EMWDA Working Group on ‘MDA is the Wrong Answer?

Discussion Report

Editor: Jos Warmer

People Attending

Oliver Sims, Andrew Watson, Jos Warmer,Tracy Gardner, Dave Pilfold, Tony Mallia, Ian de Beer, Marcus Alanen, Nelly Bencomo, Lea Kutvonen.

Introduction

Starting with “MDA is the wrong answer”, we came up with a list of problems that the participants want (or expect) to solve using MDA, and why MDa will be able to achieve this. At the same time a list was made up of perceived shortcomings of MDA. This contains arguments that the participants often encounter during their discussions on MDA with other people.

What problems do we want to solve

We came up with the following problems that the participants want (or expect) to solve using MDA: making software development less tedious, get a higher quality, documentation and application generation, solving the lack of enough skilled programmers, making less IT projects fail, structurally survive technology changes, provide better support for collaboration and integration, building better product lines, maintaining relationship between different domains, and avoiding to get into the same problem again and again.
This is quite am impressive list, suggesting that MDA has to be a silver bullet after all, if it can really solve all of these problems. The group discussed how and why MDA would be helpful in solving such a wide variation of problems.

Why would MDA solve these problems?

A crucial characteristic of MDA is that transformation specifications explicitly define the relationships between many of the artefacts that are produced during a software development project. Knowing this relationship and having tools to validate them and/or resolving conflicts between them allow for better (i.e. better quality, better productivity, better documentation, etc.) product lines. For the same reason, MDA can be the glue between all of the different issues and views within a software project.
Working from the modeling level, MDA allows people to work at a higher, technology independent, abstraction level. This provides good support for dealing with the quick technology changes that are typical for the IT world.
Current measurements are not too many, but they do indicate that savings of 40% over the complete software development life cycle can be achieved using MDA. If this is the case for MDA in its current, rather immature state, then the saving might eventually be much bigger.

Reasons why MDA would not work

Many of the participant have encountered scepticism about MDA. We have made a list of typical reasons why people don’t want to use MDA. The participants do not agree with these, but it is important to know the arguments and be able to counter argument them.

No appropriate languages. The group agreed that UML has many shortcoming. It is both too complicated, and too low level.

A solution can be found by using UML profiles to define higher level languages, or by defining new and higher level modeling languages.

Tools. There is a lack of tools and the big vendors are not offering MDA tools yet. This situation is remedied by Microsoft, who is actively building support for model driven development in their Visual Studio 2005 product. Although they do not use the term MDA (copyrighted by the OMG) they support many of the major ideas behind it.

The MDA community, needs to develop more and better tool support. The group sees both a place for open source tools, for easy experimentation and having a low threshold, and for vendor tools, to get a credible and well-supported market place. There was a slight fear in the group that vendors would try to come up with all types of proprietary model, thus making defeating the goals of MDA.

No incentive. For various reasons people simply have no incentive to use MDA. This can be a lack of the will to change and fear of losing your jobs. From a management perspective software development often isn't their major concern, day to day operations is. Therefore MDA does not seem to be relevant.

The group felt that, given the lack of enough gifted programmers, MDA will not cost jobs. It will merely change the type of work that people do. Also, when new possibilities arise, business always has an increasing demand. Therefore lose of jobs seems unlikely.

It didn't work before. People often see MDA as being CASE in different words. CASE and e.g. Shlear-Mellor failed to deliver their promises, why would MDA do any better?

An answer to this is that MDA is much more than just code generation that CASE offered. MDA offers the infrastructure such as a standardized MOF, standardized metamodels, standardized transformation languages (QVT) etc. that were lacking in the CASE era. This allows developers to look at all pieces and enables them to easily change whatever they want.

MDA is not applicable for my domain. People talk about certain domains, especially GUI, and decide MDA can never work for these.

There are already MDA tools out that do a good job in the GUI area. This type of argument can only be countered by giving concrete examples.

MDA is too vague. An often heard complaint is that MDA is too vague. Everyone and every tool is free to call itself MDA. It become unclear what MDA really is.

This is a very true complaint. However, the OMG has just started an initiative to clarify the meaning of MDA much better, including a list of requirements that a tool need to fulfil to allow it to be called an MDA tool.

Conclusion

The first and main aspect that needs work for MDA to work as promised are better tools. There was consensus in the group that MDA without tools will never work. Of course, the tools can only be made to work effectively is we also have available UML profiles, higher level modeling languages, meta-models, models of architectures, QVT standards, etc.

The group agreed that MDA is still in its early years and that we need time to make it all work as promised. Even after hearing all the reasons why it would not work, we are still convinced that MDA holds much promise and will certainly mean an important advance the IT world.