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
ssgesing@ucucsd.edu

Abstract

In an era where software plays a pivotal role in shaping technology landscapes, ensuring usabile, long-lasting and reproducible software is one that spans various domains of science and significant investment of research funding across the US, Europe, U.K, and elsewhere. Ensuring the longevity and resilience of software systems is of paramount importance for fostering innovation, maintaining functionality, and mitigating the environmental impact of constant technological evolution. The three concepts — usability, sustainability and reproducibility are interconnected with each other and cover a wide range of application areas. They affect all layers of the software process – from enabling reproducing experiments via an easy user interface to using containerization for application portability. This minitrack serves as a comprehensive exploration of diverse topics within the realm of software sustainability, encompassing intricate scenarios like containerization, strategies for enduring software solutions, usability and user interface considerations, as well as addressing challenges in data curation and provenance.

1. Introduction

The three concepts usability, sustainability and reproducibility are interconnected with each other and cover a wide range of application areas. They affect all layers of the software process - from enabling reproducing experiments via an easy user interface to using containerization for application portability. Such concepts are also relevant in the building of Science Gateways (also known as virtual laboratories or virtual research environments), which by definition serve communities with end-to-end solutions tailored specifically to their needs. Software survivability involves a wide scope that can potentially include the following topics:

- Web-based solutions (web sites, science gateways, virtual labs, etc.)
- Application Programming Interfaces (APIs)
- Computational and Data-Intensive Workflows
- Novel approaches in containerization
- Sustainability practices in software development
- System architectures for testing and continuous integration
- Emerging best practices in Machine Learning software
- Best practices and Key Success Factors for usability, survivability and reproducibility

This minitrack, Software Sustainability: Research on Usability, Maintainability, and Reproducabiliy minitrack introduces the wide variety of accepted papers to HICSS-57. It focused on the broad spectrum of submissions that deal with complex scenarios such as containerization, strategies for long-lasting software, usability and user interface issues, handling data curation and provenance and more.

2. Accepted Papers

The minitrack received three papers with an excellent breadth of topics from a literature review of sustainability to understanding the open source landscape in the data science domain.. We accepted two papers to this minitrack that introduce the following topics:

- Researching a study design in using open-source natural language processing (NLP) models and what characteristics

URI: https://hdl.handle.net/10125/107309
978-0-9981331-7-1
(CC BY-NC-ND 4.0)

determine the success or popularity of a model.

- Researches the idea behind combining the techniques of eye tracking/cognition with software trustworthiness and reusability to quantitatively analyze the general fit and finish of code. The paper described an interesting framework for assessing code quality through eye-tracking and testing various types of manipulation to determine the effects of these mechanisms on how suitable software developers found the code to be for reuse.

One paper selected for this minitrack, "Understanding Open-Source NLP Artifact Adoption Through Information Systems Success Factors", delves into the integration of natural language processing (NLP) artifacts within the realm of Open-Source Software (OSS), employing the DeLone and McLean Information Systems Success Model—a framework widely recognized for evaluating the success of information systems. Through a nuanced examination of system quality, information quality, and task types, the research seeks to discern the distinct influences on adoption concerning analysis and generation tasks. Drawing upon data sourced from Hugging Face, the study posits that system quality exerts a more pronounced impact on adoption in the context of analysis tasks, while information quality plays a more significant role in the adoption of generation tasks. These findings contribute to an enhanced comprehension of OSS success, particularly within the domain of NLP.

Lastly the authors of the paper, "Effects of Coding Norm Violations on Visual Effort, Trustworthiness Perceptions, and Reuse Intentions" assert that the escalating demand for automated systems and novel programming applications has intensified the quest for efficient strategies in computer code creation. Practices such as code reuse and writing reusable code, proven to enhance efficiency in the code creation process, is garnering significant interest. This paper explores the factors influencing a programmer's perception of code trustworthiness, identified through cognitive task analysis as performance, reputation, and transparency of the code. Performance is gauged by the code's ability to meet project objectives, reputation involves external assessments, and transparency encompasses code understandability, readability, and organization. Leveraging eye-tracking data, this study examines how introduced errors in code readability and organization, as well as alterations in the code source's stated reputation, impact a programmer's trustworthiness perceptions and reuse intentions. By analyzing eye movements during code review tasks, the research aims to infer visual effort, explore variations in eye movements based on code violations, and replicate effects observed in previous studies. The study sheds light on the driving forces behind attention capture during the code review process, contributing valuable insights into code trustworthiness and reusability.

3. Conclusion

These papers show a range of applications and impact of software sustainability in production and research software. They cover crucial aspects such as reproducibility and cultural approaches. We hope you will join us for interesting presentations and lively discussions on software sustainability, reproducibility, challenges, and solutions for our evolving landscape.
We aim at continuing with this minitrack in the future at HICSS and encourage authors to contribute their research and viewpoints on software sustainability with its many facets and areas.