

Benefits

Technical and Economical



www.mdsd-buch.de



www.mdsd-book.org

Markus Völter

voelter@acm.org

www.voelter.de



Reasons for using MDSB

- You want to provide a way for your **domain-experts to formally specify their knowledge**, and to provide a way for your technology people to define how this is implemented (using model transformations).
- You might want to provide **different implementations** (i.e. more concrete models) **for the same model**, perhaps because you want to run it on different platforms (.NET, Java, CORBA).
- You may want to **capture knowledge** about the domain, the technology, and their mapping in a clear, **uncluttered** format.
- In general, you **don't want to bother with implementation details** when specifying your functionality.
- MDSB results in a **fan-out**, i.e. one set of models can be the source for transformations to several targets.
- Another reason for using MDSB: You are working in the **context of product lines and software system families** and need to develop domain-specific assets.



MDSB Benefits I

- **Models** are free of implementation artifacts – they directly **represent domain knowledge** and are thus reusable.
- Implementations for **various platforms** can be generated in principle – the technology change problem is addressed to some extent.
- **Technology freaks and domain experts** can take care of „their business“ (transformations and models, respectively) and need to care of each other's problems only in a limited way.
- **Domain experts can play a direct role in development** since they can more easily understand models expressed with a DSL as opposed to implementation code.
→ Domain Experts play the central role they deserve!



MDSB Benefits II

- **Development becomes more efficient** since repetitive implementation code can be generated automatically.
- **Architectural** constraints/rules/patterns can more easily be **enforced**, since they are embedded in the templates rather than just being documented (and ignored).

This is especially important in really large teams, often in the context of Product-Line Engineering and Software System Families.
- Transformer/Generator can address **cross-cutting concerns** (just like an aspect weaver)



MDSB Predjudices

- MDSB **does not require UML** – any kind of modelling language is ok, graphical or textual
- Precise and complete models...
 - ... **are not the the same as „visualized code“** – the abstraction level is higher
 - ... **are not the same as analysis models** – analysis models are not computational
- **MDSB does not require – not even recommend – a waterfall.** Development of the generative infrastructure is iterative and incremental.
- **You do not need big and expensive tools** – a lot of small and useful open source tools are available.
- **You don't need to generate 100% of the code** – it is ok, to also code some aspects in a 3GL.



Benefits for Software Quality

- MDSB requires an **explicit, well-defined architecture**. Defining an architecture this way improves the quality of the system (independent of whether it is generated or not).
- **Transformations capture expert knowledge**. The generated code reflects this expert knowledge uniformly.
- An Domain Architecture defines a **strict programming model** for the manually developed parts – again, uniformity and constrained-ness improves quality.
- **Generator does not produce accidental errors** – either things are always right or always wrong. This makes finding errors easier!
- Models can serve as an **up-to-date and always current documentation**.

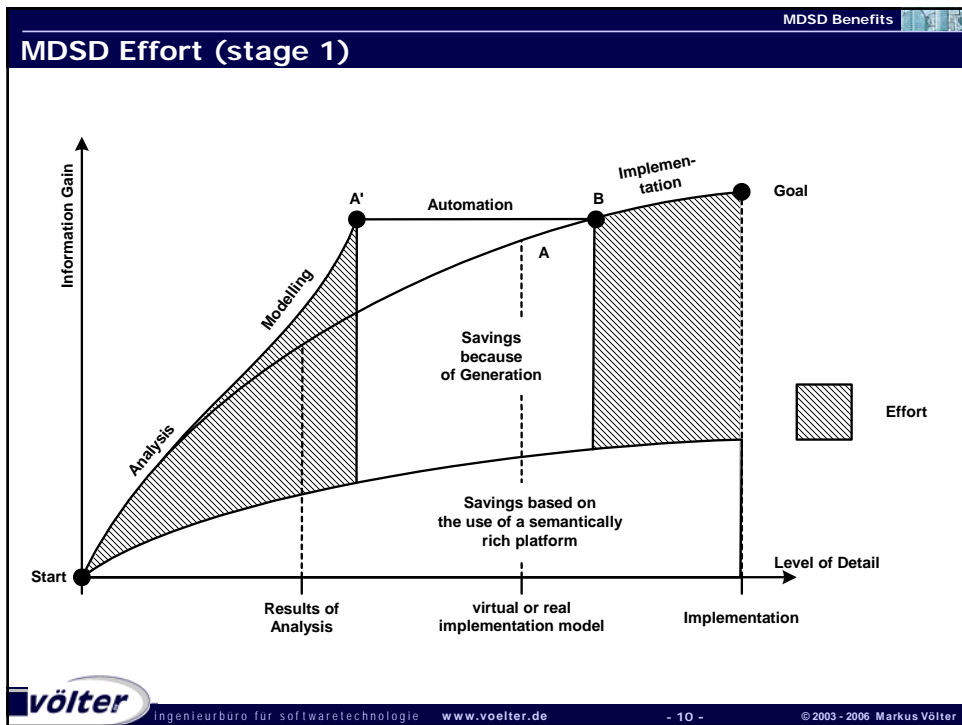
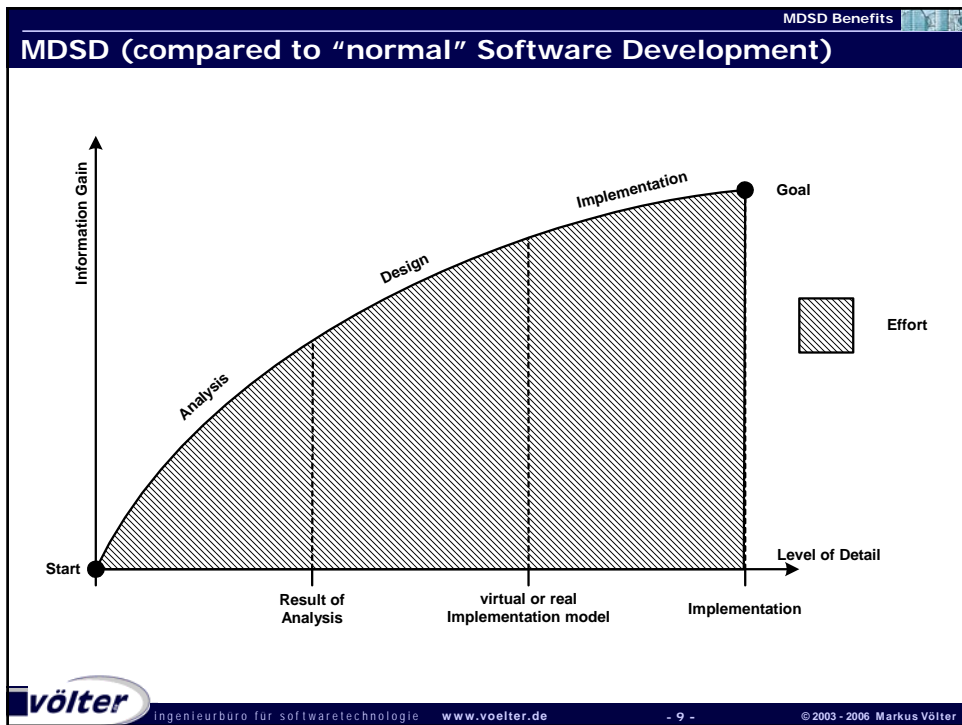


Benefits for Software Quality II

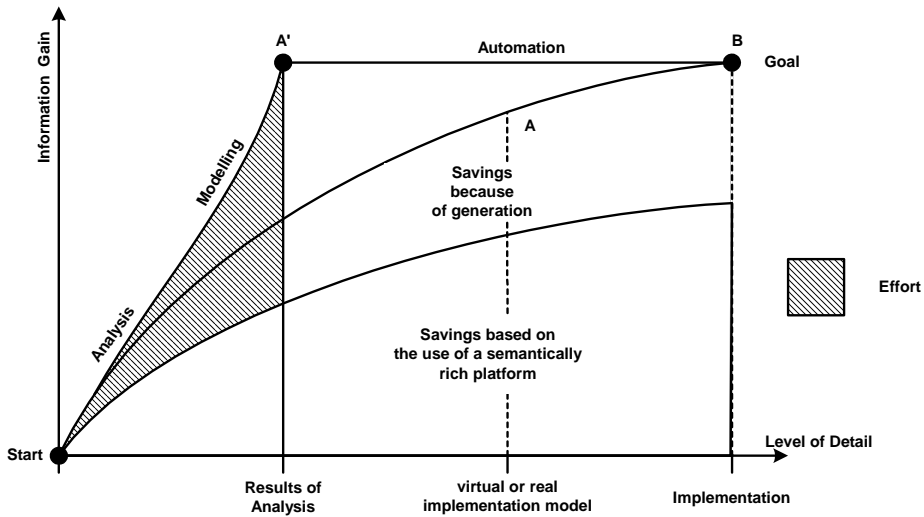
- In general, **MDSB forces you** to take care of many good things, which you'd like to have in any application development project:
 - Domain/Application Scoping
 - Variability Management
 - Well-Defined Software Architecture, Architecture Metamodelling
 - Trying to build a generator for a domain/target architecture enables your understanding of the domain/target architecture. This in itself is a huge benefit.

No Free Lunch - Challenges

- You need **additional skills** in your team (domain analysis , metamodelling, generator development, architecture)
 - You need a couple of good people who usually aren't easy to get, and who aren't cheap.
- Your **development process** needs to reflect the two-track nature of MDSB (domain architecture development, application development)
- Sometimes tools require some „creative use“ and improvisation (→ use open source!)
- Remember: a fool with a ~~tool~~^{generator} is still a fool

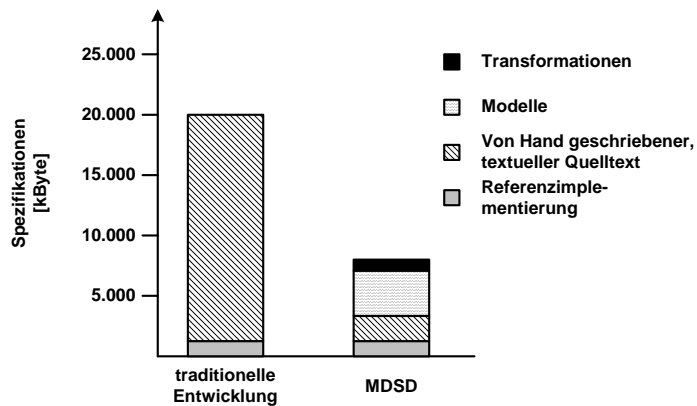


MDSB Effort (stage 2)

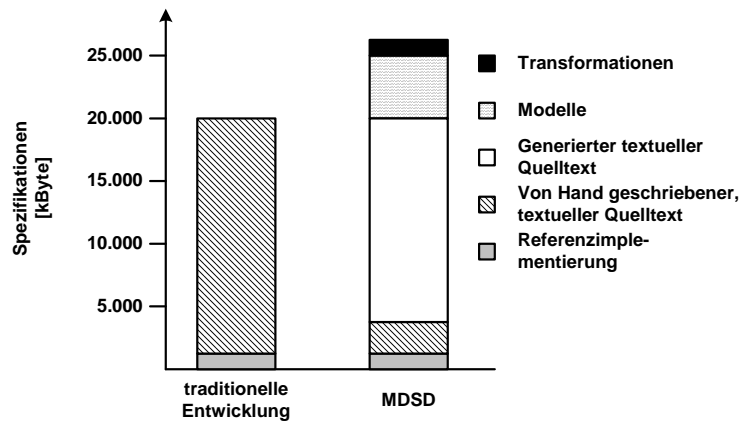


Numbers

- The following is a **specific example**, of how MDSB can reduce the amount of work in a project.
 - Other projects may have other numbers...



Numbers II



Numbers III

Menge des Quellcodes [kBytes]	Traditionelle Entwicklung	MDSB
Quellcode Referenzimpl.	1.000	1.000
Von Hand geschr. Code	19.114	2.206
Modelle	-	3.481
Transformationen	-	244
Gesamt	20.114	6.931

Entwicklungsaufwand [%]	Traditionelle Entwicklung	MDSB
Transformationen	-	4 (0,15 * 25)
Referenzimplementierung	15	15
Applikationsmodelle und -Code	85	41 (0,48 * 85)
Gesamt	100	60

