Introduction to the Minitrack Software Sustainability: Strategies for Long-Lasting and Usable Software

Maytal Dahan
Texas Advanced Computing Center,
The University of Texas at Austin
maytal@tacc.utexas.edu

Joe Stubbs
Texas Advanced Computing Center,
The University of Texas at Austin
jstubbs@tacc.utexas.edu

Sandra Gesing
San Diego Supercomputer Center
University of California, San Diego
sgesing@ucsd.edu

Abstract

As software increasingly drives technological progress, ensuring its usability, durability, and reproducibility has become essential across diverse scientific fields worldwide. Sustaining software longevity and resilience is crucial for fostering innovation, maintaining functionality, and lessening the environmental impact of rapid technological change. Usability, sustainability, and reproducibility are interconnected principles that influence all stages of the software lifecycle—from enabling reproducible experiments through accessible interfaces to facilitating portability through containerization. This minitrack provides an exploration of software sustainability, covering strategies for enduring software solutions, user experience enhancements, and solutions to data curation and provenance challenges.

1. Introduction

The principles of usability, sustainability, and reproducibility are deeply interconnected and apply across a wide range of application areas. These concepts influence every layer of the software development process - from enabling experiment reproducibility through user-friendly interfaces to achieving portability via containerization. Such principles are also crucial in building Science Gateways (also known as virtual laboratories or virtual research environments), which provide end-to-end solutions tailored to specific community needs. Software sustainability spans a broad scope and encompasses a variety of topics, including:

- Web-based solutions (websites, science gateways, virtual labs, etc.)
- Application Programming Interfaces (APIs)
- Computational and data-intensive workflows
- Novel approaches in containerization
- Sustainable practices in software development
- System architectures for testing and continuous integration
- Emerging best practices in artificial intelligence and machine learning software, including model sustainability, reproducibility, and ethical considerations
- Key success factors for usability, sustainability, and reproducibility

This minitrack, *Software Sustainability: Research on Usability, Maintainability, and Reproducibility*, introduces the range of accepted papers for HICSS-58. It showcases a broad array of submissions addressing complex scenarios such as containerization, strategies for durable software, usability and user interface challenges, and solutions for data curation and provenance.

2. Accepted Papers

This year's minitrack on Software Sustainability: Research on Usability, Maintainability, and Reproducibility presents a diverse collection of papers that explore crucial aspects of sustainable software development, with a particular focus on reproducibility, transparency, and best practices in the academic and scientific communities. Each paper offers unique insights into challenges and solutions in maintaining resilient and reusable software systems. Below is an overview of the accepted papers:

AI Readiness: A Reusability Study of Popular AI Algorithms

URI: https://hdl.handle.net/10125/109729
978-0-9981331-8-8

This paper examines the reusability and reproducibility of popular AI algorithms in computer vision, audio analysis, and natural language processing. The study highlights the challenges of recreating runtime environments and argues that publishing code and datasets alone is insufficient for reproducible AI research. By surveying 75 openly available AI algorithms, the authors identify gaps in runtime environment reusability and propose solutions, including container libraries and enhanced documentation, to support more reproducible and reusable AI research.

Making Software Work Sustainable for the Academic Research Group: A Comparative Case Study

Through a comparative case study of two research groups, this paper explores sustainable approaches to integrating software engineering practices within academic research. The authors characterize two distinct strategies—practice integration and differentiating expertise—used by research groups to ensure the sustainability of software work. Their findings underscore the importance of evaluating the impact of changes in research software practices on long-term scientific progress.

"We Provide Our Resources in a Dedicated Repository": Surveying the Transparency of HICSS Publications

This study assesses the transparency and reproducibility of HICSS publications by examining whether and how authors leverage external repositories to share supplementary research materials. Analyzing 5,579 papers from recent HICSS proceedings, the authors reveal that only 3% of papers with human subject research or technical implementations have publicly available, functional repositories. The findings emphasize the need for improved repository use to enhance transparency and reproducibility in academic research.

How to Position Your Gateway for Failure: The Ten Don’ts of Gateway Design

Science gateways are instrumental in providing access to computational resources, yet many gateway projects fail to achieve long-term sustainability. This paper identifies ten common pitfalls that lead to gateway failure, drawing on experiences from well-known frameworks like Hubzero, Tapis, and Galaxy. The authors offer practical guidance for emerging gateway projects, emphasizing strategic planning and community building as critical components for creating sustainable, user-centered research tools.

3. Conclusion

The papers presented in this year’s minitrack on *Software Sustainability: Research on Usability, Maintainability, and Reproducibility* highlight crucial themes in advancing sustainable software practices across AI, academic research software, and science gateways. The studies explore key challenges in ensuring reproducibility, such as the need for comprehensive runtime environment documentation in AI research, and the importance of transparent repositories for sharing supplementary research materials. These contributions emphasize the role of reproducibility, usability, and long-term viability as foundational elements for sustainable research software.

Furthermore, the research highlights practical strategies for integrating software engineering practices into academic and research environments. Case studies reveal how different research groups adopt varying approaches, from practice integration to differentiating expertise, to ensure the sustainability of software solutions. The work on science gateway design identifies common pitfalls and provides actionable recommendations to support long-lasting, user-centered platforms. Together, these studies provide a roadmap for developing resilient and transparent software systems, ultimately driving sustainable practices that enhance the impact of scientific discovery and innovation.