Notes about the OOPSLA'04 Workshop on Best Practices for Model Driven Software Development

Ghica van Emde Boas
Bronstee.com, the Netherlands
emdeboas@bronstee.com

ABSTRACT

At the previous workshop (OOPSLA'04) on "Best Practices for Model Driven Software Development", we had a plenary discussion at the end of the session, where the attendee's raised a number of issues concerning Model Driven Software Development (MDSD). This short document presents a selection of these, with some discussion. The purpose of this paper is to record these points and to have a yardstick against which we can measure progress of the MDSD technology and adoption in development projects.

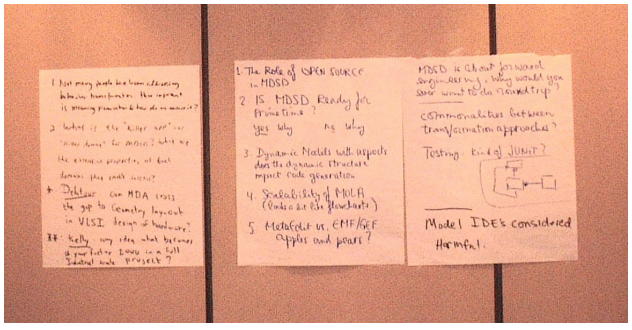
Keywords

Model Driven Application development, Roundtrip Engineering, 3 GL, Generative techniques, Java, UML, XML, XMI.

1.1 Introduction

A major objective of each years OOPSLA workshop on the topic of Model Driven Software Development is to have an open discussion about the state-of-the-art of the technology itself and the best ways of applying the technology in real-life situations.

At past-years workshop the discussion themes centered around the question of how feasible it is to apply MDSD in industrial development projects and what the problems are to do so.



1.2 Is MDSD Ready for Primetime?

The first question to ask is whether MDSD is mature enough to apply in real projects. During the discussion, the following inhibiting points were raised:

- Tool maturity
- NIH syndrome / many "small" approaches.
- Integration
- Good examples
- Is there a killer application?
- Large upfront effort

MDSD cannot be applied without extensive tool support. To-date it has not been possible to develop tools that are general enough to be useful in a large variety of domains and on most popular platforms.

There are a number of open-source and academic projects underway to improve the situation. These tend to lack on the integration issue. There are also several proprietary approaches that may do better on integration, however, this is difficult to verify because they are not open.

1.3 What is the MDSD killer application?

Many new technologies became popular because there was some kind of application that would be significantly easier or faster using the technology. Someone remarked that the killer application would kill somebody and this is not desirable. Here are some points that resulted:

- MDSD is very successful for embedded systems.
- Generation of DB schema, UI etc. is effective using MDSD.
- MDSD is useful if repetition is involved.
- MDSD shields against rapid platform evolution.

- The killer app = the app that kills somebody.

From this list it is clear that the areas of application for MDSD are still scattered and patchy. Lack of integration of tools, as raised in the previous section, prevents taking an integrated development approach.

1.4 Model Roundtrip-Engineering

The usefulness and feasibility of roundtrip-engineering was a hotly debated issue. A majority believed that roundtrip-engineering is not useful. Forward and backward engineering both have a purpose, but tooling is critical.

- MDSD is about forward engineering mostly
- Backward engineering is useful for:
    - Capturing legacy code into a model
    - Exploring the solution space
    - Multiple horizontal model aspects
    - Analysis feedback
- Compare going from Assembler to 3 GL
    - Grace period when using both.
    - Model to 3GL gap is much wider
    - Semantics of a model may be fuzzy.

1.5 MDSD and Open Source

The problems of dealing with open-source do not seem to be different for MDSD than for other development.

- Interoperability is more important than open source
- How to do mixed mode development
- Standards
- Licensing issues

Licensing deserves special attention though. Just as with a compiler, the compiler may be open source, but the source compiled with it, may be proprietary.

In MDSD, there can be many mixes of models, hand crafted-source, pieces of framework, model transformations, each of which could have a different kind of license.

1.6 Open Questions

Here follows a list of miscellaneous subjects that were raised.

- Dynamic models with aspects: Does the dynamic structure impact code generation?
- Behavioral transformation has hardly been addressed. How important is meaning preservation and how do we ensure it?
- What are the essential properties of domains that enable success?
- Commonalities between transformation approaches?
- Testing? The JUnit for model transformation?
- Model IDE's?
- Debugging of models and transformed models?

1.7 Conclusion

While we were still talking about questions like: "is MDSD ready for industrial use", and "how can we find good examples to prove our cause" a year ago, it seems a bit early to talk about *best practices*, because practice is scattered.

It would be nice to go over the same questions again now, a year later, and see how much progress is made.

Another consideration to be made is an assessment of how much the world has changed. Do we still envisage application development using MDSD in the same way as we did a year ago? What is the relationship now to the use of DSL's (Domain Specific Languages) and Software Factories?

☺

☺ Photo's by Peter van Emde Boas