Aspects of sustainability in software development

Paul Walk

p.walk@ukoln.ac.uk

UKOLN is supported by:

JISC

www.ukoln.ac.uk

A centre of expertise in digital information management

UNIVERSITY OF BATH

what do we mean by ‘sustainability’ in terms of software?

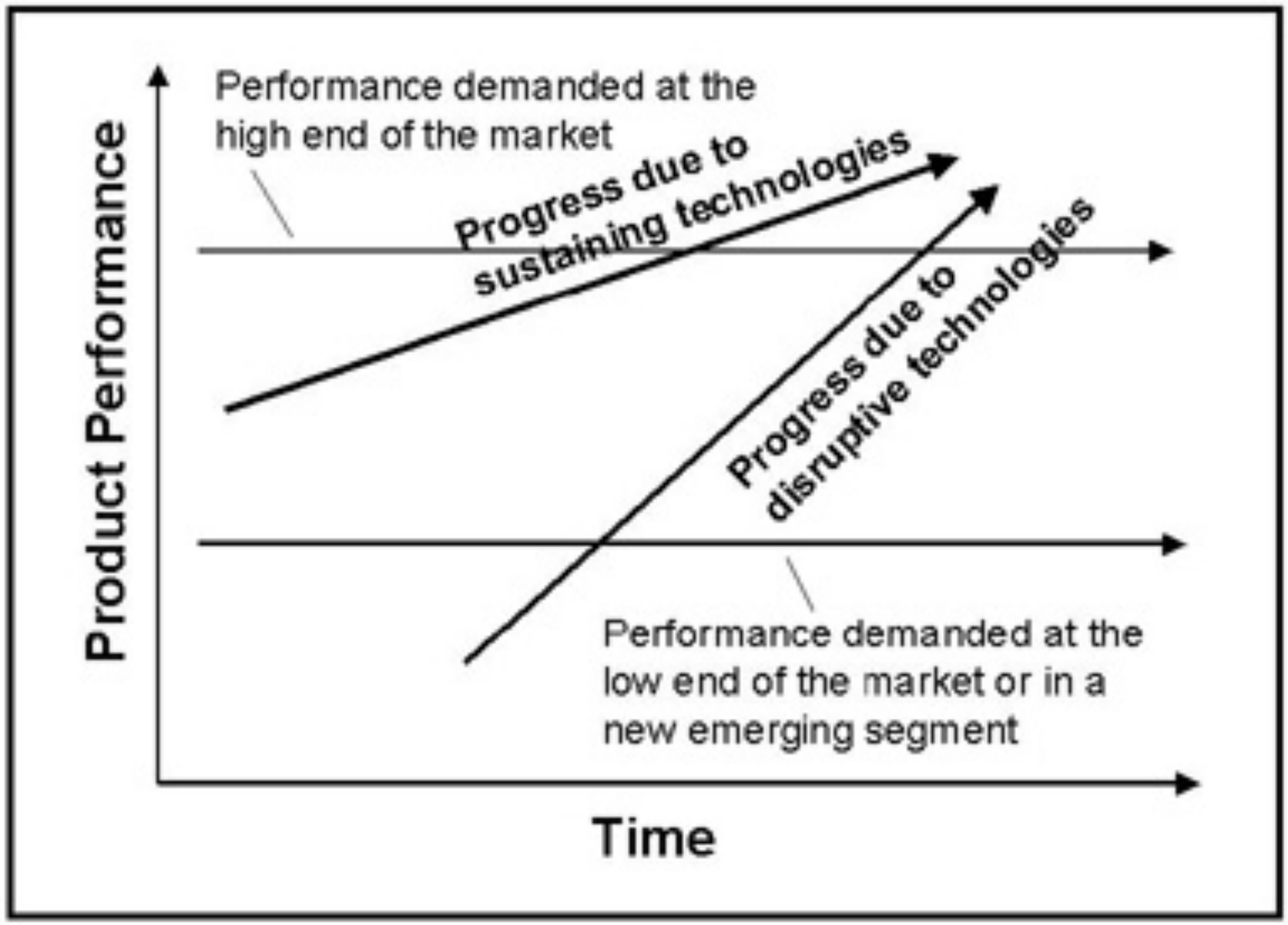
short answer:

- we mean either:
  - **preservation -** in the sense of 'archiving'
- or
  - **ongoing provision as a service (e-infrastructure)**
- debate dogged by frequent mutual misunderstanding with these two aspects being conflated or confused
- difficult (undesirable?) to entirely separate these - too many factors in common

why sustain?

- science / research
  - sustained as in *preserved* so that experiments can be repeated - reproducibility of results - software is ideally **unchanged**
- infrastructure supporting research
  - sustained as in *invested in continually* for efficiency of cost and effort - software is ideally **continuously changing**
- business
  - sustainability of software is mostly a sunk cost....
  - where we want our 'customers' to have to pay little or nothing, business wants them to pay again and again
  - much software development discipline comes from business

when to choose not to....



from The Innovator's Dilemma, Clayton Christenson
(http://web.mit.edu/6.933/www/Fall2000/teradyne/clay.html)

characterising the problem

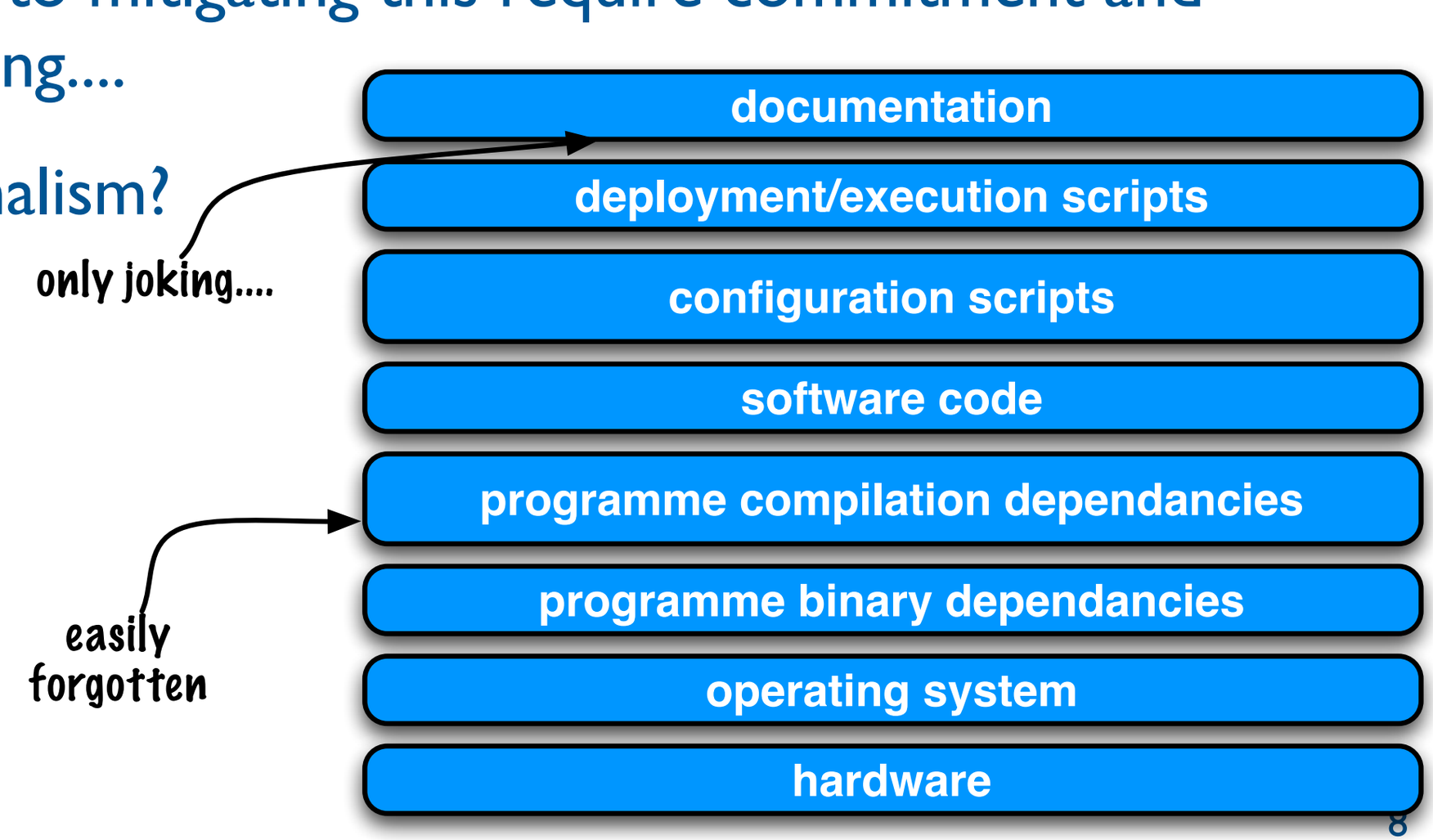
shelf life

Neil Chue Hong talks about software on the shelf having a shelf-life of **6 months**

http://www.flickr.com/photos/cogdog/

complexity

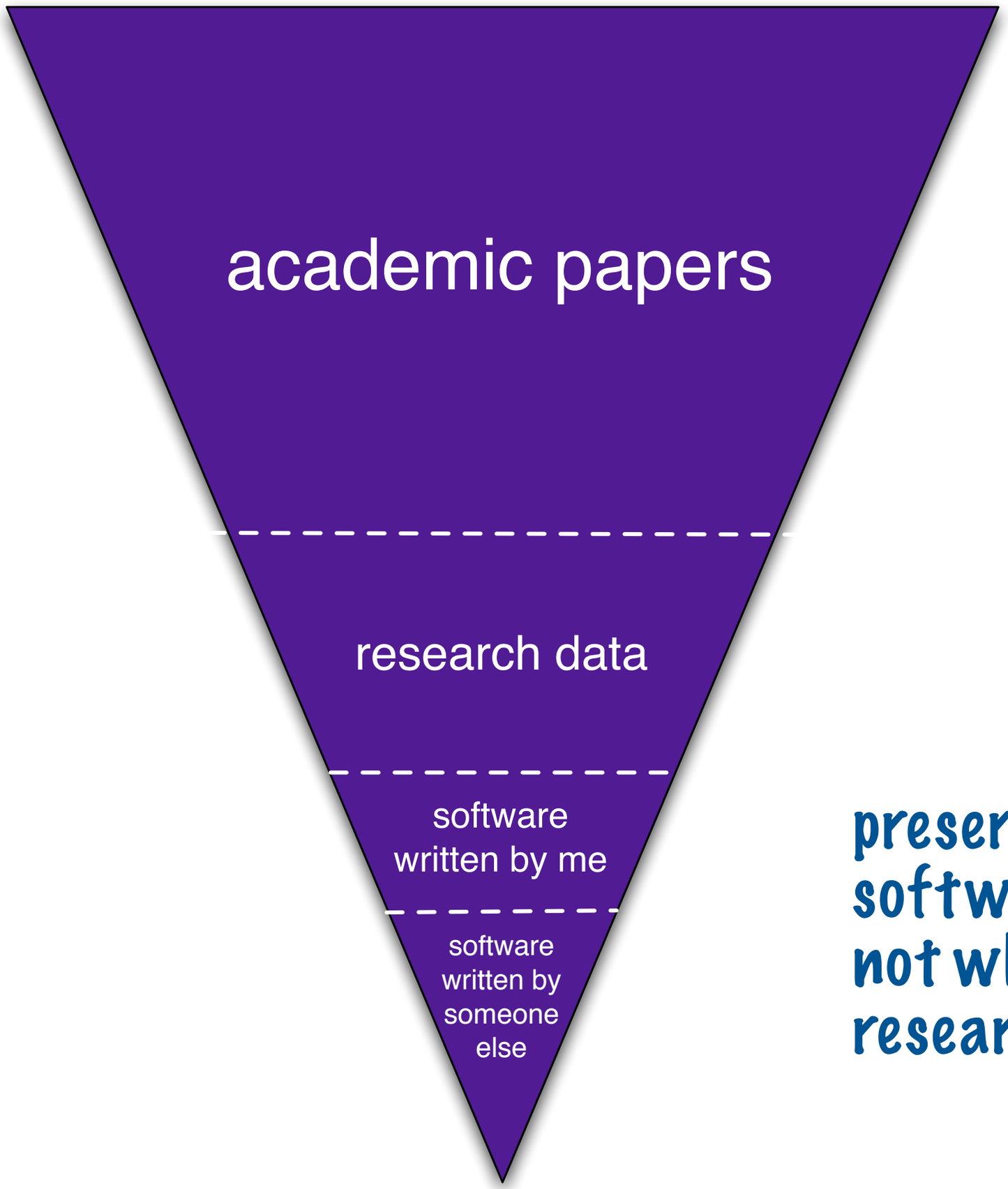
- complexity from inter-dependencies
- pretty much impossible to write software now without significant dependencies on libraries, OS features etc.
- approaches to mitigating this require commitment and understanding....
- ...professionalism?



production

- the people writing software to support e-infrastructure are sometimes not professional software developers
- where does the non-professionally produced software used to drive research come from?
  - answer ~= grad students!
- some of this then becomes tomorrow's e-infrastructure

the purple inverted triangle of vested interest



preserving software is just not what researchers do....

...or developers either

Software Preservation

- What is software preservation?
  - Storing a copy of a software package”
  - Enabling its retrieval in the future
  - Enabling its reconstruction in the future
  - Enabling its execution in the future

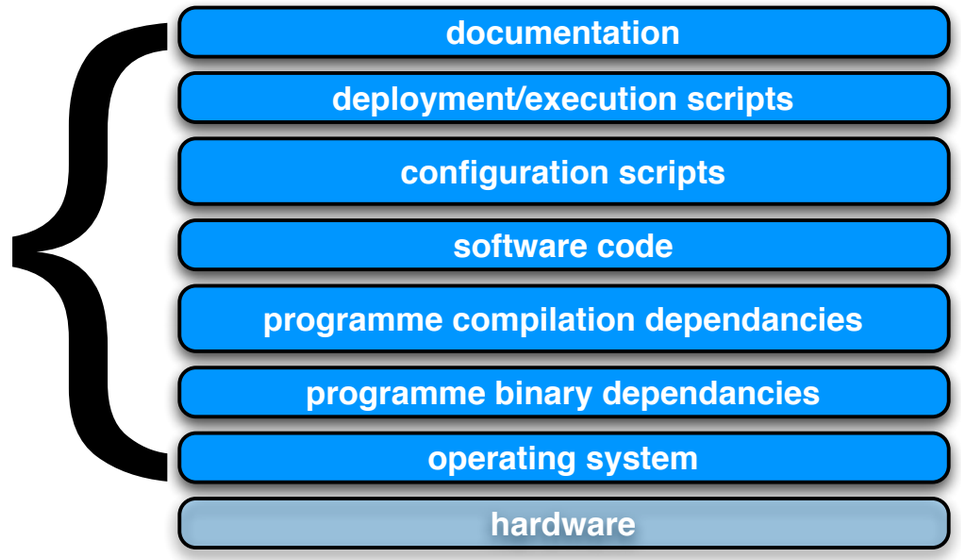
Not what most software developers and maintainers do.

Brian Matthews, *A Framework for the Significant Properties of Software*
http://www.dpconline.org/docs/events/080407sigpropsMatthews.pdf

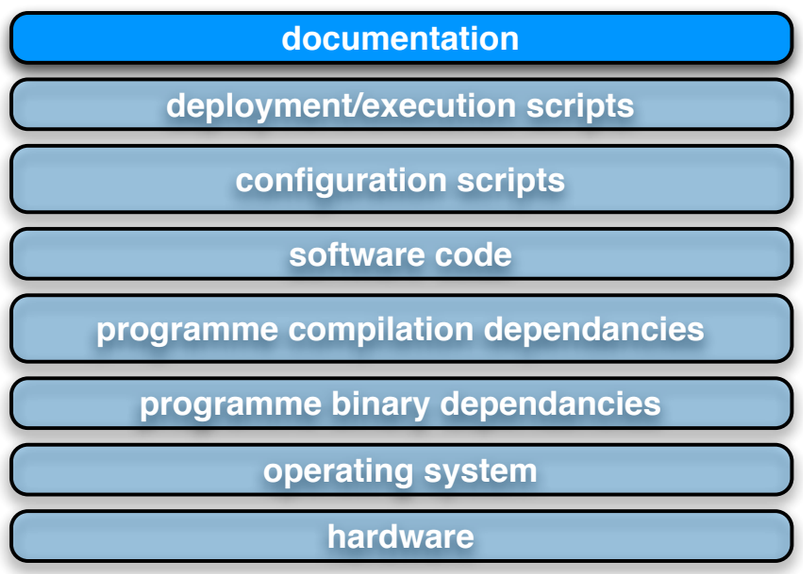
mitigating behaviour

preserving software

- two approaches based on going to either extreme of the dependency stack. Either:
- 1: bundle everything up as far as possible into one re-deployable ‘virtual machine’



- or:
- 2: abstract out the important algorithms, store as structured documentation, abandon the rest
  - “turn the software into data”



sustaining development

- better disciplined engineering?
- is software engineering an appropriate response?
- NSF argued about this
- backlash against ‘engineering’
- our CS departments also don’t often teach these skills!
- Greg Wilson and software *carpentry*
- artisan?
  - (which sounds archaic but may be nearer the truth)

intervention at a national level

more understanding

- the properties of software in terms of digital preservation
  - some good DCC and JISC-funded work in this area
  - applying this to the development process
- knowledge exchange between professional developers and researcher-developers
- business models for sustained open source development and maintenance
- how to recognise and then incubate potentially important software

insist on open-source?

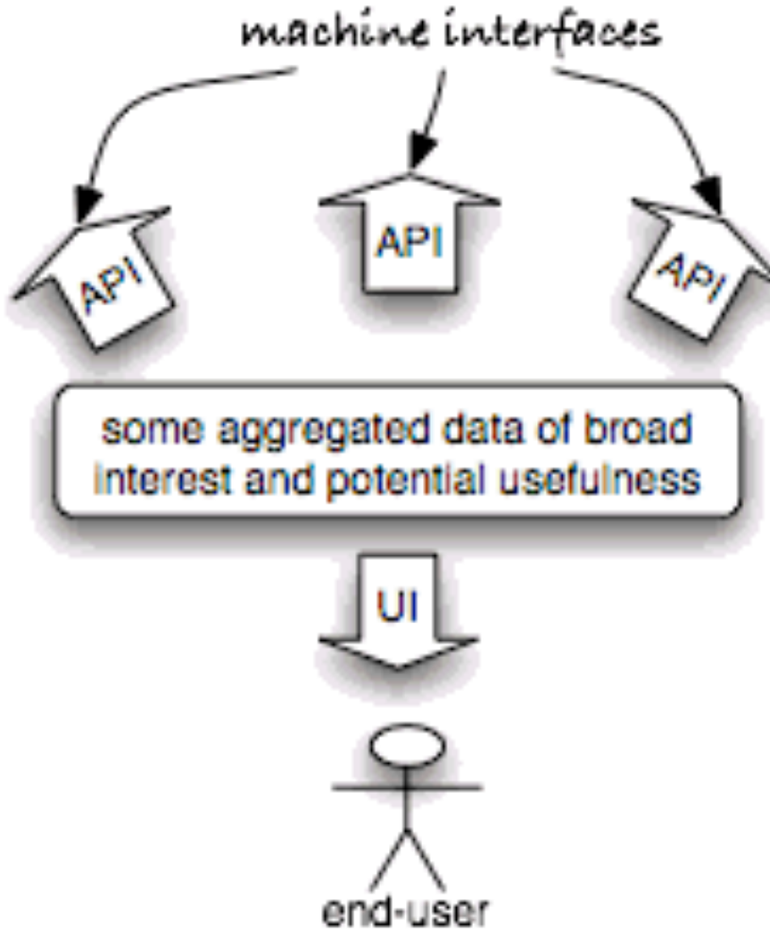
- well yes, but....
- open source is a potentially powerful approach to developing software
- it is not sufficient by itself
  - Sourceforge (and more recently Git) is a graveyard of opensource software code that no one cared to maintain
- open source would seem to be a sensible default, **when backed up with a sustainable development model**

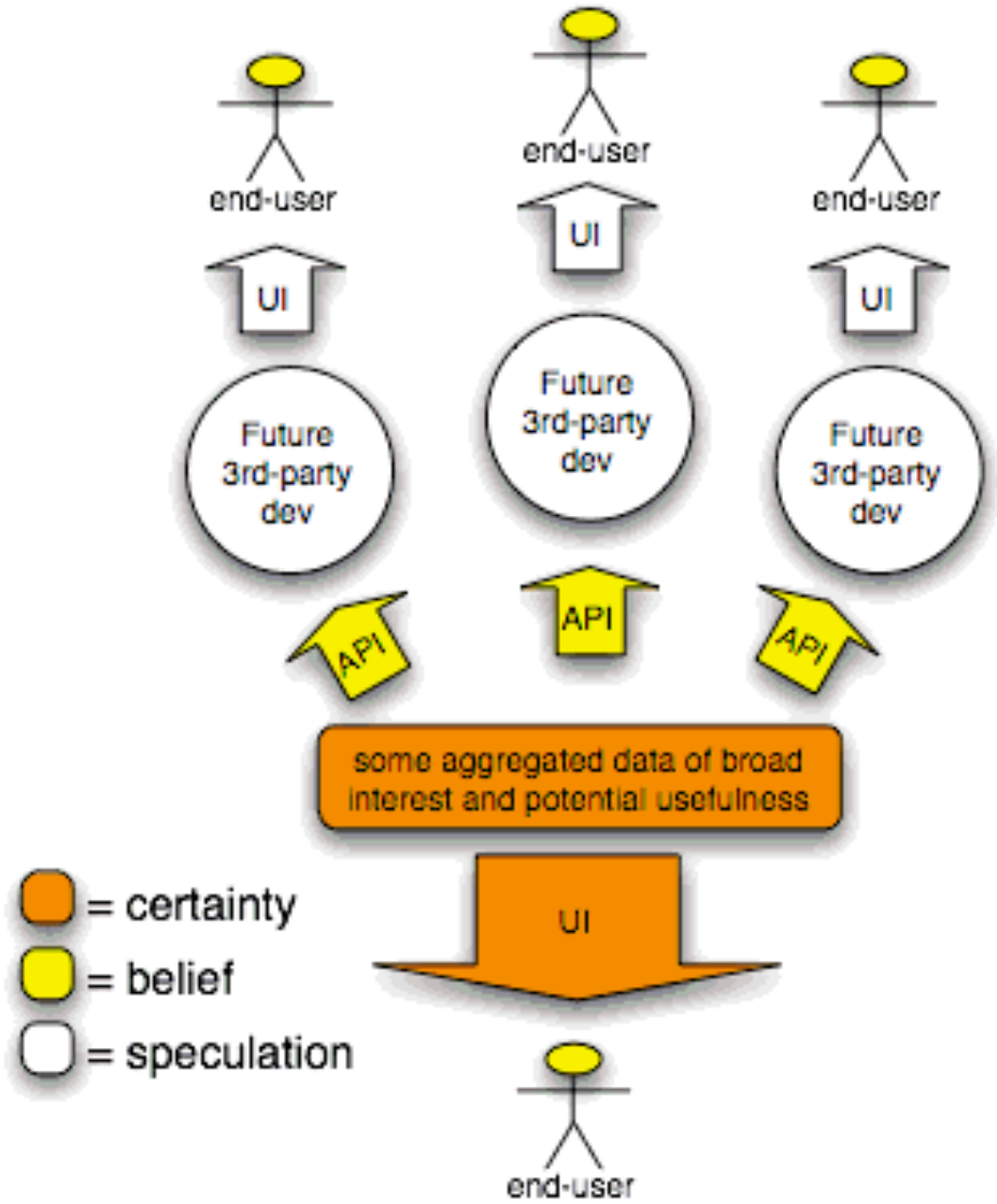
provide incubation?

- software which is ahead of its time needs to be looked after and kept alive before it is more widely adopted
- note - good software often is ahead of its time!
- Apache Foundation offer an interesting model
  - evidence of interest
  - evidence of growth of interest and commitment
  - no *explicit* need for running code to get through the incubation stage

document anti-patterns

- identify the **bad** practice

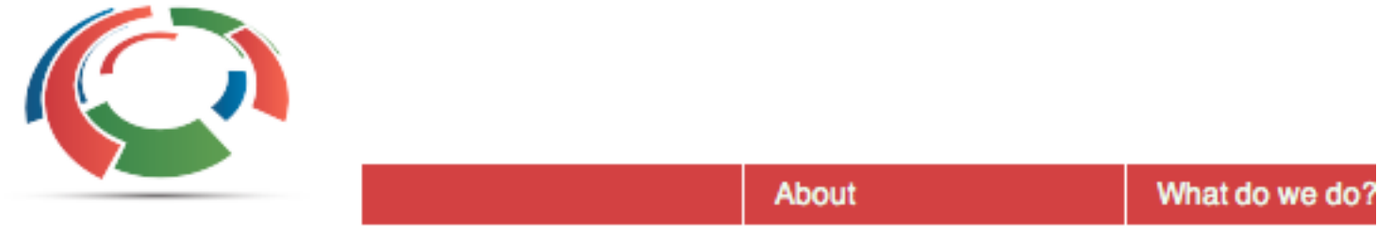




Paul Walk, An infrastructure service anti-pattern
http://blog.paulwalk.net/2009/12/07/an-infrastructure-service-anti-pattern/

provide training

- Software Sustainability Institute



DevCSI

Developer Community Supporting Innovation

- providing peer-peer training opportunities
  - £80K worth of training in two half days using community - peers training each other
- networking developers with each other and with researchers
- beginning to focus a little more on research
  - supported SSI Collaborations Workshop
- commercial sponsorship (e.g. Microsoft Research)

TransferSummit/UK

2011: Open Innovation Everywhere

Keble College, Oxford 5-8 September 2011

Two days, Three Tracks and a Gala Dinner: Endless Possibilities!

Center for Computational Thinking

Carnegie Mellon

tentative conclusions

- this space is **insufficiently understood** - to the point of occasional basic **confusion**
- software development is a **discipline**, like research, and like data-curation. Training and support are essential
- even more than with data, sustainability must be **addressed at the outset** of any development
- approaches to improving the sustainability of software in our sector may be found in addressing the **practice of production** rather than in the properties of the software itself