What’s needed to make ‘Model-Driven’ Mainstream?

CODE Generation 2013
April 10-12
Cambridge, UK

Jürgen Mutschall
Distributed Engineering Systems Software GmbH
Jürgen Mutschall

... designs and develops model driven software tools and visual programming languages for more than 20 years

1989  Designer of development tools for the visual programming of power plant automation systems for ABB.

1991  Founder of a consulting company for object-oriented software development

1996  Founder and CEO of Distributed Engineering Systems Software GmbH, Mannheim; distributed content management solutions

1999  Co-Founder of Merlin Software Engineering GmbH, Baden-Baden; a company dedicated to the development of a model-driven engineering tools, discontinued in 2005

Since  Involved in projects based on the ideas of Model Driven Software Development,
2004  active model transformation and incremental code generation.

Took part in multiple large scale international projects as project manager, architect and consultant. (Business areas Process Automation and Banking).
Some Fundamental Assumptions for Model Driven Software Development

- Models are considered equal to code (not the same!).
- Models are (semi)automatically transformed to code.
- A domain model describes the conceptual fabric of a problem domain.
- An accurate domain model ensures the agreement of all stakeholders about the scope and meaning of the concepts in the problem domain.
- The domain model should cover all layers of abstraction (separation of concern).
- Descriptive models are used for analysis and discussion.
- Prescriptive models are more formal and are used to (automatically) construct a target system.
Model-Driven Technologies and Business Awareness

One thing all participants of last year’s final panel discussion agreed on: "Model Driven is not mainstream and the business owners and the organizations are still not aware of the potential benefits".

- The MD Progress is slow.
- People are happy to create DSLs in isolation, ignoring the generic capability of standard modelling languages like UML.
- ‘Model-Driven’ seems to be a playground for programmers only. The MD community is primarily technical driven.
- Commercially the adoption of new MD tools is harder than 5 years before.
- The envisioned advances for the organizations are rarely accomplished.
- The existing (technical) MD eco-system is not enough for business adoption.
- MD technologies have to become a commodity instead of a risk for business.
- Organizational adoption (increasing productivity and quality) of MD is a task for managers and politicians.

Andrew Watson, Wim Bast, Steven Kelly, Darius Silingas and Markus Völter; panel discussion at Code Generation 2012
The (Slow) Evolution of Model-Driven Technologies
From Programming to Modelling and Back …

- ER-Diagrams (1976)
- CASE Tools (1982)
- UML (1997)
- EMF (2001)
- MDA (2002)
- Language Workbenches (2005)
- BPMN (2007)
- DSLs (2008)
- ???
Architecture Diagrams --
The Pictorial Use of Models

Graphical Modelling as an art of Communication and Documentation

- Descriptive Models
  - sometimes called “System Diagrams” or “System Architecture”
- UML-/BPMN-Tool as a drawing tool or Microsoft Visio / PowerPoint
- Often used for business specification and documentation
- Only a small part of UML is used, usually a transcription of simple box and line diagrams.
- No differentiation of classes, objects, components or systems.
- The models do not pretend to be formal, complete or consistent.
- The construction of the software solution is only loosely coupled to the models.
- On system changes the models become outdated and are usually not synchronized with the models.
Business and Data Modelling --
The Restricted Use of Models

More simple models for the business users. The promise of easy to use tools and more consistent descriptions of business requirements.

- Self-restricting to some aspects of modelling
  - e.g. persistence or workflow
- Good tool and platform support by major vendors
  - e.g. IBM, CA, Oracle
  - Often Vendor Lock-in
- Modelling as a kind of configuration
- Often used in combination with 4GL and script languages.
- “Good Enough”-approach with the good separation of concern but with a focus on system construction and not on conceptual domain modelling.
- Huge projects often uses a “tool zoo” for forward engineering.
Domain Specific Languages -- 
The Programmatic Use of Models

A “workaround” approach for developers

- Most software developers don’t want to model.
- The vision of Programming, that the distinction between modelling and programming vanishes is (still) not accomplished.
- It seems to be much more productive, and often less work, to build DSLs with “real” language engineering environments, as opposed to using UML profiles or graphical workbenches.
- The advances in building projectional/structural editors for DSLs and render an integrated, interactive visualization for different surface syntaxes (e.g. including tables) improves the modelling experience and makes it more like programming.
Hypothesis 1: Standards, especially UML, are established, but not well accepted in the software developers community

- Developers do not like to model.
- Developers like programmatic modelling approaches
  - e.g. EMF/Ecore, language tools
- Software architects, designers and programmers prefer informal box-line diagrams.
- They appreciate the freedom of “artistic interpretation”.
- In most legacy projects the programmers document the final code by UML diagrams (and powerpoint slides) and technical system manuals.
- Developers adopt MD technologies only halfhearted.

See surveys about acceptance and use of Model-Driven technologies (2010/2012)
Hypothesis 2: Usually there is no knowledge about Model-Driven technologies in big budget projects available

- Only a few team members of a software project have (potentially) MD know-how.
- A typical legacy project (e.g. JEE) resources breakdown:
  - 10% project management
  - 10% business/product management
  - 10% business specification
  - 20% hardware, infrastructure, security and database management
  - 20% test definition, management and execution
  - 10% off-shore / team coordination
  - 5% software design / technical specification
  - 15% software development

- Today’s technical challenges are handled with a lot of money and methodologies of yesterday.
- Maybe there is NO need for Model-Driven technologies.
Hypothesis 3: Business management and the organization have to initiate the adoption of MD technologies

- Money talks.
- When money becomes tight the management searches for approaches to increase productivity and to reduce costs.
- Drivers and opportunities:
  - The competition is cheaper.
  - The competition is faster.
  - Quality assurance / testing becomes more and more expensive.
  - Every next-gen product has to be developed from scratch and with a bigger budget than it’s predecessor.
- Projects: MD technologies as a coup
  - Buy consultancy
- Organizations: Build MD know-how and skills
  - Recruit and employ permanently MD experts and developers
Hypothesis 4: The adoption of MD technologies can only start at the grass-roots level.

- Detailed technical know-how is needed to understand MD technologies. So only the MD expert (developer) can initiate the adoption of new technologies and tools.
- Using MD technologies is nothing for the faint-hearted. It takes a long time to introduce a new kind of thinking.
  - New languages and tools
  - LoC are no benchmark anymore
- Levels of abstraction
  - Code generation
  - System configuration
  - Entity / persistence / service modelling
  - Business workflow modelling
  - Domain Modelling
  - DSL Modelling
  - Software factories / Feature modelling
  - Model tool modelling
Recommendations to MD experts

- Convince your (developer) colleagues
  - “Modelling is good for you!”
  - Every manual written line of code is bad!

- Help to ... leverage research results and use standards
  - Standards are good! Do not reinvent the wheel again and again
    - UML, QVT, fUML, ALF, BMNP, IFML, ...
  - New technologies
    - Model interpretation / execution / debugging
    - Bidirectional synchronization (e.g. Triple Graph Grammars)
    - Scannerless parsers, structured editors and incremental generator technologies

- Help to ... unleash the unused computing power for modelling
  - 200 times more computing power since Eclipse 1.0 (2001)
  - 100 times more memory for in-memory transformations since JDT 1.0
  - Model refactoring
  - No slow batch generator tools anymore
  - Incremental generation, background processes
  - Seamless IDE integration
  - Synchronization of visualization, model and code

- Build ... the next-gen tools for MDSD
Be .. an Evangelist for Model-Driven Technologies

- Think “Model First”
  - Programming fills the gaps
- Go for “Agile Modelling”
  - Allow model changes at any time
  - Keep your (business) model in sync with your code
- Search for the best-fitting tools
- Learn and teach methodologies, technology and standards.
- Think in the customer time scales … 5-10 years
- Help the customer to think in domain models and not in low level requirements
- See the big picture
  - Complete software product lifecycle
  - Product families
  - Software factories
References and Links

- Surveys about acceptance and use of model-driven technologies
  - Adrian Kuhn et. Al: An Exploratory Study of Forces and Frictions affecting Large-Scale Model-Driven Development (2012)

- Further information about DSL Engineering
  - [http://www.voelter.de](http://www.voelter.de), [http://mbeddr.wordpress.com](http://mbeddr.wordpress.com)

- Images and illustrations
  - [http://de.fotolia.com](http://de.fotolia.com), [http://commons.wikimedia.org](http://commons.wikimedia.org)
CONTACT INFO

http://www.desys.com

Distributed Engineering Systems
email: juergen.mutschall@desys.com
phone: +49-171-5749200