Research Software Engineering Meetups
Introduction & Tooling

2023/10/13

Overview

- Schedule
- Introduction
- Tooling/version control
- AI/LLMs
- Round up

Schedule (provisional)

Date	Location	Rough Topic	Status
16th October	34-E-3180	Introduction & tooling	In development
23rd October	34-E-3180	Sharing, publication, and collaboration	(title only)
30th October	34-E-3180	(topic TBD: based on feedback)	
6th November	34-E-3180	(topic TBD: based on feedback)	
13th November	34-E-3180	(topic TBD: based on feedback)	
20th November	34-E-3180	(topic TBD: based on feedback)	

source: *https://github.com/adamkewley/rse-meetups/*

Intro: Meetup format

- Roughly based on software meetups I attended in Cambridge UK (e.g. link)
- Geared to be casual and typically has a low production bodgeitet ;)
  - So that the content can be changed cheaply *during* the series
  - And there isn't as much pressure to go through *all* of the content/ideas (we can skip things, defer until later, etc. based on the audience)
- Typically includes the organizer providing shorter present ational bits interspersed with live demos, discussions, interactive parts, etc.

Intro: You

- You knew the title
- You knew the rough topic schedule
- You are here
- Why?

My Supervisor told me (we've all been there ;))	My grant agency told me (to practice "Open Science", but I have no idea what that actually means)	You told me (that I needed to turn up to make up the numbers)
IT KEEPS SAYING `package opensim not found` EVEN THOUGH OPENSIM IS CLEARLY INSTALLED (I'm already writing software but I keep running into platform-level technical issues)	`switch (choice) { case "alg1": return alg1(arg1); case "alg2": return alg2(arg1, arg2); default: throw "error lol"; }` (I'm already writing software but I keep running into design-level issues)	What on earth does `get_published_result(xy)` in $ {previous_code}` actually do? (I'm have to modify/use existing software)
I don't feel like I understand what's actually going on (I want to know how computers work)	I want to (know how to) engineer software projects	I want to write software in the future, and this seemed appropriate 6/13

Topic 1: Tooling

- Topic #1?
- Online platforms vs. text editors vs. IDEs
- Features to value:
  - Easy to install
  - Easy to setup with plugins/configuration etc.
  - Has a configurable button/hotkey for running your code (even if “running your code” involves things like running associated scripts first, etc.)
  - Has in-built support for (e.g.) Ctrl+Clicking functions to see their implementation
  - Has in-built support for debugging, and the debugger has support for conditional breakpoints and watches
  - (if using an interpreted language): REPL support

Interactive: Use Tools to...

- Get: https://github.com/adamkewley/rse-meetups
- Open **1_IntroAndTooling/a_UsingToolingOnTrivialCodebase**
- (debug) setting up up environment
- Run it
- Debug it
- Understand it
- Fix it
- (version control) see what we changed

Topic 1b: AI and LLMs

- They're tools.
- If you're already using them, then you probably already know where this is going
- If you aren't already using them, you should at least give it a go
- Uses in software development (**live demo**):
  - "Could you write a script that…"
  - "Can you explain what this code is doing…"
  - "Could you provide constructive feedback on this code…"
  - "What are the benefits of $library_a over $library_b…"

Topic 1b: GitHub Copilot

- It’s an LLM inside your code editor
- (**quick live demo**)

Roundup

<u>What was covered:</u>

- General format for the sessions
- Common themes in research software engineering
- General tooling considerations, how to use some features of tools
- AI/LLMs/ChatGPT/GitHub Copilot

<u>Where’d it fall short (should we cover it in a future session)?</u>

- Features X or Y of the tools presented?
- Other common development tools (trello, issue tracking, profilers, etc.)?

Next Time (up for debate)

> ℹ️ **Note:** This list is just to give you an idea (it's very very provisional).

Topic	Description
Tools	IDEs, text editors, REPL, command line, basic git usage, LLMs (ChatGPT/GH Copilot)
Sharing, Publication, and Collaboration	Shared drives, GitHub, GitLab, writing a README, managing dependencies, releasing, publishing to package managers like `pip`, etc.
Software Design	Library vs. command-line vs. UI. User-/developer-experience. What is an application? The basics of how applications work.
Software Implementation	Iterative development methods. The REPL. Incremental application design. Assertions. Tests. How to implement things done more quickly - and with fewer surprises.
Libraries	Command-line parsers, configuration parsers, plotters, renderers
Advanced/Specialized topics (later)	Languages with type systems. Native application development (C/C++/Fortran /Rust). Multithreading. GP-GPU. Julia, hardware engineering, etc.

(Bonus) Interactive: Reproduce Open Science

- I tried searching Dryad, but it's mostly datasets
- So I went through ~50-75 papers from PLOS Computational Biology's September 2023 issue
  - Most include *some* form of script or open-science
  - Around half of them use something like MATLAB or R
  - The other half use python
  - … and deliver the python via Jupyter
  - … and assume you're using anaconda to manage libraries
  - … and use many libraries
- Around 5 or so "just python, not too many dependencies" codebases from that cohort
- Lets try….. : https://github.com/John-king-zhou/COVID
- (see https://github.com/adamkewley/rse-meetups/1_IntroAndTooling README for a more comprehensive list of the stuff I tried)