Embracing your Inner Research Software Engineer (RSE)

Working Together with CTSM RSE's to improve CTSM Science

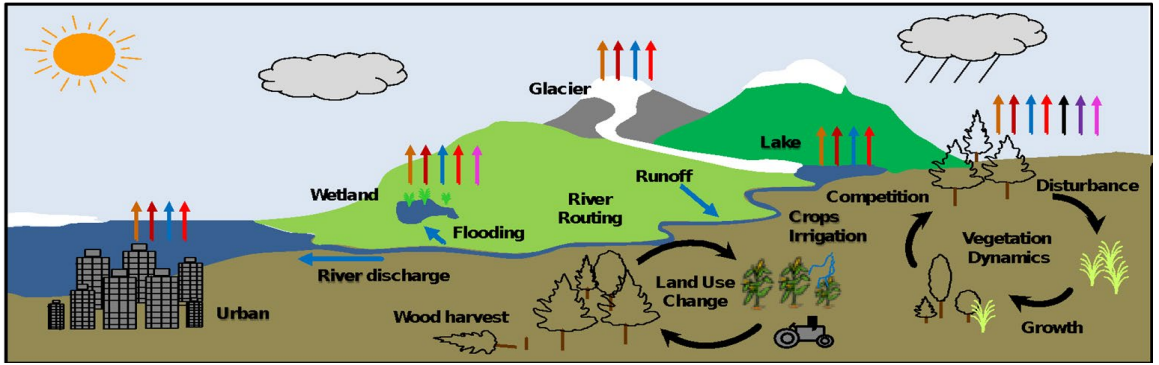
Erik Kluzek, LMWG Winter 2025 meeting, Feb 24-26

Overall Outline

- CTSM inherently is Science expressed in Software – RSE activities are vital!
- Because of limited resources we need to work together on RSE in CTSM
- Here's some things that make doing RSE things hard..
- We'll survey you on some questions regarding RSE things – please respond it's anonymous (https://app.sli.do/event/rPu6vQ3H81vHpuU1L8Qagh)
- Good news – if you are touching CTSM code – you are an RSE!
- Here's a list of things that will help YOU – and CTSM

What is CTSM?

- CTSM is Science – expressed in Software
- Therefore if you touch any CTSM code – you are an RSE!
- Bad software practices also kill our science
- Clean software and software development practices makes CTSM Science:
  - Easier
  - Flexible
  - Robust
  - Verifiable
  - Correct
  - Reproducible



In CESM what percentage of people *should* be funded as Research Software Engineers?

Rate of RSE activities for CTSM

- What percentage of CESM should be spent on RSE work? What percent is?
- In TSS in CGD NSF-NCAR 2 of 16 have a title of SE (12%)
- Since 2013 we make 40-60 tags per year (once a week)
- 2024 we opened 190 issues, and closed 261 (3 per tag)
- 2024 we closed 213 Pull Requests (oldest 6 years old)
- We have 43 open Pull Requests in CTSM right now (oldest 6 years old)

What Makes Doing RSE Tasks Hard?

What can we learn from the SE industry?

- Debugging bad code is the MOST expensive thing we do
  - Inherently intractable
  - Not possible to estimate
  - Further along in development the more expensive it is
- Brittle, poorly designed, poorly tested code keeps you constantly debugging
- Own example: 83 tasks over 2 years, 50% debugging

What Practices does the SE Industry Show Helps?

As such SE Industry and Research has found the following practices help:

- Figure out what the software needs to do ***REQUIREMENTS*** (neither too much or too little)
- Spend effort into ***DESIGN*** of the code itself
- Add automated ***TESTING*** WHILE you develop

RSE Maxim to Live By

Untested Code – IS broken (or will get broken)

When you try something in CESM and it's broken for you -- what do you do?

List of RSE Suggestions

1. Small circle JuJitsu (small cycles)
2. Trim the fat (requirements)
3. Draw the building (design code changes before starting)
4. Preserve success (git version control)
5. Practice vulnerability (openly share code/issues)
6. Trust but verify – test AS you go (Test Driven Development)
7. Improve design as you go (refactoring)

Small circle JuJitsu (small cycles)

Software Development Methodologies are based around the following steps:

- Requirements (what does it need to do)
- Design (figure out what the code should look like)
- Implement (actually do coding)
- Test (verify that the code is correct)
- Deploy (Finish it and give to others)
- Refactor (Improve design – just for ASD)

Waterfall Software Development: is one extreme with one monolithic pass

Agile Software Development: is at the other with continually doing small cycles of the above

Trim the fat (requirements)

- Before starting a development cycle
- Figure out exactly what you need to do for this cycle
- Make sure you don’t do too much
- Or too little
- Stick to it later – helps avoid scope creep

Draw the building (design code changes before starting)

- Think about the CTSM code and how your changes come in
- Go for clean interfaces that ideally isolate the changes and logic in a modular fashion
- Use helper functions to reduce code duplication

Preserve success (git version control)

- Follow advice on CTSM github wiki pages
- Start from last minor version release tag (ctsm5.3.021)
- Commit your changes as you show they work
- This allows you to go back to a working version
- Allows us to integrate your changes in easier

How embarrassed are you usually to share you code?

Practice Vulnerability (openly share code/issues)

- Who has been embarrassed to share code you've worked on?
- Have others review your code
- Best way to improve
- Code is a shared resource the team is responsible for
- Bugs are a shared problem as well

Trust but verify – test AS you go (Test Driven Development)

- Develop tests first
- Show they fail and then PASS as you implement the update
- Adrianna will go into this more

Improve design as you go (refactoring)

- With sufficient testing in place you can improve the design as you go
- Improve the design to make it easier to bring in your new code
- Improve the design to get rid of brittle code causing problems
- Refactoring means improving the code design – without changing answers

Conclusion

- Adopt some RSE practices to help your development of CTSM science
- Which practice are you most interested in?
- Which practice do you want to try first?