Dependent Types for Dependable Modelling

Jeremy Gibbons
University of Oxford
GSDP workshop, Paris, Sep 2011

1. Robustness in modelling

A trend towards greater heterogeneity in scientific research:

- large, distributed teams
- long-running collaborations
- dynamic organization
- variety of stakeholders
- interdisciplinary interests

Implicitly shared context becomes untenable.

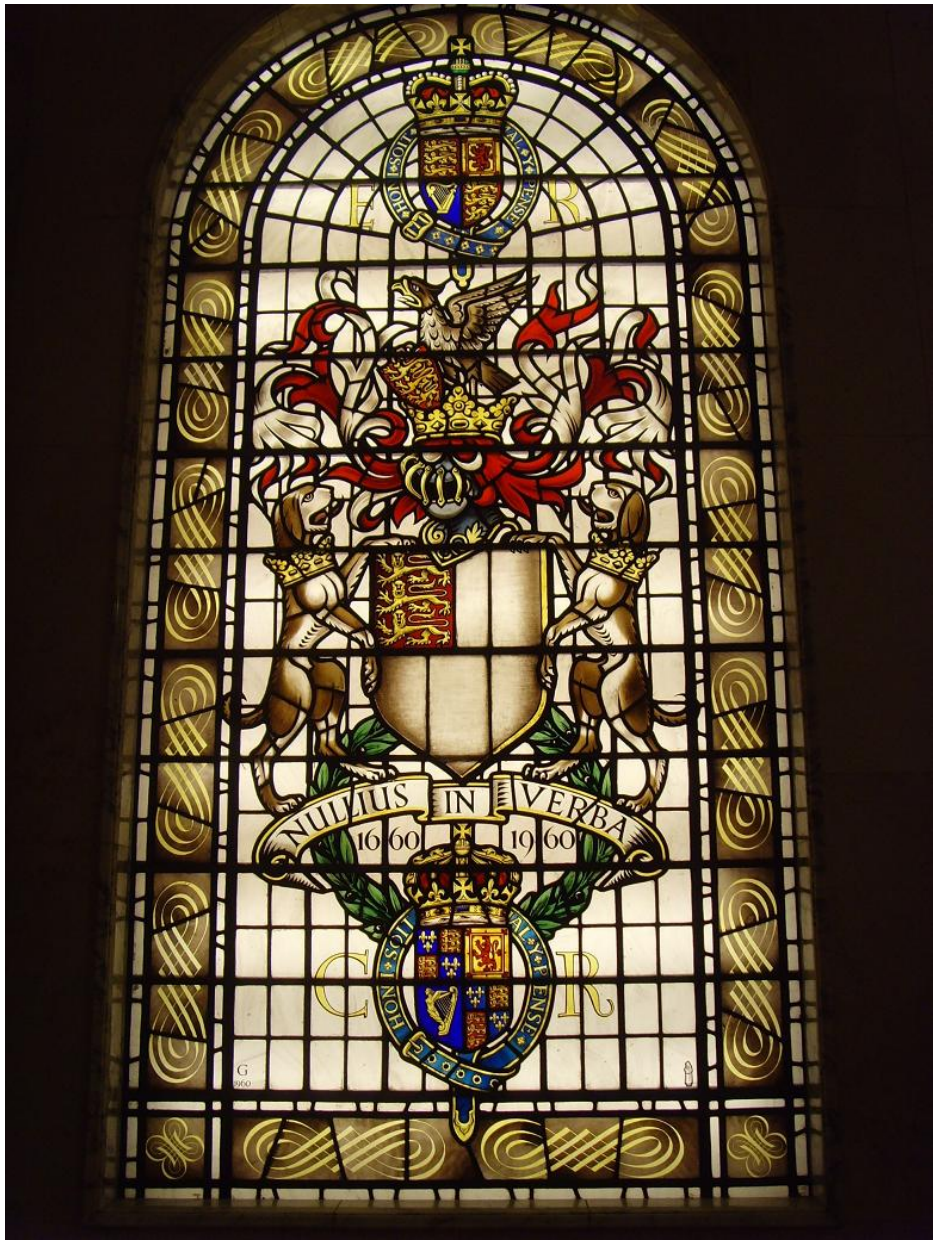
Assumptions should be *explicit*, *documented*, *transparent*, *checkable*.

1.1. Transparency and repeatability

Cornerstone of scientific method:

- model system
- perform experiments
- analyse data
- publish results
- *show working!*

(This is especially important in controversial fields, such as economics and climate change.)



1.2. Computational science

Much ‘working’ these days is digital:

- spreadsheets
- databases
- MatLab and Mathematica workbooks
- Perl scripts
- workflows. . .

‘Performing experiments’ amounts to running simulations.

Now what does ‘show your working’ mean?

1.3. Climategate

UK Parliament's review of UEA CRU:

> data disclosed in publications should be accompanied by sufficient detail of computer programmes, specific methodology or techniques used to analyse the data, such that another expert could repeat the work *(UK STC)*

1.4. Software verification and validation

Open source is no silver bullet:

- dependencies on libraries, OS, other vagaries
- bit rot
- run-time configurations: workflow
- proprietary systems

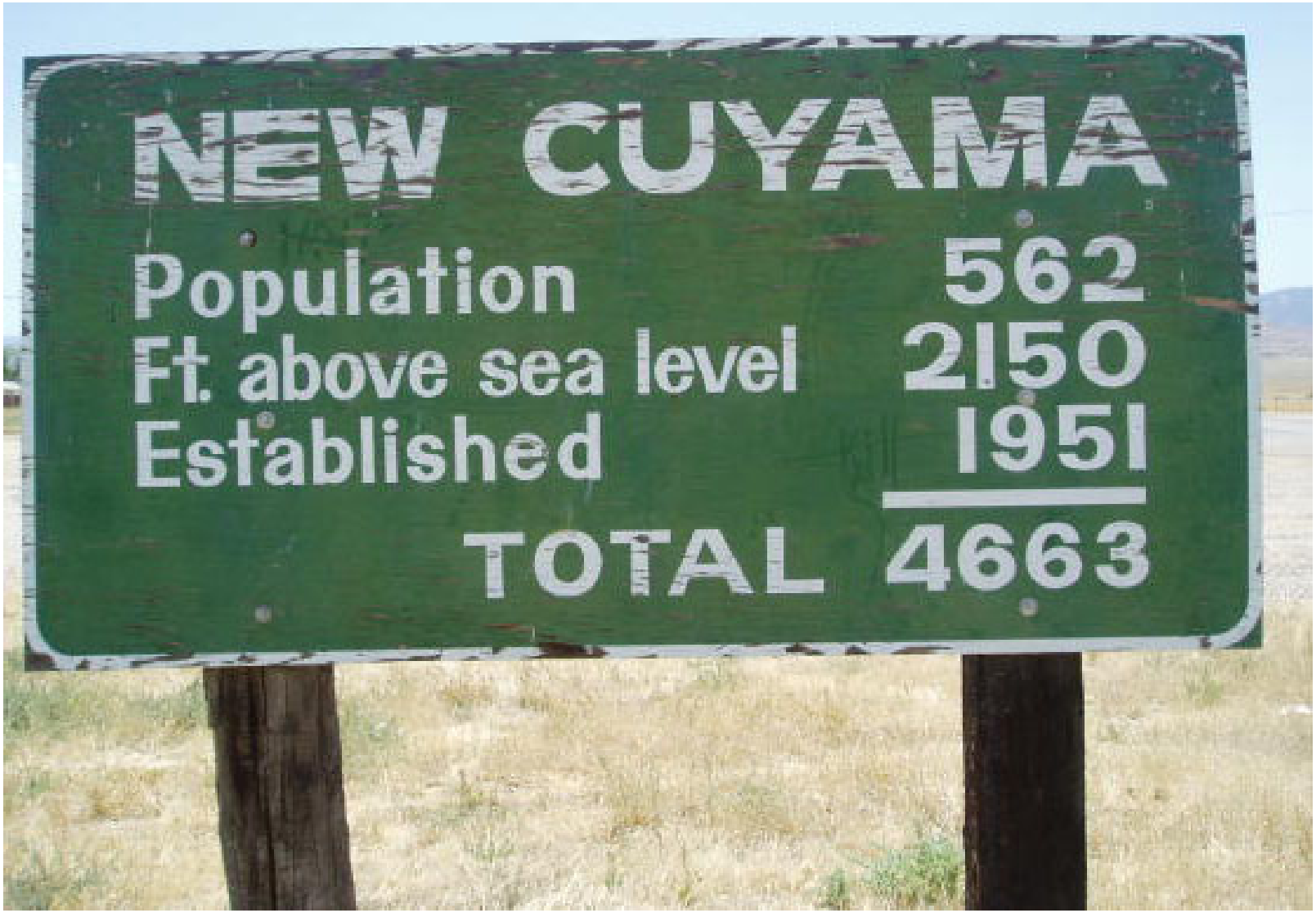
1.5. Digital preservation is difficult

Digital Domesday Book lasts 15 years not 1000

Robin McKie and **Vanessa Thorpe**
The Observer, Sunday 3 March 2002
Article history

It was meant to be a showcase for Britain's electronic prowess - a computer-based, multimedia version of the Domesday Book. But 16 years after it was created, the £2.5 million BBC Domesday Project has achieved an unexpected and unwelcome status: it is now unreadable.
The special computers developed to play the 12in video discs of text, photographs, maps and archive footage of British life are - quite simply - obsolete.

1.6. And as for digital trustworthiness...



1.7. Exploratory versus dependable development

The issue of dependability of software is even more challenging in applied fields than in professional development.

Quite rightly, scientists view programming as a means, not an end.

> Ignorance more frequently begets confidence than does knowledge *(Darwin)*

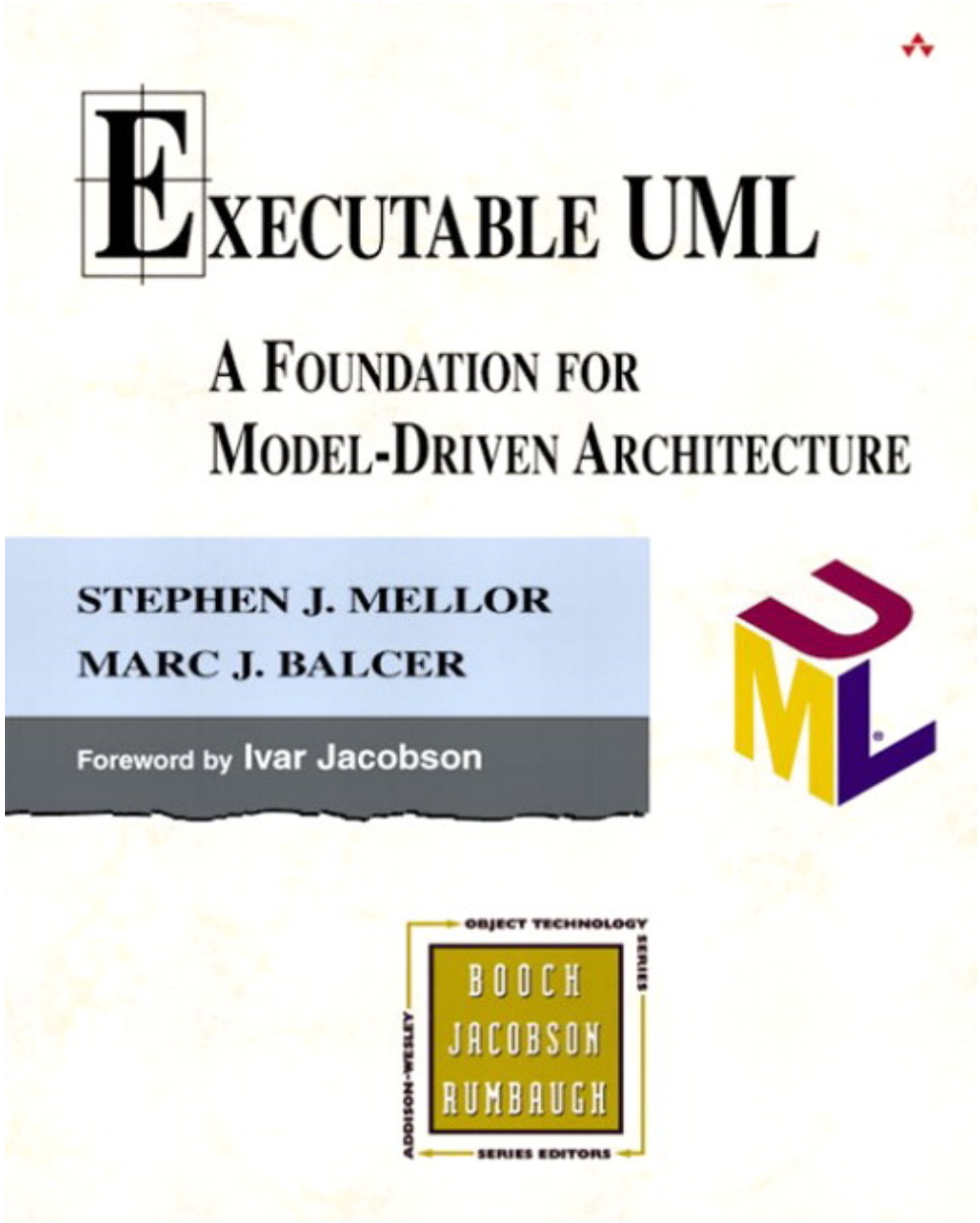
Also known as the *Dunning-Kruger effect*.

How to get dependable results without discouraging exploration of datasets?

- a light touch: MDE
- a more rigorous approach: DTP

2. A light touch: MDE

- abstract *metamodel* of problem domain
- instantiate to *model* of problem
- transformation of models to software artifacts
- translation or elaboration
- models serve multiple purposes



2.1. Semantic frameworks

At Oxford, we have been working on a series of projects developing what we call *semantic frameworks*—metadata-based MDE for

- semantically rich domains
- heterogeneous collaborations (in time, space, field...)
- often low-budget
- transparency important

Initial work in clinical trials.
But very similar concerns in eg electronic governance.

Studiously trying to do *the simplest thing that could possibly work*.

2.2. Forms-based MDE

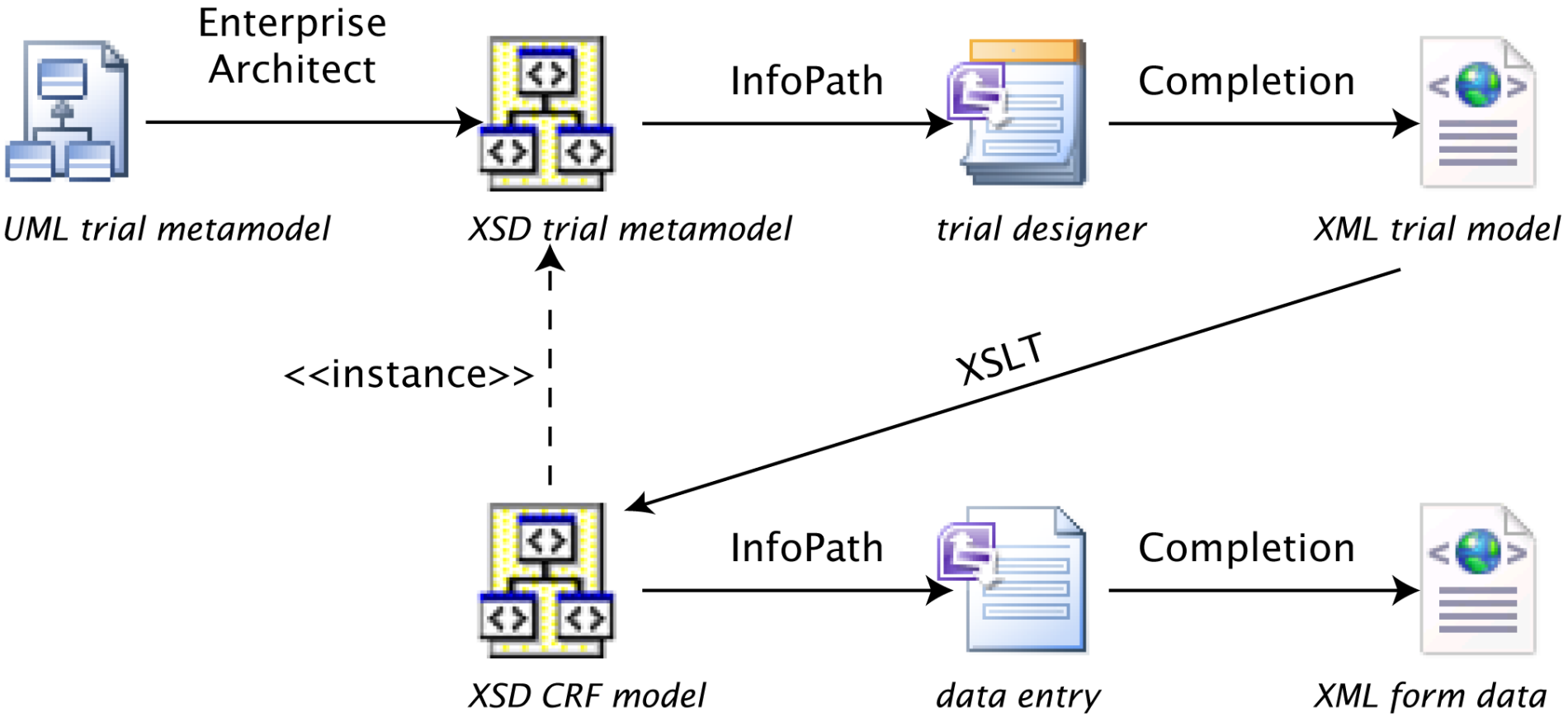
Off-the-shelf productivity software (eg Microsoft InfoPath and SharePoint) often suffices:

- *document schemas* as data models
- *conformant documents* as entities
- *form completion* as authoring
- *schema mappings* as model transformations

In some sense dual to Executable UML:

> Show me your flowchart and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowchart; it'll be obvious. *(Brooks)*

2.3. Workflow



(Note the two phases.)

2.4. Metadata for integration

- much of today's science depends on collaborative work, integrating datasets and insights from diverse heterogeneous communities
- eg meta-analysis of multiple trials in clinical medicine
- but the datasets are meaningless out of context

  units, procedures, questions, standards, conventions...

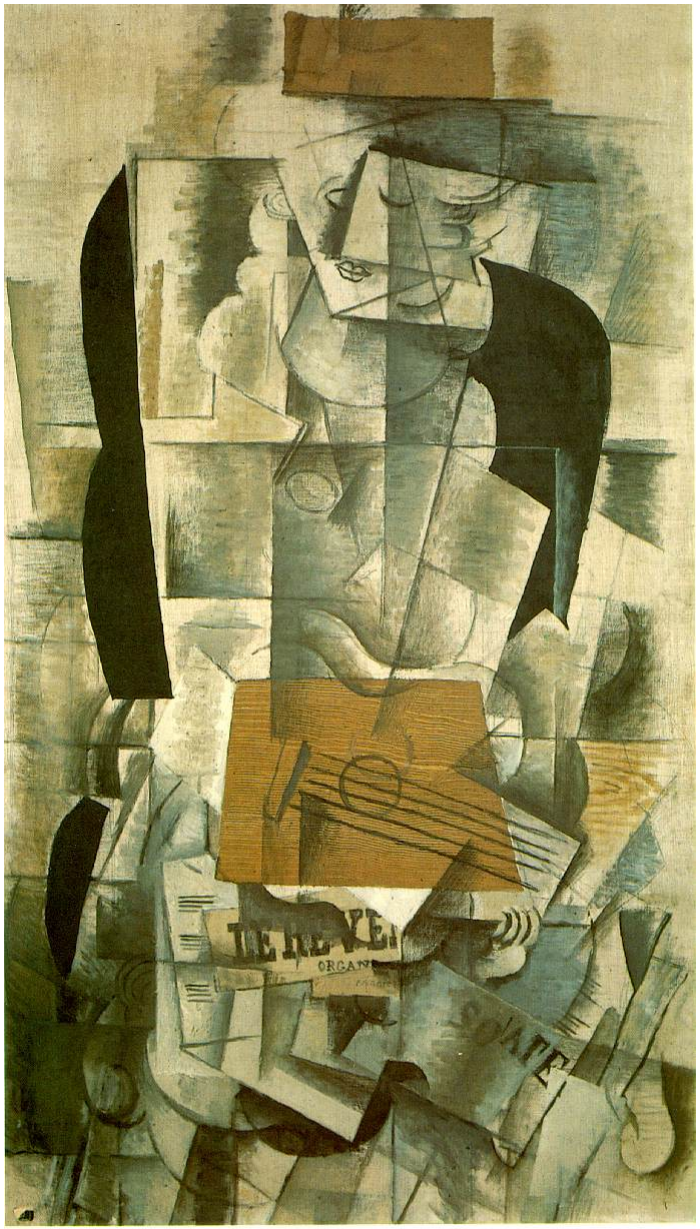
- researchers need tools to adequately capture this contextual metadata

2.5. Postmodernism

For any moderately complex system, we can't all agree on a single model; we shouldn't try to.

There is no one privileged view.

We need to allow for different data communities to document their local practices and conventions.



3. A more rigorous approach: dependent types

- MDE takes a relatively *informal* approach to modelling:
  - metamodels to express models
  - model-to-model and -to-code transformations
  - model conformance is a relatively syntactic matter
  - increased confidence through *transparency* and *reuse*
- can we do more? what if informality is unacceptable?
- *dependent types* are the state of the art in programming language technology for constructing

  software components with checkable assumptions

3.1. Static type-checking

- inside, there are only bits—it's all a matter of interpretation
- take care to avoid nonsense, eg multiplying a string by a date

  Well-typed programs don't go wrong. *(Milner)*

- gain some security via tags and *dynamic* checks

- regain some speed via *static* checks for compile-time proof

```
begin
  int i:=0                    /* once an integer,  */
  while i<10 do begin
    p(i) ; i:=i+1             /* always an integer */
  end
end
```

3.2. Hindley–Milner typing (1970s)

- types

$$\sigma, \tau ::= \mathit{Int} \mid \mathit{Float} \mid \mathit{Bool} \mid \cdots$$
$$\mid \sigma \times \tau \mid \sigma \to \tau \mid \cdots$$
$$\mid \alpha \mid \beta \mid \cdots$$

- algebraic datatypes

$$\textbf{data}\ \mathit{List} :: * \to * \ \textbf{where}$$
$$\mathit{Nil} :: \mathit{List}\ \alpha$$
$$\mathit{Cons} :: (\alpha \times \mathit{List}\ \alpha) \to \mathit{List}\ \alpha$$

- polymorphism, implicit universal quantification

$$\mathit{concat} :: \mathit{List}\ (\mathit{List}\ \alpha) \to \mathit{List}\ \alpha$$
$$\mathit{concat}\ \mathit{Nil} = \mathit{Nil}$$
$$\mathit{concat}\ (\mathit{Cons}\ (xs, xss)) = \mathit{append}\ (xs, \mathit{concat}\ xss)$$

- inference of principal type schemes

3.3. Greater precision

- but not all errors are obviously type errors
  - *off by one* (eg counting down to one rather than zero)
  - *out of bounds* (eg taking first element of empty sequence)
  - *confused comparison* (eg mixing up branches of conditional)
  - *mismatched units* (eg treating value in pounds as in newtons)
- Milner a bit optimistic—H–M typing a bit coarse
- not enough to say ‘an integer’—which integers?
- greater precision through *dependent types*

3.4. Dependent types in a nutshell

- allowing types to depend on values—eg dependent function space

$$\sigma, \tau ::= \cdots \mid (x : \sigma) \to \tau(x)$$

- eg indexed datatypes

$$\begin{array}{l}\textbf{data}\ \mathit{Vector} :: \mathbb{N} \to * \to * \ \textbf{where} \\ \quad \mathit{VNil} :: \mathit{Vector}\ 0\ \alpha \\ \quad \mathit{VCons} :: (\alpha \times \mathit{Vector}\ n\ \alpha) \to \mathit{Vector}\ (n{+}1)\ \alpha \end{array}$$

$$\begin{array}{l}\textbf{data}\ \mathit{BNat} :: \mathbb{N} \to * \ \textbf{where} \\ \quad \mathit{BZero} :: \mathit{BNat}\ (n{+}1) \\ \quad \mathit{BSucc} :: \mathit{BNat}\ n \to \mathit{BNat}\ (n{+}1) \end{array}$$

- richer types can explain dependencies between components

$$\begin{array}{ll}\mathit{find} :: & (\alpha \to \mathit{Bool}) \to \mathit{Vector}\ n\ \alpha \to \mathit{Maybe}\ (\mathit{BNat}\ n) \\ \mathit{lookup} :: & \mathit{Vector}\ n\ \alpha \to \mathit{BNat}\ n \to \alpha \end{array}$$

- (but type-checking now involves some evaluation...)

3.5. Documenting and checking assumptions

- not restricted to numeric indices
- types can depend on any kind of data
  - shape of a data structure
  - state of a process
  - provenance of an input
  - proof of a property
- cutting-edge machine-checked proofs, eg of 4-colour theorem

3.6. Continuous modelling

- floating-point arithmetic is notoriously tricky
- rounding, exceptions, signed zeroes, under- and overflow...
- usual laws of arithmetic break
- 'equivalent' equations yield different behaviour

  > Unreplicated simulation models and their results cannot be trusted. [...] An unreplicated simulation is [...] almost certainly wrong. *(Edmonds & Hales)*

- especially painful for emergent behaviour in complex systems!

3.7. Exact real arithmetic

- constructive approaches to real analysis

  God gave us the integers; all else is the work of man.
  (Kronecker)

- original inspiration for *intuitionistic* mathematics—restrict attention to proofs that yield witnesses
- 'evidence-based mathematics':
  from $\forall \epsilon{>}0, \exists \delta : |f(x{+}\delta) - f(x)| < \epsilon$ to $\Delta_f(x, \epsilon)$
- eschew *law of the excluded middle* $P \vee \neg P$

  many intuitionistically suspect constructions, such as exact comparisons, are numerically unstable anyway

- eg real arithmetic via improving intervals, or Cauchy sequences

3.8. Curry–Howard

Direct mapping from proof to computation: *constructive type theory*.

Logic	*Computation*
propositions	types
proofs	programs
satisfiability	non-emptiness
demonstration	inhabitation
conjunction	pairing
labelled disjunction	disjoint union
implication	function
well-foundedness	induction
…	…

3.9. Enrich-EM

- *An IT-based Framework to Enrich Economic Models for the Governance of the Globalized Economy and its Markets*
- aiming to overcome

  the serious limitations of existing economic and financial models *(Trichet, ECB)*
- EU FP7, rejected :-(
- constructive type theory, multiscale modelling, service-oriented architecture
  - model alignment and integration
  - documenting and checking modelling assumptions
  - justified (or at least justifiable) real arithmetic

4. Conclusions

- in collaborative research, make assumptions explicit
- lightweight approach: model-driven engineering
- rigorous approach: constructive type theory
- still hoping to make this fly!

- any questions?