Sandia National Laboratories

Exceptional service in the national interest

Challenges and Strategies for Testing Automation Practices at Sandia National Laboratories

Miranda Mundt, Jonathan Bisila, Reed Milewicz, Joshua Teves, Michael Buche, Jonathan Compton, Jason Gates, Kirk Landin, Jay Lofstead

17 October 2023

First Conference of the US Research Software Engineer Association (US-RSE'23)

SAND2023-10769C

Icons used from Flaticon by artists Uniconlabs, wanicon, Iconjam, Freepik; Illustrations taken from undraw.co

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Introduction: Automation for Software Development

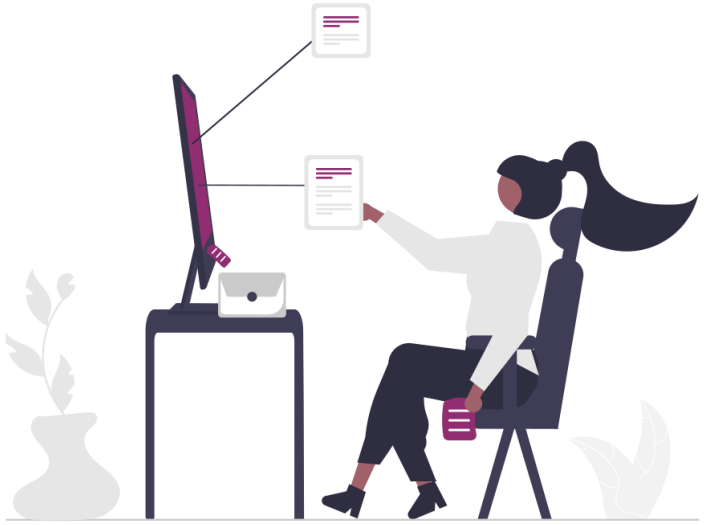
- Automation has been a topic of conversation in software development for decades
  - Automate **repetitive**, **labor-intensive tasks** in order to improve overall **developer and project productivity**
  - It enables **reproducibility** through automated workflows, **verification** through testing, and **better interdisciplinary teaming**

The Challenges in Computational Science and Engineering

- Scientific software **benefits from a range of testing practices** – but have **limited effectiveness**
- Only **half** of CSE teams have software engineering training
- Many institutions have **heterogeneous software environments**

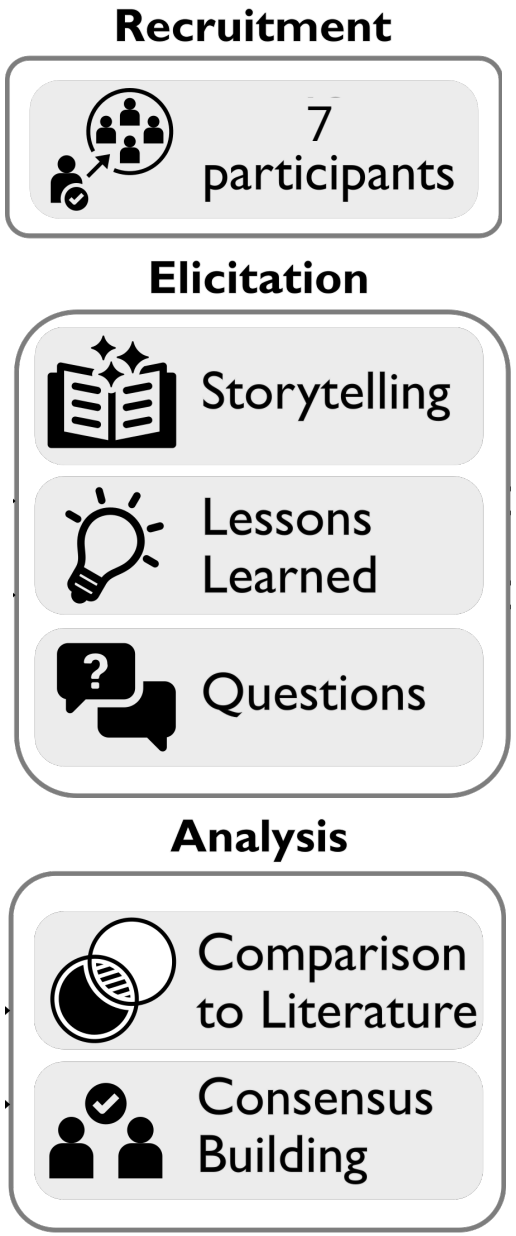
Our Research Questions

- **RQ1**: What are the **challenges** experienced by Research Software Engineers (RSEs) in testing automation at a large US national laboratory?
- **RQ2**: What **strategies** have been employed by Research Software Engineers (RSEs) to address these challenges?



Our Approach

- Conducted a **participatory action research study** to collect and analyze the experiences of RSEs at Sandia National Laboratories.
- Recruited RSEs to **share experience stories**: detailed narratives of challenges faced and accomplishments made in automation work at the laboratories.
- Analyzed their challenges and lessons learned through the lens of the **scholarly literature on automation** in industry contexts, and iterated on results with participants to **build consensus**



Results

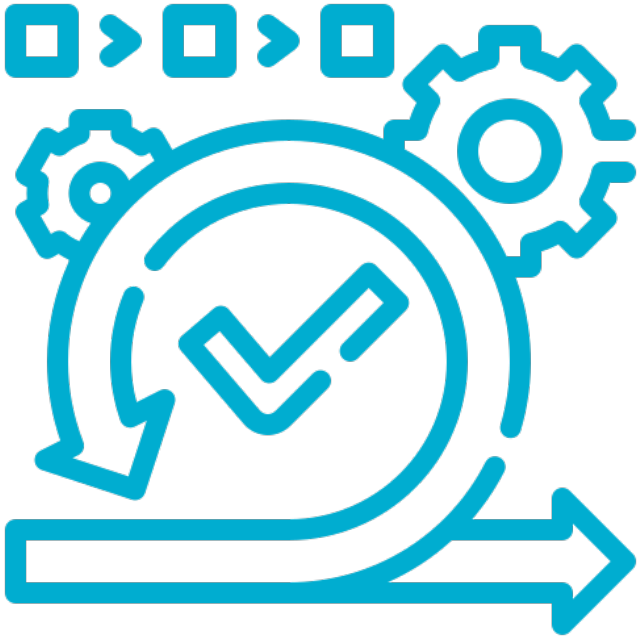
Summary of Experience Stories
(S1) Using pytest, nbmake, and GitLab pipelines to handle a specific dataset and environment
(S2) Using Rust to provide a computational back-end while providing an interface for Python and Julia
(S3) Creating a pipeline layer and a machine orchestration layer to manage a separation between the two
(S4) Creating randomized tests that check an invariant property
(S5) Distributing software components across two hosting services
(S6) Writing a simplified test with no compute for speed
(S7) Testing machine learning code by using small unit tests and checking expected invariants; running the program to see if it will crash as a basic test
(S8) Creating an automated pipeline to get performance benchmarks

- Collected and analyzed 8 experience stories
- Compared results to industry automation studies

Themes

Continuous Integration

The development of exascale codes on bleeding-edge hardware requires testing across a **variety of heterogeneous machines**. For each machine, there may be **multiple supported programming environments**, and for each environment, there may be **multiple ways to configure the code**. Ensuring the code clones, configures, builds, tests, installs, and runs successfully for the plethora of desired permutations is a daunting task. When considering the testing of multiple long-lived branches, and the desire to have both development and production versions of the CI infrastructure, you're looking at maintaining **hundreds of jobs**.

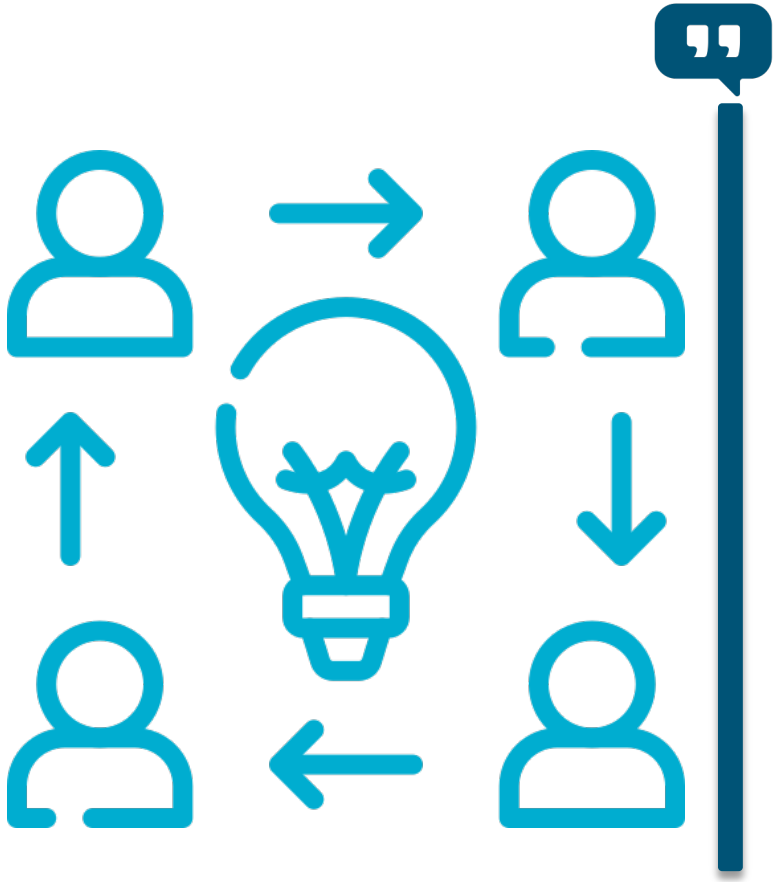


Heterogeneous Computing Environment

We developed one set of testing routines for Stitch-IO in Python that focused on ensuring that things functioned (not quite unit testing, but slightly more complex). We also had a test written in C that was supposed to represent how the application works in practice, but without the physics so it would be fast.

In spite of both the Python and C tests **all working correctly**, the application was having **data corruption issues**. The tests should have revealed the source of the errors, but they did not. After some analysis, we determined that the C application representation was not moving through the computational space exactly as it would for a production run. **The simplification should not have mattered, but it turns out that it did**.

Interdisciplinary Collaboration Requirements



Creating a “one build script to rule them all” in Python **removes cognitive load** from scientific subject-matter expert developers. **Making it easy for them to do the right thing helps everyone**. Also, providing a means for the team to contribute back to the “one script” allows **flexibility** to explore outside the box while still **controlling things as much as possible**.

Lack of Professionalization

Randomized property-based testing, despite all of its successes, is still not widely known in the software engineering world. I think that is mainly due to **lack of education**, and our schools need to do a better job of including it in their curricula. It is still viewed as an “Advanced Topic” despite being very **accessible**. I think that part of this view is that successfully employing this testing requires the developers to formulate invariants, etc., and this is another skill that is **not taught very well in schools**.

Addressing the Research Questions

RQ1: Challenges in Test Automation

Challenge Category[1]	Challenges[1]	Stories
Behavioural	• Process adherence • Organizational change • Too high expectations • Process deviations	• S3, S5 • (N/A) • S1, S3, S4, S6, S8 • S3, S5
Business and Planning	• Cost of ownership and operation • Automation too expensive for small projects • Lack of time, people, and funding	• S1, S3, S4, S5, S8 • S1 • S1
Skills	• Diversity • Steep learning curve	• S1, S2, S3, S5, S8 • S1, S2, S3, S4, S5, S6, S7, S8
Test System	• Inadequate development practices • Testware architecture • Untested test environment • Limitations in externally sourced tools • Environment configuration	• S1, S3, S4, S5, S7 • S1, S3, S4, S5, S8 • S1, S3, S6 • S1, S2, S3, S4, S5, S6, S7, S8 • S3, S5
System Under Test (SUT)	• SUT speed • SUT testability • Platform limitations	• S3, S4, S6 • S1, S2, S3, S4, S6, S7 • S1, S2, S3, S5, S6, S8

[1] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Impediments for software test automation: A systematic literature review," Software Testing, Verification and Reliability, vol. 27, no. 8, p. e1639, 2017.

RQ2: Recommendations for Improvement

Challenges[1]	Recommendations[2]
• (Behavioural) Too high expectations	✓ Involve key stakeholders in strategy development ✓ Keep test professionals motivated about automation ✓ Define an effective test automation strategy ✓ Adjust the test automation strategy to the changes
• (Business and Planning) Cost of ownership and operation	✓ Provide enough resources
• (Skills) Diversity • (Skills) Steep learning curve • (Test System) Inadequate development practices	✓ Share available test automation knowledge ✓ Allow time for training and the learning curve ✓ Have competent test professionals ✓ Promote collaboration
• (Test System) Testware architecture • (Test System) Limitations in externally sourced tools	✓ Select the right test tools ✓ Arrange testware in good architecture
• (SUT) SUT testability	✓ Design the SUT for automated testability
• (SUT) Platform limitations	✓ Define test automation requirements ✓ Have control over changes of test automation requirements ✓ Arrange testware in good architecture

[1] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Impediments for software test automation: A systematic literature review," Software Testing, Verification and Reliability, vol. 27, no. 8, p. e1639, 2017.
[2] Y. Wang, M. V. Mäntylä, Z. Liu, J. Markkula, and P. Raulamo-jurvanen, "Improving test automation maturity: A multivocal literature review," Software Testing, Verification and Reliability, vol. 32, no. 3, p. e1804, 2022.

Conclusion

- As software becomes more integral to the advancement of science, so do the processes and procedures used to create scientific software.
- In our study, we collected experiences from RSEs at a US national laboratory. We analyzed the commonalities and differences between industry and scientific software testing automation practices.