Managing Academic Software Development
Dr Sam Mangham

ORCID: 0000-0001-7511-5652
Who Am I

- Senior RSE @ University of Southampton
- Trustee @ Society of Research Software Engineering
- RSE @ Software Sustainability Institute
- Generalist, interdisciplinary RSE, training, community
Background

- PhD in Astrophysics
  - HPC monte carlo radiation transfer code for supermassive black holes
- Neutronics @ Culham Centre for Fusion Energy
  - HPC monte carlo radiation transfer code for fusion
- Both large legacy HPC codes!

Mangham et al, 2019, ESO/M. Kornmesser
Why?
Enterprise

- Often large teams
- Formal training
- Formal project management frameworks & staff
- Software is the product

Academic

- Small/single teams
- Large numbers of loose collaborators
- Limited training
- Ad-hoc management (by other researchers) or self-management
- Papers are the product

Research Institutes

- Somewhere in-between
- Vary with scale, focus, discipline
Outline

- Development
- Usage
- Publication
Managing Development
Project Boards

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

- Break a project into components
- Subdivide as you go!
- Track progress publicly
Project Boards

- Document process on tasks
  - GitHub/GitLab etc. let you turn issues into lab books
- BUS FACTOR
  - Collaboration
  - Future You is a collaborator
  - Knowledge decays quickly
Prioritisation

- Time estimates
- MoSCoW

<table>
<thead>
<tr>
<th>Must</th>
<th>Should</th>
<th>Could</th>
<th>Won't</th>
</tr>
</thead>
<tbody>
<tr>
<td>(60%)</td>
<td>(20%)</td>
<td>(20%)</td>
<td></td>
</tr>
</tbody>
</table>

- Consider and revise!
Prioritisation

• Won'ts aren't forever
• Typical won'ts
  - Future research avenues
  - Features you don't need right now
  - Bugs that don't stop work
• Acknowledge them publicly
  - Help others plan around you
• Leave time for testing & documentation!
Version Control

- Protection against disaster
- Test and verify changes are *intended*
- Avoid having to rerun entire papers' worth of analysis to avoid version mismatches
Branching Workflows

• New branches for new features
  ▪ Link branches to tasks
  ▪ Easy to parallelise work
  ▪ Easy to switch to working on another feature

• Regularly merge branches back to development!
  ▪ Otherwise each developer ends up with a divergent version

• Review pull requests
Write Sustainable Code

• Proactively avoid technical debt
• Share and collaborate more easily
  ▪ No code worth writing is disposable!
• Write for collaborators and community
• Can't reproduce results if the code isn't sustainable
  ▪ HPC-BP talk on this
Write Readable Code

- Easier onboarding
- Follow community standards
  - E.g. **PEP 8** for Python
    - **pylint**, **flake8**
  - E.g. **C++ Core Guidelines**, **LLVM** for C++
    - **clang-tidy**
- Pick a style and stick to it!
Write Readable Code

- Descriptive variable names
  - Minimise potential for collision!
  - Not 'c', 'e', 'hb'
- Code completion & IDEs
  - CLion, PyCharm, Visual Studio Code
- Modular code
  - You **will have to** refactor!
  - You can't predict your code's future
Document Your Code

- Bus factor again
- Optionally: Document then design
  - Test-driven development
- Automated tools
  - Sphinx
  - Doxygen
- Automatic hosting
  - ReadTheDocs for Sphinx
  - CodeDocs.xyz for Doxygen
- Call graph generation
- docs-like-code

Call graph, Christina Jacob, 2020
Test-Driven Development

- Continuous Integration
- Many more detailed talks on this!
Questions?
Managing Usage
Public Documentation

- Easy onboarding
- Quick reference for yourself
- Online documentation platforms
  - ReadTheDocs again
  - GitHub Pages
  - GitHub wikis
Public Issues

- Facilitate problem solving
  - Searchable if possible!
- Own up to the code’s limitations
  - Benefits far outweigh embarrassment!
- Issues are a dialogue with your users
  - Even non-issues!
  - Structure it with issue templates
Questions?
Managing Release
Release Your Software

- Majority of research relies on software
- Much is paperware
- Public release is required for reproducibility!


Randall Munroe, XKCD - Dependency
Structured Releases

- GitHub Releases
- Citation.CFF
- Zenodo
  - Provides citeable DOIs
- Include *all* info
  - Library versions
  - Compiler versions
  - Compiler flags
Software Licenses

- Previous HPC-BP talk
- No License
  - Automatically copyrighted
  - No rights for others to do *anything*
- Open-Source
  - Copyleft (e.g. GPL3)
  - Permissive (e.g. MIT)
- Proprietary License
  - Lawyers are expensive
- choosealicense
Thank you for your time

⭐ s.mangham@soton.ac.uk / rsg.soton.ac.uk ⭐

s.mangham@society-rse.org / society-rse.org
s.mangham@software.ac.uk / software.ac.uk

Advertisements


rsg.southampton.ac.uk/jobs

Any questions?