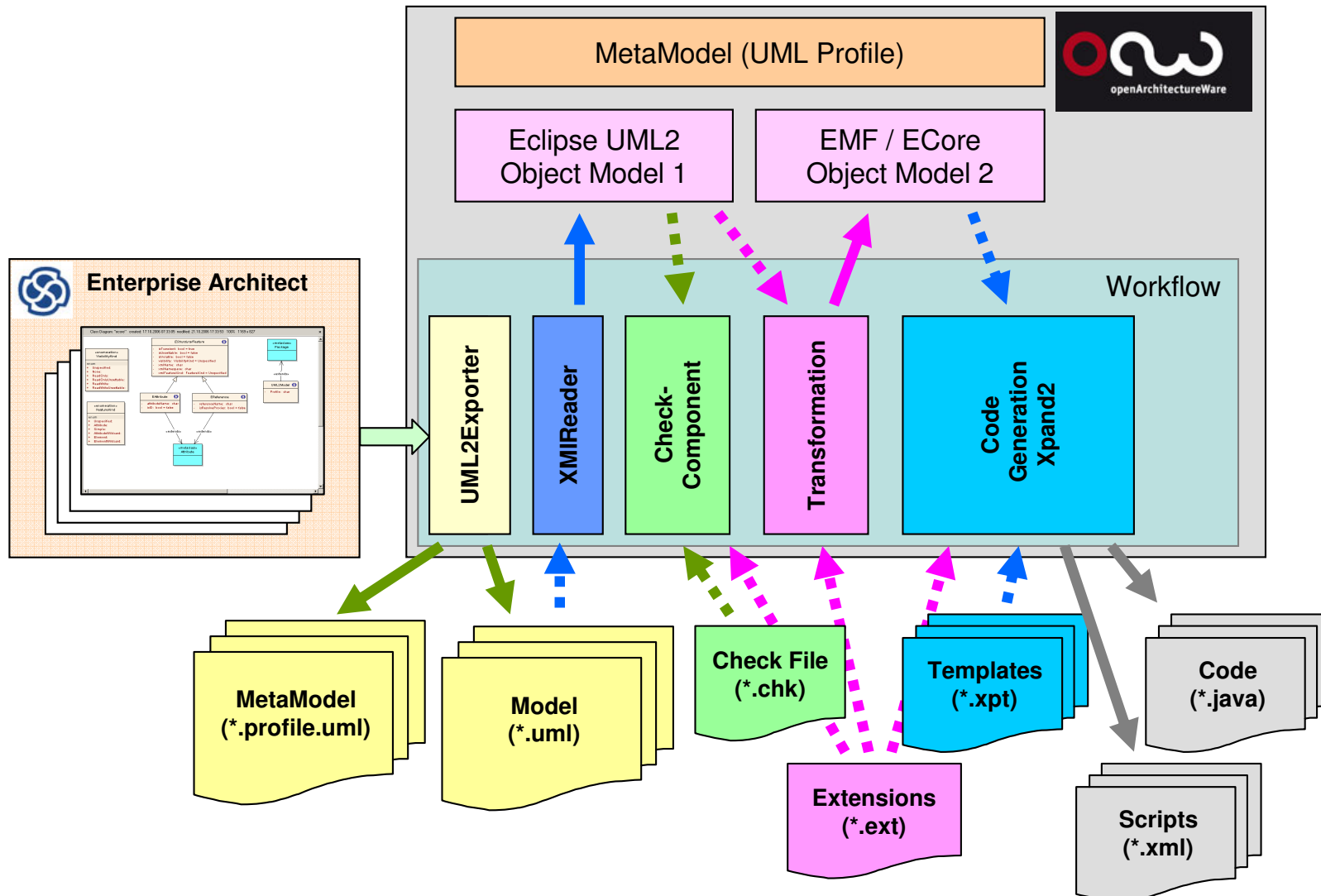


MDSD mit OpenSource

Slide 37
18. März 2008

Überblick oAW

aktuelle Konfiguration



MDSO mit OpenSource

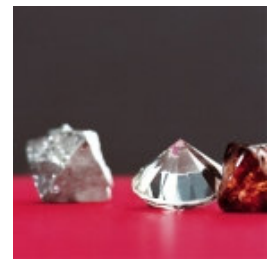
Slide 38
18. März 2008 18. März 2008

Ulrich Brawand
© Zühlke 2007

Testen

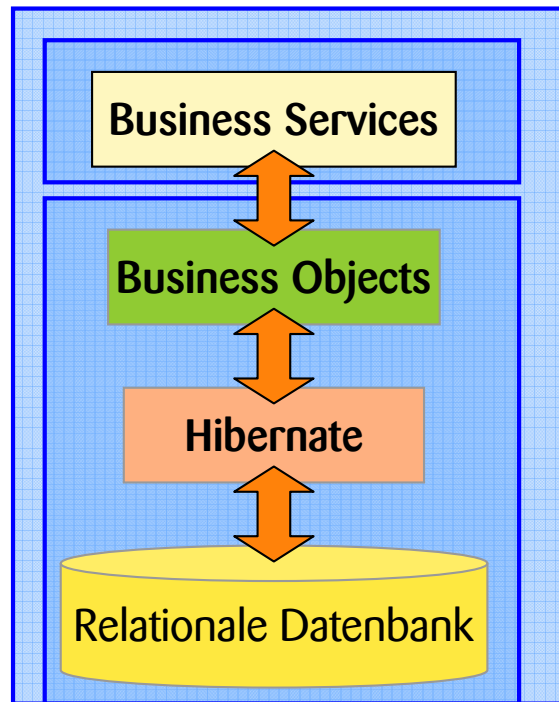
Funktioniert es so wie ich möchte?

- **Generierter Code**
 - Sinnlos, da nur Generator getestet wird.
 - Muss durch Architektur-Team sichergestellt werden.
- **Kein Testcode aus dem gleichen Modell generieren!**
 - Schlägt immer fehl oder ist immer richtig.
- **Was soll ich testen?**
 - Manuell erstellter Code
 - Integration generierter und manuell erstellter Teile
 - Integration von Code aus unterschiedlichen Generatoren
- **Testaufwand ist deutlich geringer!!**



MDSD mit OpenSource

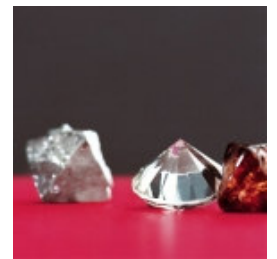
Slide 40
18. März 2008 18. März 2008



3. Integrations-Test

2. Unit-Test

1. Integrations-Test



MDS mit OpenSource

Slide 41
18. März 2008 18. März 2008

Zusammenfassung

... und was haben wir nun davon?

- **Realisierungsaufwand wird massiv reduziert**
 - 50% bis 70% in Realisierung
 - 30% bis 50% in Projekt
- **Qualität wird deutlich verbessert**
 - Technologie-bedingte Redundanzen werden automatisch synchronisiert
 - Architektur wird eingehalten
- **Einsatz von Middleware wird einfach**
 - Konfiguration von Hibernate
 - Konfiguration von EJB2 oder EJB3
 - Einsatz von Web-Services
- **Wartung und Unterhalt wird vereinfacht**
 - Erweiterungen der Struktur
 - Upgrades in Technologien
 - Änderungen in der Architektur

- **Komplexität nimmt zu**
 - Generierbare Architektur
 - Erweiterung Tool-Chain
 - Modelle und Meta-Modelle

- **Applikationsfamilien notwendig**
 - Gleiche oder ähnliche Architekturen

- **Generator erstellen und pflegen**
 - Referenzimplementierung
 - Trennung von generiertem und handgeschriebenem Code
 - UML-Profil erstellen und pflegen
 - Validierungen und Transformationen

- **Investition notwendig**

- **Management-Support notwendig**

Sehr interessanter und flexibler Ansatz, um Projektaufwand zu reduzieren und Qualität zu steigern.

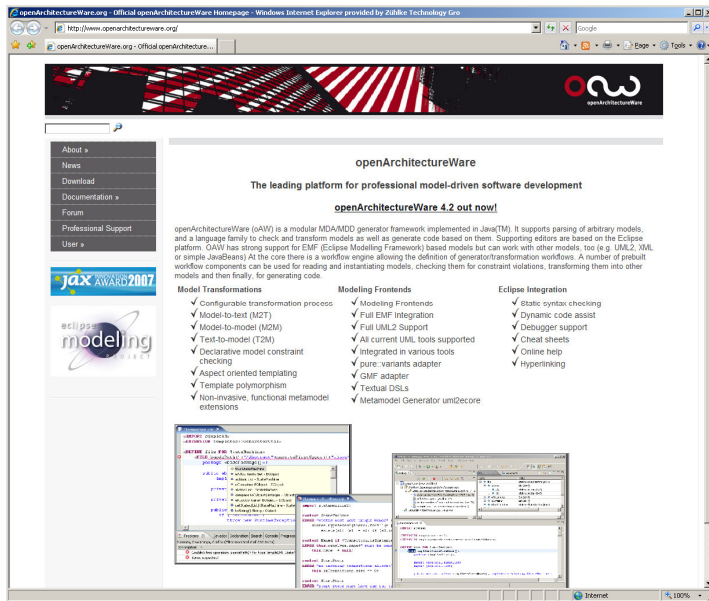
MDA wird brauchbar !!

Referenzarchitektur und –applikation notwendig

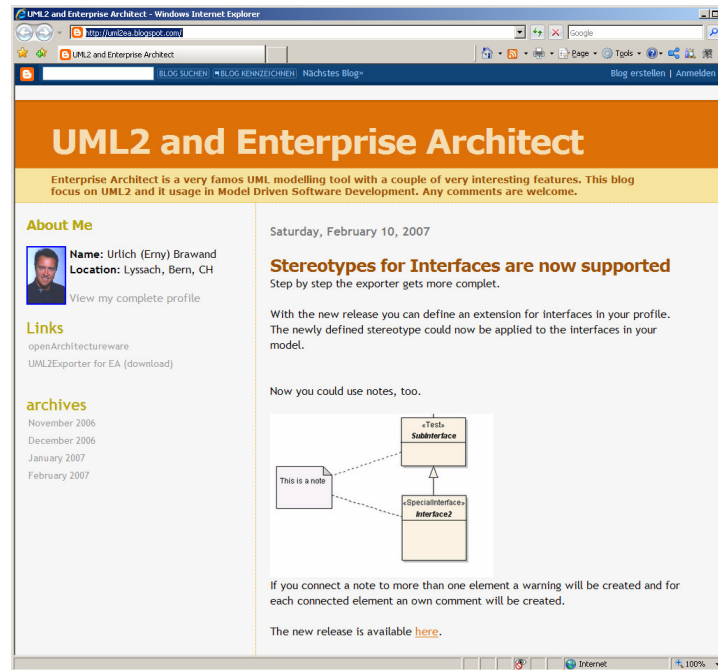
- Problem und Lösung vorher durchdenken
- Verantwortliche Person und Team definieren

Für 60% – 70% der Funktionalität eignet sich generierte Lösung sehr gut . Restliche Funktionalität muss manuell erweitert oder individuell erstellt werden.

Varianten der Generierung möglich (Product Lines)



<http://www.openarchitectureware.org/>



<http://uml2ea.blogspot.com/>