Model-Driven Software Development: Does anyone really use it?

Jon Whittle
joint work with John Hutchinson, Mark Rouncefield

School of Computing & Communications
Lancaster University, UK
j.n.whittle@lancaster.ac.uk
2001 AD
I'm sorry, Dave. I'm afraid I can't do that.
model-driven architecture

aka.
model-driven development
model-driven engineering
model-based software development
model-based design
model-integrated computing
domain-specific modeling
model-based testing

A Typical Tester
model-driven maintenance

A Typical Maintainer
model-driven integration

A Typical Integrator
agile modeling
model-driven development with Modelica

A Typical Modelica-er?
maintainability
portability
interoperability
all is well with the world
except for…

those darned naysayers
2013 AD
who was right?
talk outline

• Project EAMDE
  – Social/organizational/technical factors

• Discoveries
  – Some observations
  – Some lessons
  – Some tips
Project EAMDE

- Widely-distributed questionnaire on how MDD is used
  - 35 questions pertaining to MDD application
  - 449 responses
- In-depth interviews with MDD practitioners
  - 22 interviews; 17 different companies
  - 150,000 words of transcribed data
  - >360 years of cumulative experience
- On-site ethnographic studies (ongoing)
  - 2 done (by Steinar Kristoffersen); more planned
What EAMDE is **not**

- an attempt to quantify the penetration of MDD in industry
- a study on UML
  - deliberately broad view of MD*
- an attempt to evangelise or promote MDD
  - interested in failure as much as success
Diversity
Examples of MDD
Examples of MDD

“the broader the domain you try and cover, the less the productivity increase… with DSM, you’re not looking at building a modeling language for embedded applications … you’re not looking at building one for mobile applications… mobile embedded, or even mobile phones or even Brand X mobile phones, but a particular product family of Brand X mobile phones…”
First discovery: a lot of MDD success is hidden
Which modeling languages do you use? (tick all that apply)
DSLs favored over general-purpose modeling

- mostly, companies write their own code generators for very specific tasks
  - In contrast, companies often ditch commercial tools because they cannot modify them the way they want or because they “don’t do everything”
  - Multiple references to the fact that off-the-shelf tools could have killed an MDD effort
- Generation of whole systems is not widespread
Second discovery: code generation is a red herring
“I guess at the end of the day, this dream of code generation from models doesn’t exist – I mean everything ends up being done by hand because either we don’t trust code generators or they just don’t generate the code we need… it’s actually impossible to get in non-functional requirements into code generators – it’s too difficult”
Offsetting gains without realising it

• “sometimes the code generated makes it necessary to use a larger CPU which costs more money than the efficient code of an experienced programmer”

• 8x more expensive to certify generated code
Don’t obsess about productivity

- 65-100% code generated
- Figures on productivity gains differ
  - 20-800% gain
  - 27% loss
Third discovery: the real benefits of MDD are holistic
So if not productivity, then what?

- The **real** benefits of MDD are quality, architecture, reuse

- “whenever you name any single advantage...you can always achieve the same advantage with another approach...”
  - “it tends to be that a model-driven approach is more likely to have a well articulated design and architecture”
Example

- An organisation finds itself developing lots of little (modeling) languages over time:
  - “we were generating 70% of the system off these little XML languages… we would try to separate out the pieces that were generateable and the pieces that weren’t… it motivated us to have better separation of concerns”
Fourth discovery: MDD must enable new things, not just speed up old things
Doing things faster and better is not enough

• If the status quo works (or is perceived to work), there will be insufficient buy-in to change
  – MDD should be sold not based on how it can do things (slightly) better, but in terms of how it can fix things that are broken
  – “software is no longer a bottleneck”
People/Organizations
The Psychology of MDD
Architects love MDD
The code guru hates it
as does the hobbyist developer
It’s bad news for offshoring
Middle-managers are usually the bottleneck
The MDD guru is likely to be a developer and domain expert
MDD works best in companies that are not in the software business.
MDD Works Best in Non-Software Companies

- domain experts already model
  - “they already have an established way to design in Powerpoint”
  - “I think they are more open than a company that has very long years of experience in software development”
What would 449 MDD practitioners say?
The “Community” of Respondents
Experience with MDD

• Initial exploration: 10%
• Prototyping: 10%
• First major project: 18%
• ~ 1/3: extensive experience of MDD on many projects and/or over many years
Use of multiple languages

- 62% of those using custom DSLs also use UML
- UML is the most popular ‘single use’ language
  - 38% of all respondents
- UML used in combination with just about every combination of modeling languages
  - 14% of UML users combine with vendor DSL
  - 6% with both custom and vendor DSL
Unraveling points of contention related to MDD
Benefits of code generation

Q: Is your use of code generation an important aspect of your MDD productivity gains?

75% agree, 10% disagree

Q: Is integrating generated code into your existing projects a significant problem?

36% agree, 40% disagree
Changes on model or code?

Q: Do you mainly make updates on the model rather than the code?

70% agree, 15% disagree

Q: Do you spend a lot of time synchronizing the model and the code?

35% agree, 45% disagree
MDD and Flexibility

Q: Does MDD make you faster at implementing new requirements?

75% agree, 12% disagree

Q: Does MDD prevent you from responding to business opportunities?

32% agree, 38% disagree
Effect of MDD on Training Costs

Q: Does using MDD allow you to employ developers with less software engineering experience?
46% agree, 34% disagree

Q: Does using MDD require you to carry out significant extra training in modeling?
74% agree, 9% disagree
Complexity of UML

Q: Is UML too complex?
44% agree, 32% disagree

Q: Is UML powerful enough?
52% agree, 31% disagree
Tooling

Q: Are MDE tools too expensive?  
45% agree

Q: Do organizations attempt to deploy MDE using inappropriate and/or cheap tools?  
55% agree
Tools used


Over 100 tools
Education, education, education
• Are we teaching modeling the wrong way?
  – bottom-up versus top-down
  – combine teaching modeling & compilers/code optimization
Top 10 Tips for Practitioners
1

Keep Domains Tight and Narrow
Target well-documented, well-understood domains
3

Put MDD on the Critical Path
MDD works best when driven from the ground-up
5

Be Careful About Gains Offset Elsewhere
Don’t Obsess About Code Generation
Not everyone can think abstractly
Most projects fail at scale-up
Match tools/processes to the way people think; not the other way around
OK, there were only 9…
If you do all this, you might just find that MDD is your holy grail after all.
Bedtime reading…

- Model Driven Engineering Practices in Industry, ICSE 2011
- Empirical Assessment of MDE in Industry, ICSE 2011
- A Survey of Practitioners’ Use of MDE (ask me for a copy)
Dataset

http://www.comp.lancs.ac.uk/~eamde/site/content_qres.php
What Next?

Actively partnering with companies
Interviews
Focus groups
Observational studies