MANCHESTER 1824
The University of Manchester

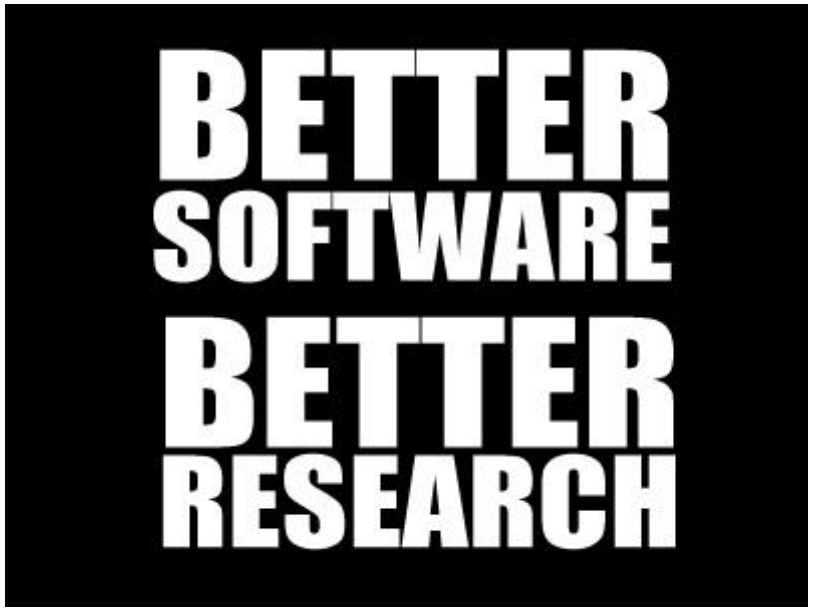
Software Engineering for Software Sustainability

Robert Haines

Head of Research Software Engineering

University of Manchester, UK

Workshop on Sustainable Software Sustainability (WOSSS19)



MANCHESTER 1824
The University of Manchester

State of the art?

I'm not going to teach you how to suck eggs…

"To try to tell or show someone more knowledgeable or experienced than oneself how to do something" - http://idioms.thefreedictionary.com/teach+grandmother+to+suck+eggs

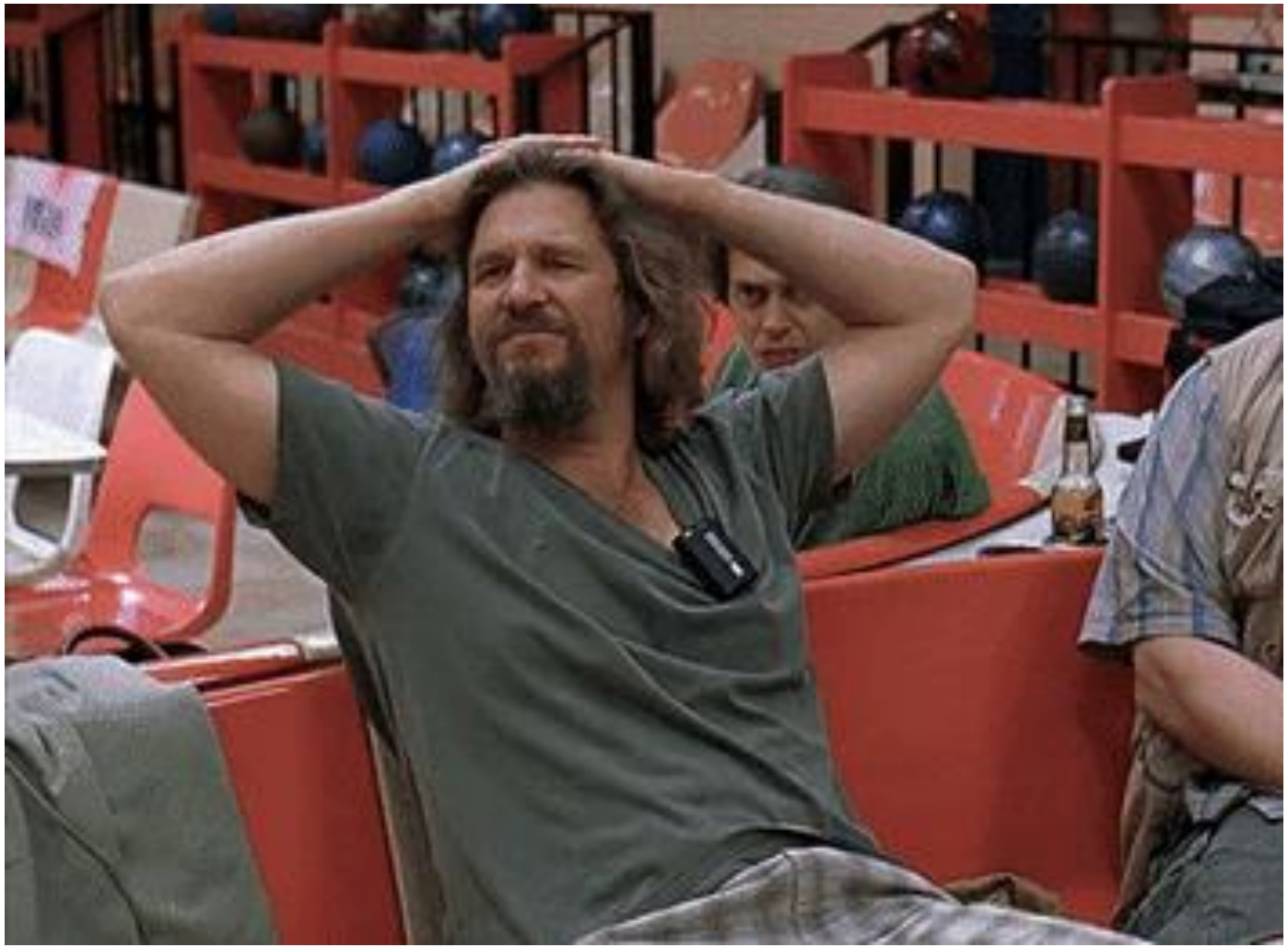
These are all useful things to know about:

- Version control; git-flow; trunk-based-development
- Clean, readable code; idiomatic code; coding standards
- Code review; pair programming
- Testing; code coverage; continuous integration; continuous delivery
- Technical debt; refactoring
- Documentation; tests as documentation; semantic versioning
- Use of standards; correct before optimized

- But they will only ever get us so far…

MANCHESTER 1824
The University of Manchester

Opinions? We've all got one (or two)…



MANCHESTER 1824
The University of Manchester

So what do RSEs think?

- We did an interview study with Research Software Engineers:

1. *From your point of view, what is sustainability in terms of software?*
   - (probe) *What are the attributes or features of the software that lead you to believe that it is sustainable?*

2. *Regarding the software you've developed: was sustainability a consideration?*
   - (probe) *If yes, at what point in time did it become a consideration? If no, why not?*

3. *Have you worked on any projects that were not sustainable?*
   - (probe) *Were there any consequences of it not being sustainable?*

MANCHESTER 1824
The University of Manchester

RSE sustainability perspectives

Intrinsic Sustainability

Of the software artefact itself…

Extrinsic Sustainability

Of the environment in which the software is developed and/or used…

MANCHESTER 1824
The University of Manchester

Intrinsic sustainability

- Documented
  - “I [spent] … months writing documents, analysing every single module … doing reverse documentation … just to demonstrate [that] this software was usable…”
- Testable
  - “…test automation and CI … that helps a lot with keeping it sustainable.”
- Readable
  - “…if [someone] finds my code and … the effort of learning to use my code is going to be more difficult than the actual benefit [they’d] probably throw it away and write [their] own stuff”
- Modular
  - “It turned out that the software was impossible for anyone to actually deploy in full, and it would only work if all the pieces were deployed. Funny, that didn’t work.”

MANCHESTER 1824
The University of Manchester

Intrinsic sustainability

- Standardized
  - “It’s important to [use] technologies that people generally understand, reusing as much as you can, so don’t write your own things, [when] there’s good solutions already.”
- Useful
  - “…it’s coupled to the software doing something useful, which either there isn’t an alternative for, or that it is much better in its niche than the alternatives.”
- Scalable
  - “… it’s also [got] to be usable long term, because if it’s just the simple cases … then as soon as they start using it in anger, a lot blows up because it doesn’t scale.”

MANCHESTER 1824
The University of Manchester

Extrinsic sustainability

- Openly available
  - “Usually I would look online in a repository for libraries…”
- Shared/co-owned
  - “Whether that community is composed of volunteers or people that are actually paying for your product it doesn’t matter. But basically you do need to have a community.”
- Resourced
  - “You see it in a lot of research, I mean the [REDACTED] stuff I did – [it’s] completely just gone. The minute I left it, still sitting on GitHub but no one even looked at it.”

MANCHESTER 1824
The University of Manchester

Extrinsic sustainability

- Actively maintained
  - “Physically the software lies there … you find software to do something, [you think] OK that looks good, and then you look – last updated three years ago. Most people won’t touch it.”
  - “So it’s about having this kind of momentum to the project, so that it keeps moving. That you have further development, even if you have maintenance mode—that is, not many new buttons being added—but at least there is someone [keeping] it alive.”
  - “I guess it is around sort of maintainability, the fact that codebases, if you don’t sort of keep them up-to-date and keep developing them, they tend to go stale.”

MANCHESTER 1824
The University of Manchester

Extrinsic sustainability

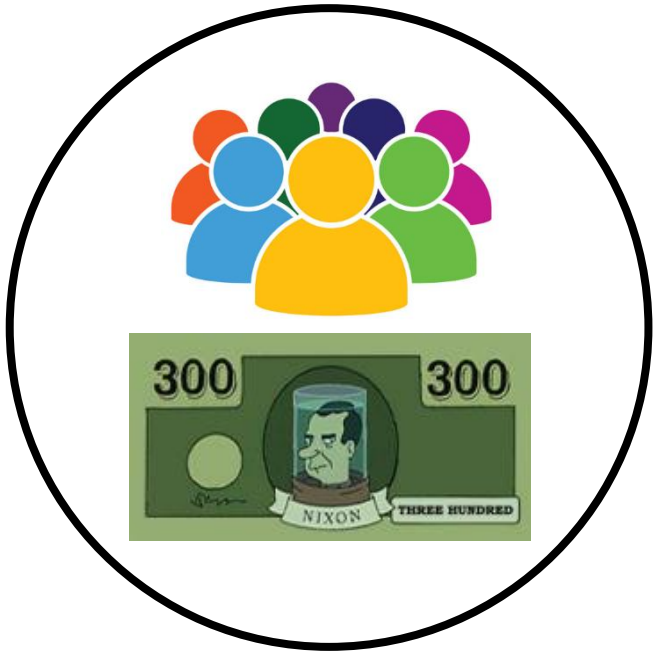
- Independent from infrastructure
  - “[REDACTED] did most of that, and it’s one of these things that will probably stay alive for as long as the server that it’s on stays alive, and if that server crashes they will probably not bother rebuilding it into another machine.”
- Supported
  - “So this tends to mean things like [email] support, or automated tools of various kinds.”
- Version control
  - “I would say sustainability is partly about you know this … re-usability of research … to keep track of, your developments because you want to see exactly which version of the code produced what results that went into some application.”

MANCHESTER 1824
The University of Manchester

Extrinsics distilled: sustainable != sustained

- Those intrinsic factors help make software *sustainable*
- But they aren't enough to actually *sustain* the software

- Focus on these extrinsic factors:
  - Resourced
  - Actively maintained
  - Supported
- In effect: Software is *sustained* while it is *sustained*!!



- We can't write fabulous software and sit back and watch it sustain itself

MANCHESTER 1824
The University of Manchester

Software == Puppies

"Open source is free like a puppy is free"

– Scott McNealy, 2005

https://www.zdnet.com/article/open-source-is-free-like-a-puppy-is-free-says-sun-boss-3039202713/

"Tell your funders and PI's that software is like a puppy. Puppies aren't free, you've got to feed them, they will get old, and they will die.

Funders don't understand software – but they know about puppies."

– Carole Goble, 2019

https://www.biosustain.dtu.dk/nyhedsbase/Nyhed?id={EE8FDD3C-E814-492E-A9BE-1D79A1283CBC}

"Puppies sometimes get sick. Sick enough that one has to make a difficult decision."

– Me, today

https://doi.org/10.5281/zenodo.2649774

MANCHESTER 1824
The University of Manchester

Preventing new legacy

- Is it all really just about money/funding?
- Was our cohort of RSEs on to something with the intrinsic/extrinsic factors?
  - Which are the most important? Can we prioritize a subset?
- Given that we know how to do all the intrinsic factors *and it's not enough*, how do we get better at the extrinsic factors?
  - Which extrinsic factors should we focus on as a priority?
- Should we be quicker to stop sustaining software?
  - Is there a point at which we accept that we have sustained something as long as possible, but now it's time to start again?
  - How do we spot that point and plan for it properly?
- What are the levels at which we should aim to sustain software?
  - Project – end-to-end processes that are repeatable, but may be less flexible?
  - Code – reusable pieces that might not do much on their own but can be combined?
  - Can we know which is appropriate early enough to do so?
- Can standardization be the driver for sustainability?

MANCHESTER 1824
The University of Manchester

References/Acknowledgments

Mário Rosado de Souza, Robert Haines, Markel Vigo, Caroline Jay; *What Makes Research Software Sustainable? An Interview Study With Research Software Engineers.*; accepted at 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) @ ICSE 2019; https://arxiv.org/abs/1903.06039

Thanks – *people I've talked to about (R)SE regularly/recently*:

- Caroline Jay, Carole Goble, Dan Katz, Jeff Carver, Sandra Gesing, Ian Cottam, James Howison, Simon Hettrick, Neil Chue Hong, Jeremy Cohen