Problems Faced in Programming for Research

Causes, Effects and Some Strategies

(v0.1.0)

Cristian Dinu

March 2023

Introduction

- Scientific programming is still programming
- The main issues of general programming still apply
    - Unclear requirements
    - Tough stakeholder management
    - Bad planning
- But I will talk about problems specific to programming for research in my non-Large Hadron Collider experience

Both research and coding are hard

So that one individual can only do well one or the other

- Effects:
  - It difficult to collect requirements and understand the problem domain
  - (civilised) Friction and frustration between the researcher and the coder.
  - Researchers stuck with old tools and methods (Excels with 100 sheets and colour codes)
- Strategies: "Embed" the developer in the team. Iterate fast, show and validate early, using friendly formats (Jupyter, Excel), avoid computer/programming slang, assume educator role

Research projects are, well... research

Requirements, goals, methods, change continuously as they progress. Ideas come up, real life is muddy, data is unstructured.

- Effects: Nearly impossible to estimate and meet deadlines, frustration as a lot of time needs to be spent on menial task
- Strategies: Prepare "plan B"-s, embrace changes, accept this is life, be happy with it, as it means progress.

The "Paper" is more important than the code

The incentives and priorities are conflicting. Recognition not always given.

- Effects: Quick and dirty, duct-tape coding, neglected documentation, testing, maintainability, security and ethics reviews, and reproducibility
- Strategies: Recognise software contributions, emphasise code quality, make code as important as the paper

People who code seem ”in transit” to better roles or self-taught. . .

I saw most of the code is done ”for free” by masters or PhD students or enthusiastic self-taught young academics.

- Effects: Lack of good practices, inconsistent code quality, slow skills improvement. Lots of technical debt
- Strategies: Educate beyond coding – e.g. source control, documentation, and establish some standards (e.g. everything goes on departments' GitHub account, no Excel allowed)

. . . because good developers are expensive and industry pays well

Also, it's unfair to pay a developer three times as much as a researcher, just because the IT pays well.

- Effects: Difficulty attracting and retaining skilled engineers, reliance on inexperienced developers
- Strategies: Allow people to be paid directly from grants, or projects with the industry.

Ethics, Security, and Privacy are a second thought

In research programming, ensuring ethical conduct, maintaining security, and respecting privacy are crucial, but they are often overlooked or neglected.

- Effects:
    - Inadequate protection of sensitive data or private information
    - Unintended consequences of research findings or technology use
    - Misuse of research results or tools for malicious purposes
- Strategies: educate (tell scary stories), create checklists so that important issues are not missed, check where code and data are stored and deployed.

More and more blackboxes (libraries, AI models)

- Effects: Reduced transparency, harder debugging, validation issues, over-reliance on external libraries
- Strategies: Encourage open-source development, require explainable AI, prioritize validation and testing

Conclusions

- Programming for research shares difficult problems with ”general” programming
- Competition with the industry is unfair [compared to academia]
- Hopefully a ”software is an investment not an expense” mindset will be adopted by the majority of people involved in research
- But keep a WindowsXP machine handy. You might need it for next project.