Enhancing Transformation Adaptability Through Generalized Platform Models

Prof. Dr. Colin Atkinson
University of Mannheim

Symposium - How Adaptable is MDA?
University of Twente
20th May 2005

Role of the “Platform” Concept in MDA

- the concepts of “platform” and “platform model” lie at the heart of MDA
- however, these concepts are very vaguely defined in the MDA literature
  - “... a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.” MDA Guide.
- general consensus is that a platform is a vehicle for execution
  - CORBA, J2EE, .NET and Web Services or other Web-based platforms
What is Platform Independence?

- prevailing view of the MDA transformation chain

100% independent of Z
Model of A based on W → W

80% independent of Z
Model of A based on X → X

60% independent of Z
Model of A based on Y → Y

0% independent of Z
Model of A based on Z → Platform Z

Alternative View of Transformation Chain

- platforms do not have to support execution, just description

100% independent of Z
W specific model

80% independent of Z
X specific model

60% independent of Z
Y specific model

0% independent of Z
Z specific model
Platforms as Domain Specific Languages

- in many approaches “Domain Specific Languages” (DSMs) implicitly play the role of platform models
  - Microsoft Software Factories, Xactium XMF, ..
- platforms are therefore described (modeled) through classic language definition concepts
  - abstract syntax
  - static semantics
  - semantics (mapping to semantic domain)

Model Element Classification

- Two dimensions of classification

**Ontological Classification** (domain types)
- Animal
- Dog
- Collie
- Celebrity
- Movie Star
- Four Legged Object

**Linguistic Classification** (representation form)
- Model Element
- Instance
- Object
Classification Dimensions

Classification Dimensions – Alternative View
**General Platform Model**

- Core language (abstract syntax, constraints etc.)
- Predefined languages instances at multiple logical meta levels
  - Instance level
  - Type level
  - Meta type level
  - ......
- patterns

**Java Platform(s)**

- J2SE Platform
- EJB Platform
.NET Platform(s)

1. .NET Intermediate Language
   - Common Types
   - Console IO
   - Debug

   - W-based model of A
   - Platform W

   - X-based model of A
   - Platform X

   - Y-based model of A
   - Platform Y

   - Z-based model of A
   - Platform Z

Parameterization via Output GPMs
Parameterization via Input and Output GPMs

- W-based model of A
- X-based model of A
- Y-based model of A
- Z-based model of A
- Model of W
- Model of X
- Model of Y
- Model of Z
- Platform W
- Platform X
- Platform Y
- Platform Z

Spectrum of Transformation Tools

- Model Transformation Tools (MTTs) will offer various levels of generality w.r.t. platform models
  - specific (hardwired) with respect to source and target platforms
  - generic (data driven) with respect to source and target platforms
  - all combinations in between

Platform Model richness

- Compiler
- General MTT tool

Platform genericity
Conclusions

- The notion of platforms should be generalized to include any model representation vehicle (not just execution vehicles).
  - Implies that the UML is a platform.
  - All models are specific to exactly one platform and independent (to some degree or other) of all others.
- The accepted wisdom that platforms are best described as DSL’s is too restrictive.
  - Fails to accommodate predefined artifacts and patterns which are found in almost all modern platforms.
- The GSM captures all these aspects in a clean and simple way.
  - Separates the logical and physical classification dimensions.
  - Maintains strictness in each dimension.
  - Foundation for fully general MTTs parameterized by both the source and target platforms as well as by the desired mapping.

References