Model-Driven Development and the Future of Software Engineering Education

A. J. Cowling

Department of Computer Science

University of Sheffield

The Development Process

Products	Processes	People	Change?
Business Problem			
↓	Requirements Analysis	Analysts	
Requirements			
↓	Design	Designers	
Architecture			**Suppose this is automated! What then?**
↓	Construction	Coders	
Implementation Code			
↓	Compilation etc	Automated	
Executable System			

Timescales (1)

For This Change

- It is an active research area:
  - Model-Driven Development;
- Some systems exist already, for limited domains:
  - Itanium Appcelerator;
- The general problem is a complex one, so
  - general systems may be some years away – perhaps 2020?
- But the economic benefits from them will be huge.

A Prediction

- For our current undergraduates, before the ends of their careers,
- The activity of writing conventional program code will become as rare and exotic
  as writing large pieces of code in assembly language is now.

Timescales (2)

An Historical Analogy

- Development of problem-oriented languages;
- Automating the production of executables:
  - ie the process of compilation, etc.

Timeline

1958 to 1960 – First versions of Algol, Cobol and Fortran;
1964 – Compilation methods well-established:
- eg Randell & Russell, "*Algol 60 Implementation*";

1968 – ACM's Curriculum 68:
- assumed that problem-oriented languages would be used,
- and assembly language programming covered later;

1973 – C used to implement almost all of unix:
- so all application domains now covered.

Impact on the Curriculum (1)

For the SE 2004 SEEK

- Some knowledge areas will be relatively unaffected:
  - they are effectively independent of this activity;
  - eg Software Evolution (EVO), Mathematical & Engineering Fundamentals (FND), Software Management (MGT), Professional Practice (PRF), Software Process (PRO), Software Quality (QUA);
- But, almost one third is Computing Essentials (CMP):
  - mainly CS concepts – programming, OS, databases, etc,
  - and construction activities, from a programming perspective,
  - this will change significantly.

For the CS Body of Knowledge

- This change will have even bigger effects than for SE.

Impact on the Curriculum (2)

For the SE 2004 SEEK (continued)

- The focus will have to become models of software structures:
  - as in Software Modeling & Analysis (MAA),
  - which currently says little about actual models,
  - but may need to cover relationships between models and code;
- The associated process activities will be affected:
  - they will need to take model-centric approaches,
  - which currently they hardly do,
  - eg Software Design (DES),
  - and Software Validation & Verification (VAV), which has some very code-centric units.

Impact on the Curriculum (3)

For the SE 2004 Introductory Courses

- How much programming will SE101 or SE102 need?
  - arguably, none!

For the SE 2004 Core Courses

- Currently two sets of courses, with different approaches:
  - one top-down, the other bottom-up;
- Will the bottom-up approach still be valid?
  - arguably not!

Conclusions

This Change is Coming

- Bits of it have happened already;
- We can not predict how quickly it will mature;
- When it does, its impact will be very rapid:
  - as with the adoption of problem-oriented languages;
- And it will be very significant for SE education.

Likely Impacts

- Some parts of the SE 2004 SEEK may not need to change much;
- Programming-related parts will change substantially:
  - Reflecting even bigger changes for the CS curriculum;
- Other parts will become much more model-centric;
- The approaches of the courses will change substantially.

The End

Thanks for your attention!!

Questions and Discussion to follow.