2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) | 979-8-3503-2263-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSE-COMPANION58688.2023.00095

SAIN: A Community-Wide Software Architecture INfrastructure

1st Joshua Garcia
University of Calfornia Irvine
joshug4@uci.edu

2nd Mehdi Mirakhorli
Rochester Institute of Technology
mxmvse@rit.edu

3rd Lu Xiao
Stevens Institute of Technology
lxiao6@stevens.edu

4th Sam Malek
University of Calfornia Irvine
malek@uci.edu

5th Rick Kazman
University of Hawaii
kazman@hawaii.edu

6th Yuanfang Cai
Drexel University
yfcai@cs.drexel.edu

7th Nenad Medvidović
University of Southern California
neno@usc.edu

***Index Terms*—software architecture, reproducible, empirical software engineering**

I. Introduction

Software Architecture is the most important determinant of the functional and non-functional attributes of a system [1]–[3]. Put simply, software systems "live and die" by their architectures [4]. Despite the importance, the architecture of a software system is often not explicitly documented, especially in the prevalent Agile methods in the past decades. Instead, the architecture of a system often becomes hidden in the myriad system implementation details, and gradually decays and accumulates grime—causing significant challenges to its long-term evolution and maintenance [5]–[8]. Recovering, understanding, and updating a system's architecture is an important facet of overcoming this challenge to support the evolution and maintenance of long-lived software systems.

Responding to the above challenge, software architecture research has yielded many different tools and techniques in the past two decades. However, the disjoint research effort and diverse lab environments where different tools and techniques are created have impeded technology transfer for reproducible empirical studies for the community. In other words, there is a lack of shared infrastructure with available tools and datasets for systematic synthesis and empirical validation of new or existing techniques. As such, researchers and practitioners in need of cutting-edge tools tend to re-invent, re-implement research infrastructure, or ignore particular research avenues altogether.

To address these challenges of reusability, reproducibility, and replicability of software architecture research, we have produced **S**oftware **A**rchitecture **IN**strument (*SAIN*), a first-of-its-kind framework for assembling tools in support of *architecture-based software maintenance*. *SAIN*'s capabilities have been motivated by directly engaging the software researcher and practitioner communities, in the form of three workshops as well as a survey conducted by the authors. *SAIN* is delivered as a web-based platform consisting of three principal components: 1) a *catalogued library* of cutting-edge tools for reverse engineering and analyzing software systems' architectures; these tools are either provided by their original authors or reproduced from literature; 2) a *plug-and-play instrument* for integrating the tools and techniques to facilitate empirical studies of software architectures; and 3) *reproducibility wizards* to set up experiment templates, produce replication packages, and release them in easy-to-run and modify formats.

Identify applicable funding agency here. If none, delete this.

In this technical briefing, we will present *SAIN*, demonstrate how to employ the three components of *SAIN* mentioned above for repeating existing as well as developing new research ideas, and provide participants with a hands-on experience/tutorial guided by members of our team. The technical briefing will contain three sessions:

- We will introduce the basic functions of *SAIN*, as well the tools and the datasets available on *SAIN*. The latter include, 13 architecture recovery components, 8 components for computing architectural metrics or analyses, 2 fact extractors, and 9 utility components from those tool suites.
- We will then demo two case studies that can be easily replicated using the *plug-and-play instrument* of *SAIN*. These include one compact case study of *SAIN* run on a game engine project called Mage and another detailed case study of *SAIN* run on Hadoop 2.5.0; and the empirical results of the detailed case study, which analyzes the relationships between architectural smells, architectural tactics, and error-proneness.
- We will let the participants reproduce case studies themselves by using the *reproducibility wizards* of *SAIN*.

In the end, we will collect feedback from the participants regarding how we could keep improving *SAIN* to better serve the community.

II. Relevance to Software Engineering Community

SAIN and its capabilities have been motivated by directly engaging a large segment of the software researcher and practitioner communities. They included three workshops—conducted in Los Angeles, CA in January 2017; in Buenos

979-8-3503-2263-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICSE-Companion58688.2023.00095

Authorized licensed use limited to: Rochester Institute of Technology. Downloaded on December 16,2023 at 07:46:42 UTC from IEEE Xplore. Restrictions apply.

Aires, Argentina in May 2017 (collocated with ICSE 2017); and in Urbana-Champaign, IL in November 2017 (collocated with ASE 2017)—that led to the identification of SAIN's core requirements and its baseline architecture. These activities were followed by a survey of an additional 130 researchers to prioritize and identify any additional requirements.

SAIN will provide a critical resource to software engineering *researchers*, enabling extensive empirical research. By providing a large repository of architectural artifacts, researchers will be able to compare and contrast their techniques using the same datasets. This will, in turn, enable researchers to establish a common understanding of the relative accuracy of different techniques, identify gaps and sources of inaccuracy, and develop new solutions to incrementally or fundamentally improve the results. *SAIN* will also be an important resource to *practitioners*. *SAIN* will provide practitioners an infrastructure where they can obtain and try the various tools, provide feedback, contribute to the repository of architectural artifacts, and generally influence the research conducted in academia and research labs.

In terms of the usage of *SAIN* so far, we ran a school allowing users to learn more about *SAIN*, which included over 40 registrants and attendees from across the world (including the US, Europe, and China). Additionally, the *SAIN* user base has grown to nearly 100 users from all over the world.

III. Technical Briefing Agenda

We plan to host a 180 minute program with two modular 90 minutes sessions. Following we describe the agenda for our sessions:

a) Introduction and General Discussions: We are proposing to have the first 90-minute session to start with an introduction to software architecture, and a brief presentation of the state of the art and practice on this topic from three complementary perspectives of (i) software engineering problems addressed, (ii) existing tools and techniques previously developed and used, and (iii) types and characteristics of available datasets. The introduction will take roughly 45 minutes.

After the introduction, we will hold a *General Discussions*. This technical briefing will be highly interactive in order to encourage discussion between participants. It will feature "three-minute madness" talks from participants to share their experiences and initial thoughts on the topic. The idea is that the participants will be given three minute to talk to the audience about their interests, challenges, or questions. This part will take roughly 45 minutes (assuming 15 participants). We plan to adjust the time given to each participant, depending on the number of participants we get.

b) Demo and Experiment Session: The secnond 90 minutes session will include a demo of *SAIN*, as well as the "hands-on" exercises for the participants. We will demo the three key components of *SAIN*: 1) a *catalogued library* of cutting-edge tools for reverse engineering and analyzing software systems' architectures; these tools are either provided by their original authors or reproduced from literature; 2) a *plug-and-play instrument* for integrating the tools and techniques to facilitate empirical studies of software architectures; and 3) *reproducibility wizards* to set up experiment templates, produce replication packages, and release them in easy-to-run and modify formats.

c) Experiment Session: We will engage the participants to replicate experiments on *SAIN*, levering its *reproducibility wizards*. These include one compact case study of *SAIN* run on a game engine project called Mage and another detailed case study of *SAIN* run on Hadoop 2.5.0; and the empirical results of the detailed case study, which analyzes the relationships between architectural smells, architectural tactics, and error-proneness.

d) Feedback: Finally, we will invite participants to share their feedback of using *SAIN*. We will ask participants to submit a survey to evaluate the functions of *SAIN*. We will also invite participants to share their insights regarding how we should improve and maintain *SAIN* as a community.

IV. Target Participants

The target audience includes researchers interested in software architecture, software maintenance, empirical software engineering, and reproducibility. Researchers and practitioners who are developing/using methods, techniques, and tools to analyze software architecture will be particularly interested in this technical briefing. We also target practitioners who face the challenge of finding empirically validated solutions for the software engineering problems they are addressing. Furthermore, our target audience includes SE educators interested in developing course materials in this area. No specific background will be required. Participation in the technical briefing will be open to all ICSE 2023 participants.

References

[1] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall Englewood Cliffs, 1996, vol. 1.

[2] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software architecture: foundations, theory, and practice*. Wiley Publishing, 2009.

[3] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Addison-Wesley, 2013.

[4] "Analysis: It experts question architecture of obamacare website," http://www.reuters.com/article/us-usa-healthcare-technology-analysis-idUSBRE99407T20131005, 2013.

[5] A. Telea and L. Voinea, "Visual software analytics for the build optimization of large-scale software systems," *Computational Statistics*, vol. 26, no. 4, pp. 635–654, Dec. 2011. [Online]. Available: http://link.springer.com/10.1007/s00180-011-0248-2

[6] T. A. Standish, "An Essay on Software Reuse," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 5, pp. 494–497, Sep. 1984, conference Name: IEEE Transactions on Software Engineering.

[7] T. A. Corbi, "Program understanding: Challenge for the 1990s," *IBM Systems Journal*, vol. 28, no. 2, pp. 294–306, 1989, conference Name: IBM Systems Journal.

[8] S. Yau and J. Collofello, "Some Stability Measures for Software Maintenance," *IEEE Transactions on Software Engineering*, vol. SE-6, no. 6, pp. 545–552, Nov. 1980, conference Name: IEEE Transactions on Software Engineering.

Authorized licensed use limited to: Rochester Institute of Technology. Downloaded on December 16,2023 at 07:46:42 UTC from IEEE Xplore. Restrictions apply.