Computer Science

UCL

THE NEXT 10 YEARS: THE SHAPE OF SOFTWARE TO COME AND WHAT IT MEANS FOR SOFTWARE ENGINEERING

Anthony Finkelstein

Brief outline of talk …

The Discipline of Software Engineering…

- Sustained relevance of 'big agenda'
- Substantial scientific progress but (perhaps) receding impact on practice
- Significant advances in some areas
  - Testing
  - Automated verification (model-checking)
  - (largely outstripping capacity to absorb innovation)

The Discipline of Software Engineering…

- Uncertain directions in other areas
  - Software architecture
  - Software design
  - Software requirements
- Difficulties in making progress in some areas
  - Software development tools
  - 'Methodologies' (modelling and process combos)
  - Middleware
- Grounds for optimism but …


© FERRADURA MUZENZA
WWW.CAPOEIRACADEMYUK.COM


© FERRADURA MUZENZA
WWW.CAPOEIRACADEMYUK.COM

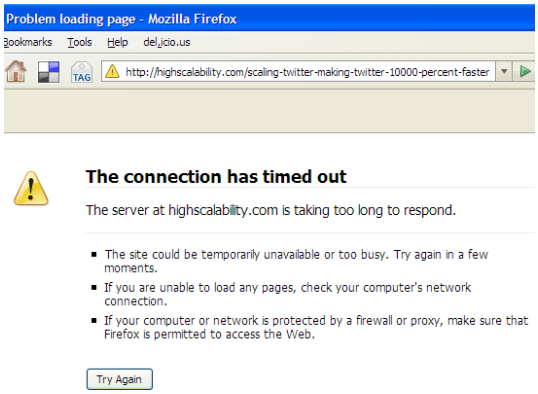
Our largest risk … not recognising the game has moved

‘Internet-scale’ Services

- Characterised by …
  - Large and rapid variations in the demand for resources
- Existing practice
  - Some high level patterns for limited classes of application
  - Resource profligacy
  - Suck it and see (dimension by dimension)

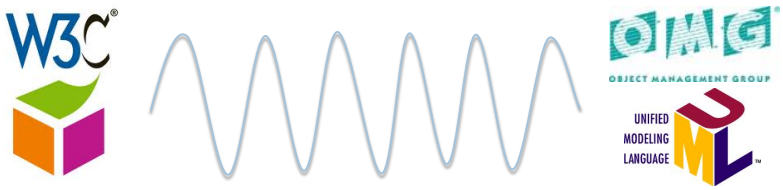
What it Means …

- Building large-scale testbeds
- Understanding scaling ‘in the wild’
- Architectural breakdowns
- Dynamic systems models



Convergence of Content & Infrastructure

- Separation of the semantic models
- Existing practice
  - Web standards and software engineering standards moving in different incompatible directions
  - Wasteful of effort and of technical opportunity



What it Means …

- Stop playing at the periphery and pull back to fundamental requirements, a fudge probably will not work
- Devise and test shared schemes
- Identify quick wins
  - For example smart semantic tagging of software artefacts
- Start the 'hard grind' of engagement with standards bodies

Marginal Business Advantage

- From enabling to improvement
- Existing practice
  - We are unable to reliably predict the cost/effort required to build a system. We may be fortunate and have built a very similar system before.
  - Function Points are precious little assistance. 'Jelly Beans' only work for small systems, relatively 'late' in the process.

What it Means …

- Nothing even on the horizon here!
- We are probably going to have to:
  - Rethink software economics
    - Making money a 'first class object' in software engineering
  - Get a much better handle on 'programmer productivity'
  - Provide an appropriate data-sharing infrastructure

SaaS

- New models around SaaS
- Existing practice
  - We know how to build SaaS (sort of) but we don't know how to:
    - buy it
    - manage QoS
    - achieve interoperability

What it Means …

- Stop 'wasting time' with fine grained software services (wake up and smell the cocoa)
- Enterprise mash-ups
- Requirements methods based on balancing mutability
- 'Security in the cloud'
- 'Walk away' methods

The Apotheosis of 'Apps'

- Existing practice
  - Highly-tuned, device-specific interfaces across to services with 'sync' to clients
  - Because a viable payment model exists …

What it Means …

- Requirements engineering for mass-markets
- New types of 'product-family' engineering
- App Stores SM
- App management
- App assembly

Towards Channels

- Continuing development, continuing change
- Subscription-based business model
- Engagement & retention
- Channel packages

What it Means …

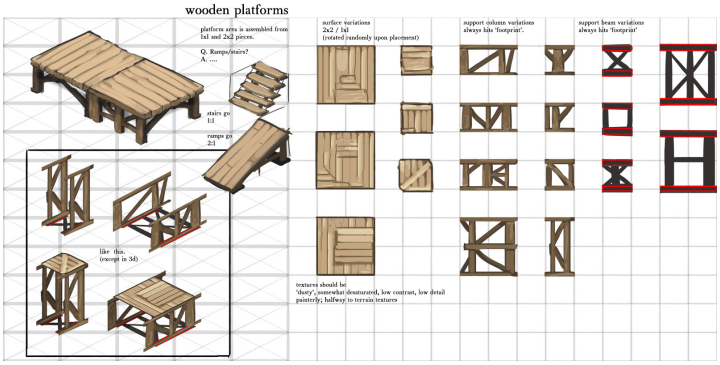
- Relationship focus
- Continuing development
- New paradigms

Platforms & Ecosystems

- Operational platforms (upperware)
- Functional clusters
- Interdependence between platforms and plugin, app, adapter ecosystems
- Developer ecologies

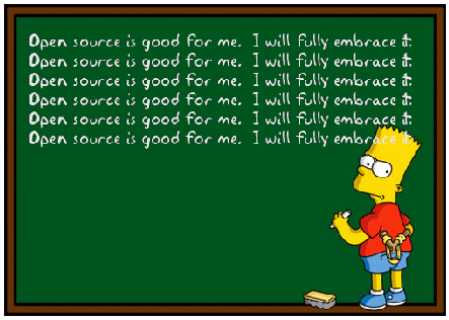
What it Means …

- API design
- Intertwining of commercial and technical architecture



Transformation of Open Source

- Open / Community source model changing
- Unable to drive innovation
- Take over by large organisations
- Interplay with service-model



What it Means …

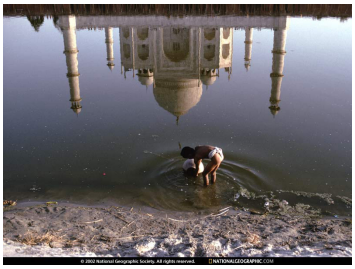
- Unclear …
- Advantage is service wrap
- … and capacity to leverage ecosystem

'Adaptive' Systems

- Systems that must adapt to context
- Existing practice
  - Problems with systems embedding significant COTS/ Community Sourced independently evolving components
  - Problems with systems that involve user scripting and 'plug-ability'

What it Means …

- Moving reflection from being a programming language level mechanism to software systems that can 'account for themselves'
- Can reflect their requirements and (through monitoring) the extent to which those requirements are being satisfied



Governance

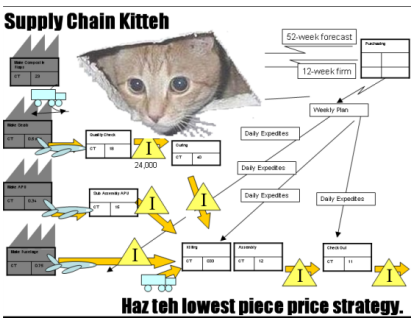
- Mismatches at the boundaries between business and software engineering give rise to many of the problems we encounter
- Changing business structures … more dynamically assembled

What it Means …

- Reengaging with the business interface – IT/IS communities
- Much more serious study of allocation of decision rights
- Governance design as part of development

Supply Chains

- Addressing complex inter-product and inter-supplier dependencies
- Existing practice
  - None to ad-hoc



What it Means

- Rethinking software production
- From garage 'design and make' to … globalised interdependent business

A Conclusion

- The dangers of not reacting quickly enough to changes in business structures and technical capabilities
- We can 'catch-up' but we lose credibility