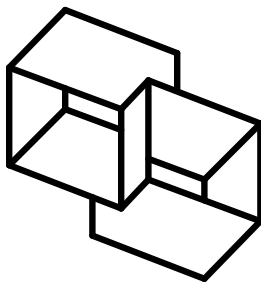


Alternatives to Software Engineering



Pete McBreen, McBreen.Consulting
petemcbreen@acm.org

The *software engineering* metaphor allows us to talk about software development

This metaphor allows us to identify the major activities in software development

Analysis, design, coding, testing and integration

Once these activities are identified they start to shape our thinking

Software engineering is the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software; that is, the application of engineering to software -- IEEE definition

This idea of a systematic, disciplined and quantifiable approach originated in 1968

Overall the software engineering approach has been successful for large projects

Structured Analysis and Structured Design was a great leap forward compared to previously

The Space Shuttle software is a great poster child for this approach

- *The last three versions of the program – each 420,000 lines long-had just one error each. The last 11 versions of this software had a total of 17 errors.*

Similarly, the **Software Engineering Institute's** Capability Maturity Model has great results

Unfortunately the implications of the software engineering are not all positive

Fred Brooks wrote *The Mythical Man Month* to address one set of problems

Parnas and Clemens wrote *A Rational Process: How and Why to Fake It* to address another set of problems

Methodology wars have been fought over the *one best way* to develop software

The *Waterfall lifecycle* still dominates thinking about software development

For most projects, software engineering is espoused rather than actually practiced

The resulting disconnect between thought and action is the cause of many problems

Projects overrun because the team did not take the time to realistically estimate or plan the project

Design is started with a sketchy understanding of what is actually needed

Designers do not put enough details into their CASE tool models to enable coding to be a mechanical task

Testing is cut short because coding overruns the original schedule

The software engineering metaphor is limiting our ability to deliver software

The time has come to start looking for alternate metaphors that work better

**The test for a new metaphor is simple:
*Appropriate metaphors invoke appropriate behaviour***

Software engineering manifestly fails this test

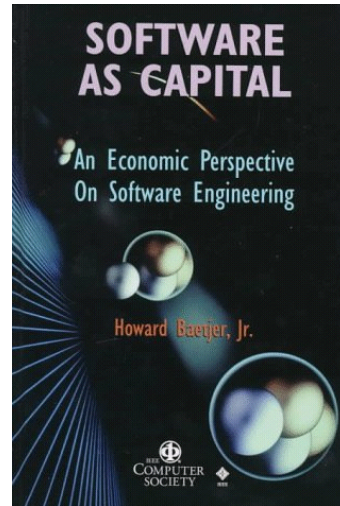
- We think of **software projects** not **software assets**
- We forget about **emergent behaviour**
- We argue against **incremental development** even though **all software is developed incrementally**
- We do a lousy job of **nurturing new developers**

Howard Baetjer has shone a spotlight on the economics of software development

Thinking of software projects downplays the role of maintenance

Software as capital assets puts maintenance in a central role

In what other field do we have very expensive artifacts maintained by the cheapest labor we can find?

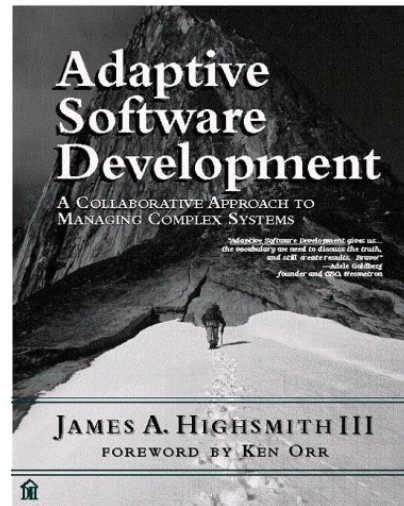


Jim Highsmith has looked at how emergent behaviour affects software development

Managing the *workstate* rather than the workflow is the key difference

Collaboration and self-organization enable *emergence*

**Describes a new lifecycle
Speculate, Collaborate,
Learn**



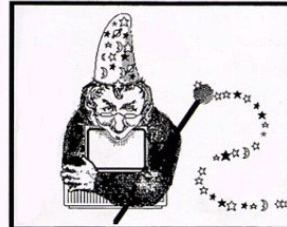
DeGrace and Stahl dissected the waterfall model and looked at incremental development

Recognized that getting complete requirements is hard

Changing requirements are a natural consequence of software delivery

Attempted to place people at the center of the software development process

Wicked Problems, Righteous Solutions



A Catalogue of Modern Software Engineering Paradigms

Peter DeGrace
Leslie Hulet Stahl

YORKEN PRESS COMPUTING SERIES

Putting people at the center of software development is becoming a recurring theme

The Agile Alliance (www.agilealliance.org) is a group of methodologists who "value *Individuals and interactions over processes and tools*"

Scrum, eXtreme Programming and Feature Driven Development are all Agile approaches

Software development is not a *mechanical* task, it is a *social, intellectual* task

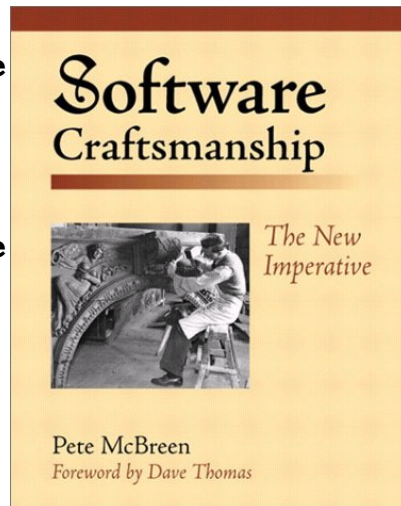
Unfortunately the software engineering metaphor hides this aspect of software development

Software Craftsmanship is a new metaphor that directs attention to skilled developers

Software Craftsmanship explicitly looks back to the craft tradition

Apprenticeship and situated learning are the way to pass on esoteric knowledge

Experienced people are more important than the specific process used



Many other metaphors are possible for software development

The GNU/OpenSource community talks about *the Cathedral and the Bazaar*

Richard Gabriel talks about *Habitable Software* as the place developers live

***eXtreme Programming* is attempting to create a new vocabulary for development**

***Gamasutra.com* is attempting to articulate how to create great game software**

Closing thoughts

Software engineering is not going to go away any time soon

but we need to distinguish *working harder* and *working smarter*

We all need to act congruently with what we know about software development
and *push back* against unrealistic constraints

Software Craftsmanship may be a way to get mentoring and coaching happening

Sources and references

Space Shuttle software

<http://www.fastcompany.com/online/06/writestuff.html>

www.agilealliance.org

www.xprogramming.com

www.gamasutra.com