

Model Driven Architecture (MDA)

What is MDA?

MDA provides an approach for, and enables tools to be provided for:

- specifying a system independently of the platform that supports it,
 - specifying platforms,
 - choosing a particular platform for the system, and
 - transforming the system specification into one for a particular platform.
-

Characteristics of MDA

- Using *models* to direct the course of understanding, design, construction, deployment, operation, maintenance and modification.
 - Separating the *specification* of system functionality from the specification of the *implementation* of that functionality on a specific technology platform.
 - architectural separation of concerns.
-

Promise of MDA

- To allow definition of machine-readable application and data models which allow long-term flexibility of:
 - implementation
 - integration
 - maintenance
 - testing and simulation
-

Primary Goals of MDA

- Portability
 - Interoperability and
 - Reusability
-

Models, Architectures, and Viewpoints

- A *model* is a representation of a part of the function, structure and/or behavior of a system.
 - The *architecture* of a system is a specification of the parts and connectors of the system.
 - A *viewpoint* is an abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns.
-

Formal Models

- A specification is said to be *formal* when it is based on a language that has a well-defined form ("syntax"), meaning ("semantics"), and possibly rules of analysis, inference, or proof for its constructs.
 - In the MDA, a specification that is not *formal* in this sense, is not a model.
-

Platforms and Views

- *Platform* -- technological and engineering details that are irrelevant to the fundamental functionality of a software component.
 - *View* -- a representation of that system from the perspective of a chosen viewpoint.
-

MDA Viewpoints

- *Computation Independent Viewpoint*
 - focuses on the on the environment
 - computation independent model (CIM)
 - *Platform Independent Viewpoint*
 - focuses on the operation of a system while hiding the details of a particular platform
 - Platform Independent Model (PIM)
 - *Platform Specific Viewpoint*
 - focuses on the detail of the use of a specific platform
 - Platform Specific Model (PSM)
-

PIM and PSM

- *Platform-independent model (PIM)* -- a formal specification of the structure and function of a system that abstracts away technical detail.
 - *Platform-specific model (PSM)* – a model expressed in terms of the specification model of the target platform.
-

Mappings of Models

□ **PIM to PIM.**

- enhancing, filtering, or specializing models
- no platform dependent information

□ **PIM to PSM.**

- PIM is sufficiently refined to be projected to the execution infrastructure.
-

Model Transformation

□ **PSM to PSM.**

- component realization and deployment.

□ **PSM to PIM.**

- abstracting models of existing implementations in a particular technology into a PIM.
-

Mapping of Models

- An MDA mapping provides specifications for transformation of a PIM into a PSM for a particular platform.
 - *Model type mappings*
 - types in PIM → types in PSM
 - *Model Instance Mappings*
 - Elements in PIM → elements in PSM
-

Marks for Mapping

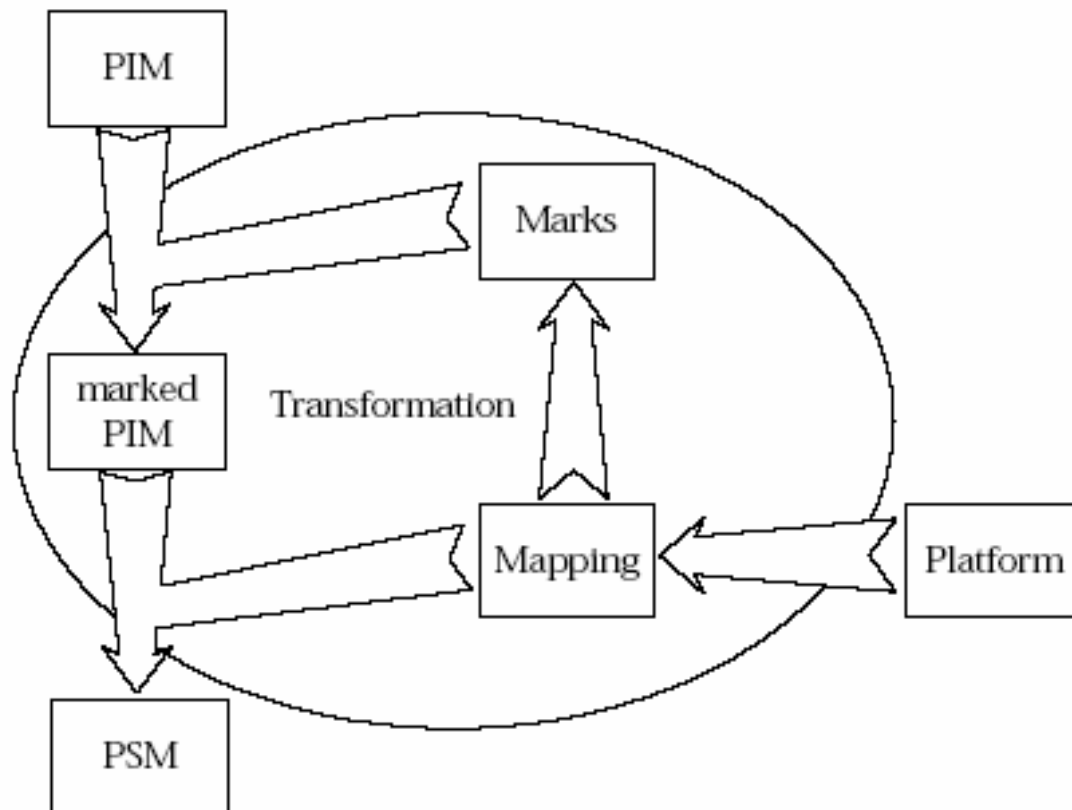
- Represents a concept in the PSM
 - Applied to an element of the PIM
 - Indicate how the element in PIM is to be transformed
 - types from a model, specified by classes, associations, or other model elements
 - roles from a model, for example, from patterns
 - stereotypes from a UML profile
-

Mapping Language

- A language to describe a transformation of one model to another.
 - in natural language,
 - in an algorithm in an action language, or
 - in a model mapping language
 - a formal language to specify the model mappings
-

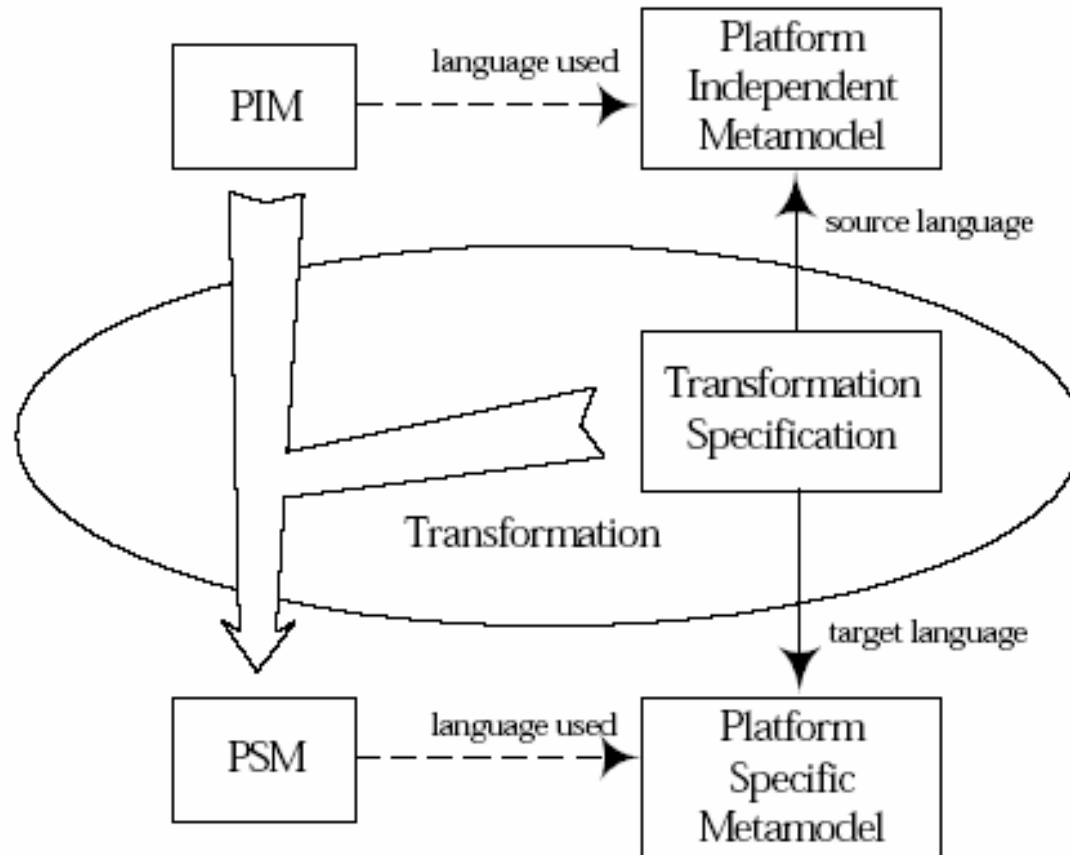
Model Transformation Approaches

□ Marking



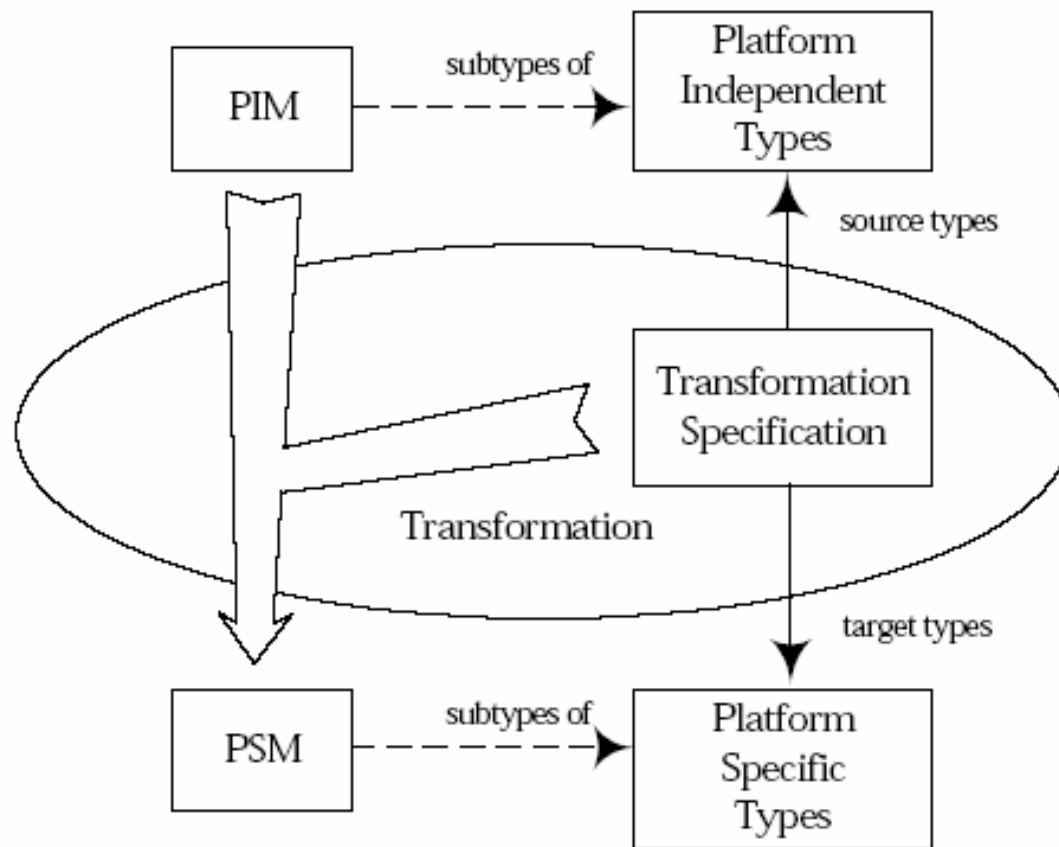
Model Transformation Approaches

- Metamodel Transformation



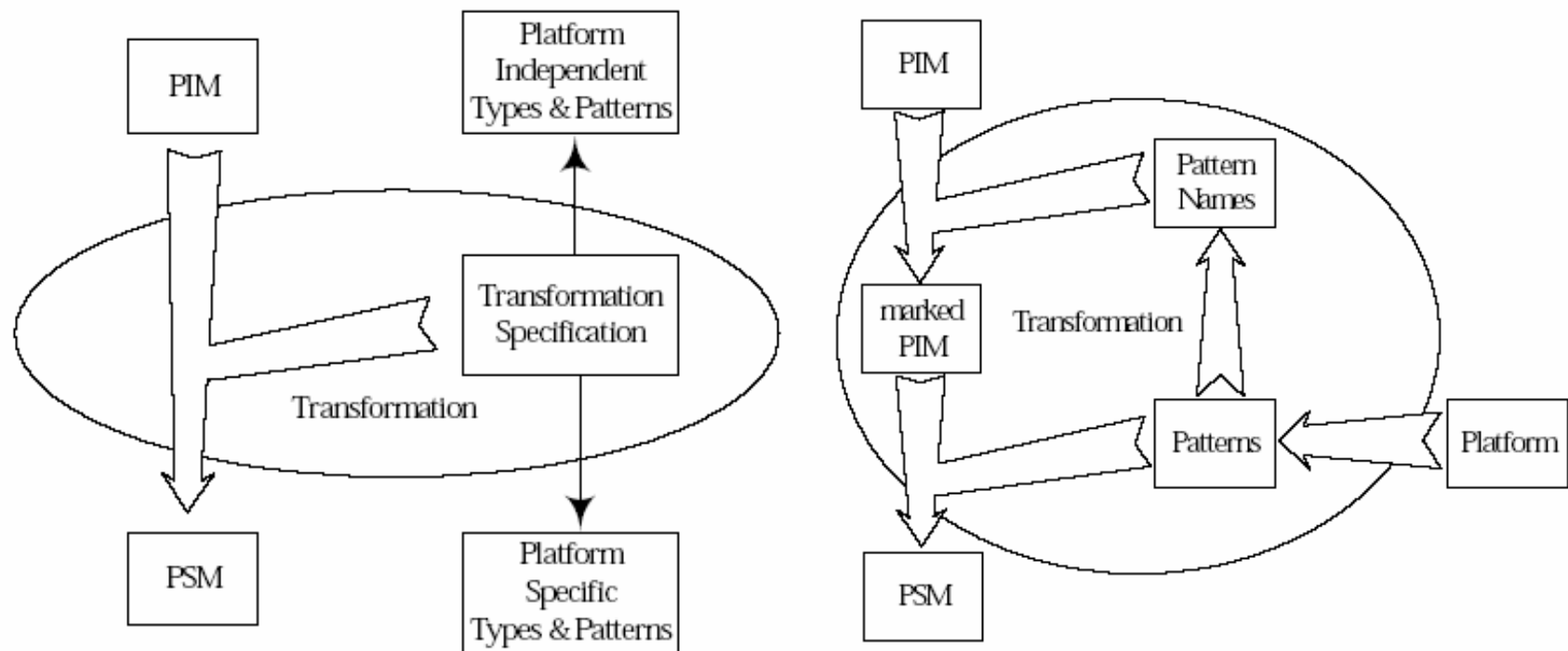
Model Transformation Approaches

□ Model Transformation



Model Transformation Approaches

□ Pattern Application



Model Transformation

□ Methods

- Manual transformation
- Using profiles
- Using patterns and markings
- Automatic transformation

□ Input

- Patterns
 - Technical choices
 - Quality requirements
-

MDA and Standards

- UML and MOF (Meta-Object Facility)
 - Meta-modeling layers
 - M0: user objects
 - Address **addr** = new Address(...)
 - M1: domain models
 - class **Address** { ... }
 - M2: meta-models
 - UML
 - M3: meta-meta-models
 - MOF
-