Software as Engineering Material

Keith Braithwaite, SPA2014, Monday 30th June, 2014 2:30pm – 4:00pm

Introduction

Zuhlke was set up as a product innovation company. Most business is in software development.

Engineering vs. Design vs. Manufacturing. Engineering is about adapting existing art into new kinds of solution. Design or Product development is iterative, incremental, evolutionary. Very rarely does anyone invent anything brand new. Manufacturing is completely different – series production.

Feedback is provided by the marketplace to the product engineering / development cycle. For example, the coffee capsule machine on sale in the US gave rise to a market for third-party "generic" capsules (some of them refillable). The next generation machine will have DRM technology to enable it to refuse to accept these capsules. A nice example of engineering reuse, too.

Material Properties in Product Design

Example: snap-fit fasteners for cases all use the principle of an extruded cantilever arm that snaps into a recess. Googling for designs leads you to pages of pre-packaged designs published by the chemical companies for your use in designing a case. They come with formulae to let you calculate the parameters of the shape based on criteria such as the amount of force needed to push the case closed, how hard you want it to resist attempts to separate the parts, the stiffness of the material etc.

Design patterns should have provided this kind of library for software engineering. So far, nobody has identified the right parameterisation. Dave Cleal: software libraries might be another way to apply this principle. In Dave's view most libraries are far too extensive – they should concentrate on doing one thing well.

Things to note

- A lot of the parameters in the data sheet are analogous to "non-functional characteristics" in a software system. The actual function of the fastener is taken as read.
- Nothing was proved here – all closed-form equations

Software "Engineering"

Would we want to build software this way? Maybe…

- Reliable
- Safe
- Quick
- Interchangeable/interoperable parts
- Other desirable properties?

What would be the "material properties" of software that could guide us? Plastics: easy – graphs showing deformation vs. force, etc.

Jason: there are no S.I. units of software properties. Everyone interprets them differently (even metrics like cohesion and complexity). These measurements need to be reusable beyond one single producer and consumer.

Example of the Nespresso milk frother: simulation was used to predict how fluids would move in the vessel etc. before any prototype was built.

Nat: there is some interesting research contrasting different programming languages in terms of their robustness to "fat finger" programming errors (what proportion of lines will still compile, run, and run without causing an obvious error). Interesting results: Java does better than Haskell, thanks to the amount of redundancy in it; C was no better than a randomly generated syntax.

Jason: graphs have two dimensions, which is more valuable than a single number (coefficient). Nat: frameworks are sold on the basis of how easy they make it to write software – yet 90% of software costs come in maintenance.

Nat: presented a feedback loop example.

Jason: predictive properties are more like they might be in medicine – where you can predict that of a given population, a certain proportion will succumb to a particular disease within a given time period.

Keith: we lack a "thermodynamics" of software. Jason: is this the right question? What we need to measure is determined by the questions about it we need to answer. E.g. if we want the project to be releasable at a moment's notice, what do we need to know about the software to ensure that? Typical questions are "how much will it cost" and "when can I have it".

Immo: volatility might be a useful measure when it comes to predicting maintenance costs.

Keith: most engineering disciplines are enabled to think in bigger units because they have available a library of design ideas. The first cantilever analysis was done by Galileo – so physical engineering has had a fair while to build up this body of parameterised knowledge.

Historically, most things we have measured about software have not proved useful.

Conclusion

Can we do anything to get more data?

Immo: Engineers need incentives to feed back (e.g. into pattern web sites) implementation and maintenance implications of using the patterns.