LANCASTER UNIVERSITY

Software Modeling and Reuse: the good, the bad, and the ugly

Jon Whittle

joint work with John Hutchinson, Mark Rouncefield

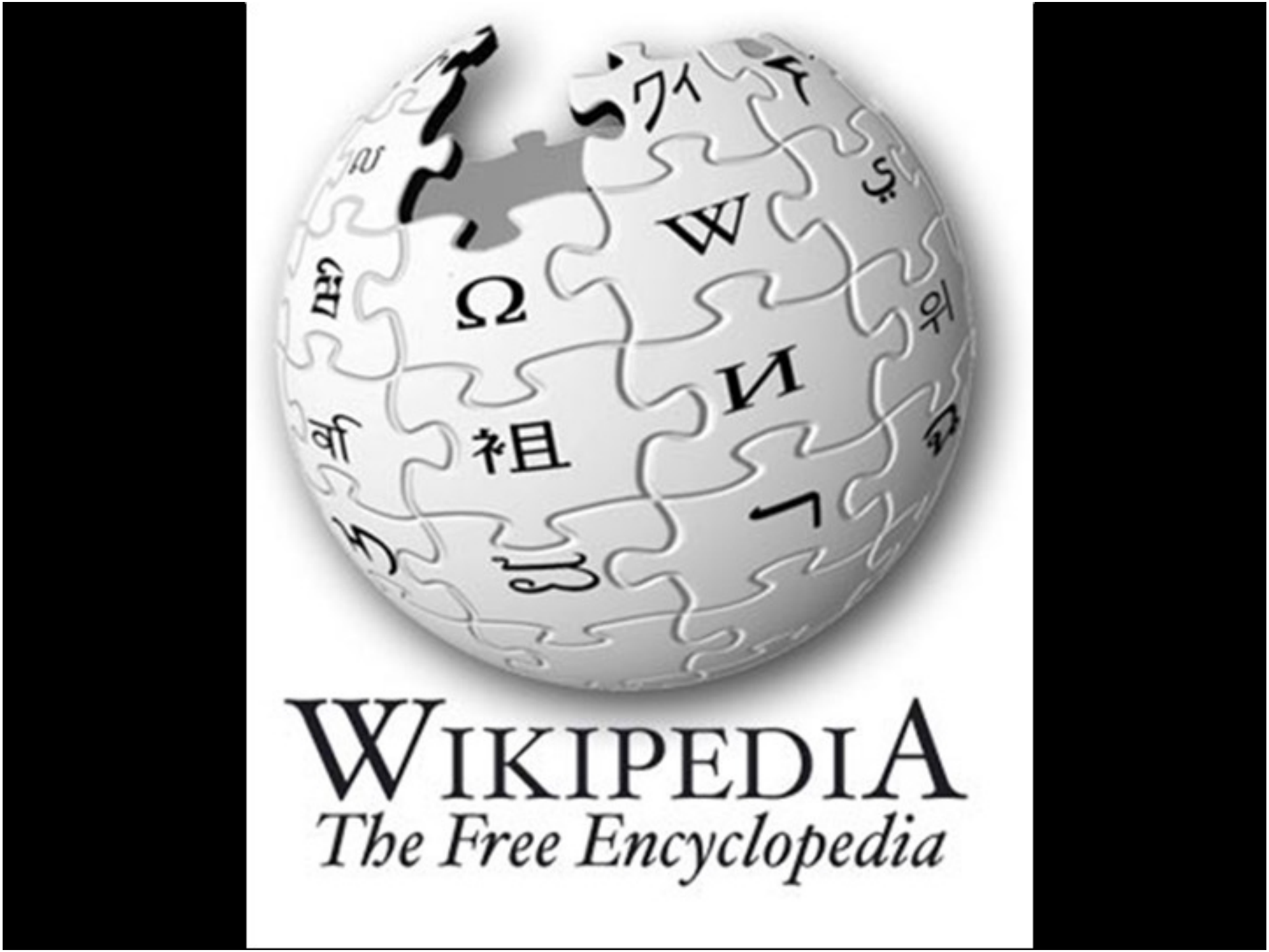
School of Computing & Communications

Lancaster University, UK

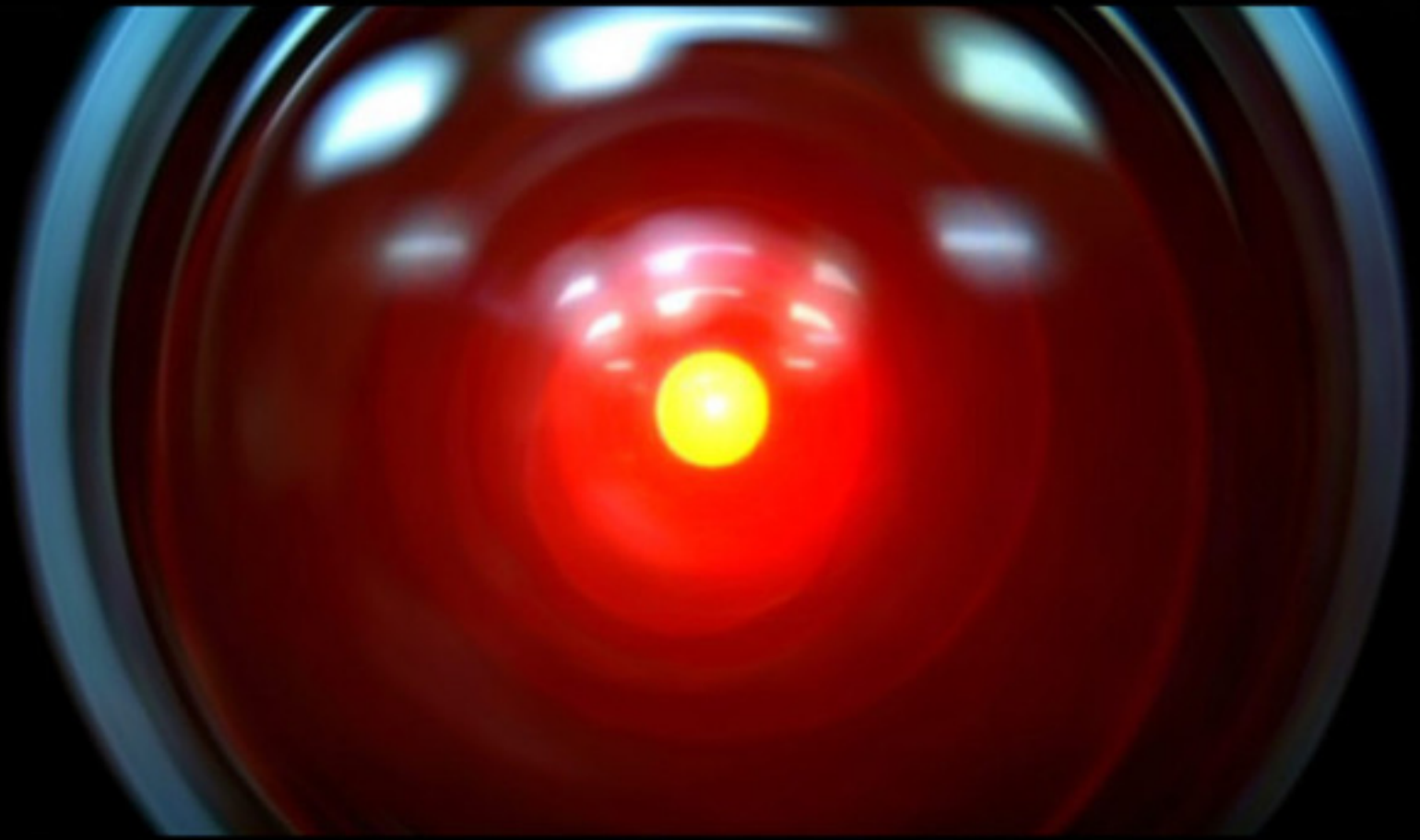
whittle@comp.lancs.ac.uk

InfoLab21

2001 AD


WIKIPEDIA
The Free Encyclopedia

I'm sorry, Dave. I'm afraid I can't do that.

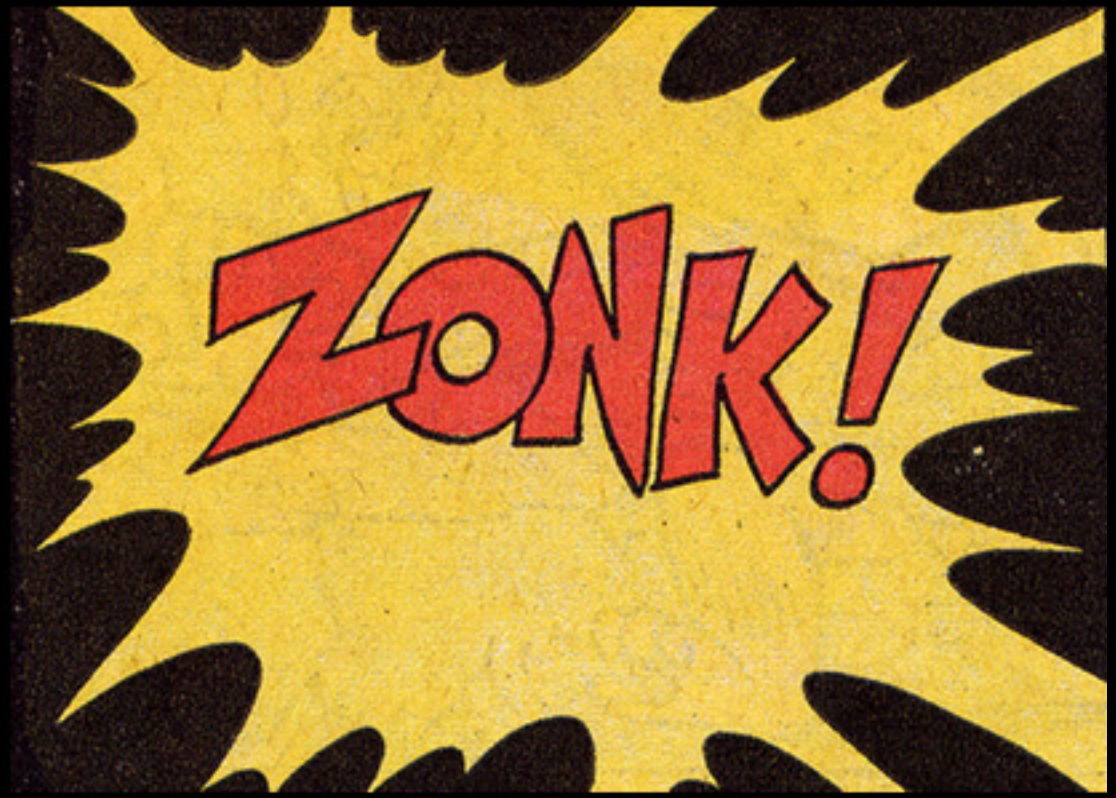


model-driven architecture

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

productivity

maintainability



portability

interoperability

all is well with the world

except for…

those darned naysayers

2011 AD

who was right?

talk outline

- Project EAMDE
- Discoveries
  - Modeling Practice
  - Reuse/DSLs/Architecture

Project EAMDE

LANCASTER UNIVERSITY

- 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

InfoLab21

What EAMDE is **not**

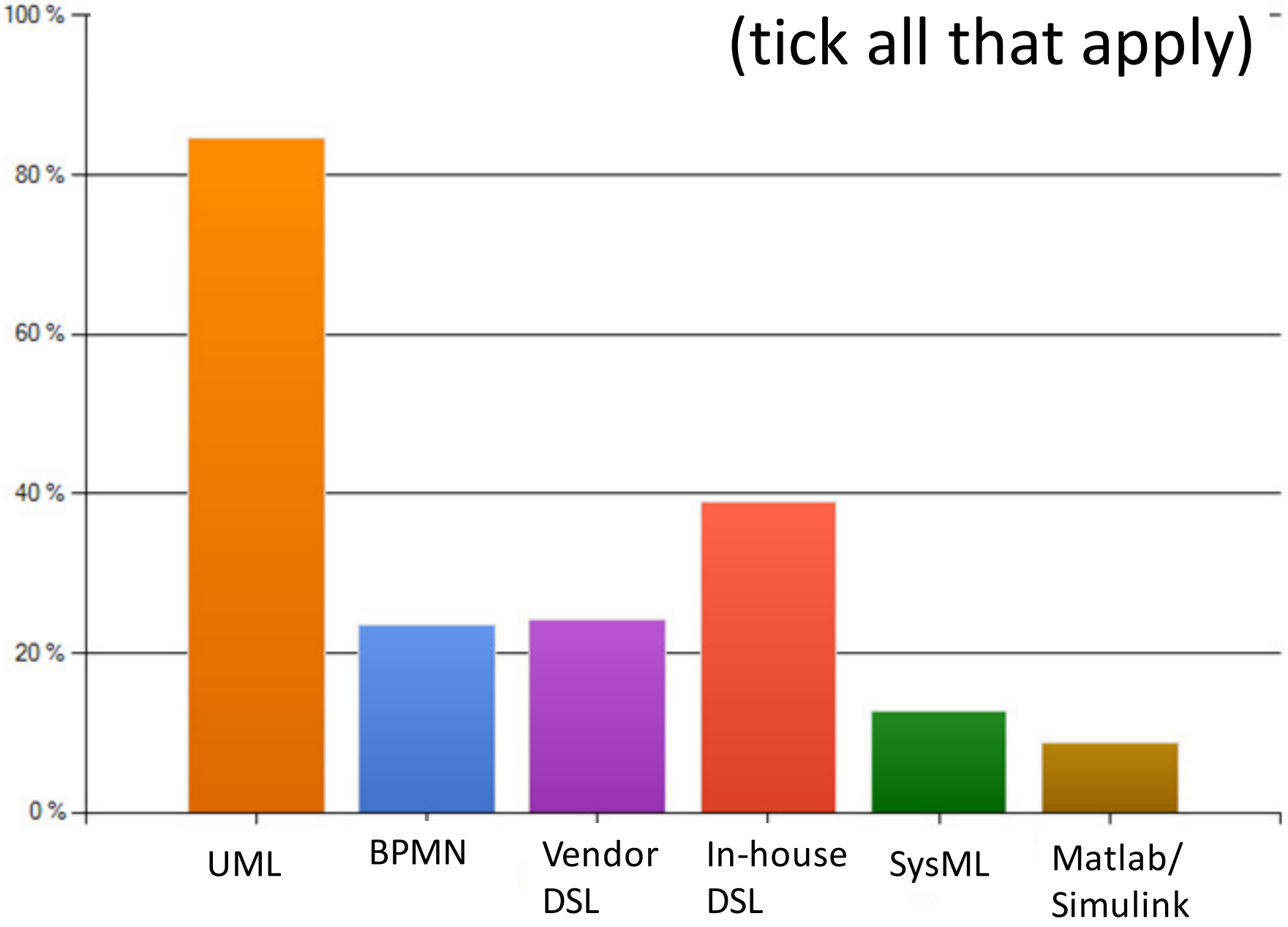
LANCASTER UNIVERSITY

- 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

InfoLab21


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

LANCASTER UNIVERSITY

- 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

Code generation

LANCASTER UNIVERSITY

"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

LANCASTER UNIVERSITY

- *"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

InfoLab21

Don’t obsess about productivity

LANCASTER UNIVERSITY

- 65-100% code generated
- Figures on productivity gains differ
  - 20-800% gain
  - 27% loss

InfoLab21


Third discovery:
the real benefits of
MDD are holistic

So if not productivity, then what?

LANCASTER UNIVERSITY

- 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"*

InfoLab21

Example

LANCASTER UNIVERSITY

- 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”*

LANCASTER UNIVERSITY

Fourth discovery:
MDD must enable new things, not just speed up old things

Doing things faster and better is not enough

LANCASTER UNIVERSITY

- 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"*

LANCASTER UNIVERSITY

People

InfoLab21

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

LANCASTER UNIVERSITY

Organisation

InfoLab21

Current Business Practice Doesn’t Support MDD

Business Practice Doesn't Support MDD

LANCASTER UNIVERSITY

- *"lead architects need to understand that just because their models are simple, they weren't going to be put out of a job…"*
- Developers are defined by their expertise for a technology not a domain; so it is not in their interests to innovate

MDD works best in companies that are not in the software business

MDD Works Best in Non-Software Companies

LANCASTER UNIVERSITY

- 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"*

InfoLab21

The Good, The Bad, and The Ugly

The Good


LC900M

“we put it directly in the line of the product”

“we are not allowed to fail”

“the benefit was not only because we introduced model-driven design”

“we also started a re-use group”

“if you didn’t introduce model-driven design, the reuse initiative would have failed”

“at this moment, embedded software is not bottleneck in any project”

“56% of all our code is from re-used building blocks, but in the beginning it was only 5 or 10%”

“normally, decisions are made as low in the organisation as possible

The Good

- critical path
- non-software context
- real business driver
- MDD an enabler for reuse

The Bad-Ass

“the same electrical design in a full size truck as in a Cadillac”

“somebody would be writing the spec”

“they were actually outsourcing”

“there was an order of magnitude difference in terms of number of people involved from generation to generation”

“you couldn’t find a computer scientist if you went on a search party”

“vendors try to push out-of-the-box code generation”

“very, very challenging to do… so we wrote our own code generator”

“if we did modeling just for code generation”

“you’ll likely not get the right abstraction”

The Bad-Ass

- mature domain
- non-software context
- ground-up effort
- real business driver

The Ugly

“it was a totally new concept of switching system”

“he made this huge decision ”

“50 developers all using this CASE tool”

“if there is a problem, they just contact the [CASE tool] engineers.. That was kind of their strategy”

“and suddenly the tool doesn’t do something expected”

“they try to contact the vendor but they don’t really know what’s going on”

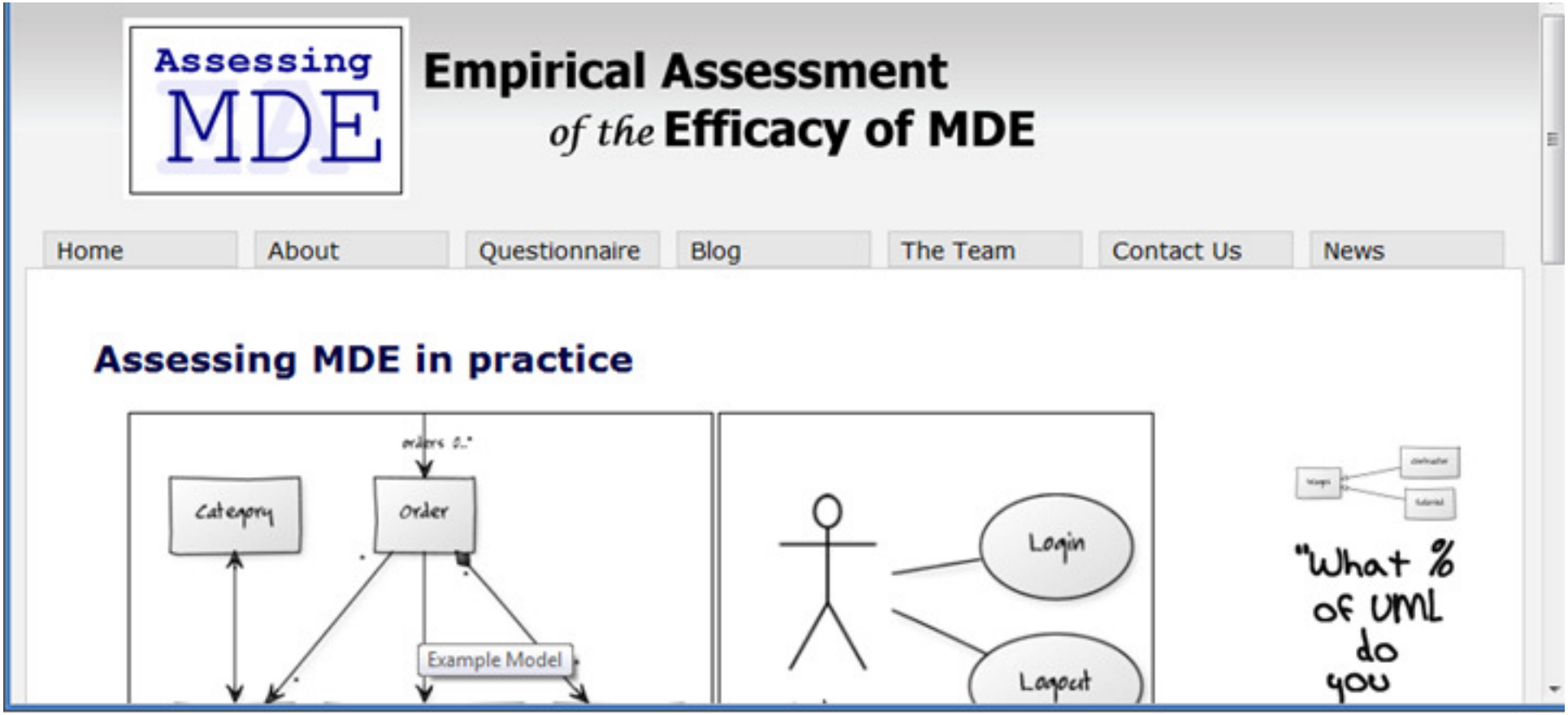
“they couldn’t optimize the generated code”

“so the way they had to do it was asking the hardware guys to have more hard disc, more memory, because of the tool”

“it was a nightmare to them to add all the features”

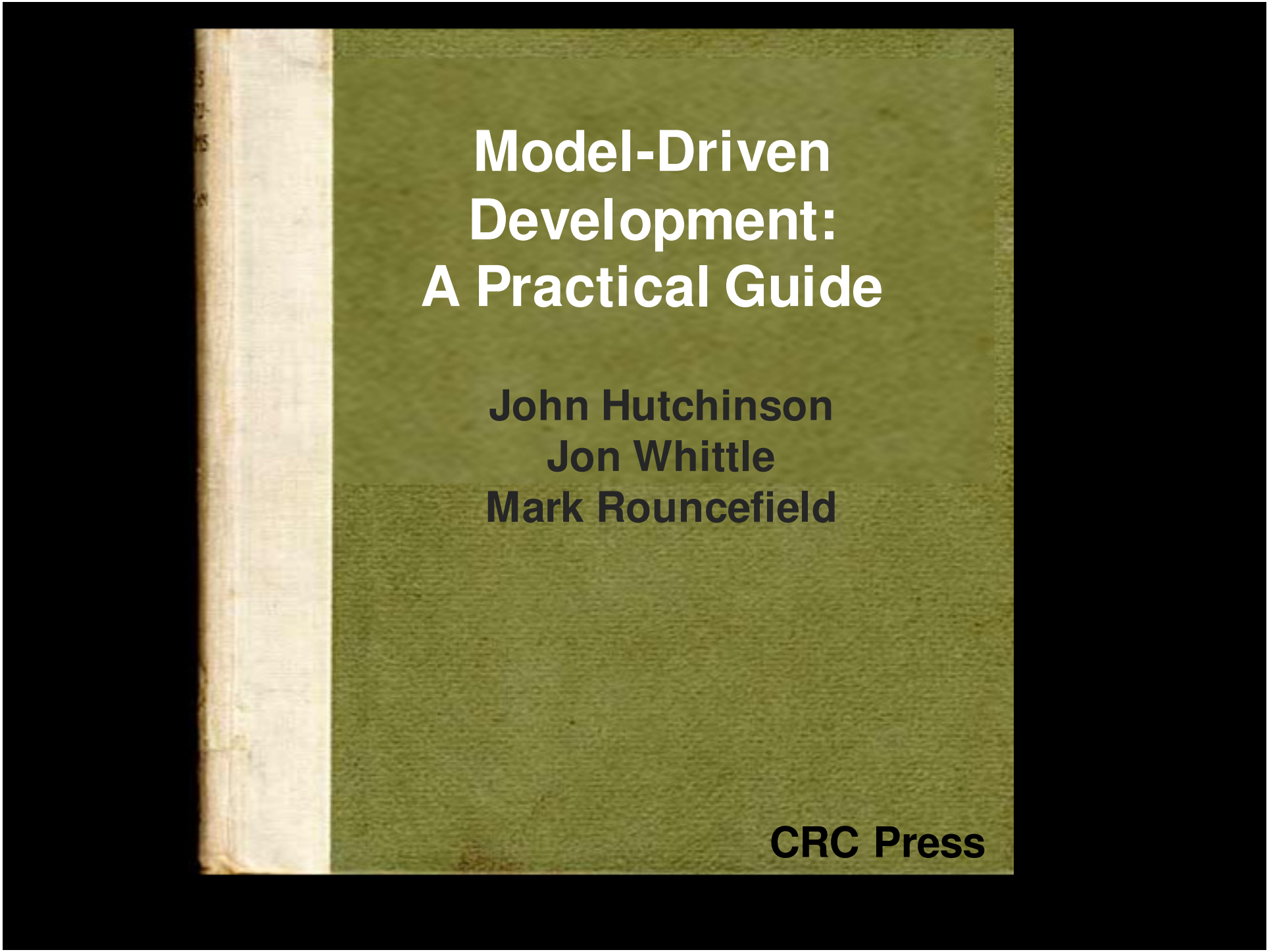
The Ugly

- immature domain
- top-down effort
- no real business driver
- lack of control



http://www.comp.lancs.ac.uk/~eamde/

whittle@comp.lancs.ac.uk


Model-Driven
Development:
A Practical Guide
John Hutchinson
Jon Whittle
Mark Rouncefield
CRC Press

Thank you for listening

LANCASTER UNIVERSITY

- "**Model Driven Engineering Practices in Industry**," 2011 International Conference on Software Engineering (ICSE)
- "**Empirical Assessment of MDE in Industry**," 2011 International Conference on Software Engineering (ICSE)

InfoLab21