A Doomsday Vault of Software Engineering Tools

Archiving Software Engineering Tools from ICSE and FSE 2011 through 2014

Emerson Murphy-Hill
Department of Computer Science
NC State University
Raleigh, NC, USA 27695
emerson@csc.ncsu.edu

ABSTRACT

Many innovative software engineering tools appear at the field's premier venues, the International Software Engineering Conference (ICSE) and the Foundations of Software Engineering (FSE). But what happens to these tools after they were presented? In this paper, we spend 10,000 hours trying to obtain, download, use, and repackage 150 tools from ICSE and FSE's tool demonstration tracks. Our results enumerate the practical and accidental reasons that software engineering tools fail to work over time, and provide practical implications for creating lasting tools.

CCS Concepts

•**General and reference** → *Empirical studies;*

Keywords

Software engineering tools; replication

1. INTRODUCTION

Sofware engineering research seeks to better understand software and how it is built and constructed. While such understanding in isolation can be insightful, ultimately a substantial amount of such research aims to impact the practice of software engineering. To do so, researchers can create recommendations for new software engineering practices, can create educational techniques and materials, and can create tools.

Arguably creating new tools is the most common way that sofware engineering researchers attempt to influence practice. Broadly speaking, tools are software that can help design and build software. Examples include a tool that helps mobile application developers choose which devices to target [9], a tool that checks the use of locks in multithreaded programs [4], and a tool that creates code from natural language text [3].

FSE '16 November 13–19, 2016, Seattle, WA, USA

DOI: 10.1145/1235

While papers describe tools in software engineering venues, there are several reasons why software engineering researchers should make the tools themselves available. First, the full details of how the tool works, both in terms of its internals and from an end-user perspective, may not be clear from the paper. Second, making the tools available can help pracitioners try and adopt the tools in their work. Third, it helps facilitate reproducibility by enabling future researchers to perform studies with the original tools. Finally, it helps the advancement of the field by allowing others to build on existing tools, rather than re-buiding them from scratch.

Despite the benefits of making tools available, in this paper we catalog the practical difficulties in doing so. We describe a study that examined the tools presented at the premier venues for software engineering research over the past several years. As part of a graduate university course on software engineering, we spent 10,000 hours trying to obtain, download, use and repackage tools from the International Conference on Sofware Engineering (ICSE) and the Symposium on the Foundations of Software Engineering (FSE). In doing so, we make the following three main contributions in this paper:

- A study that evaluates the difficulty in getting tools working across a variety of software engineering research.
- A synergistic course project for graduate software engineering students that gives students meaningful educational value *and* provides the research community value.
- XXX existing tools repackaged in automatically-built virtual machines to make it easier for others to use these tools.

2. RELATED WORK

timeliness

General scientific interest in reproducibility. How it fits with other Rs. State of practice.

Software reproducibility outside SE. Systems [2, 1]

Others [8, 13, 12, 7].

Arguably as SE researchers we should be the best!

Other fields: - Data science (doing better - http://zenodo.org) - MPC journal requires code submission (tarball) - In general, Math not doing a lot with code sub submissions - Security, should be able to do VM, but don't (but see Will's

paper) - HPC doesn't (and maybe can't) - RTC is starting to do it, but maybe shouldn't

" We argue that, with some exceptions, anything less than the release of source programs is intolerable for results that depend on computation." [6].

Constant worry is protecting IP in security and HPC

Repeatability in AI-SE: "depends on who does it" Reproducibility assessment for 2 papers, plus reproducibility overview [5]

While no studies of software reproducibility in our field, practical interest.

Artifact evaluation committees. (SIGPLAN, where else?)

3. RESEARCH QUESTIONS

Henceforth, we will simply say *tool* to refer to software engineering tools presented at the International Conference on Software Engineering or Foundations of Software Engineering in their respective tool demonstration tracks.

1. How much effort is required to get tools to work?
2. What are the barriers to get tools to work?
3. How much effort is required to get tools to work in virtual machines?
4. What are the barriers to get tools to work in virtual machines?
5. To what extent is this study practical to implement in a classroom context?

4. COURSE DESCRIPTION

We conducted the study described in this paper as part of a graduate software engineering course in the Computer Science department at North Carolina State University. The department has about 200 PhD students and 450 MS students in its graduate program. While graduate students are not required to take the course, it is one of seven core systems courses, from which students must take one course. The department does have a specialty MS degree track in software engineering,[1] which does require this course.

Course content covers software engineering processes, software architecture, design patterns, software security, verification and validation, estimation, project managment, requirements, certification, and formal methods.[2] Apart from the course content, involving lectures and a final exam, the other major component of the course is the project. Next, we describe the project for this course in prior offerings of the course (Section **??**) and in the offering described in this paper (Section **??**).

4.1 Prior Course Project and Criticism

In prior offerings of the course, students were essentially given two project offerings. In one version, students could create a software engineering tool, such as a plugin for Eclipse. In the other version, students chose several existing, similar software engineering tools, and applied them to open source software, then reflected on what they learned about the tools and the projects. Most students opted for the second version. In both cases, students were required to write up their results. This project is similar to a course project assigned by David Notkin at the University of Washington.[3]

After offering this project to students for several years, the instructor recognized several problems the way he had executed it:

- The course did not have enough time to teach students technical writing skills, and thus final papers were of poor quality, on average. Moreover, while technical communication is a course objective, other types of communication would likely be more valuable to most students, who are non-thesis and industry-focused.
- Students had limited exposure to state-of-the-art tools; the tools they chose to study were typically quite basic.
- Students appeared to rarely gain technical skills during the project.
- Most students were not doing *novel* work; each semester, different students wrote similar reports on similar tools, which had no value beyond their educational value.

While the last point may seem odd – why would a course project need value beyond its educational value? – it's not unusual for some software enginering courses to have added value, such as contributing to open source projects [11, 10]. In the case of the present course, the instructor thought that such system development would not be useful to most students in the course, who typically come in with 1 to 2 years of industrial software engineering experience.

4.2 New Course Project

In Fall of 2015, the instructor changed the course project to allieviate the problems described in the last subsection. In short, student teams were assigned tools described in a a prior research paper at ICSE or FSE, the premier venues for software engineering research. Students were required to obtain the tools, get them running, and redistribut them.

The instructor chose tool demonstration papers, rather than full techncial papers, for practical reasons. Full technical papers may not present a tool; for instance, a purely qualitative study that reports on empirical findings may have no software to go along with it. In contrast, tool demo papers almost certainly had a working tool when the paper was presented at a conference.

There were several learning goals of the course project:

- Gain deep experience with several state-of-the-art software enginering tools, and broad overview of many others;
- Effectively read research papers;
- How to build virtual machines that contain custom software;
- How to script virtual machine creation; and
- Oral communication skills.

In the remainder of this section, we describe project activities, requirements, and deliverables.

[1] https://www.csc.ncsu.edu/academics/graduate/degrees/se.php

[2] Syllabus: TODO

[3] https://courses.cs.washington.edu/courses/cse503/

4.2.1 Team Formation, Tool Selection, and Tool Assignment

4.2.2 Obtaining Tools

4.2.3 Getting the Tools Working

4.2.4 Presenting the Tools

4.2.5 Getting the Tools Working in a Virtual Machine

4.2.6 Building the Virtual Machine Automatically

5. PROCEDURE

No cost, but demos ok. (How many did this happen with?)

Tools where tool could convievably be put into a virtual machine. How many didn't fit?

6. RESULTS

6.1 RQ1 and RQ2

6.1.1 Effort

Average person-hours per tool, distribution (min, max, box plot)

Time invested in evaluating non-working tools.

6.1.2 Challenge: Tool Cannot Be Obtained

What percent of tool links were dead? What percent of tools said they were available in the paper, but the tool could not be obtained? What percent of tools were being planned to be commercialized? What were actually commercial? What percent of tools could we use if we had paid for them?

6.1.3 Challenge: Disappearing Tools

One author said tool just doesn't exist anymore.

Another author had to dig tool out of long term archive.

Several authors (what percent?) has tools hosted on Google code, even though it was dying. Did we save 'em?

6.1.4 Challenge: Author non-responsive

6.1.5 Challenge: Technical Difficulties

6.1.6 Challenge: Inconsistencies

One tool required to different VMs because two features needed different prereqs.

Some tools versioned differently.

6.2 RQ3 and RQ4

6.2.1 Challenge: Tool Licensing

University grey area.

6.2.2 Challenge: Technology Stack Licensing

6.2.3 Challenge: Author Doesn't Want Redistribution

Even when tool is available

7. CONCLUSIONS

Some other things.

8. ADDITIONAL AUTHORS

Additional authors: Shabbir Abdul (sabdul@ncsu.edu), Varun Aettapu (vaettap@ncsu.edu), Sumeet Agarwal (sagarwa6@ncsu.edu), Sindhu Anangur Vairavel (sanangu@ncsu.edu), Rishi Avinash Anne (raanne@ncsu.edu), Haris Mahmood Ansari (hmansari@ncsu.edu), Ankit Bhandari (abhanda3@ncsu.edu), Anand Bhanu (bhanua@ncsu.edu), Aditya Vinayak Bhise (avbhise@ncsu.edu), Saikrishna Teja Bobba (sbobba3@ncsu.edu), Vineela Boddula (vboddul@ncsu.edu), Venkata Krishna Sailesh Bommisetti (vbommis@ncsu.edu), Dwayne Christian (Chris) Brown (dcbrow10@ncsu.edu), Peter Morgan Chen (pmchen@ncsu.edu), Yi Chun (Yi-Chun) Chen (ychen74@ncsu.edu), Nikhil Chinthapallee (nchinth@ncsu.edu), Karan Singh Dagar (kdagar@ncsu.edu), Joseph Decker (jdecker@ncsu.edu), Pankti Rakeshkumar Desai (prdesai2@ncsu.edu), Jayant Dhawan (jdhawan2@ncsu.edu), Yihuan Dong (ydong2@ncsu.edu), Sarah Elizabeth Elder (seelder@ncsu.edu), Shrenuj Gunvant Gandhi (sgandhi4@ncsu.edu), Jennifer Michelle Green (jmgree17@ncsu.edu), Mohammed Hasibul Hassan (mhhassan@ncsu.edu), Satish Inampudi (sinampu@ncsu.edu), Pragyan Paramita Jena (ppjena@ncsu.edu), Bhargav Rahul (Bhargav) Jhaveri (bjhaver@ncsu.edu), Apoorv Vijay Joshi (avjoshi@ncsu.edu), Nikhil Josyabhatla (njosyab@ncsu.edu), Sujith Katakam (skataka@ncsu.edu), Juzer Husainy Khambaty (jhkhamba@ncsu.edu), Aneesh Arvind Kher (aakher@ncsu.edu), Craig Kimpel (ckimpal@ncsu.edu), Siddhartha Kollipara (skollip@ncsu.edu), Asish Prabhakar Kottala (akottal@ncsu.edu), Abishek Kumar (akumar21@ncsu.edu), Harini Reddy Kumbum (hkumbum@ncsu.edu), Nitish Pradeep Limaye (nplimaye@ncsu.edu), Apoorv Mahajan (amahaja3@ncsu.edu), Sai Sindhur Malleni (smallen3@ncsu.edu), Sudha Manchukonda (smanchu@ncsu.edu), Kavit Maral Mehta (kmmehta@ncsu.edu), Justin Alan Middleton (jamiddl2@ncsu.edu), Ramakant Moka (rmoka@ncsu.edu), Eesha Gopalakrishna Mulky (egmulky@ncsu.edu), Gauri Naik (gnaik2@ncsu.edu), Shraddha Anil Naik (sanaik2@ncsu.edu), Yashwanth Nallabothu (ynallab@ncsu.edu), Yogesh Nandakumar (ynandak@ncsu.edu), Kairav Sai Padarthy (kspadart@ncsu.edu), Pulkesh Kumar Yadav Pannalal (ppannal@ncsu.edu), Sattwik Pati (spati2@ncsu.edu), Kahan Prabhu (kprabhu@ncsu.edu), Shashank Goud Pulimamidi (spulima@ncsu.edu), Gargi Sandeep Rajadhyaksha (gsrajadh@ncsu.edu), Priyadarshini Rajagopal (prajago4@ncsu.edu), Venkatesh Sambandamoorthy (vsamban@ncsu.edu), Mohan Sammeta (msammet@ncsu.edu), Shaown Sarker (ssarker@ncsu.edu), Anshita Sayal (asayal@ncsu.edu), Vrushti Kamleshkumar Shah (vkshah@ncsu.edu), Esha Sharma (esharma2@ncsu.edu), Saurav Shekhar (sshekha3@ncsu.edu), Sarthak Prabhakar Shetty (spshetty@ncsu.edu), Manish Ramashankar Singh (mrsingh@ncsu.edu), Ankush Kumar Singh (asingh21@ncsu.edu), Vinay Kumar Suryadevara (vksuryad@ncsu.edu), Sumit Kumar Tomer (sktomer@ncsu.edu), Akriti Tripathi (atripat4@ncsu.edu), Jennifer Tsan (jtsan@ncsu.edu), Vivekananda Vakkalanka (vvakkal@ncsu.edu), Alexander Valkovsky (avalkov@ncsu.edu), Rishi Kumar Vardhineni (rkvardhi@ncsu.edu) and Manav Verma (mverma4@ncsu.edu).

9. REFERENCES

[1] C. Collberg, T. Proebsting, and A. M. Warren. Repeatability and benefaction in computer systems

research. Technical report, University of Arizona, 2015. TR 14-04.

[2] C. Collberg and T. A. Proebsting. Repeatability in computer systems research. *Communications of the ACM*, 59(3):62–69, 2016.

[3] A. Desai, S. Gulwani, V. Hingorani, N. Jain, A. Karkare, M. Marron, S. R, and S. Roy. Program synthesis using natural language. In L. K. Dillon, W. Visser, and L. Williams, editors, *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pages 345–356. ACM, 2016.

[4] M. D. Ernst, A. Lovato, D. Macedonio, F. Spoto, and J. Thaine. Locking discipline inference and checking. In *ICSE'16, Proceedings of the 38th International Conference on Software Engineering*, Austin, TX, USA, May 18–20, 2016.

[5] J. M. González-Barahona and G. Robles. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering*, 17(1-2):75–89, 2012.

[6] D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–488, 2012.

[7] C. Klein, J. Clements, C. Dimoulas, C. Eastlund, M. Felleisen, M. Flatt, J. A. McCarthy, J. Rafkind, S. Tobin-Hochstadt, and R. B. Findler. Run your research: on the effectiveness of lightweight mechanization. In *ACM SIGPLAN Notices*, volume 47, pages 285–296. ACM, 2012.

[8] J. Kovacevic. How to encourage and publish reproducible research. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1273. IEEE, 2007.

[9] X. Lu, X. Liu, H. Li, T. Xie, Q. Mei, D. Hao, G. Huang, and F. Feng. Prada: prioritizing android devices for apps by mining large-scale usage data. In *Proceedings of the 38th International Conference on Software Engineering*, pages 3–13. ACM, 2016.

[10] A. Meneely, L. Williams, and E. F. Gehringer. Rose: a repository of education-friendly open-source projects. In *ACM SIGCSE Bulletin*, volume 40, pages 7–11. ACM, 2008.

[11] M. Pedroni, T. Bay, M. Oriol, and A. Pedroni. Open source projects in programming courses. *ACM SIGCSE Bulletin*, 39(1):454–458, 2007.

[12] V. Stodden. Enabling reproducible research: Licensing for scientific innovation. *Int'l J. Comm. L. & Pol'y*, 13:1, 2009.

[13] P. Vandewalle, J. Kovačević, and M. Vetterli. Reproducible research in signal processing. *Signal Processing Magazine, IEEE*, 26(3):37–47, 2009.