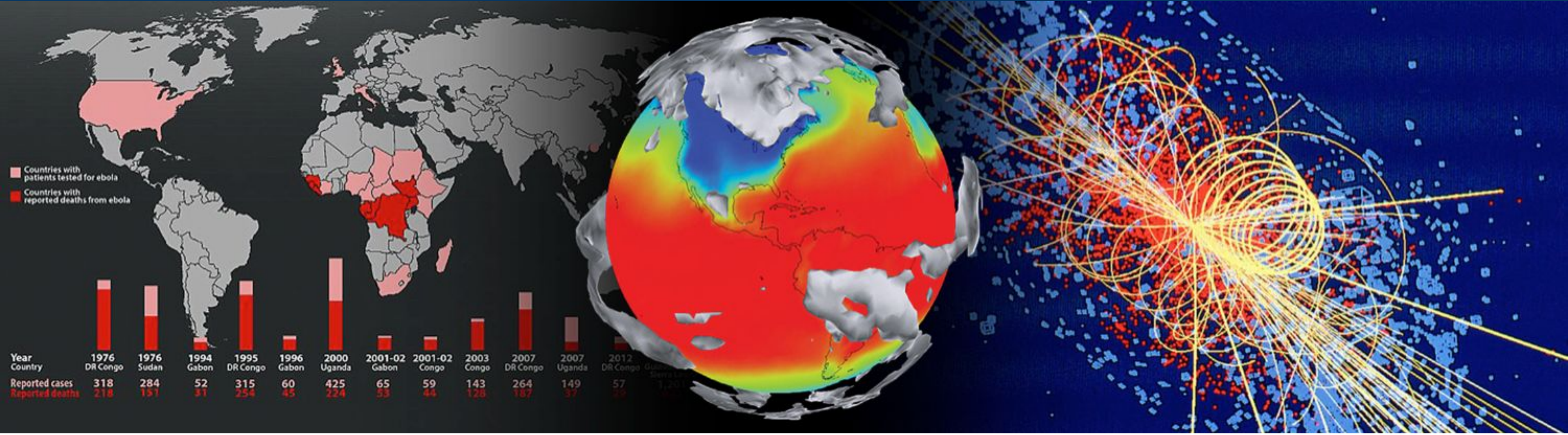
EPSRC and MRC Systems Approaches to Biomedical Science CDT

Introduction to Software Engineering Concepts

Steve Crouch
Software Sustainability Institute
s.crouch@software.ac.uk

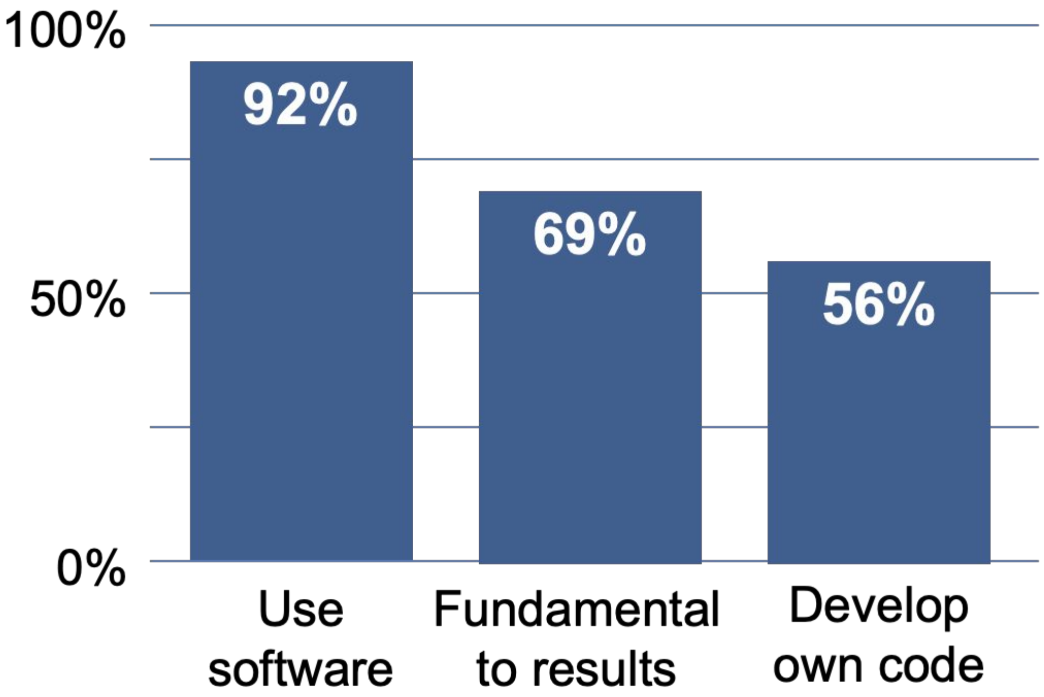
14th October 2019

Modern research is impossible without software



From thrown-together scripts, through an abundance of complex spreadsheets, to the millions of lines of code behind large-scale infrastructure, there are few areas where software does not play a fundamental part in research

Why should we care about software?



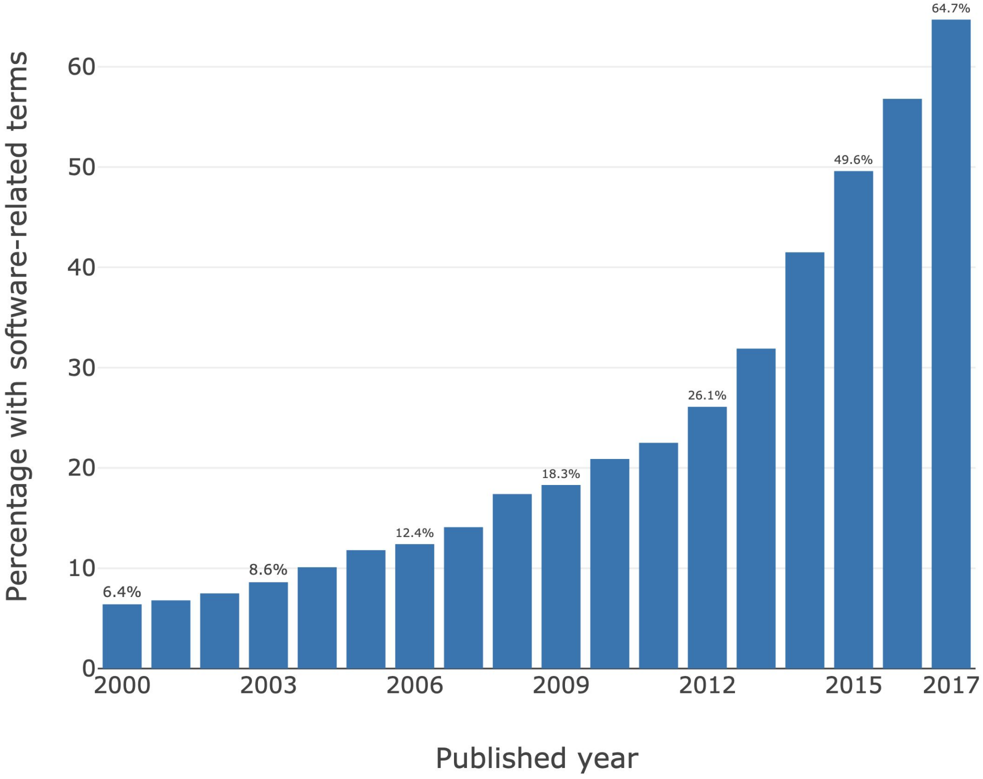
SSI survey of researchers, 2014[1]

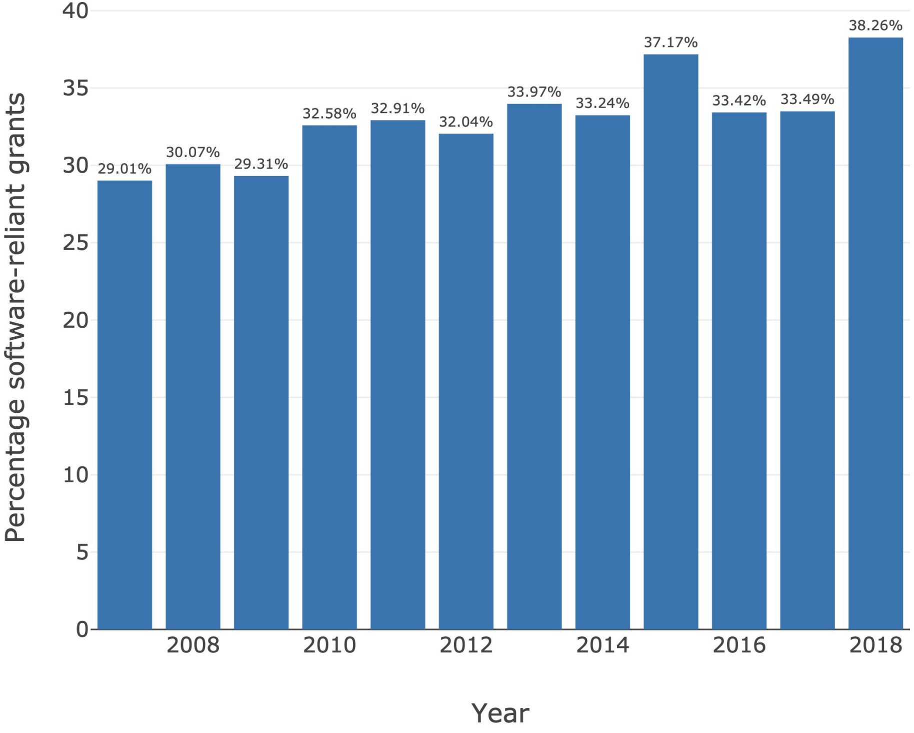
15 Russell Group Universities

Their software use and background

417 respondents

Why should we care about software?

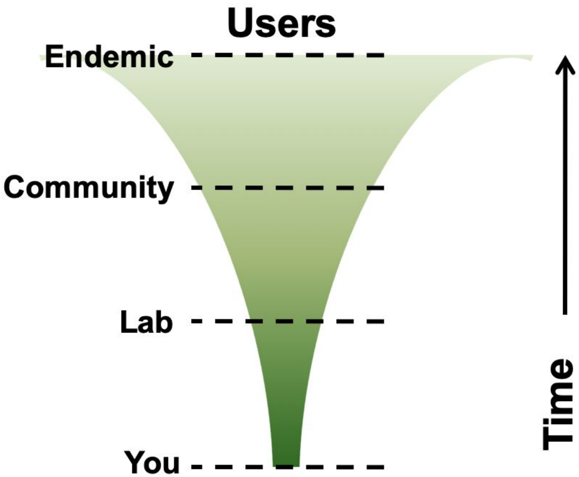




[2]

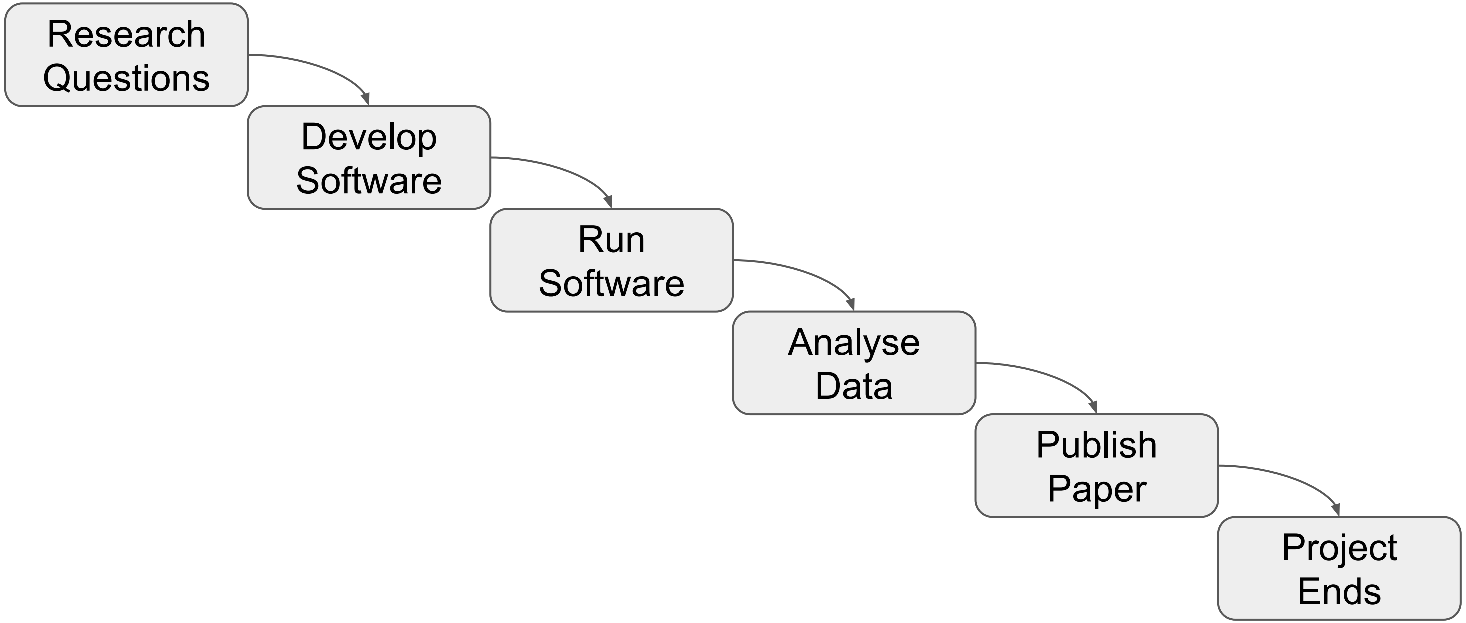
The software you write is important!

- Software inherently contains *value*
  - Produces results, contains lessons learnt, effort
- Difficult to gauge to what extent it might be used in the future
  - By who?
  - Which parts?
  - Which projects?
  - Reproducibility – from publications!



Can it/should it be reusable by others?
...including yourself?

A typical research software lifecycle



Research Questions

Develop Software

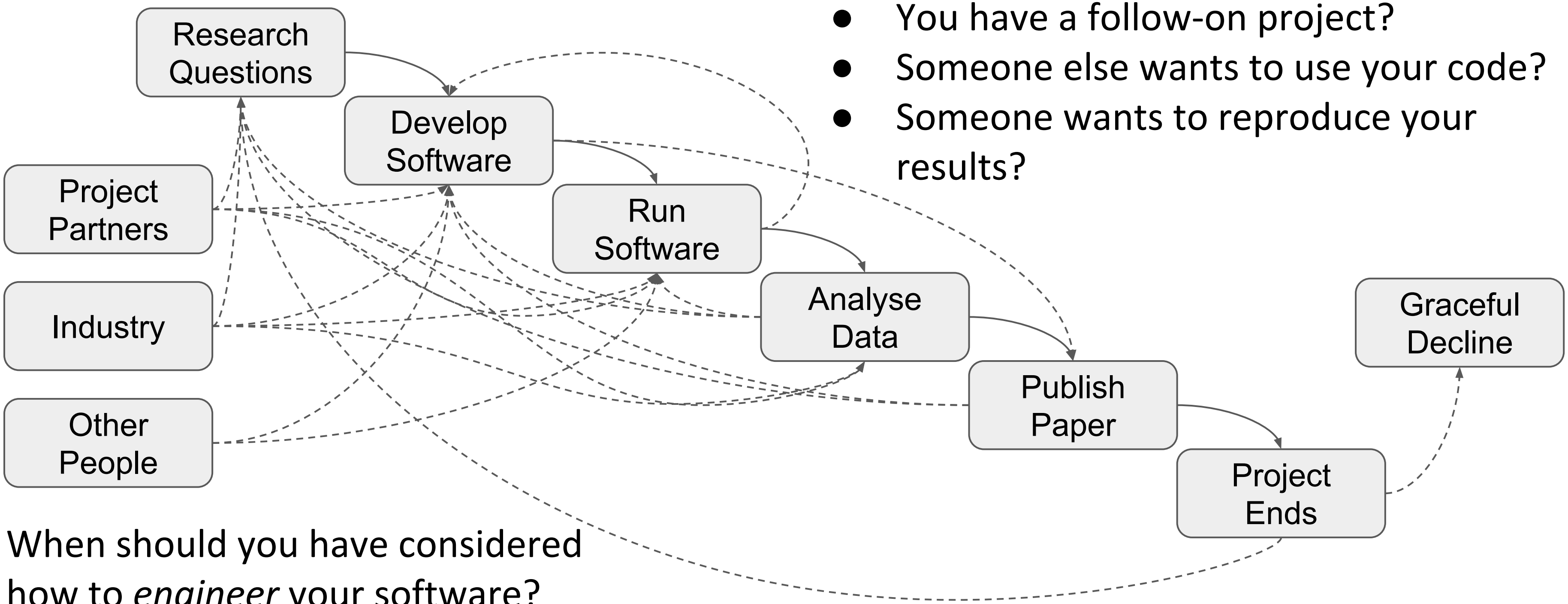
Run Software

Analyse Data

Publish Paper

Project Ends

In reality...



What happens when…

- You have a follow-on project?
- Someone else wants to use your code?
- Someone wants to reproduce your results?

When should you have considered how to *engineer* your software?

What could go wrong?

- Ariane 5
- $7B d... $500 ... rocket
- Used Arian 4

EXCEPTION HANDLER DISABLED

de info → **64-bit FP converted to 16-bit signed integer**

Programming vs Engineering

Programming / Coding

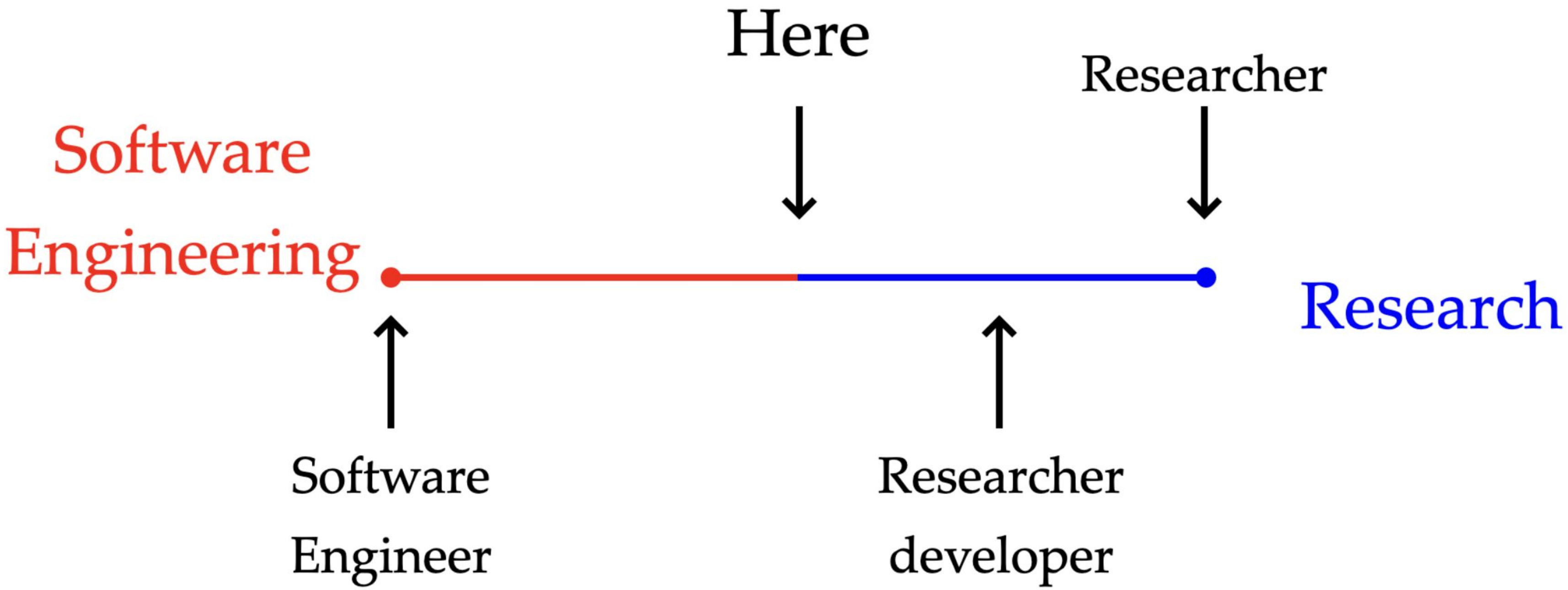
- Focus is on one aspect of software development
- Writes software for themselves
- Mostly an individual activity
- Writes software to fulfil research goals (ideally from a design)

Engineering

- Considers the *lifecycle* of software
- Writes software for *stakeholders*
- Takes *team ethic* into account
- Applies a *process* to understanding, designing, building, releasing, and maintaining software

> *"Programmers tend to start coding right away. Sometimes this works."*
> - Eric Larsen, 2018

Where are you?



Beyond building a 'sequence of instructions'

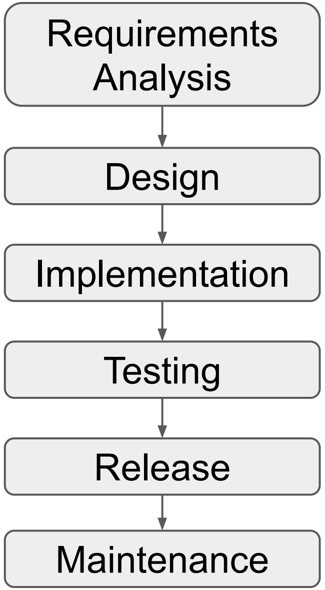
Software is far more than that…

- **Outcome of a *development process***

But also...

- Architecture
- Implementation of algorithms
- Data model
- Programming paradigm
- Documentation
- *Best practices* and *conventions ...*

Waterfall Model



Requirements Analysis

Design

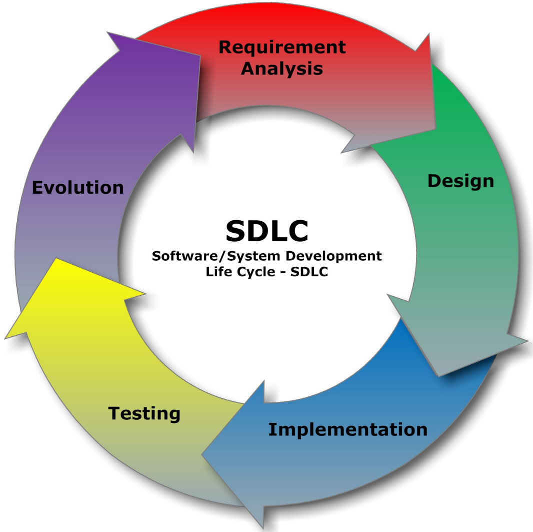
Implementation

Testing

Release

Maintenance

Agile Model

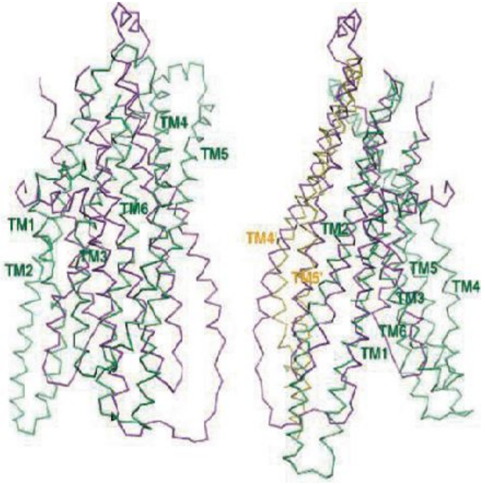


Testing

- Humans are fallible! Our software will contain defects
  - In requirements, design, as well as code

- Validation: are we building the right product?
- Verification: are we building the product right?
  - Manual testing, unit testing, automated testing, code reviews

- Highly-cited papers published on multidrug resistance transporters between 2001 - 2010
- Results couldn't be reproduced - 5 retractions
- Caused by error in an *internal software utility*
  - Flipped two columns of data, inverting electron-density map used to derive protein structure

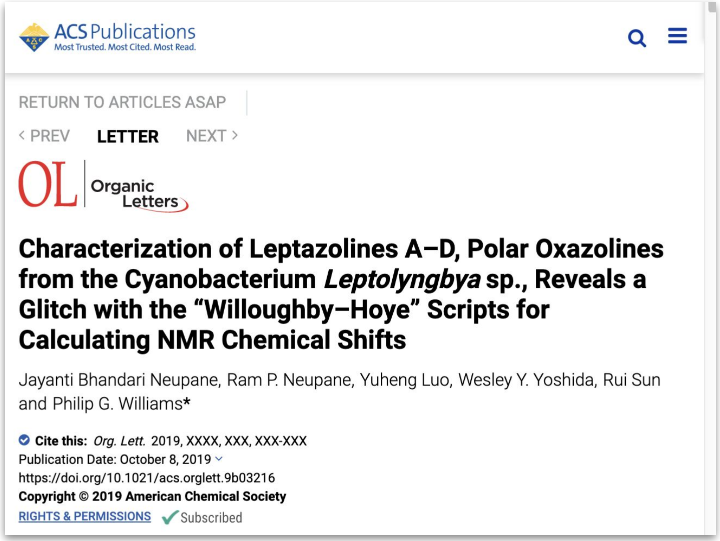


"I didn't question it then. Obviously now I check it all the time."
- Geoffrey Chang[3]

… STOP PRESS ...

… Density functional theory nuclear magnetic resonance calculations established the relative configurations of 1 and 2 and revealed that **the calculated shifts depended on the operating system when using the "Willoughby–Hoye" Python scripts to streamline the processing of the output files, a previously unrecognized flaw that could lead to incorrect conclusions**.

- Just *last week*
- Due to *different sorting of file names* on different operating systems



Characterization of Leptazolines A–D, Polar Oxazolines from the Cyanobacterium *Leptolyngbya* sp., Reveals a Glitch with the "Willoughby–Hoye" Scripts for Calculating NMR Chemical Shifts

Jayanti Bhandari Neupane, Ram P. Neupane, Yuheng Luo, Wesley Y. Yoshida, Rui Sun and Philip G. Williams*

Cite this: *Org. Lett.* 2019, XXXX, XXX, XXX-XXX
Publication Date: October 8, 2019
https://doi.org/10.1021/acs.orglett.9b03216
Copyright © 2019 American Chemical Society
RIGHTS & PERMISSIONS ✓ Subscribed

Organic Letters, October 8 2019
https://doi.org/10.1021/acs.orglett.9b03216

Optimisation

*“Three orders of magnitude in **machine speed** and three orders of magnitude in **algorithmic speed** add up to six orders of magnitude in solving power. A model that might have taken a year to solve 10 years ago can now solve in less than 30 seconds.”*

– Robert Bixby, review of linear programming solvers from 1987-2002

- Faster code, faster results!
- Understanding trade-offs
  - Maintainability, accuracy
- When & where to optimise?
  - 80/20 rule, code profiling

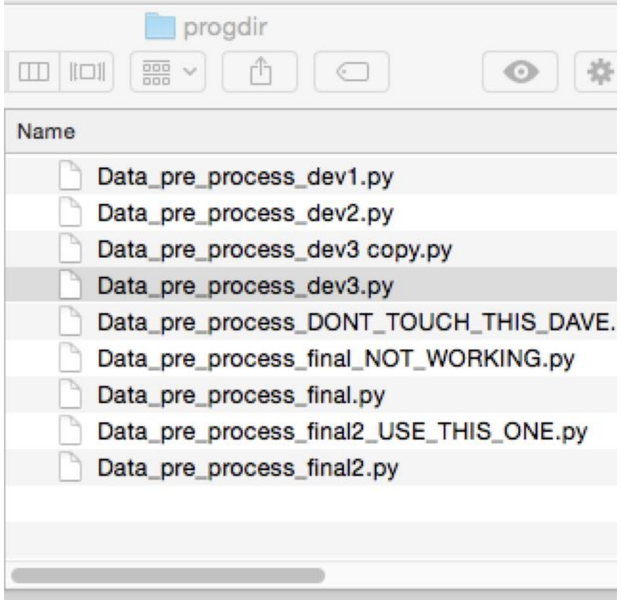
Amdahl's Law:

Time to result = Develop Time (D) + Time to Run (R)

As effort is put into reducing R, overall time required to get new result is dominated by writing, testing, maintaining, installing, configuring software.

Code management & collaboration

- *Version control* provides a full history of your project's software and other assets
- Makes for easy:
  - Backups
  - Collaboration
  - Recovering from dead-ends
- What should be in version control?
  - Code, documentation, tests, test data, analysis scripts
  - Reports, papers, etc.
- Packaging and deployment



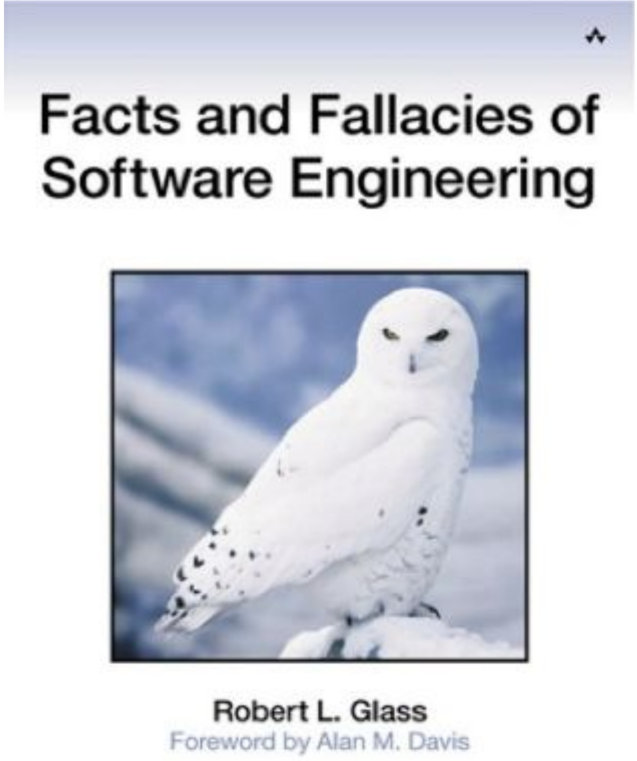
Data_pre_process_dev1.py
Data_pre_process_dev2.py
Data_pre_process_dev3 copy.py
Data_pre_process_dev3.py
Data_pre_process_DONT_TOUCH_THIS_DAVE.
Data_pre_process_final_NOT_WORKING.py
Data_pre_process_final.py
Data_pre_process_final2_USE_THIS_ONE.py
Data_pre_process_final2.py

> *"If you're not using version control, whatever else you might be doing with a computer, it's not science."*
>
> - Greg Wilson, SWC

Other key points

- These skills will **save you time**
- Always assume others will use and develop your software
- Be clear on requirements and assume they will change
- Funders are increasingly expecting software outputs to be sustainable and reusable

More on software engineering



Facts and Fallacies of Software Engineering

Robert L. Glass

Foreword by Alan M. Davis

Robert L Glass, Addison-Wesley Professional

Group projects

1. Mathematical model to quantify pharmacokinetics & pharmacodynamics [Roche]
   - Existing PK-PD solutions difficult to use - a user-friendly interface is required
   - Design and develop fast ODE solving and robust parameter estimation/inference
   - To promote PK-PD modelling to wider pharma community and its wider application

2. MRI brain segmentation for elderly neurodegenerative disease [GE Healthcare]
   - Need for accurate and robust MRI segmentation tool for problematic brain regions
   - Generalisable methodology to perform well with variance in MRI acquisition parameters
   - Should be available/usable by broad community & comparable in accuracy and computation time to commonly available methods

3. Expansion/improvement of Fragalysis for early stage drug discovery [Diamond]
   - Addition of new Fragalysis algorithms and their integration into HPC/cloud infrastructure
   - Python API needed to allow users to access underlying algorithms in open source fashion
   - Documentation, tutorials, and improved UI design required

References

[1] "It's impossible to conduct research without software, say 7 out of 10 UK researchers”, http://www.software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers

[2] "An investigation of the funding invested into software-reliant research", https://github.com/softwaresaved/software_in_grants_GTR

[3] "Retractions unsettle structural bio", https://www.the-scientist.com/daily-news/retractions-unsettle-structural-bio-46891

Need help?

Say hi to your neighbours!

Need help?

Sticky notes