Types in MDA

EWMDA-2 Workshop

September 8, 2004

People

- Jim Steel
- Ed Willink,
- Andrew Watson
- Laurie Tratt
- Rasmus Fogh
- Girish Maskeri
- Marcus Alanen
- Val Jones

What's the Problem?

- MDA says a lot about languages, but not very much about their type systems
- So:

- What do we want to type?
- What is a type?
- How do we know when one type will do in the place of another?

- Motivating example: How do we know if the output of one transformation is acceptable as input to another?

The Solution?

- Generally, type systems consist of:
- What is a type?
- What is the substitutability relationship between 2 types?

- Further things like type induction

- There is a perceived gap between mathematical type theory and “real” programming languages

What is a type?

- “Domain of interesting instances”
- Extensional versus intensional definition
  - Simplifying/limiting closed-world assumption
- “suitability for some purpose”
- Typing at a structural level – generic approaches
  - Inheritance-based, structural conformance, etc
- Typing based on semantic domains
  - Is one process definition substitable for another

Typing models for transformations

- What is a model? Its a graph of objects, or is there an entrypoint?
- Constraints as types – very general
- Patterns as types – very popular
- Classes as types – very usable
- Additional information:
  - Minimum, maximum numbers of objects
  - Additional constraints
- Also, typing transformation implementations
  - Are they incremental?
  - Are they reversable?
- “No match” versus “something is wrong”

Getting meta

- Mixed type systems
- What type system is appropriate?
- Do we need to reason about target platform type systems?
- Example, in numerical analysis, does the type system of a target platform provide adequate numerical precision, etc?
- Types of type systems? Substitutability
- Can we model type systems using MOF now? If not, how?

Future work

- All of the above!
- Modelling type systems?