

47. Werkzeuge für die modellgetriebene Architektur (Model- Driven Architecture, MDA)

Prof. Dr. rer. nat. Uwe
Aßmann

Institut für Software- und
Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

<http://st.inf.tu-dresden.de>

Version 11-0.1, 29.12.11

1) MDA


2) QVT

3) ATL

- ▶ Alan Brown. An introduction to Model Driven Architecture. Part I: MDA and today's systems
 - <http://www.ibm.com/developerworks/rational/library/3100.html>
- ▶ Frédéric Jouault and Ivan Kurtev. On the Architectural Alignment of ATL and QVT. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06). ACM Press, Dijon, France, chapter Model transformation (MT 2006), pages 1188—1195.
 - <http://atlanmod.emn.fr/bibliography/SAC06a>
- ▶ Tutorial über ATL “Families2Persones”
- ▶ http://www.eclipse.org/m2m/atl/doc/ATLUseCase_Families2Persons.ppt
- ▶ ATL Zoo von Beispielen
 - <http://www.eclipse.org/m2m/atl/atlTransformations>
- ▶ K. Lano. Catalogue of Model Transformations
 - <http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf> /
- ▶ Implementation in ATL
 - <http://www.eclipse.org/m2m/atl/atlTransformations/EquivalenceAttributesAssociations/EquivalenceAttributesAssociations.pdf>

47.1 Modellgetriebene Architektur (MDA)



- 
-
-
- ▶ Modelle und Spezifikationen der MDA sind Graphen
 - ▶ MDA-Werkzeuge benötigen Graph-Mapping, Graph-Querying und Graphtransformation
 - Gewöhnlich bieten sie Unterstützung für eine oder mehrere der besprochenen Sprachen
 - Typisierung durch Metamodelle
 - ▶ Oft werden die Sprachen QVT oder ATL unterstützt

Modell-Verknüpfung am Beispiel INNOVATOR

- ▶ Innovator kann gleichzeitig für Analyse-, Entwurfs- und Implementierungsmodelle eingesetzt werden, sowie für Transformationen dazwischen
- ▶ Wie kann man diese Modelle systemisch mit einander verknüpfen?

	Status		Name	Typ	Änderungsdatum
1	0	A	Ausleihe	Sec...	22.11.2003 00:48:02
2	0	A	Kunde_anmelden	Koll...	10.11.2003 01:21:54
3	0	A	Rückgabe	Sec...	22.11.2003 00:21:47
4	0	A	Tonträger_Einkauf	Sec...	10.11.2003 01:23:59
5	0	A	Kunden_neu_anlegen	Sec...	10.11.2003 01:26:19
6	0	A	AnalysisClassDiagram	Klas...	09.11.2003 15:29:14
7	0	A	Verwaltung_AS	Klas...	09.11.2003 15:25:56
8	0	A	Tonträger_AS	Klas...	09.11.2003 15:20:08
9	0	A	Kunde_AS	Klas...	09.11.2003 15:27:32
...	0	A	: Kunde_AS	Obji...	09.11.2003 13:20:05
...	0	A	: Tonträger_AS	Obji...	09.11.2003 13:20:16
...	0	A	VerwaltungUI_AS	Klas...	09.11.2003 15:16:32
...	0	A	: VerwaltungUI_AS	Obji...	09.11.2003 13:23:08
...	0	A	: Kunde_UC	Obji...	09.11.2003 14:05:54
...	0	A	: Bibliothek_UC	Obji...	09.11.2003 15:44:35
...	0	A	: Verwaltung_AS	Obji...	09.11.2003 16:14:14

PIM und PSM gemäß der MDA

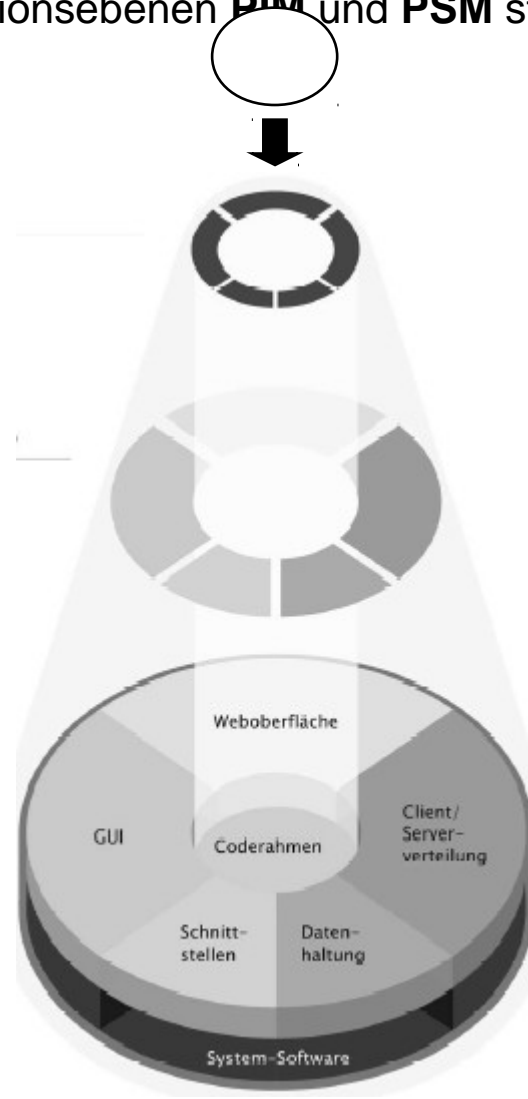
Für die unterschiedlichen Abstraktionsebenen **PIM** und **PSM** stehen verschiedene Beschreibungsmittel zur Verfügung:

Fachkonzept auch CIM
(Computation independent model)

Plattformunabhängiges Modell
(UML, OCL, XMI)

Plattformspezifisches Modell
Basiskomponenten (JB)
Steuerungskomponenten
Infrastrukturkomponenten (EJB,
CCM, COM+, .NET)
Anwendungskomponenten

Programmierung (ooP)



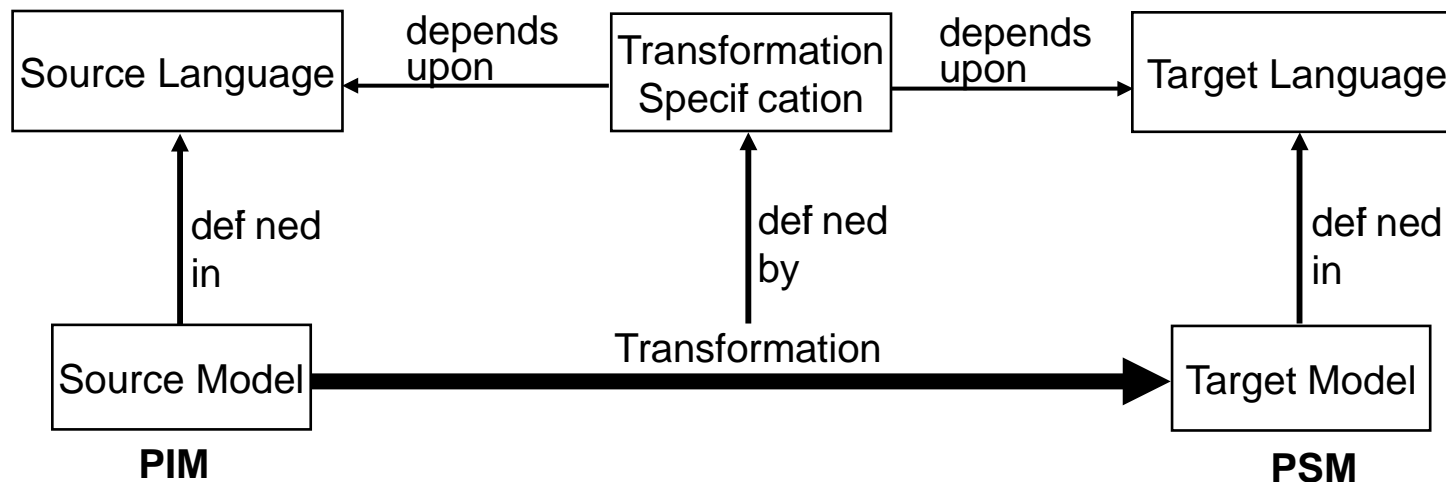
Ein **PSM** berücksichtigt die jeweilige Basistechnologie, auf der ein **PIM** zum Einsatz kommen kann (CORBA-Broker, .NET-Spezifika oder das Web-Service-Protokoll SOAP).

Auch **PSMs** können mit der UML modelliert werden. In jedem Fall werden aus den **PSMs** die **Codegerüste** erzeugt, die die Komponenten-Entwickler dann weiter bearbeiten.

MDA-Transformationsprozess

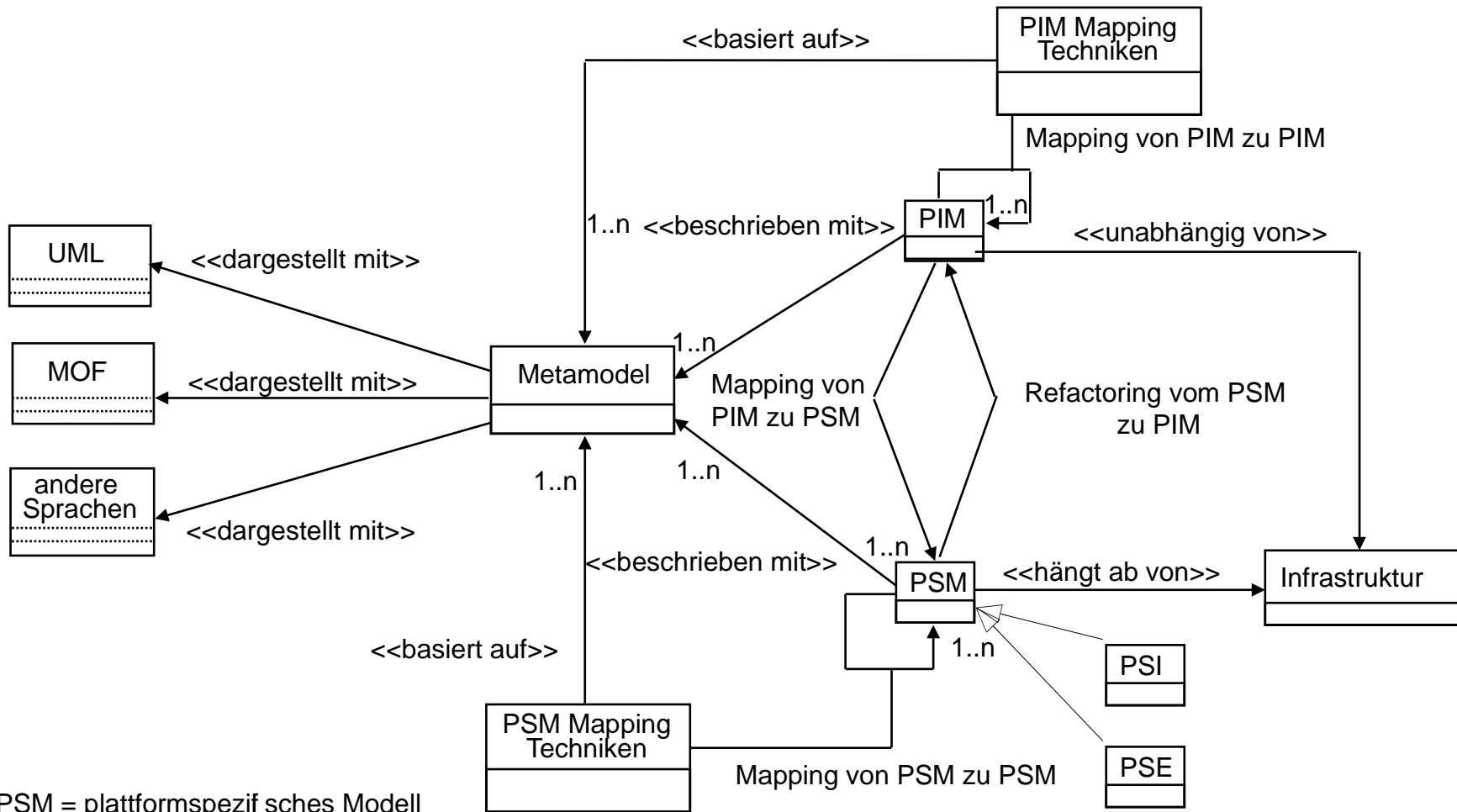
Aus plattformunabhängigem (*independent*) Metamodell **PIM** sind mittels Regeln, Techniken plattformspezifische (*specific*) Modelle **PSM** zu entwerfen, zu generieren, oder abzuleiten, um neue Anwendungen für eine bestimmte (Komponenten-)Plattform zu erhalten.

Ein weiteres Ziel von MDA ist die Integration solcher Technologien wie CORBA, J2EE, .Net und XML als *Plattform*.



Quelle: Kleppe, A., Warmer, J., Bast, W.: MDA Explained - Practice and Promise of the Model Driven Architecture; Addison Wesley 2003 (Draft 25.10.02)

Das MDA-Metamodell



PSM = plattformspezifisches Modell
 PSE = plattformspezifische Erweiterung
 PSI = plattformspezifische Implementation

Transformationen bezeichnet man auch als **Abbildungen (mappings)**. Mapping von PIM zu PIM schafft neue „Business Viewpoints“, von PSM zu PIM Abstraktionen aus plattformabhängigen Implementierungen und zwischen PSM weiteren Verfeinerungen oder Zielplattformen.

Model Management

- ▶ In der MDA müssen Modelle (Graphen) verwaltet werden:
 - Modellalgebren
 - Lookup
 - Diff, comm, union, compose
 - Versionsmanagement
 - Konfigurationsmanagement
- ▶ Das führt auf metamodellgesteuerte Repositorien/Modellinfrastrukturen (siehe Kapitel “Repositories” und “Modellmanagement”)

Bewertungsaspekte von MDA-Tools

- ▶ Unterstützung der Metamodellierung
 - Metamodelle der Sprachen UML 2.0, OCL, CWM (MOF 2.0-basiert)
 - Metamodellgesteuerte Repositorien
 - Erweiterungsmöglichkeiten der UML-Profile durch explizite Metamodellierung sowie Modellprüfung
 - Austauschformate: Import, Export und Validierung von Modellen auf Basis ihres Austauschs mit XMI 2.0
 - Validierung der Modelle mit OCL
- ▶ Model-to-Model Mapping bzw. Transformation (z. B. PIM zu PSM) mit QVT, ATL oder einer proprietären Sprache
- ▶ Forward-, Reverse- bzw. Roundtrip-Engineering auf der Code-Ebene
 - Codegenerierung (Model-to-Code Transformation, PSM zu PSI)
 - Mapping zu einer Programmiersprache wie z. B. JMI
- ▶ Modellierung von Testfällen und automatische Generierung der Testdaten (Model-driven Testing)

Quelle: Petrasch, R., Meimberg, O.: Model Driven Architecture - eine praxisorientierte Einführung in die MDA; dpunkt-verlag 2006

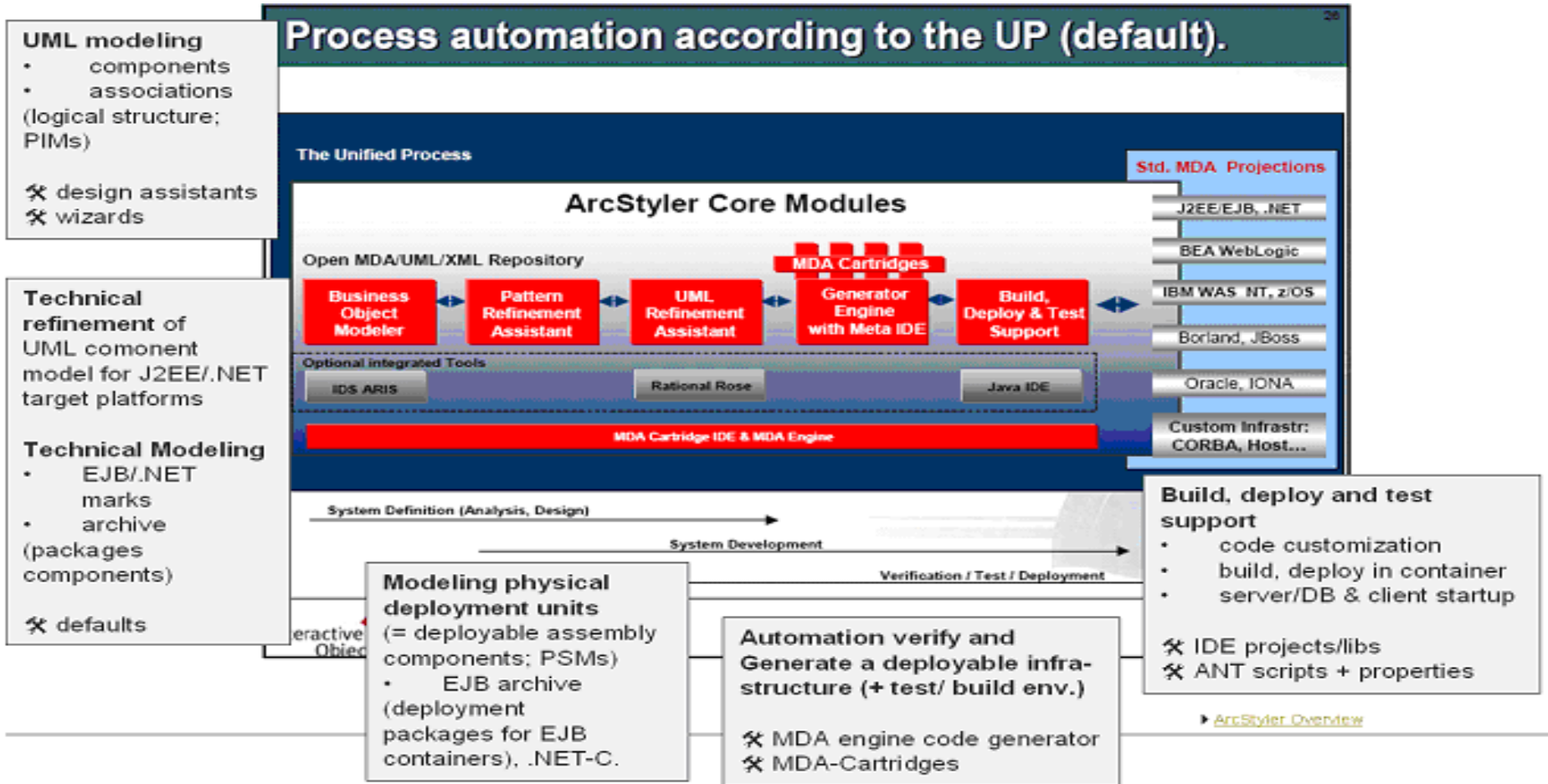
Werkzeugfunktionen am Bsp. ArcStyler

Das Werkzeug ArcStyler ist im Zusammenspiel mit einem UML-Editor wie MagicDraw (oder Rational Rose...) ein leistungsfähiges Werkzeugsystem (MDA-Suite), mit dem sich zum Beispiel J2EE-Applikationen gemäß den Konzepten der MDA entwickeln lassen.

- ▶ Object Modeler erfasst Anforderungen unabhängig von Plattform (funktionale, essentielle Anforderungen) Basis CRC-Cards Technologie
- ▶ Pattern Refinement Assistent überführt Fachmodell interaktiv in PIM UML-Modell (Basis MagicDraw oder Rational Rose) mit Annotation der essentiellen Design-Entscheidungen
- ▶ Verfeinerung des Fachmodells top down in untergeordnete UML-Diagramme und Quellcodegenerierung ebenfalls mit UML-Tool (MagicDraw)
- ▶ Codevervollständigung und Optimierung für jeweiligen Applikationsserver mit Cartridges (Codegenerierungs-Plugins)
- ▶ Komponentengenerierung für Oberfläche sowie weitere Projekt- und Konfigurationsdateien mit JBuilder.
- ▶ Schnittstelle zu IDE ist Standard „Ant Build Process“
- ▶ Datenbankgenerierung über Skripte zum Erstellen der DB-Schemas möglich.

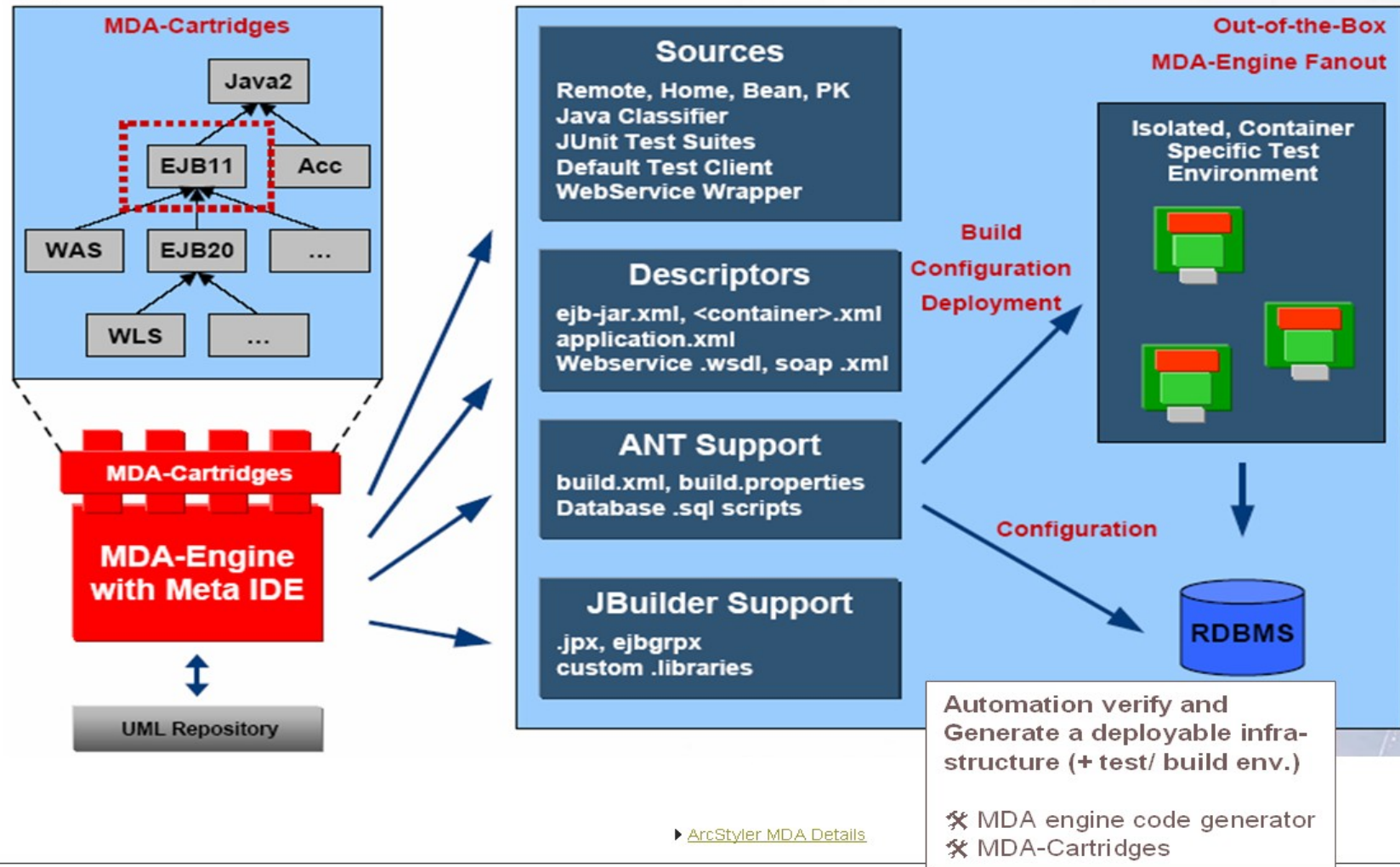
Quelle: Versteegen, G.: Wege aus der Plattformabhängigkeit - Hoffnungsträger Model Driven Architecture;
Computerwoche 29(2002) Nr. 5 vom 1. Febr. 2002

Vorgehen und Unterstützung beim ArcStyler



<http://www.interactive-objects.com/products/arcstyler/supportdocumentation.html>

Cartridges und generierte Artefakte



Quelle: Butze, D.: Entwicklung eines Praktikums für die werkzeuggestützte Softwareentwicklung nach der Model-Driven-Architecture; Großer Beleg an der Fakultät Informatik der TU Dresden 2004

Kurzbeschreibung weiterer MDA-Tools

	Integriert in	URL
AndroMDA	Eclipse	http://www.andromda.org/
XText, Xpand	Eclipse	http://www.eclipse.org/Xtext/
IBM Rational Suite Software Architect	Eclipse	
BITplan smart Generator	Eclipse	http://www.bitplan.com/

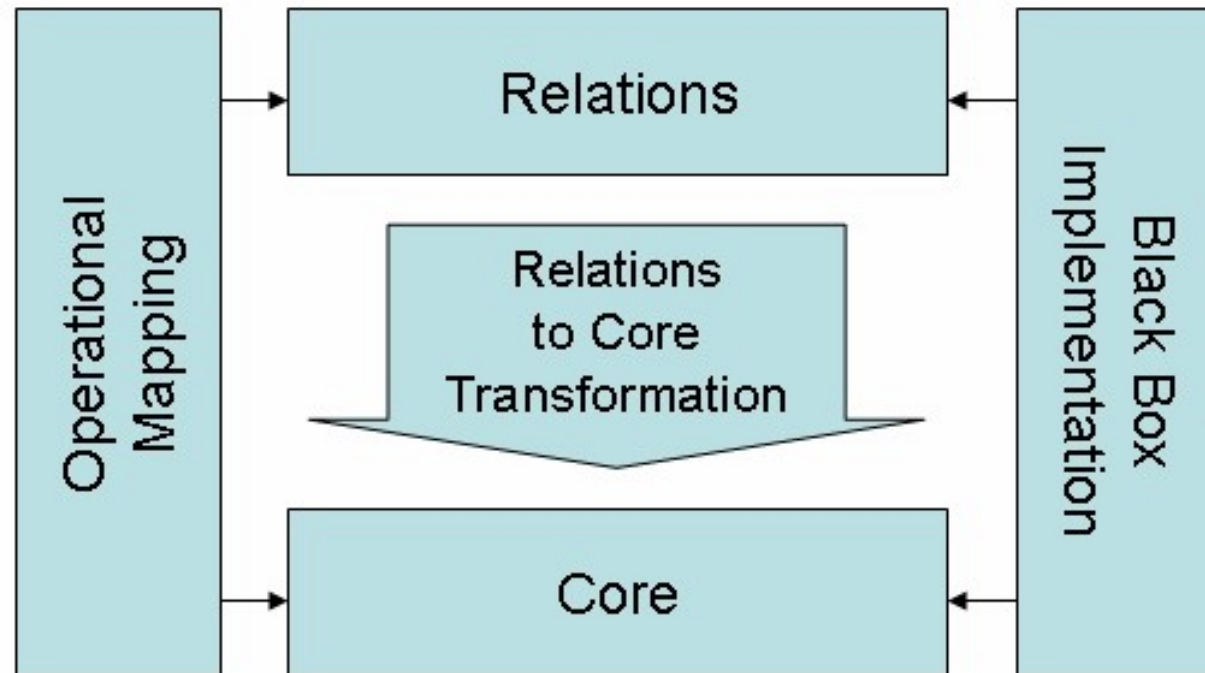
Quelle: Petrasch, R., Meimberg, O.: Model Driven Architecture - eine praxisorientierte Einführung in die MDA; dpunkt-verlag 2006





47.2 Query-View-Transformations (QVT)

The language of the OMG for model transformations within MDA



Transitive Hülle mit QVT Relations

// Transitive Hülle in QVT relations, mit Verwendung einer rekursiven Relation

```
relation transitiverelation {
  domain node:Node {
    // matching attributes
    name = sameName;
  }
  domain node2:Node {
    // node2 must have the same name as node
    name = sameName;
  }
  domain node3:Node {
    // node3 must also
    // have the same name
    name = sameName;
  }
  when {
    // conditions: base relation must exist
    baserelation(node,node2) or
    // or a transitive relation to a base relation
    (transitiverelation(node,neighbor)
    and baserelation(neighbor,node2));
  }
  where { // Aufruf einer Transformation
    makeNodeSound(node);
  }
}
```



QVT und EARS

- ▶ QVT Relational ist eine Sprache, die Kantenadditionssysteme realisiert.
- ▶ Werkzeuge, die QVT-Relational unterstützen, sind in Wirklichkeit einfache Graphersetzungssysteme

Tool			
Eclipse M2M Project	Operational	http://www.eclipse.org/m2m/	
Magic Draw	Operational		
MediniQVT	Relational	http://projects.ikv.de/qvt/wiki	

47.3 ATL - eine QVT-alike Sprache für Modelltransformationen

ATLAS Transformation Language
Statt QVT ist in der Praxis auch ATL sehr beliebt
<http://www.eclipse.org/atl/>

ATL integriert OCL

```
// Transitive Hülle in ATL, mit Verwendung einer rekursiven OCL Query
rule computeTransitiveClosureBaseCase {
  from node: Node (
    // possible to call OCL expressions
    node->baserelation.collect( e | e.baserelation)->flatten() );
  )
  to newNode mapsTo node (
    // set new transitive relation
    newNode->transitiverelation <- node->baserelation
  )
}
rule computeTransitiveClosureRecursiveCase {
  from node: Node (
    node->transitiverelation.collect( e | e.baserelation)->flatten() );
  )
  to newNode mapsTo node (
    // set new transitive relation
    newNode->transitiverelation <- node->transitiverelation
  )
}
```

- 
-
- ▶ OCL ist eine Graph-Query-Sprache, ähnlich zu EARS und .QL

```
rule checkNoDoubleFeatureInSuperClasses(name:String) {  
  from node: Class (  
    node->TransitiveClosure()->collect().exists(s | s.name() = name);  
  )  
  to  
    System.out.println("Error: super class has doubly defined feature: "+s.name());  
}
```

End