



Some thoughts about MDD

Agenda

- When to use MDA/MDD ? Does it always make sense ? Is it always possible ?
- MDD Process
- What do we need for MDD ?
- How to build the Model / What should be considered
- How to build Transformations
- MDD Tool Chain
- “Ideal” MDD Tool
- Difficulties with MDD, Why *Not* MDD
- Samples
- “Ideal” MDD Process

When to use MDD ?

- New Development
- Legacy Transformation - reverse engineering. Automate discovery of models for re-integration on new platform
- Automating bridges.

- Project Characteristics must be considered
 - Project size
 - Team organization and size
 - Architecture, Technology
 - Skills

Model driven development isn't just about producing software faster, though it can certainly do that. MDD is also about producing better software.

When to use MDD ?

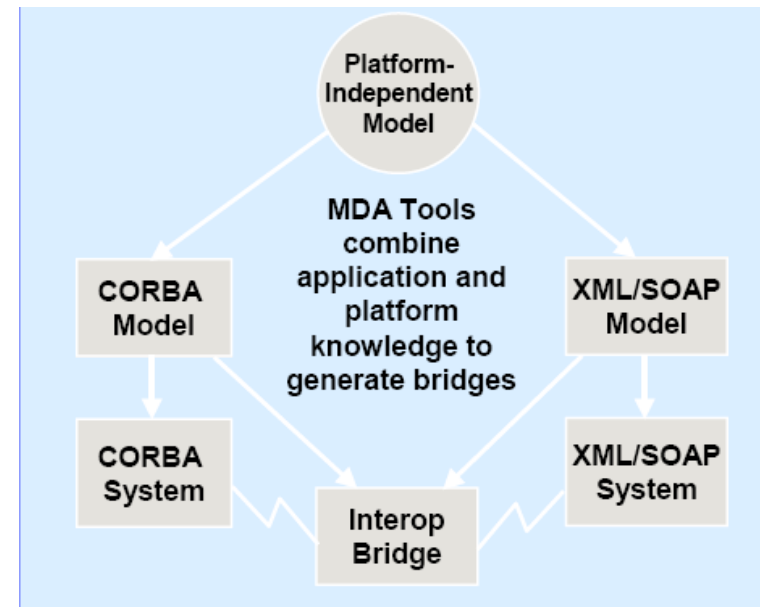
- Say something about
 - Costs
 - Standard Frameworks
 - Team organization
 - Outsourcing
 - Project life-cycle
 - Change Management
 - Software Quality (Design Quality – can be validated)
 - **"lightweight" MDA: produce** code for limited scope portions of an overall design, like communications protocols, state machines (for UIs and embedded systems), data models, and data flow engines.

When to use MDD ?

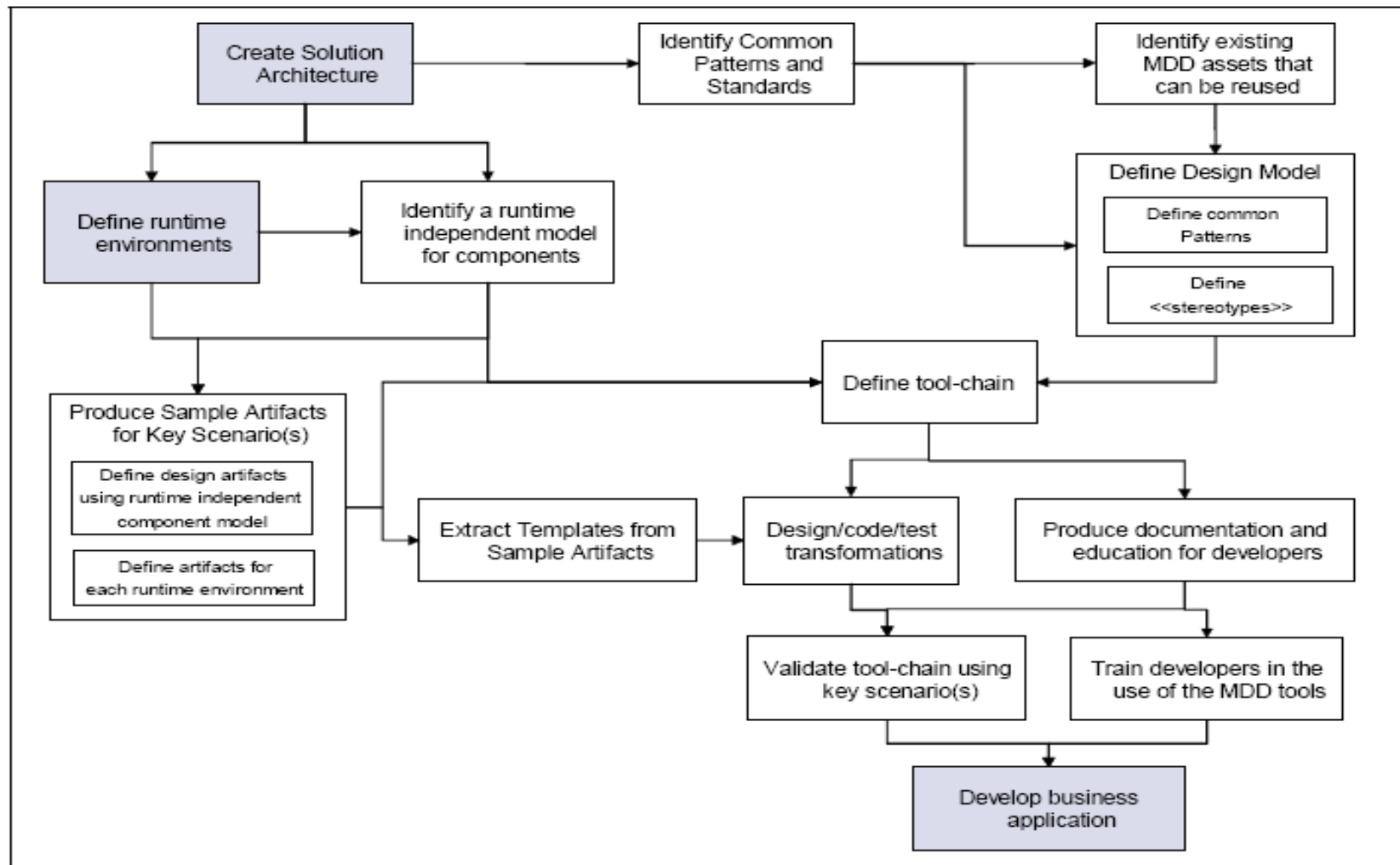
Activity	% of Savings
<i>Business Requirements Gathering</i>	10%
<i>Design and UML Modeling</i>	-20%
<i>Application Construction</i>	60%
<i>Application Iterations</i>	70%
<i>Testing</i>	80%
<i>Documentation</i>	70%
<i>Runtime Environment Tuning</i>	85%
<i>Project Management</i>	60%

Bridge Generation

- Simplify creation of integrated applications.



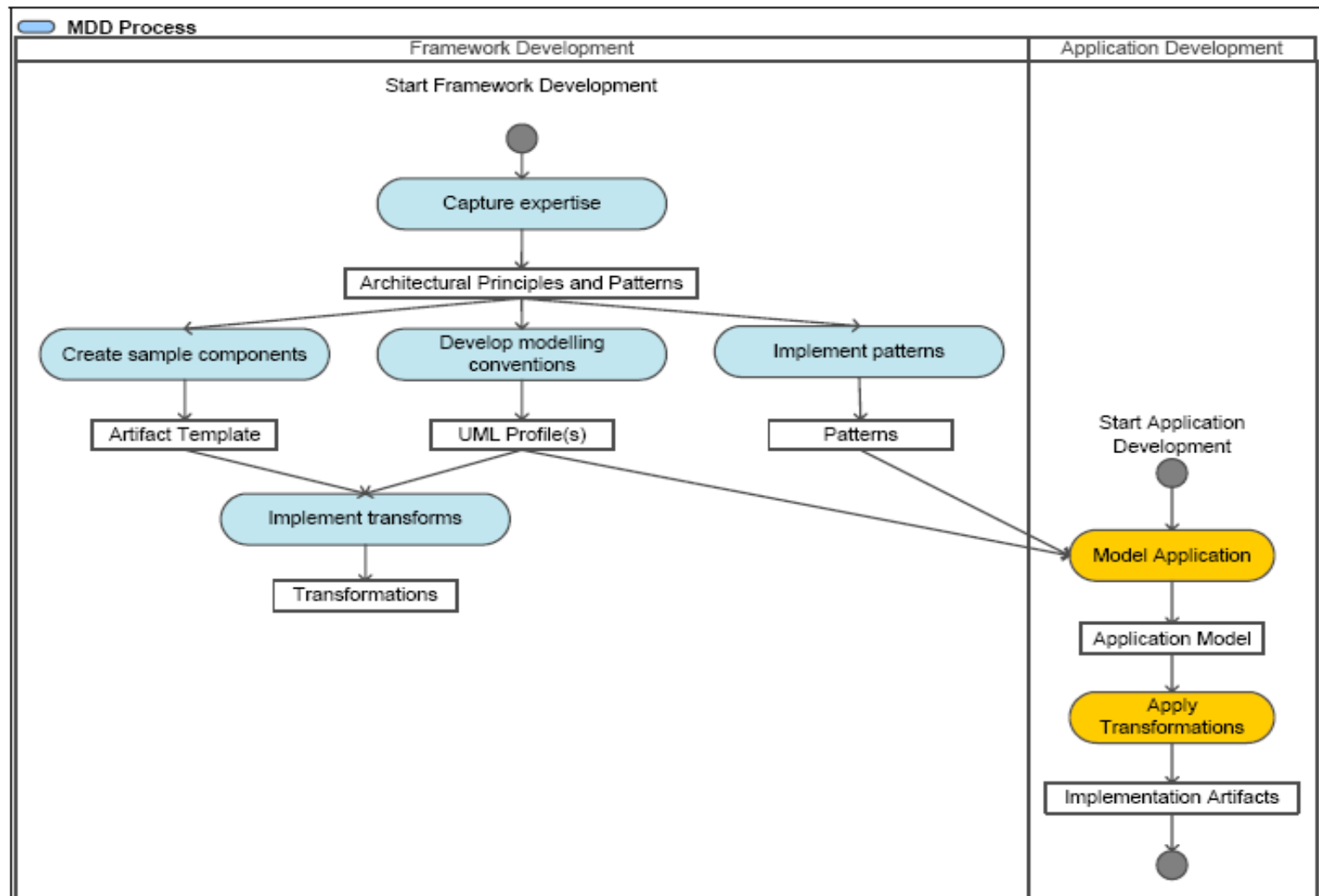
MDD Process



MDD Process

- **Normal development process:**
 - formalize knowledge
 - render the knowledge as an implementation
- **MDD**
 - **Build the model of a platform in a modelling language that can be transformed into others**
 - **Map these models onto target platform**
 - **Perform transformations**

MDD Process (There is no magic here. Somebody must do it.)



What do we need for MDD

- **Precise modeling**
 - Abstraction is not the same as imprecision
- **Automation. MDD and Tools. Achieved using two main techniques:**
 - **Transformations: generation of artefacts from models.**
 - **Patterns: creation and modification of model elements within a model.**
- **Architectural style**
 - **Category of software platform that is to be used (e.g. service-oriented, aspect-oriented).**
 - **Subject matter of the software (e.g. telecoms, finance, retail).**
 - **Other factors that influence the concepts that are appropriate for software modeling in a particular context (e.g. used frameworks).**

What do we need for MDD

- **Modeling language**
- **Expertise capture**
 - **Logical architecture expertise**
 - **Technical architecture expertise**
- **Patterns**

How to build the Model / What should be considered

MDA speaks about PIM and PSM

- PIM vs. PSM
 - **PSM has to use the platform concepts: exceptions, tapes, notation**
 - **PIM does not need these distinctions. It can use a simpler, more uniform model. It is easily validated.**
- Abstraction
- Do not be too specific. Avoid Technical Debt.

- “Everything” can be used.
- There is no standard for modeling.

- (Show different samples)

What to use to build Models

- UML, UML Profiles, OCL
- XMI (XML Metadata Interchange)
- MOF (Meta Object Facility)
- CWM (Common Warehouse Metamodel)

- Think about good visualization

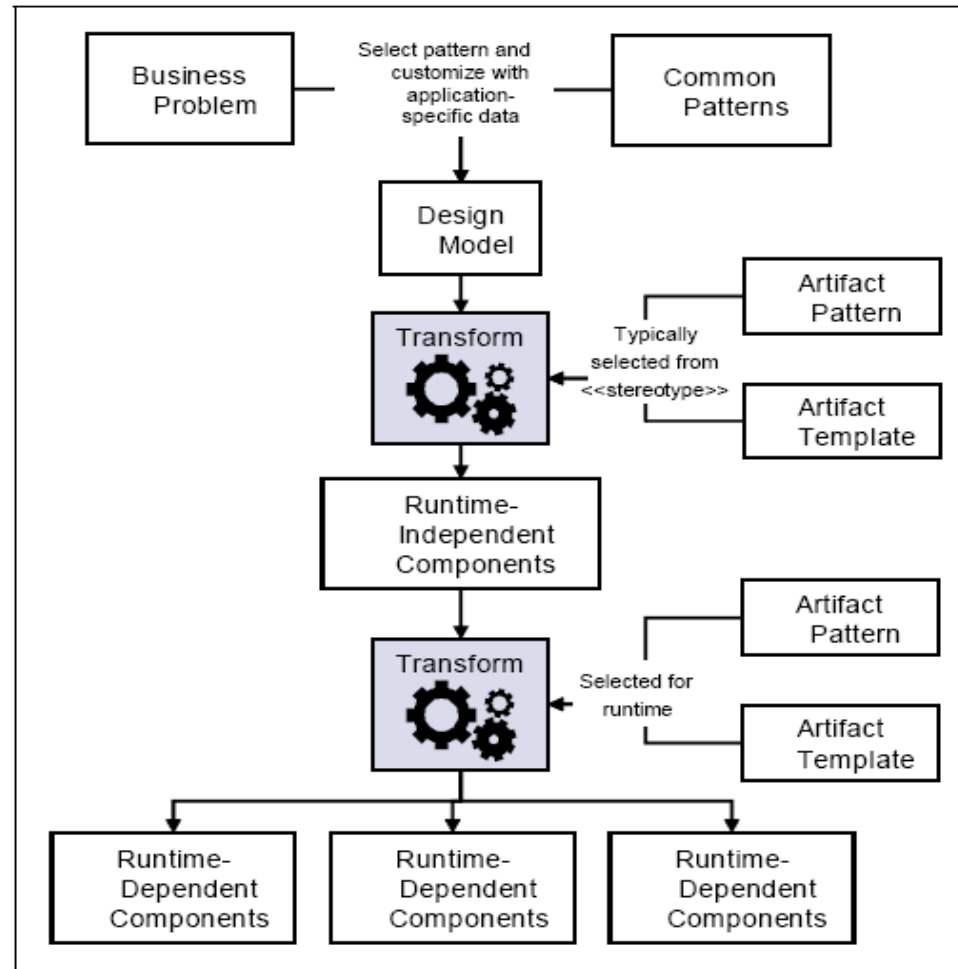
How to build transformations

- Transformation language
 - Eclipse Jet (Java Emitter Templates)
 - Velocity
 - QVT (Query/View/Transformations) language
- Rules mapping
 - XML configuration
 - UML Models as configuration elements
 - Rule-Based Configuration

How to build transformations / what to consider

- Develop transformations only when the semantic connections between the model elements are well understood.
- Not all semantic connections make for good MDA transformations.
- Writing MDA transformations should be treated as a software development project itself.
- In most MDA situations, the model-to-model mappings are complex and require careful design and implementation.
- All transformations should be created such that they can participate in an iterative process, and hence, be repeatedly applied without loss of unrelated information.
- Think about manual changes.

MDD Tool / Transformation Chain



MDD Tool / Transformation Chain

- What can be done:
 - Produce Documentation
 - Generate Tests
 - Validate Models
 - Invoke Patterns
 - Invoke “standard” transformations
 - Java Annotations, XDoclets, different Wizards and tools

“Ideal” MDD Tool

- “Standard” capabilities
 - UML modelling
 - Models (or parts of models) can be web-published
 - Code generators for major (and some other) platforms
 - XMI model interchange
- Transformation Language and OCL Support
- Integrated IDE
- Integration of modules at the PIM level
 - Reuse previously-built PIM modules
- Reverse engineering
- QVT (Query/View/Transformations) language

Difficulties with MDD, Why *Not* MDD

- Traceability
- Poor granularity. Models too specific. Examples: method calls, persistence, behaviour parameters, etc. Often these details don't even show up on the screen.
- Crappy Tools
- Resulting code is so large and complex that any meaningful analysis of it is virtually impossible.
- It is impossible to try out quick changes or do any kind of rapid prototyping.
- The architecture and tools are so complex that new staff required long training periods in order to be productive.

Difficulties with MDD, Why *Not* MDD

- The implementation requires a number of different "one-off" solutions that can't easily and naturally be incorporated in the model (for instance, where the model is based on an early, simplistic view of how the application should behave).
- Late-arriving requirements vs. existing MDD Tool.
- Domain-Specific Modeling vs. Solution domain (classes, attributes, operations).
- Sociological issues.

“Ideal” MDD Process

- Take a model of an application subject matter off the shelf.
(Enterprise Patterns)
- Subset the model as necessary.
- Take models of the implementation technologies off the shelf.
- Describe how the models are to be linked.
- Mark the source models.
- Generate the system.

- Asset-based development (ABD). Reuse of models, patterns and transformations.
- OMG Reusable asset Specification (RAS): about packaging, documentation, discover and retrieve software assets. (something like UDDI - **Universal Description, Discovery and Integration**)