Model Driven Architecture

Wilhelm Stephan

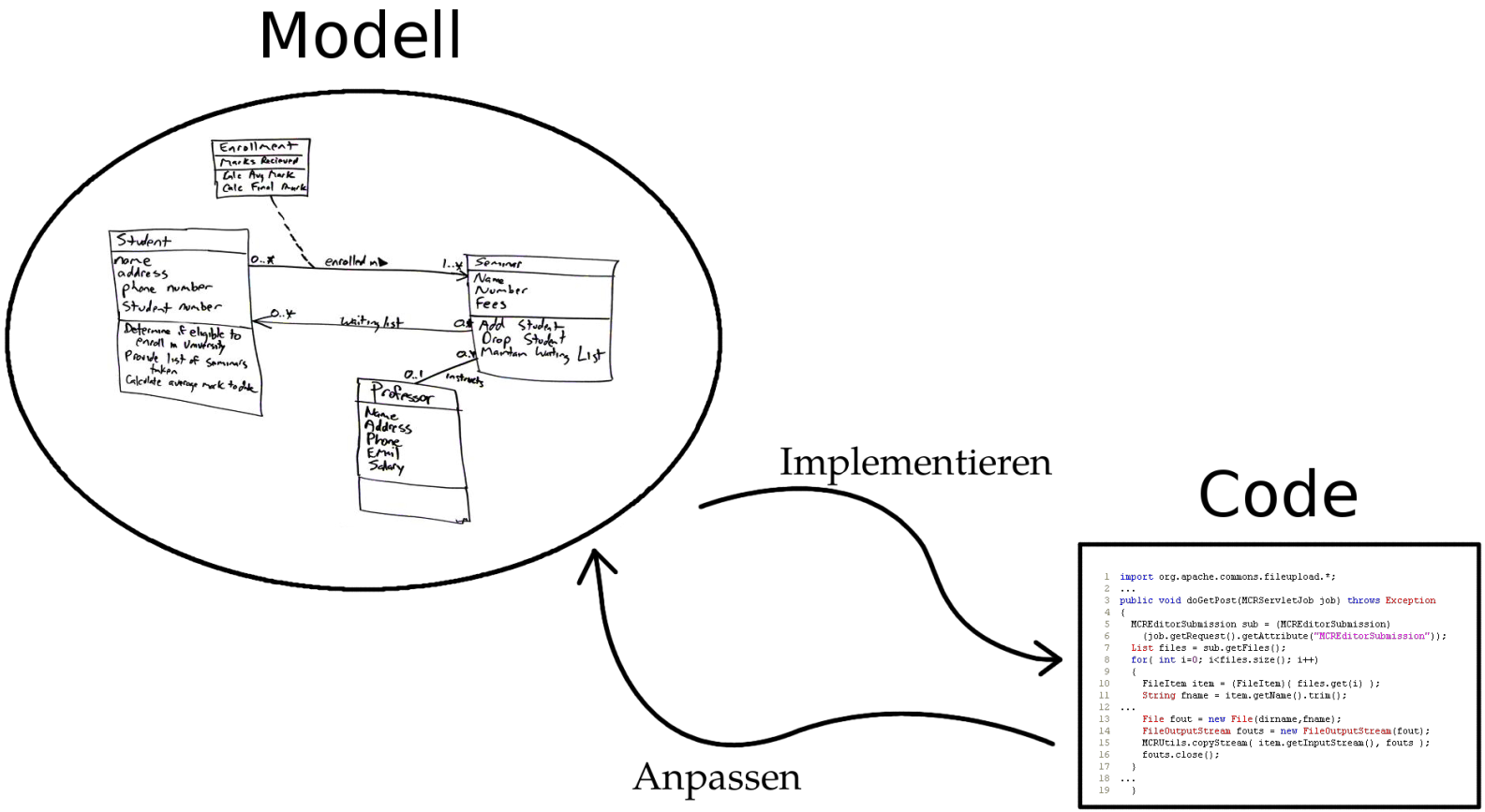
Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Seminar Softwareentwicklung in der Wissenschaft
Betreuer: Julian Kunkel
SommerSemester 14

25. Mai 2014

Agenda

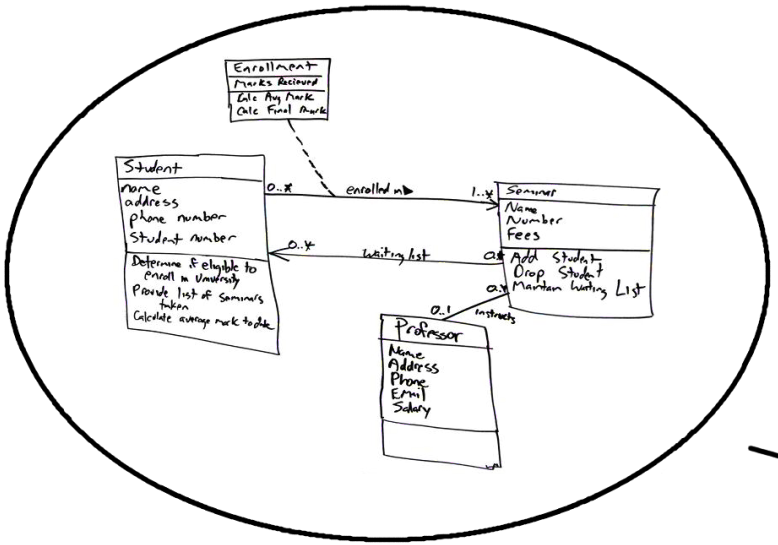
1. Was ist MDA
2. Entwicklungsprozess mit MDA
3. Nutzen von MDA
4. Zusammenfassung

Modellbasierte Softwareentwicklung



Modellgetriebene Softwareentwicklung

Modell



Generiert

Code

```
1  import org.apache.commons.fileupload.*;
2  ...
3  public void doGetPost(MCRServletJob job) throws Exception
4  {
5    MCREditorSubmission sub = (MCREditorSubmission)
6      (job.getRequest().getAttribute("MCREditorSubmission"));
7    List files = sub.getFiles();
8    for( int i=0; i<files.size(); i++)
9    {
10     FileItem item = (FileItem)( files.get(i) );
11     String fname = item.getName().trim();
12 ...
13     File fout = new File(dirname,fname);
14     FileOutputStream fouts = new FileOutputStream(fout);
15     MCRUtils.copyStream( item.getInputStream(), fouts );
16     fouts.close();
17   }
18 ...
19   }
```

Was ist besser?

Vorteile Modellgetrieben

- Modell verständlicher als Code
- Modell dokumentiert Verhalten und Struktur
- Codegenerierung
- Plattformunabhängigkeit

Vorteile Modellbasiert

- Flexibilität

CHAOS Report - Softwareprojekte scheitern

Source: CHAOS Report 2009, Standish Group

Gründe für das Scheitern

- Fehlende Dokumentation
- Fehlendes fachliches Verständnis
- Kosten
- Zeitdruck

Die OMG und ihre Lösung

OMG

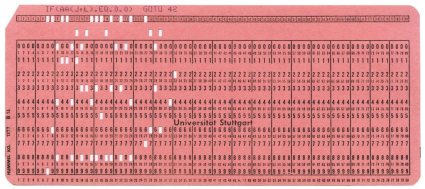
- Object Management Group
- Konsortium von 11 Firmen
- 1989 gegründet
- Entwicklung von Standards
  - UML
  - CORBA
  - MDA

Der Lösungvorschlag MDA

- Abstrakter Lösungsvorschlag
- Keine konkrete Implementation
- Implementation ist den Toolherstellern überlassen

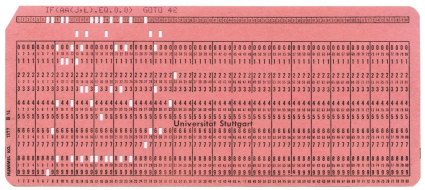
MDA - bevorstehender Schritt in der Evolution

Lochkarte

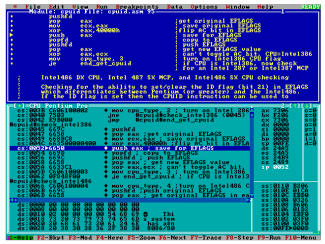


MDA - bevorstehender Schritt in der Evolution

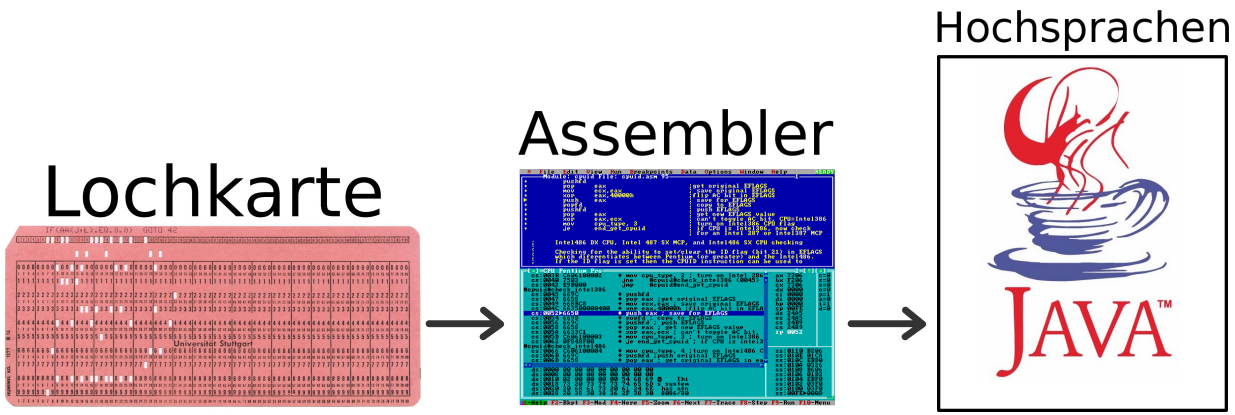
Lochkarte



Assembler

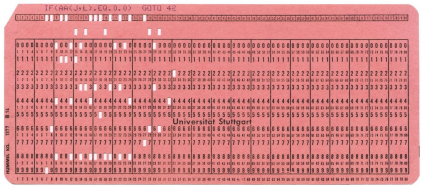


MDA - bevorstehender Schritt in der Evolution

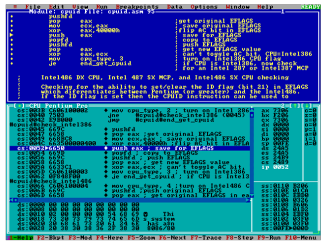


MDA - bevorstehender Schritt in der Evolution

Lochkarte

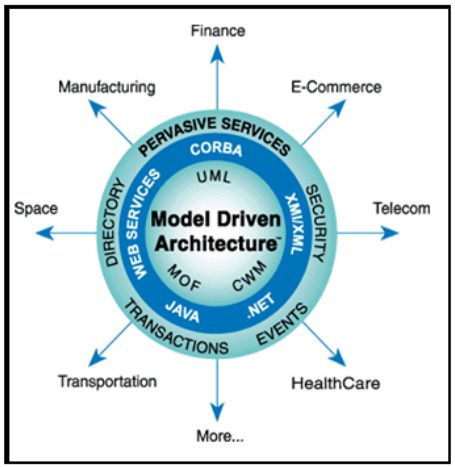


Assembler



Hochsprachen

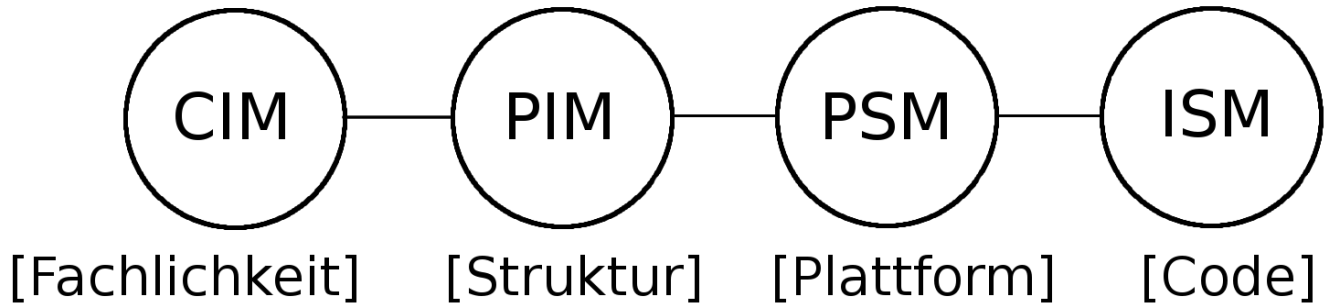
MDA



Grundkonzept von MDA

Die Idee

- Anwendung modellieren
- Modelle transformieren
- Code generieren
  - Wichtig: MDA ist mehr als Codegenerieren



Einführung des begleitenden Beispiels

Der Weblandwirt

- Verwaltung eines Landwirtschaftlichen Betriebes
- Der Landwirt hat
  - Ländereien
  - Tiere
  - Kunden und Lieferanten
  - Produkte
  - Gebäude

Vorgehen

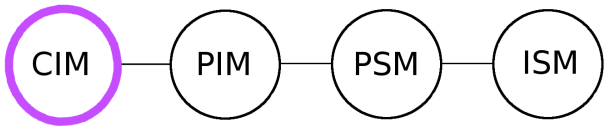
- Vom abstrakten Modell zum Code

Quelle [ZW05]

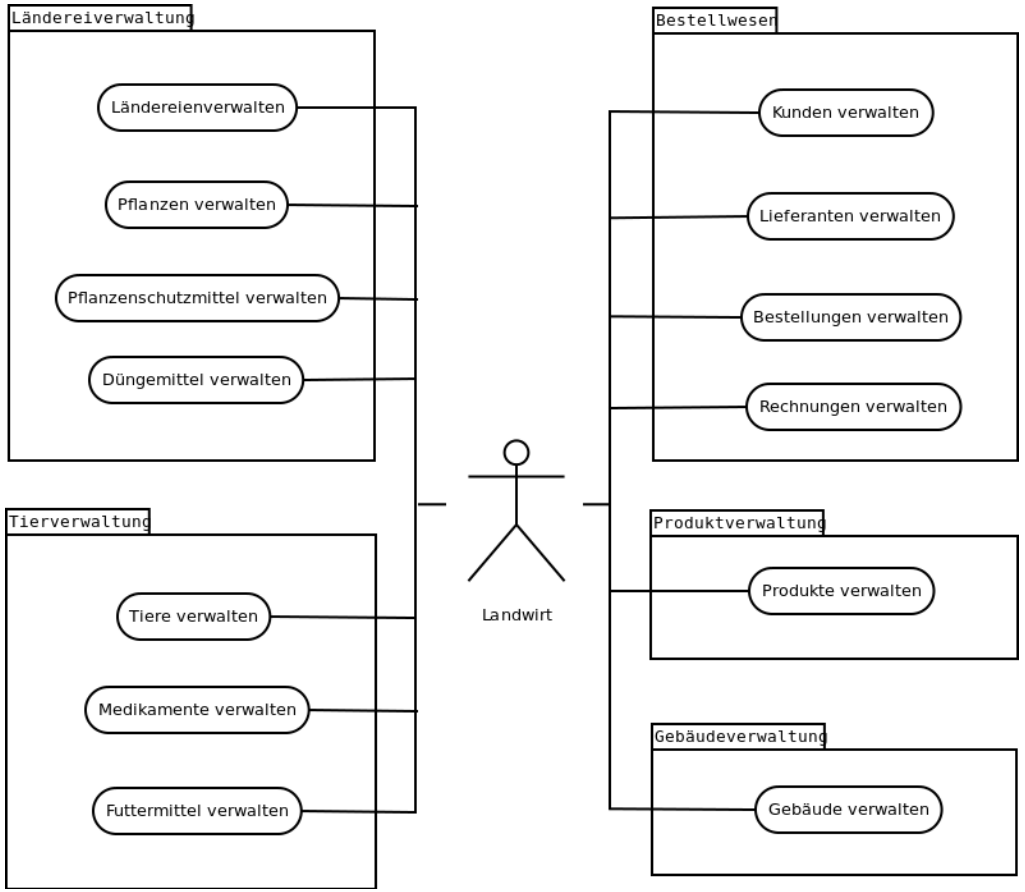
Modelltyp CIM - Computation Independant Model

Abstraktestes Modell

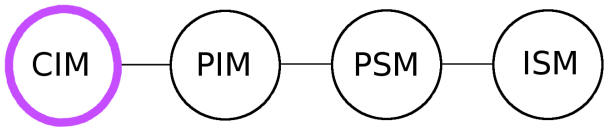
- Unabhängig von der Realisierung
- Spezifiziert Anforderungen und Verhalten
- Verständnis der Aufgabe des Systems
- Use-Case-Diagramme, Aktivitätsdiagramme
- Entspricht noch modellbasierter Softwareentwicklung



Weblandwirt - Beispielhaftes Use-Case-Diagramm



Quelle [ZW05]



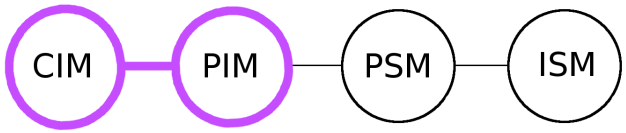
Modelltyp PIM - Platform Independant Model

Strukturmodell/ Berechnungsmodell

- Unabhängig der Implementationstechnologie
- Fachliche Spezifikation
- Modelliert Struktur
- Resistent gegen Technologieveränderung
- Klassendiagramm, Aktivitätsdiagramm, Use-Case-Diagramm

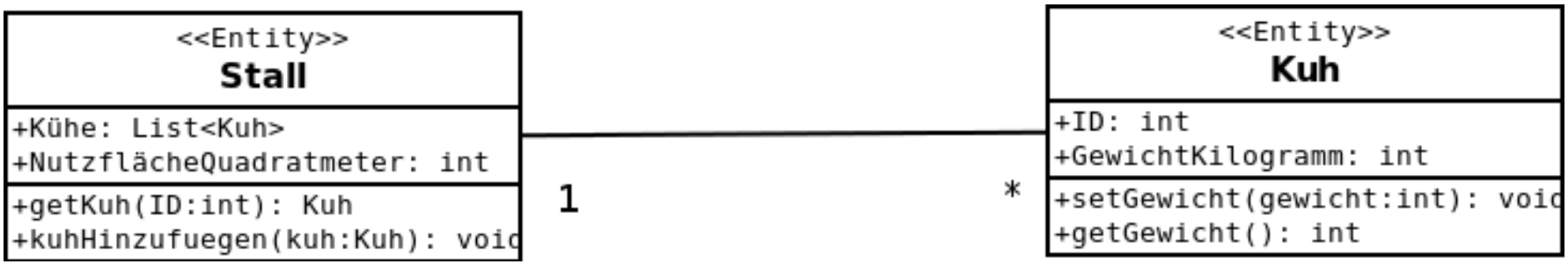
Transformation von CIM in PIM

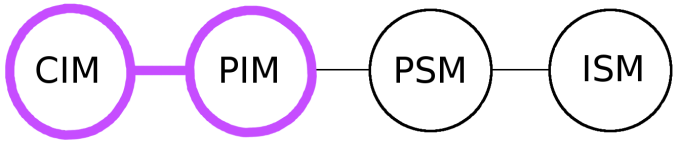
- Modell zu Modell Transformation
- Keine Tools bekannt - Manuelle Transformation



Weblandwirt - Transformation CIM zu PIM

Beispielhaftes Klassendiagramm

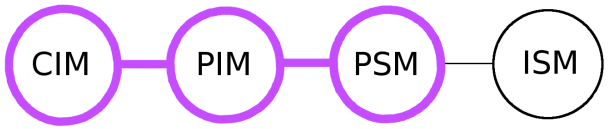




Modelltyp PSM - Platform Specific Model

Plattformmodell

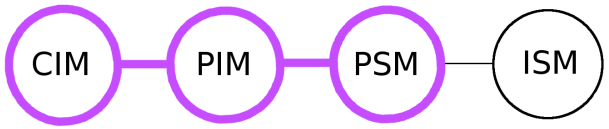
- Durch CIM und PIM modelliert
- Modelliert Struktur
- Plattformspezifische details
  - Programmiersprache
  - Datenbank
  - Kommunikation zwischen Komponenten
- Plattformen z.B.: CORBA, J2EE



Modelltyp PSM - Platform Specific Model

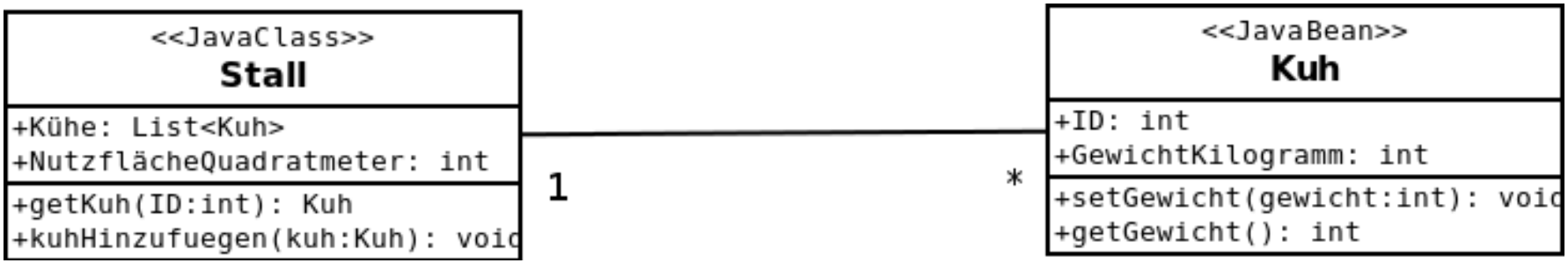
Transformation von PIM in PSM

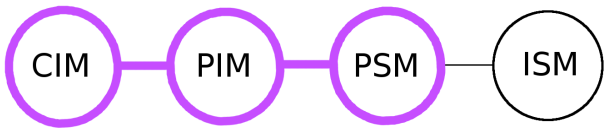
- Modell zu Modell Transformation
- Schlüsselkonzept der MDA im Sinne der OMG
  - Konzepte sind stabiler als Technologien
  - Formale Modelle - automatisierte Tranformationen möglich



Weblandwirt - Transformation PIM in PSM

Beispielhaftes Klassendiegramm





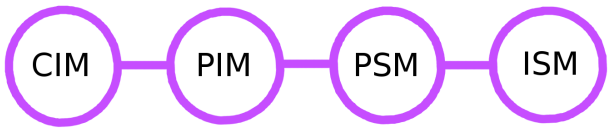
Modelltyp ISM - Implementation Specific Model

Endresultat Quellcode

- Keine funktionale Implementierung
- Nur Quellcode-Skelette

Transformation von PSM in ISM

- Modell zu Code Transformation
- Automatisierte Transformation möglich
- Viele Tools (Codegeneratoren)

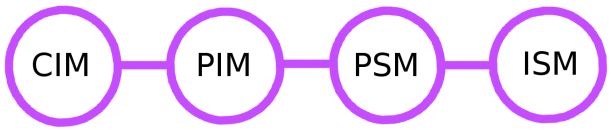


Weblandwirt - Transformation PSM in ISM

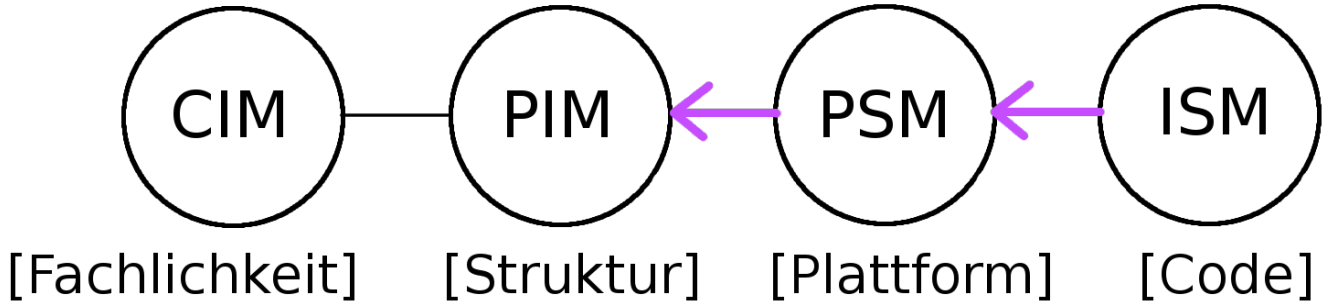
Automatisch generierter Code

```java
import java.util.List;

public class Stall
{

    private List<Kuh> Kuehe;
    private int NutzflaecheQuadratmeter;

    public Kuh getKuh(int id)
    {
        return null;
    }

    public void kuhHinufuegen(Kuh kuh)
    {

    }
}
```

```java
public class Kuh
{
    private int _ID;
    private int _gewichtKilogram;

    public int getGewichtKilogramm()
    {
        return 0;
    }

    public void setGewichtKilogramm(int gewicht)
    {

    }

}
```



Rücktransformation/ Synchronisation



ISM zu PSM

- Änderungen im Code - ändern das Modell
- Modell bleibt korrekt
- Können viele Tools (Codegeneratoren)

PSM zu PIM

- Kaum automatisierbar
- Erfordert manuelle, intelektuelle Arbeit

Lohnt es sich MDA zu benutzen?

Ziele und Vorteile

- Plattform Unabhängigkeit
- Konsistenz zwischen Code und Modellen
- Steigerung der Entwicklungsgeschwindigkeit
- Verbesserung der Softwarequalität
- Bessere Dokumentation

Idealismus vs Realität

- Nur teilweise automatisiert
- Toolhersteller beschränken sich auf Codegeneratoren
- Nur Code Skelette - Manuelle Vervollständigung nötig

Evaluation des Beispiels

Einheit	Gesamtzeilenzahl	generierte Zeilen	manuell erstellte Zeilen
generierte ejb-Dateien	1087	1087	0
manuell erstellte ejb-Dateien	411	218	193
Exception	15	0	15
Sprachdatei	95	0	95
WebPages	384	315	69
WebForms	462	462	0
WebActions	498	427	71
Gesamt	2952 (100%)	2509 (85%)	443 (15%)

Quelle [ZW05]

Stand der Technik

Allgemein

- Interoperabilität
- Cartridges
- Modellierung
- Transformationen

Tools

- dia und dia2code
- AndroMDA
- ExecutableUML

Zusammenfassung

MDA hat Potential

- Mächtige Abstraktion von Hochsprachen
- Codegeneratoren nützlich
- Einarbeitung lohnt sich
  - Zeitgewinn im Laufe eines Projektes
  - Mehrgewinn durch bessere Dokumentation

Quellen


Keefer, Andreas: *MDA auf Basis Open Source*.
Diplomarbeit, Hochschule der Medien Stuttgart, 2007.


Stahl, Thomas und Markus Völter: *Modellgetriebene Softwareentwicklung - Techniken, Engineering, Management*.
Dpunkt-Verlag, Köln, 1. Aufl. Auflage, 2005.


Wimmer, Manuel: *Model Driven Architecture in der Praxis - Evaluierung aktueller Entwicklungwerkzeuge und Fallstudie*.
Diplomarbeit, TU Wien, 2005.


Zeppenfeld, Klaus und Regine Wolters: *Generative Software-Entwicklung mit der MDA -*.
Spektrum Akademischer Verlag, Heidelberg, 2005. Aufl. Auflage, 2005.