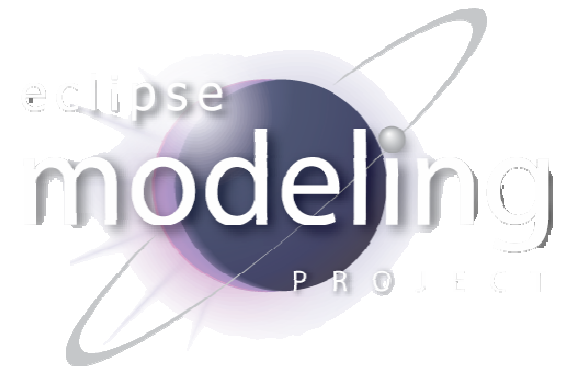


Eclipse Modeling Project Webinar

March 27, 2008

Richard Gronback & Ed Merks
Modeling PMC



Agenda

- DSL Overview
- Developing a DSL
 - Abstract Syntax (domain model)
 - Graphical Concrete Syntax (diagram)
 - Textual Concrete Syntax
- Model Transformation
 - Model-to-Model using QVT OML
 - Model-to-Text using Xpand
- Demo
- Summary

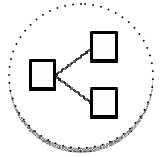
Domain-Specific Languages

- Definition
 - A language designed to be useful for a specific set of tasks, as opposed to a general purpose language
- Syntax
 - Abstract : defined using BNF, EBNF, XSD, MOF, **Ecore**, etc.
 - Concrete : typically graphical or textual
- Tooling
 - Can be largely generated, even bootstrapped
 - Defined using the *abstract* syntax:
 - constraints, validation, grammar (textual notation), graphical notation, model-to-model transformation, model-to-text definition
- Motivation
 - Focus is on the problem domain (the *purity of essence*)
 - Rigorous, as required by adhering to the *abstract* syntax
- Downside
 - Model-centric DSL tooling relatively immature
 - You may end up with something as large/complex as the UML

Eclipse Modeling Project

- A Range of Model-centric DSL Capabilities:
 - Eclipse Modeling Framework (EMF)
 - Core, Query, Validation, Transaction, Teneo, Net4j, CDO
 - Graphical Modeling Framework (GMF)
 - Generative tooling and runtime for diagramming
 - Textual Modeling Framework (TMF)
 - Generative IDE for textual modeling languages: TCS, Xtext
 - Model-to-Model Transformation (M2M)
 - ATL, QVT (OML)
 - Model-to-Text Transformation (M2T)
 - Xpand, JET
 - Model Development Tools (MDT)
 - UML2, OCL, UML2 Tools, XSD, . . .
 - Generative Modeling Technologies (GMT)
 - Research and emerging technology
 - Amalgamation
 - Aims to improve packaging, integration, and usability

Abstract Syntax Development

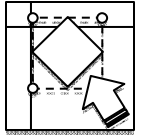


- EMF is used to develop the abstract syntax
 - Using the Ecore metamodel (similar but != EMOF)
 - **Generation** capabilities using JET/JMerge
 - Model code (API), Edit code, Editor, Tests
 - Runtime provides reflection, serialization, notification, etc.
 - Apply **constraints** with MDT OCL
 - Write custom templates to enforce at runtime
 - Model **query** support
 - Using OCL or SQL-like query language
 - Model **validation** framework
 - Using OCL or Java, batch and “live” processing
 - Model **transaction** support



Object Constraint Language (OCL)

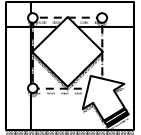
- It's Everywhere (*time to learn it!*):
 - EMF model constraints, invariants, pre/post-conditions, etc.
 - Used in Validation Framework
 - Used in EMF model Query
 - Used in GMF for link constraints, initializers, audits, metrics
 - Used as the basis of QVT Operational Mapping Language
 - OCL-ish languages in MOFScript, Xtend, Xpand



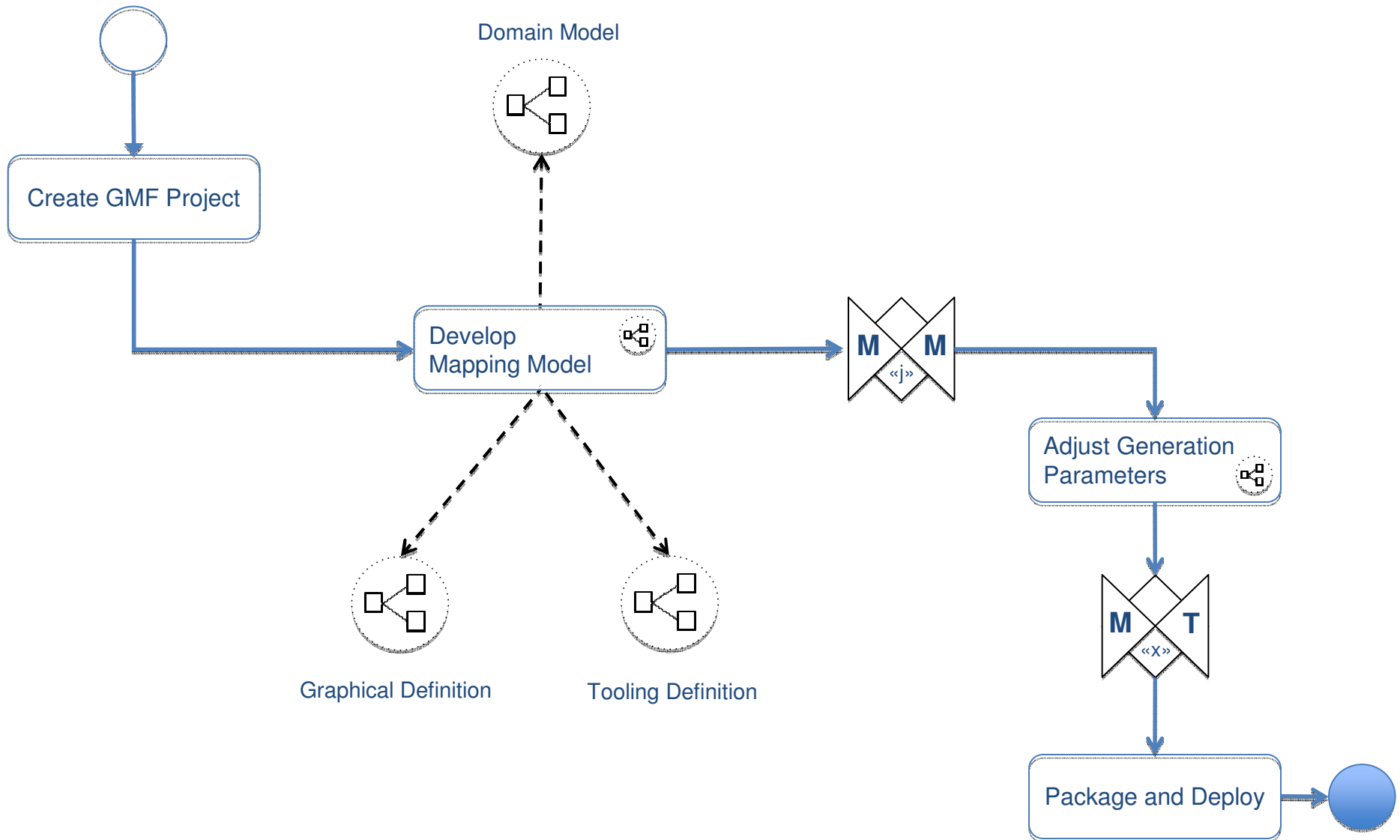
Graphical Concrete Syntax

- GMF is used to develop a graphical concrete syntax
 - GMF provides the tooling and runtime; you provide the notation
- Considerations:
 - Is your DSL well suited for graphical representation?
 - Do you need editing, or just visualization?
 - Where is the best place to map notation to domain?
 - GMF mapping model, or using QVT? (think BPMN and BPEL)
 - Read Tufte <http://www.edwardtufte.com>
 - Yes, I know I've disregarded his advice regarding PowerPoint :-)





GMF Overview



Textual Concrete Syntax



- Textual Syntax benefits
 - Familiar, well-established technologies
 - Syntax highlighting, code completion, outline, refactoring, versioning, comparison, merge, etc.
 - Not all languages have sensible graphical notation
 - Text used instead, or in combination with graphical
- Textual Modeling Framework (TMF)
 - Two components:
 - Xtext: grammar-based, with optional metamodel references
 - TCS: grammar derived from model
 - Should eventually target Eclipse IDE Meta-Tooling Platform (IMP)
 - Formerly called “SAFARI” – currently migrating to target Eclipse 3.3
 - Similar to what Martin Fowler calls a “language workbench”
- Emfatic provides a textual syntax for Ecore
 - Available on AlphaWorks <http://www.alphaworks.ibm.com/tech/emfatic>

Model Transformation

- Two forms: [Model-to-Model](#) and [Model-to-Text](#)
- Used for:
 - Integrations
 - Code generation
 - Reporting
 - Model exchange
 - Model migration and refactoring
 - ...
- Considerations:
 - Transformation languages can be complex (OML)
 - Complex metamodels make for complex transformations
 - Transform to dedicated model for code generation
 - Or, straight to templates? (think Java, C#, XHTML, etc.)

Model-to-Model Transformation



- QVT is used for M2M transformations
 - Implementation of the OMG's Query/View/Transformation
 - Operational Mapping Language (OML)
 - Used to define a set of mappings and queries
 - Based on extension to OCL (+ side effects)
 - Operates on input EMF model to produce output EMF model(s)
 - Output can be the same as input for in-place transformations
 - Core and Relations languages are coming...
 - “Higher level” transformation languages
- Alternative to QVT is ATL
 - Another component within the M2M project



Model-to-Text Transformation

- Xpand : a template engine for code generation
 - Straightforward syntax

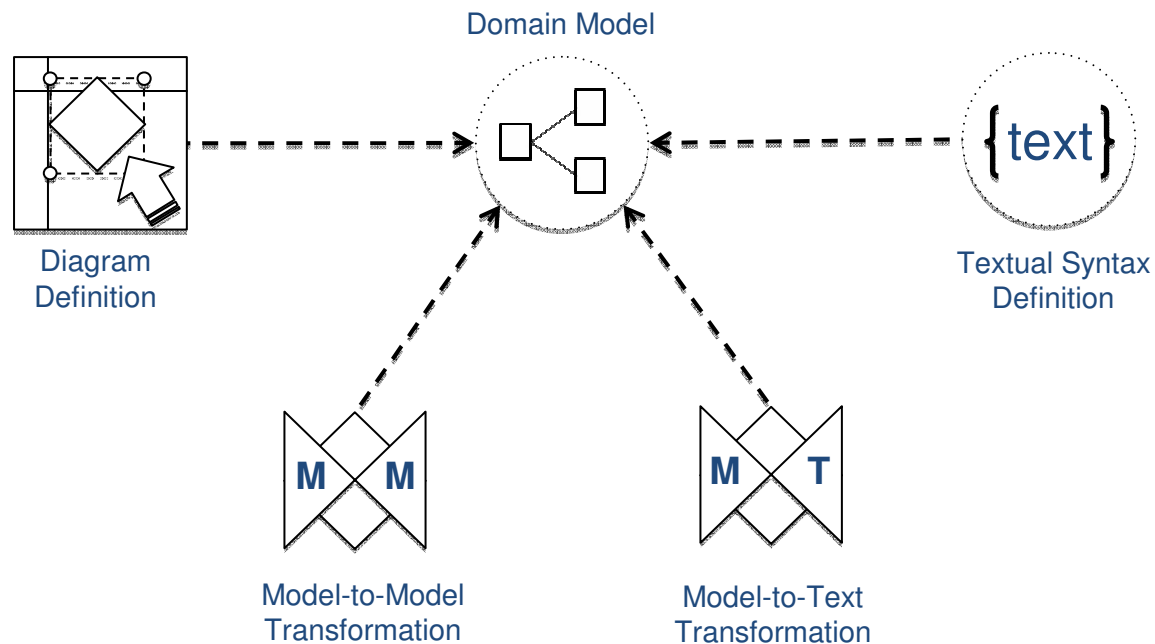
```
«IMPORT meta::model»
«EXTENSION my::ExtensionFile»
«DEFINE javaClass FOR Entity»
  «FILE fileName()»
    package «javaPackage()»; public class «name» {
      // implementation
    }
  «ENDFILE»
«ENDDFINE»
```
 - Aspect-Oriented capabilities
 - Used extensively in GMF
- Alternative is JET (JSP-like syntax)
 - Both JET and Xpand are in the M2T project

DSL Development Process

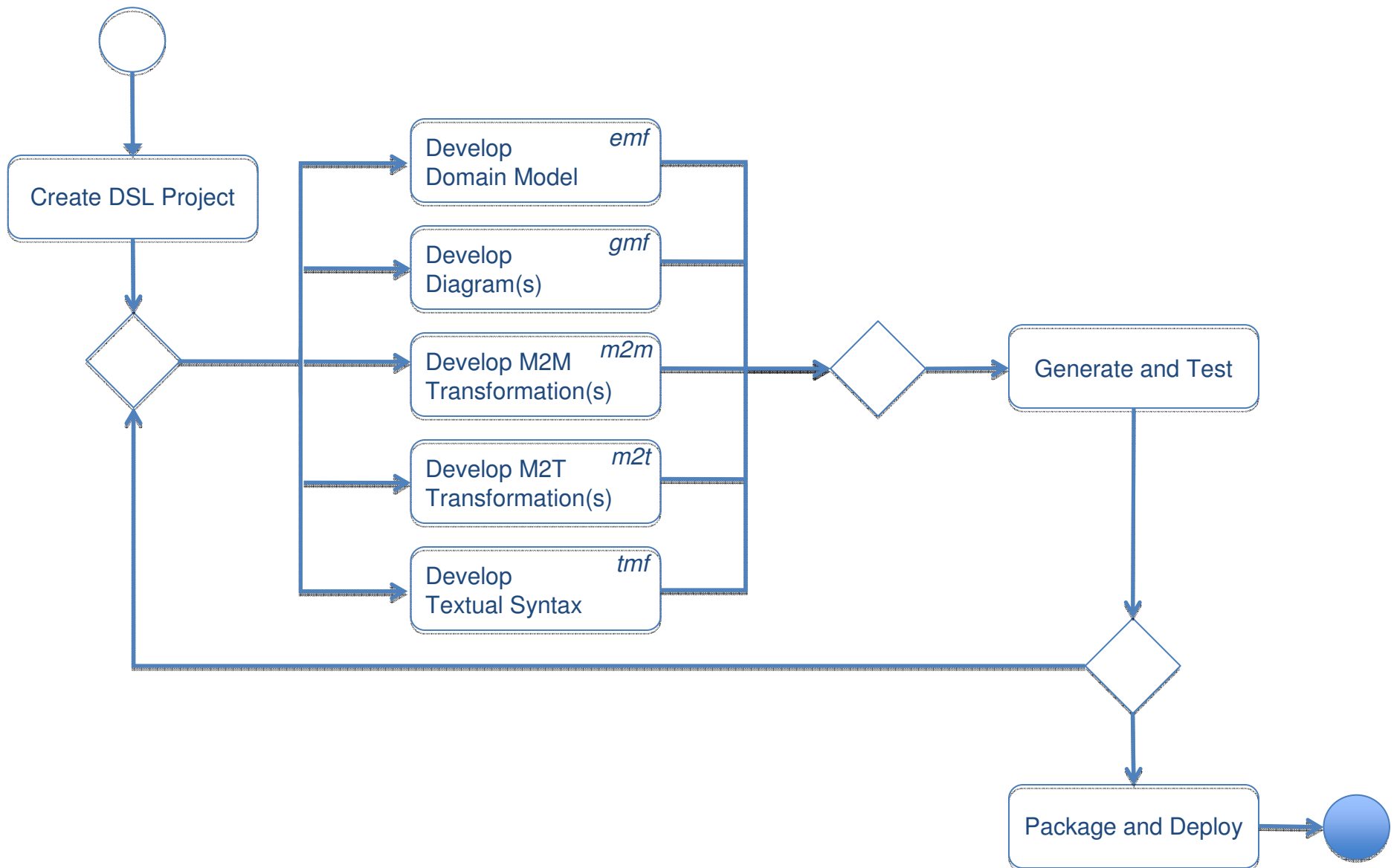
- Roles
 - Domain Expert
 - Provides input on structure and semantics of the DSL
 - Toolsmith
 - May also be the domain expert
 - Develops DSL artifacts for generation and deployment
 - Authors M2M and M2T transformation definitions
 - Practitioner
 - The “end user”

DSL Toolkit Overview: Toolsmith

- Development centered on Ecore-based domain model
 - Diagram definition using GMF for graphical concrete syntax
 - Model-to-Model transformations using QVT or ATL
 - Model-to-Text transformations using Xpand or JET
 - *Textual concrete syntax defined using TMF*

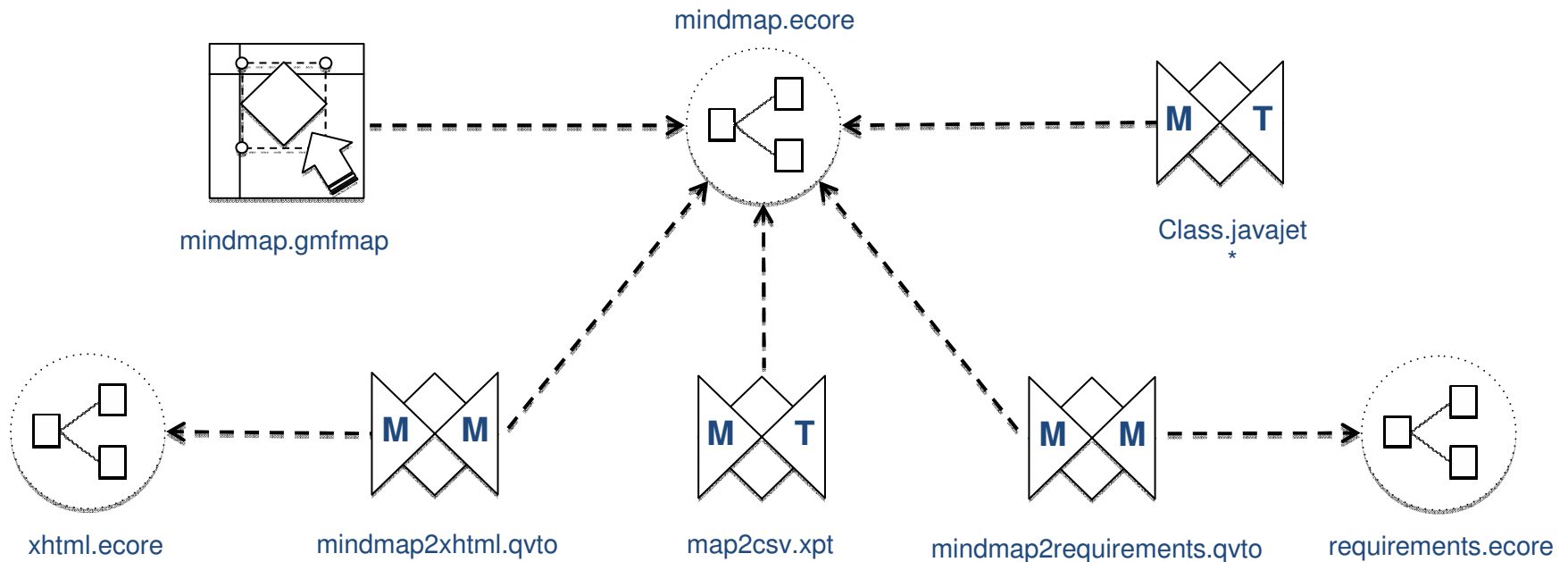


Toolsmith Process Overview

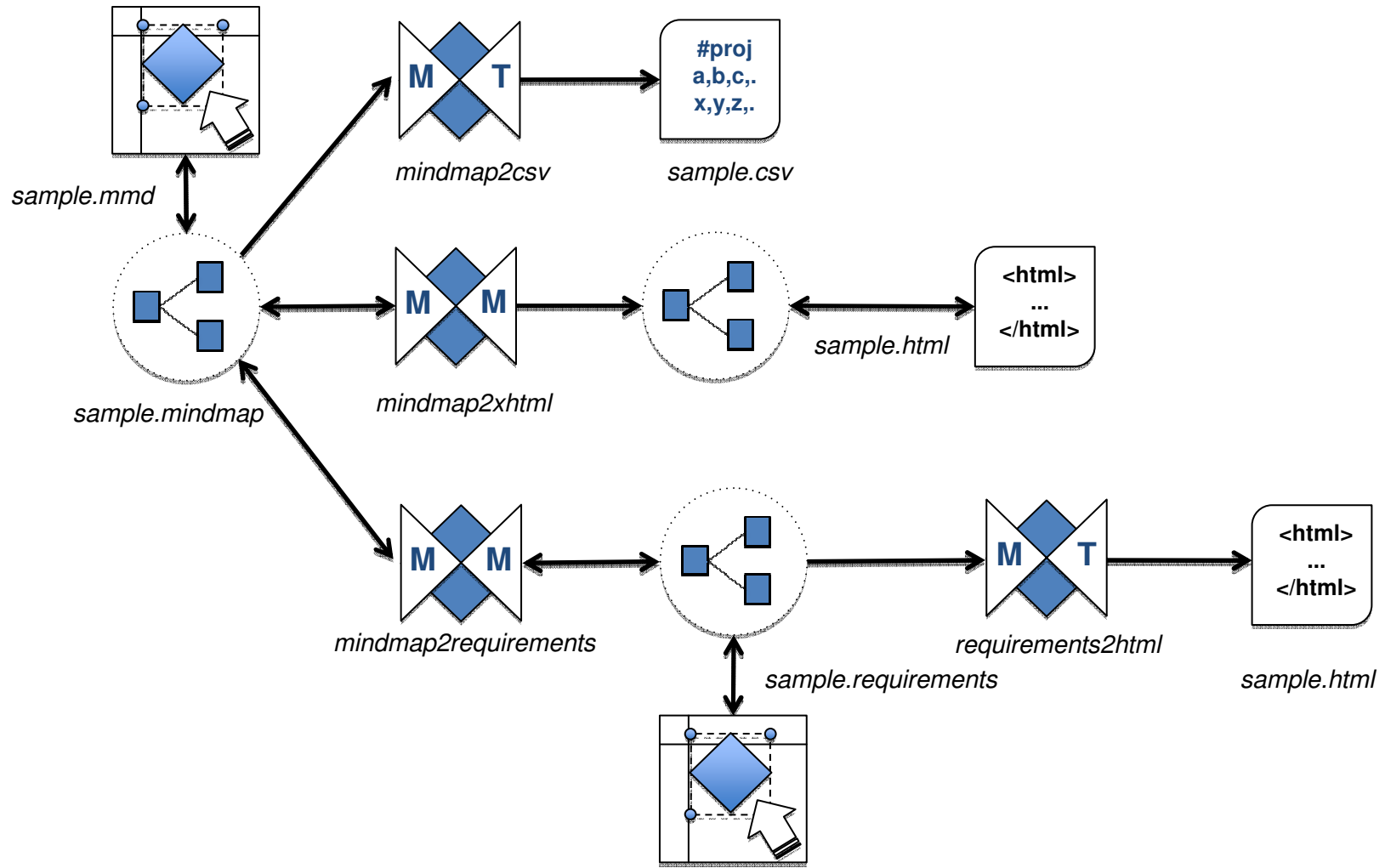


DSL Toolkit Overview: Toolsmith

- An Example:



DSL Toolkit Overview: Practitioner



Demo

- Scenario:
 - Toolsmith to create a mindmap application
 - Graphical concrete syntax only
 - Model-to-Model transformation to requirements model
 - Model-to-Model transformation to XHTML report
 - Model-to-Text transformation to CSV file

Summary

- Developing a Domain-Specific Language is not a trivial exercise
 - But, there are many possible advantages
 - Requires a domain expert [+ a toolsmith]
- Developing custom tooling it not a trivial exercise
 - But it's feasible, using EMP technologies
 - Reuse potential in common models and transformations
 - It *should* only get easier

The End

Thank you!

Richard C. Gronback
richard.gronback@borland.com

Questions?

References

- Eclipse Modeling Project <http://www.eclipse.org/modeling/>
- Eclipse Modeling Framework (EMF) website <http://www.eclipse.org/emf>
- Graphical Modeling Framework (GMF) website <http://www.eclipse.org/gmf>
- Model-to-Model Transformation (M2M) website <http://www.eclipse.org/m2m>
- Model-to-Text Transformation (M2T) website <http://www.eclipse.org/m2t>
- Model Development Tools (MDT) website <http://www.eclipse.org/mdt>
- Eclipse Modeling Framework Technology (EMFT) website <http://www.eclipse.org/emft>
- Emfatic <http://www.alphaworks.ibm.com/tech/emfatic>
- Martin Fowler on Language Workbenches
<http://www.martinfowler.com/articles/languageWorkbench.html>
- Model-Driven Software Product Lines (Krzysztof Czarnecki)
<http://swen.uwaterloo.ca/~chpkim/pp46-czarnecki.pdf>

Software Product Lines

- Likely the best application of a DSL Toolkit
 - DSLs can be customized to suit the needs of a customer
 - Generation output (templates) highly configurable
 - A mature, extensible target platform is key
 - Optionally, provide full generation
 - Feature trees used to select variations
 - Generate only what is required
 - Alternatively, enable only what is required
- Complemented by Framework-Specific Modeling Languages (FSMLs)
 - <http://gp.uwaterloo.ca/fsmls>

UML™/MDA® vs. DSL/MDD

- UML is a general purpose modeling language
 - Similar to general purpose programming languages (e.g. Java)
 - Can be seen as a collection of DSLs
 - Can be used to define a DSL (i.e. using profiles)
- MDA™ is a trademark the OMG
 - A collection of standards
 - Models defined in MOF or UML, refined/constrained/queried (OCL), transformed (QVT), and used to generate text (MOF2Text),...
 - MDA is often synonymous with Model-Driven Development™ (MDD)
 - *and Model-Driven Engineering (MDE), and MDSD, and...*
- The UML metamodel can be the starting point of your DSM tool
 - How important are standards to you?
 - How much complexity do you need in a metamodel or language?