



Sue Maurizio and Tullio Vardanega

On the Goodness of Fit for High-Integrity Systems: Reflections on the Design and Evolution of Model-Driven Tools

ECMDA 2008 – 3rd Q2M Workshop:

“From code centric to model centric software engineering: Practices,
Implications and ROI”



Outline

- Why MDA?
- Coping with User Demands
 - Enhancing the Expressive Power: Alternative Choices
 - Example: the NGC Problem
- Product Evolution: Action Semantics and Learning Curve
- Optimisation
 - Pros and Cons
 - The ASSERT Case: Lessons Learned
- Conclusions

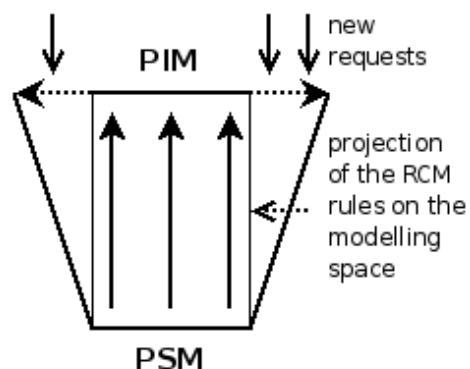
Why MDA?

- A score of aspects must be taken into account in the design of a model-based tool
 - Optimisation of the generated source code and of the resource requirements
 - Learning curve for the tool users
 - Management of increasing user demands
- How should the tool (design and implementation) respond to those issues?

Coping with User Demands

The ASSERT Project:

- A modelling tool for high-integrity applications
- A domain-specific platform: the RCM
- A design space constrained by the RCM



New user requirements led to the expansion of the design space beyond the RCM limits

Enhancing the Expressive Power: Alternative Choices

“Library” support

Solutions are built using the **building blocks** already provided by the modelling tool

- Improved usability
- Improved maintainability
- Correctness proofs are easier to provide

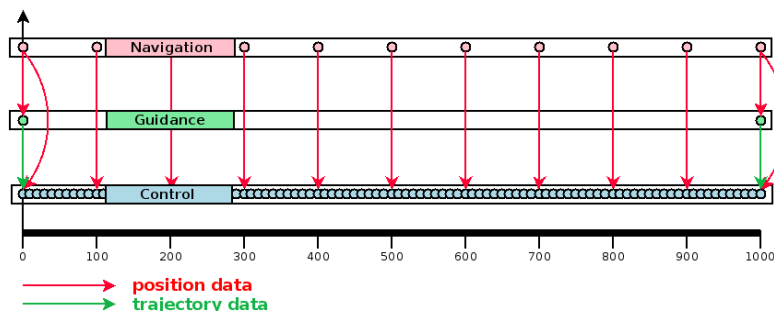
Native support

Solutions are integrated as entirely **new** provided features, leading to changes:

- In the model transformations
- (Possibly) In the metamodel(s)

Example: the NGC Problem

- Automated scheduling of actions
 - To implement data exchanges between three entities
- Minimise the number of tasks involved



Example: the NGC Problem

- The solution to the NGC problem was embedded in the framework as a native feature
- Lessons learned
 - The new feature **did not solve a common problem**
 - No proper “design pattern” → No reuse
 - Other solutions could have better exploited the existing infrastructure
 - **The way** a solution is provided is important

Product Evolution: Action Semantics

- The ASSERT tool allowed to specify
 - The structure of the system
 - Part of its execution semantics
 - Timing, synchronization and concurrent behaviour
 - Deployment and configuration
 - The sequential part of the functional model was to be treated outside
- The code generation was therefore not complete
 - Full understanding of the generated code was required for inserting the functional code (whether manually or by a foreign tool)
 - The lack of information deeply affected the transformation logic → Impact on the tool evolution
- This problem could be solved using an **action language**

Product Evolution: Learning Curve

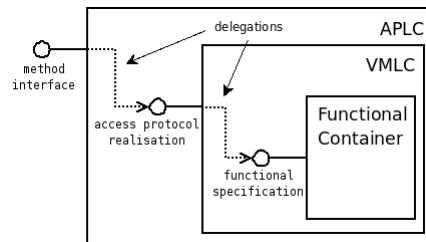
- MDA should ideally allow people to model systems without needing full mastery of the employed technologies
 - A typical application is composed of multiple dimensions, aspects, concerns
 - While user expectations increase, action languages can become very complex
- **Do MDA tools really make life simpler?**
 - Expressiveness and usability must be balanced
 - Balance must be maintained during the **evolution** of the tool

Optimisation: Pros and Cons

The RCM modelling tool was designed for the creation of high-integrity real-time applications → high efficiency is required	The ability to tune the non-functional properties of the system greatly influences performance
Any MDA tool can provide a way to re-generate a given system using a score of technologies	Fast prototyping aids in finding out the best solution, improving efficiency and avoiding contact with the underlying technologies

Optimisation: the ASSERT case

- Some of the model views provided by the RCM toolset
 - Functional
 - Interface (APLC)
 - Concurrency (VMLC)
- What if we simplify the code representation?
 - Is this a convenient vector for tool evolution?



Conclusions

- Expressiveness vs. usability
 - ASSERT → the importance of action semantics
 - Choosing which functionalities must be provided and **how** they must be presented is important
- Tool evolution is a key problem
 - Both in HRT and general-purpose contexts
 - What are the indispensable components of a model-driven tool?
 - How can the evolution be disciplined?
 - How should we facilitate tool optimisation?