Repeatability and Benefaction in Computer Systems Research

—

A Study and a Modest Proposal

University of Arizona TR 14-04

Christian Collberg collberg@gmail.com
Todd Proebsting proebsting@cs.arizona.edu
Alex M Warren amwarren@email.arizona.edu

February 27, 2015

Abstract

We describe a study into the extent to which Computer Systems researchers share their code and data and the extent to which such code builds. Starting with 601 papers from ACM conferences and journals, we examine 402 papers whose results were backed by code. For 32.3% of these papers we were able to obtain the code and build it within 30 minutes; for 48.3% of the papers we managed to build the code, but it may have required extra effort; for 54.0% of the papers either *we* managed to build the code or the authors stated the code would build with reasonable effort. We also propose a novel *sharing specification* scheme that requires researchers to specify the level of sharing that reviewers and readers can assume from a paper.

1 Introduction

Reproducibility is a cornerstone of the scientific process: only if my colleagues can reproduce my work should they trust its veracity. In the wet sciences reproducing someone's experiment can be difficult, often involving expensive laboratory equipment and elaborate processes. In applied Computer Science, however, this should seldom be so: unless esoteric hardware were used, reproducing the work published in a systems conference or journal should be as simple as going to the authors' website, downloading their code and data, typing "make," and seeing if the results correspond to the published ones.

Anecdotally, this is not so. In our own experience we have multiple times contacted the authors of a particularly interesting paper, asked for source code, either to be met with a negative response or stone cold silence. We present two such anecdotes in Appendix A. It is our belief, and the beliefs of many [10, 21, 28], that not sharing ones work with fellow researchers hampers the progress of science and leads to needless replication of work and the publication of potentially flawed results.

In order for Computer Science research to be reproducible, several hurdles have to be cleared: the source code and test case data have to be available, the code has to build, the execution environment has to be replicated, the code itself has to run to completion, and accurate measurements (with respect to performance or other metrics) have to be collected. In this paper we concentrate on the two most basic of these requirements: *is the source code available, and does it build?*

Not only researchers themselves, but the agencies who fund them and, ultimately, the general tax-paying public, have an interest in research being reproducibible and research artifacts being

freely shared. The government's newfound interest in the state of reproducible science is demonstrated by this recent *request for comments* put out by the White House [9]:

> Given recent evidence of the irreproducibility of a surprising number of published scientific findings, how can the Federal Government leverage its role as a significant funder of scientific research to most effectively address the problem?

1.1 This Study

To investigate the extent to which Computer Science researchers share their code and data, and the extent to which this code will actually build with reasonable effort, we performed the following study. We downloaded 601 papers from the latest incarnations of eight ACM conferences (ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12) and five journals (TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34), all with a practical orientation. For each paper we determined whether the published results appeared to be backed by code or not. Next, we examined each such paper to see whether it contained a link to downloadable code. If not, we examined the authors' websites, did a web search, examined popular code repositories such as `github` and `sourceforge`, to see if the relevant code could be found. In a final attempt, we emailed the authors of each paper for which code could not be found, asking them to direct us to the location of the source. In cases when code was eventually recovered, we also attempted to build and execute it. At this point we stopped—we did not go as far as to attempt to verify the correctness of the published results.

The details of how this study was conducted can be found in Section 3 and the outcomes can be found in Section 4. Our conclusion is *not* to blame authors for an inability to share code—we discovered many cases where such sharing was made impossible by licensing or commercial constraints. Similarly, our intent is *not* to blame authors for instances when we were unable to build their code—we discovered that research software often have complicated dependencies on specfic versions of external libraries, compilers, and other software, and sometimes require domain expertise to build and operate. Rather, our study is investigative in nature; our goal is to find out the extent to which researchers share their code, the effort required to build such codes, and to propose procedures to improve sharing and, hence, repeatability, in Computer Systems research.

Related work, described in Section 2, falls in three categories: some authors describe the steps that need to be taken in order to produce research that is truly reproducible; some authors describe tools and websites that support the publication of reproducible research; and some authors propose licensing frameworks that ensure that researchers who willingly share their artifacts will receive propper attribution. Our recommendations (detailed in Section 5) are more modest. We recognize that, as a discipline, we are a long way away from producing research that is always, and completely, reproducible. But, in the interim, we can require authors to conscientiously inform us of their intent with respect to publishing their code and data: will it be made available, how will it be made available, in what form will it be made available, on what platforms will it run, what support will be provided, etc. We believe this information should be provided by the authors *when their work is submitted for publication*, allowing reviewers to account for the expected level of reproducibility in their decision to accept or reject.

Thus, this paper makes the following contributions: we describe a process for determining the level of *weak repeatability* in Computer Systems research (Section 3), we describe the outcome of a repeatability study of 601 papers published in top ACM conferences and journals (Section 4), and we make a recommendation for adding *sharing specifications* to publications that represent a contract between authors and their readers as to the promised level of sharing (Section 5).t

All the code and data from this project can be downloaded from our website[1].

1.2 History of this Document

The first version of this document [7] was placed on our web site around December 10, 2013, along with the raw data collected from our study. This was done in order to support a journal submission of a shorter version of the technical report. Though never publically announced, the existence of our study leaked to the community, and resulted in sprited discussions on social media platforms such as Facebook and Twitter. We also received direct feedback from authors over email.

While most of the feedback was supportive of the goals of the study, much critique was directed at the methodology we employed. In particular, there was a feeling that restricting the student workers to 30 minutes to build each downloaded software package was too limiting. A common argument was that such a short time is unreasonable for software written in a research setting and that "your students should have tried harder."

While we do not necessarily agree with this argument (many of the packages we downloaded *did* build cleanly out-of-the-box, and it is unclear to us why, as a community, we should have lower standards than that), it was clear that some of our negative results were the consequence of the inexperience of our student workers[2] and the limited time and direction they were given by us. We therefore embarked on *Phase 4* of the study (Phase 1-3 being download and analyze papers, send email requests to authors, and build and run the code under a tight time constraint), and hiring new students, and engaging the services of a post doctoral scholar, giving them "unlimited" amount of time to try to build and more hands-on direction. This process proceeded over the summer and fall of 2014.

In parallel with this fourth phase of the study, a group of independent researchers set out to verify our results, through a crowdsourced effort[3]. Their effort, and ours, essentially attempted the same task: we both revisited our failed builds from Phase 3 of the study.

To further verify the data we collected, in the fall of 2014 we distributed a questionnaire to all authors, asking them to comment on the data we gathered about their paper. We call this *Phase 5* of the study.

This revised document reflects the Phase 4 and 5 efforts. We have also made many minor changes throughout the document. For example, to reflect the terminology put forth by Vitek and Kalibera [30], we have replaced "reproducibility" with "repeatability."

1.3 Definitions

In Computer Systems, research is typically carried out by building a system, running experiments over a set of benchmarks, measuring certain properties (such as performance, energy use, etc.), and publishing the results. We say that this research is *repeatable* [30] (see Figure 1) if you are able to run the same system, under the same environment, on the same benchmarks, with the same methodology, and are able to verify that you get the same results:

Definition 1 (Vitek [30]) Repetition *is the ability to re-run the exact same experiment with the same method on the same or similar system and obtain the same or very similar result.*

A reviewer, for example, may want to repeat the experiments in a submitted paper before recommending it for publication, to ensure that, at the very least, the experiments produce the data

[1] `http://reproducibility.cs.arizona.edu`.

[2] As Principal Investigators we, of course, take full responsibility for the study.

[3] `http://cs.brown.edu/~sk/Memos/Examining-Reproducibility`

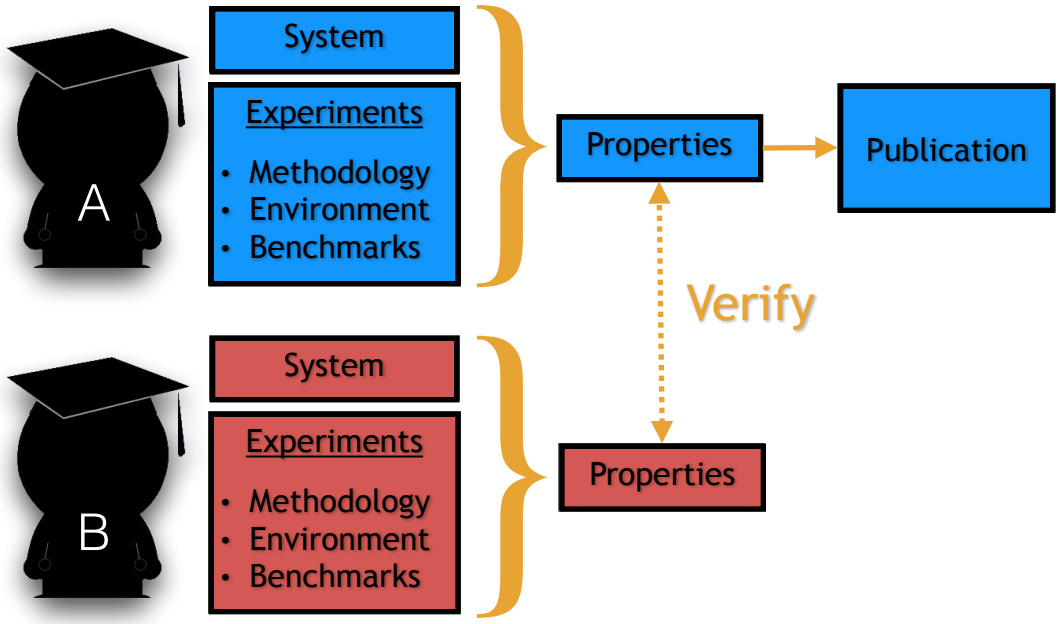


Figure 1: Repeatability. A builds a system, runs experiments on the system (using a particular methodology, in a particular computing environment, using a particular set of benchmarks), measures a set of properties (time, space, energy, etc.), and prepares the results for publication. B runs the same system, on the same experiments, in the same environment, in order to verify the properties in the paper.

presented in the paper. Similarly, a reader might repeat the experiments if he finds the published data suspicious.

A test for *reproducibility* [30] (see Figure 2), on the other hand, is carried out *after* publication, by running different experiments on a different system, and getting results that verify the claims in the original paper:

Definition 2 (Vitek [30]) Reproducibility *is the independent confirmation of a scientific hypothesis through reproduction by an independent researcher/lab. The reproductions are carried out after a publication, based on the information in the paper and possibly some other information, such as data sets, published via scientific data repositories or provided by the authors on inquiry.*

In practice, in Computer Systems, so many choices are made in the design and implementation of the system that never make it into the published paper, that completely understanding the claims and arguments in the paper without access to the original system may be impossible.

Benefaction (see Figure 3), finally, is to share research artifacts with your colleagues, in order for them to use it in their own work, to make new discoveries:

Definition 3 Benefaction *is the avoidance of needless replication of work in order to better advance scientific progress.*

In addition to sharing code, benefaction can also include the sharing of benchmarks and data sets, etc.

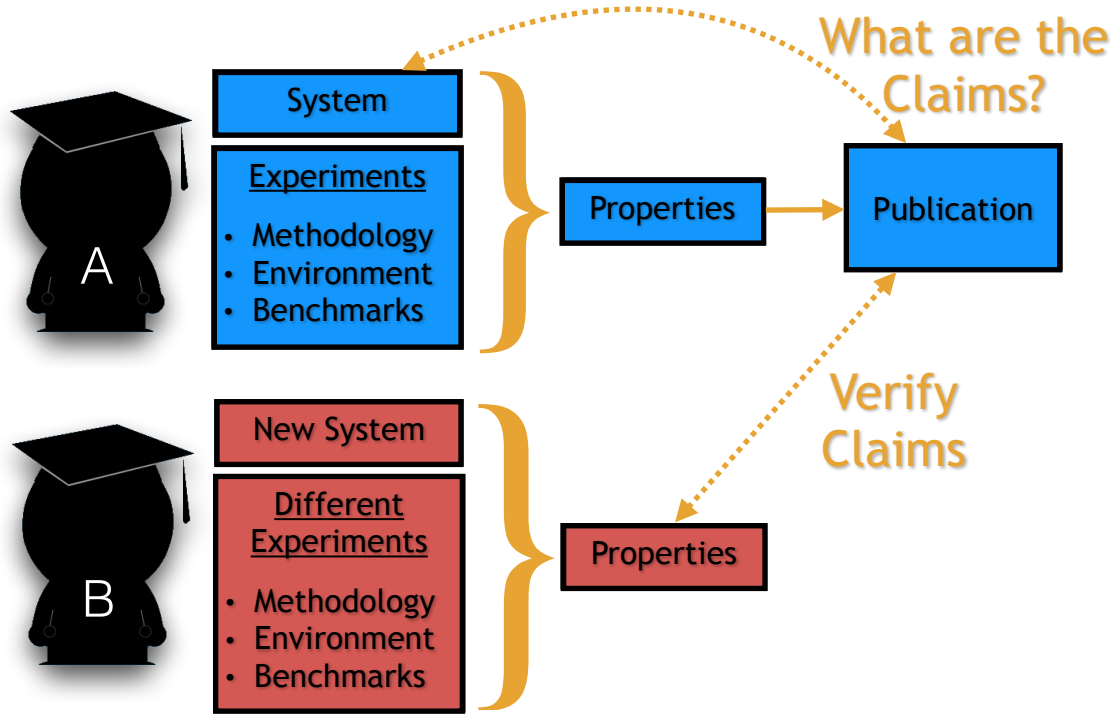


Figure 2: Reproducibility. B builds a different system than A, and runs different experiments, in a different environment using a different set of benchmarks, collecting a different set of properties, using these properties to verify the claims made in the paper.

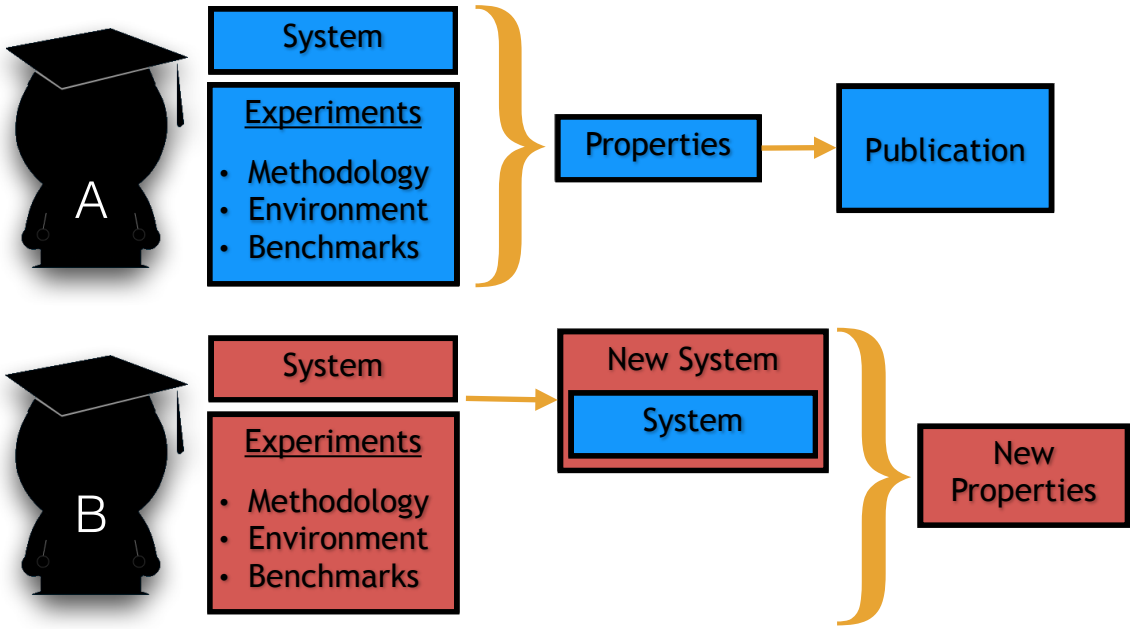


Figure 3: Benefaction. B uses the system (and benchmarks, etc.) originally produced by A to build his own system. The motivation for A's and B's systems, and the properties they derive from them, may be completely different.

1.4 Weak Repeatability

When you set out to measure the rate of repeatability in a scientific community, the definitions above are not precise enough. Not only do you need a definition that resonates with the community that is being measured but also one that is practical enough to collect data given the resources at your disposal. Even minor changes to the definition are likely to affect the outcome of the study.

For the purposes of this study, we defined three measures of what we have come to call *weak repeatability*:

Definition 4 (Weak Repeatability) *Let*

- BC *be the number of papers whose results are backed by code;*
- $\text{OK}^{\leq 30}$ *be the number of systems our research assistants successfully built in* ≤ 30 *minutes;*
- $\text{OK}^{>30}$ *be the number of systems our research assistants built in* > 30 *minutes;*
- OK^{Auth} *be the number of systems for which our research assistants were unable to build the system, but the author says the code builds with "reasonable effort."*

Then

$$\begin{aligned} \textit{Weak repeatability rate } A &= \frac{\text{OK}^{\leq 30}}{\text{BC}} \\ \textit{Weak repeatability rate } B &= \frac{\text{OK}^{\leq 30} + \text{OK}^{>30}}{\text{BC}} \\ \textit{Weak repeatability rate } C &= \frac{\text{OK}^{\leq 30} + \text{OK}^{>30} + \text{OK}^{\text{Auth}}}{\text{BC}} \end{aligned}$$

Weak repeatability A measures the extent to which code builds more or less out-of-the-box. In 30 minutes, our research assistants were able to unpack the code, follow directions in a README file (if provided), download and install compilers and libraries (if obvious from the README file), and resolve minor issues such as removing hardwired paths in makefiles. Weak repeatability A models scenarios where limited time is available to examine an artifact, and when communicating with the author is not possible. A reviewer of a conference paper examining a submitted research artifact is one example.

Weak repeatability B models situations where ample time is available to resolve issues, but the lead developer of the code is no longer available for consultation. The latter turns out to be quite common: we saw situations where the student responsible for development had graduated, when the main developer had passed away, when the authors' email addresses no longer worked, or when the authors expressed that they were too busy to provide assistance.

Weak repeatability C, finally, measures the extent to which we are able to build the code or the authors believe their code builds with reasonable effort. This models the situation where ample time is available to examine the code, and where the authors are responsive to requests for assistance.

1.4.1 Caveats

Definition 4 leaves out one crucial aspect of repeatability: versioning. In order to repeat the results in a paper the exact version of the code that was used to produce those results must be available. In this study, it was often not clear to us whether the code we gained access to was identical to the code that generated the results in the published paper, or was a later version with bug fixes, etc.

In some cases, the authors themselves were also unsure. We therefore leave versioning out of our definition.

In our study we classified papers as belonging to one of three categories: backed by code, not backed by code, and requiring non-standard hardware. We only considered papers in the first category. Thus, Definition 4 only applies to papers whose result is backed by code that runs on stock hardware.

Definition 4 refers explicitly to *our* research assistants, who were mostly undergraduates students of Computer Science or Engineering at the University of Arizona, and who also included one Master's student of Electrical and Computer Engineering and one Postdoctoral Researcher. Whether someone is able to resolve issues that come up when building a particular piece of software is dependent on their general abilities, their background and training (specifically, their familiarity with the operating system, programming language, programming environment, and build system assumed by the software), and their previous experience with this particular type of software. Thus, the results presented in this report have to be taken in context not only of *how* the study was carried out, but also by *whom.*

We initially considered including *executability* in the definition, i.e. we wanted to see if we could not only build the code but also run it. However, during the study we often found that it was not possible for us to determine the cause of a program crash: it could be because of internal issues, because the program was invoked with the wrong arguments, or because a file was missing, etc. Without being familiar with the actual use of the software—which our research assistants were not—it is not easily determined whether a program runs properly or not. For this reason, we leave executability out of the definition.

2 Related Work

In this section we review some of the prior work related to reproducibility. In particular, we examine previous studies into reproducibility, tools for supporting reproducible research, and legal and licensing frameworks.

2.1 Empirical Studies into Reproducibility

In this paper we conduct a comprehensive study into the extent to which Computer Systems researchers will share their code with colleagues. Two similar studies have been conducted in the past. In 2007 Kovacevic [17] examined 15 papers published in the *IEEE Transactions on Image Processing*. She read the papers and rated them on how well algorithms were explained and whether code and data were available. She found that while all algorithms had proofs, none had code available, and 33% had data available.

Vandewalle et al. [29] distinguish 6 levels of reproducibility, starting at level 5 where *"The results can be easily reproduced by an independent researcher with at most 15 min of user effort, requiring only standard, freely available tools (C compiler, etc.)"*, to level 4 where *"proprietary source packages (MATLAB, etc.)"* are required, down to level 0 where *"The results cannot be reproduced by an independent researcher."* They then repeated the study from Kovacevic [17] with a larger sample size, all the 134 papers published in *IEEE Transactions on Image Processing* in 2004. Each paper was scored on its reproducibility by two or three reviewers. They found that *"code (9%) and data (33%) are only available online in a minority of the cases, with data being available more often thanks to the frequent use of standard image data sets, such as Lena."*

While the Kovacevic and Vandewalle studies consider many aspects of reproducibility, such as whether algorithms (subjectively) are described clearly enough to be independently implemented,

our study focuses entirely on whether code and data are made available. We also consider a larger sample size (601 papers) and a larger number of journals and conferences (13). We furthermore attempt to actually build and execute the source and contact the authors of code not found on public repositories to see if they would provide it when asked directly.

Stodden [26] reports on a survey of 638 registrants at the NIPS machine learning conference as to their willingness to reveal code, data, and ideas. The survey showed that 74% were willing to share post-publication code, and 67% post-publication data. In contrast, Stodden found that "30% of respondents shared some code and 20% shared some data on their own websites." The most common reasons for not sharing code were found to be "The time it takes to clean up and document for release," "Dealing with questions from users about the code," "The possibility that your code may be used without citation," "The possibility of patents, or other IP constraints," and "Competitors may get an advantage."

2.2 Public Repositories of Code and Data

Stodden et al.'s [27] RunMyCode web site[4] provides "computational infrastructure" allowing authors to publish the code and data related to their papers. In addition to making code and data available for download, the site also allows users to execute the code in the cloud. Unfortunately, the *"if you build it they will come"* paradigm does not always work: so far (November 2014) only 82 papers appear to have been registered on the site, none of them in the Computer and Information Sciences category. ResearchCompendia[5], the open source version of RunMyCode, has a total of 235 artifacts registered (November 2014).

Gavish and Donoho [11] describe a system whereby data and scripts are stored on (journal) web repositories, and every computational result (tables, figures, charts, datasets, etc.) is given a *Verifiable Result Identifier* (VRI). VRIs are included directly into the article source, allowing reviewers and readers to interact with the scripts and data (verifying results, examine data, re-execute computations) without having to explicitly download anything. Gorp and Mazanek [13] describe a similar system, called SHARE, a web site that allows authors to share remote virtual machines that contain a complete execution environment, code, data, and the paper text itself. Readers can click on links in a paper to connect to these virtual machines where they can interact with computational artifacts. Using VMs to ensure reproducibility is an attractive proposition since it allows code and data to be bundled with the exact versions of operating system and libraries with which experiments were run [22]. However, it comes with its own problems, such as how to perform accurate performance measurements, and how to future-proof computations: for how long will there exist VM monitors that will run my VM image, and for how long will it be safe to run an image that contains an operating system and applications to which security patches have not been applied?

Paper Mâché [6] is yet another system that supports the *executable paper*[6] concept, also using virtual machine technology to replicate the execution environment in which scientific computations were run.

CARMEN [5] (Code, Analysis, Repository and Modelling for e-Neuroscience) is a web based portal which allows users to share code and data in neuroscience.

Koop et al. [16] describe a provenance-based system that captures workflows, i.e. exact histories of how data was captured, experiments performed, and results were generated. Similarly to the systems described above, clicking on a plot in a paper will open up the original workflow from which

[4] `http://RunMyCode.org`.

[5] `http://researchcompendia.org`.

[6] `http://www.executablepapers.com`.

Table 1: Outcome of Artifact Evaluations.

Venue	Accepted Papers	Submitted Artifacts	Accepted Artifacts
ESEC/FSE 2011 [1]	34	?	7
ECOOP 2013[a]	29	9	8
OOPSLA 2013[b]	50	21	18
ESEC/FSE 2013	51	22	12
SAS 2013 [2]	23	22	11 ?
ISSTA 2014[c]	36	9	7
HSCC 2014[d]	29	?	5
ECOOP 2014 [14]	27	13	11
OOPSLA 2014	?	?	?
PLDI 2014 [3]	52	20	12

[a] Erik Ernst, personal communication.
[b] `http://evaluate.inf.usi.ch/artifacts/aea/statistics`
[c] Milos Gligoric, personal communication.
[d] `https://sites.google.com/site/hscc2014repeatability/home/results`

it was generated. Additionally, the system described by Koop et al. integrates version control so that authors can revert to previous versions of data and code.

Muller et al.'s [4] A-R-E (Author-Review-Execute) environment provides similar facilities as in the systems described above, but also provides anonymization during the review process. The idea is to place an anonymizing proxy between the reviewer and the author's web service that provides access to data and code.

2.3 Processes, Procedures, and Publication Practices

Schwab et al. [23] describe how to use `make` and related tools along with naming conventions for input and generated files to ensure that readers as well as authors can reproduce computations.

Lepton [18] is a literate programming system that makes papers self-contained, by integrating code and data within the paper itself. Gentleman [12] presents a similar idea based on literate programming, specifically for the R language.

Limare [19] describes experiences gained from editing the journal *Image Processing On Line* where "Each article contains a text describing an algorithm and source code, with an online demonstration facility and an archive of online experiments. The text and source code are peer-reviewed and the demonstration is controlled." In particular, he points out the risk of obsolescence of code stored in the repository, as programming languages and libraries change over time.

Several conferences run an "artifact evaluation process" independently of the paper submission evaluation.[7] Table 1 summarizes the outcome of the process for the 10 conferences that participated during the years 2011–2014. For example, in the first instance of the PLDI conference that included an artifact evaluation, PLDI 2014, the conference accepted 52 papers. Of these, 20 submitted artifacts for evaluation, and of these, 12 were classified as "above threshold."[8] Since only accepted papers may submit an artifact, the outcome of the artifact evaluation cannot influence whether a paper is accepted or not. The evaluation is typically carried out by PhD students and postdocs (22, in case of PLDI 2014).

[7] `http://www.artifact-eval.org`.
[8] `http://pldi14-aec.cs.brown.edu`.

2.4 Licensing Frameworks

Stodden [24, 25] has proposed "The Open Research License" to incentivize researchers to share their code and data without fear that their work will be expropriated by others, without proper attribution. Her licensing scheme combines the Creative Commons license[9] for the media portion of the research and the BSD[10] license for the code portion. The hope is that universal use of such licensing schemes will "encourage reproducible scientific investigation, facilitate greater collaboration, and promote engagement of the larger community in scientific learning and discovery" by ensuring "each scientist is attributed for only the work he or she has created" [24].

2.5 Reproducibility vs. Repeatability

In this paper we examine what we call *weak repeatability*, the extent to which computer code used to create the research results in published papers can be located, and the extent to which it builds. While access to buildable code is a prerequisite for an attempt at repeating the work, such a study is, of course, a far cry from an actual attempt at *reproducing* the work. It was been shown, across many disciplines, that attempts at reproducing well-known results often fail. Such studies appear to be less common in Computer Science than in other disciplines, but we mention here work by Klein et al. [15] who, through formal modeling and mechanized reasoning, found mistakes in nine papers published in the ICFP 2009 proceedings.

3 Process

Figure 4 shows the process by which our study was conducted. It was carried out in five phases: download and analyze papers to find the location of any related code, email authors in cases where code could not be found, attempt to build and run the code under a tight time constraint, attempt to build and run the code without a time constraint, and survey the authors to validate the results of the study. These phases are described in detail below.

3.1 Study Design Goals, Constraints, and Ethics

At the beginning of the study we identified multiple potential issues related to bias and ethics:

Author bias. An author who is aware of the fact that they are taking part in study on repeatability might either be less likely to respond to email requests for artifacts (because responding would be seen as a waste of time), or *more* likely to respond (so as not to look bad).

Longitudinal bias. The skills of investigators will improve over time. Thus, artifacts that are examined towards the end of the study period might be treated better than similar artifacts examined early on.

Investigator bias. The more outrageous the results of this study (i.e. the fewer artifacts that could be located and built) the more *interesting* it is, and the more likely it is that the results will lead to a publication in a prestigious venue. Thus, there is an inherent risk of investigators favoring negative results over positive ones.

[9] `http://creativecommons.org/licenses`.

[10] `http://opensource.org/licenses/BSD-3-Clause`.

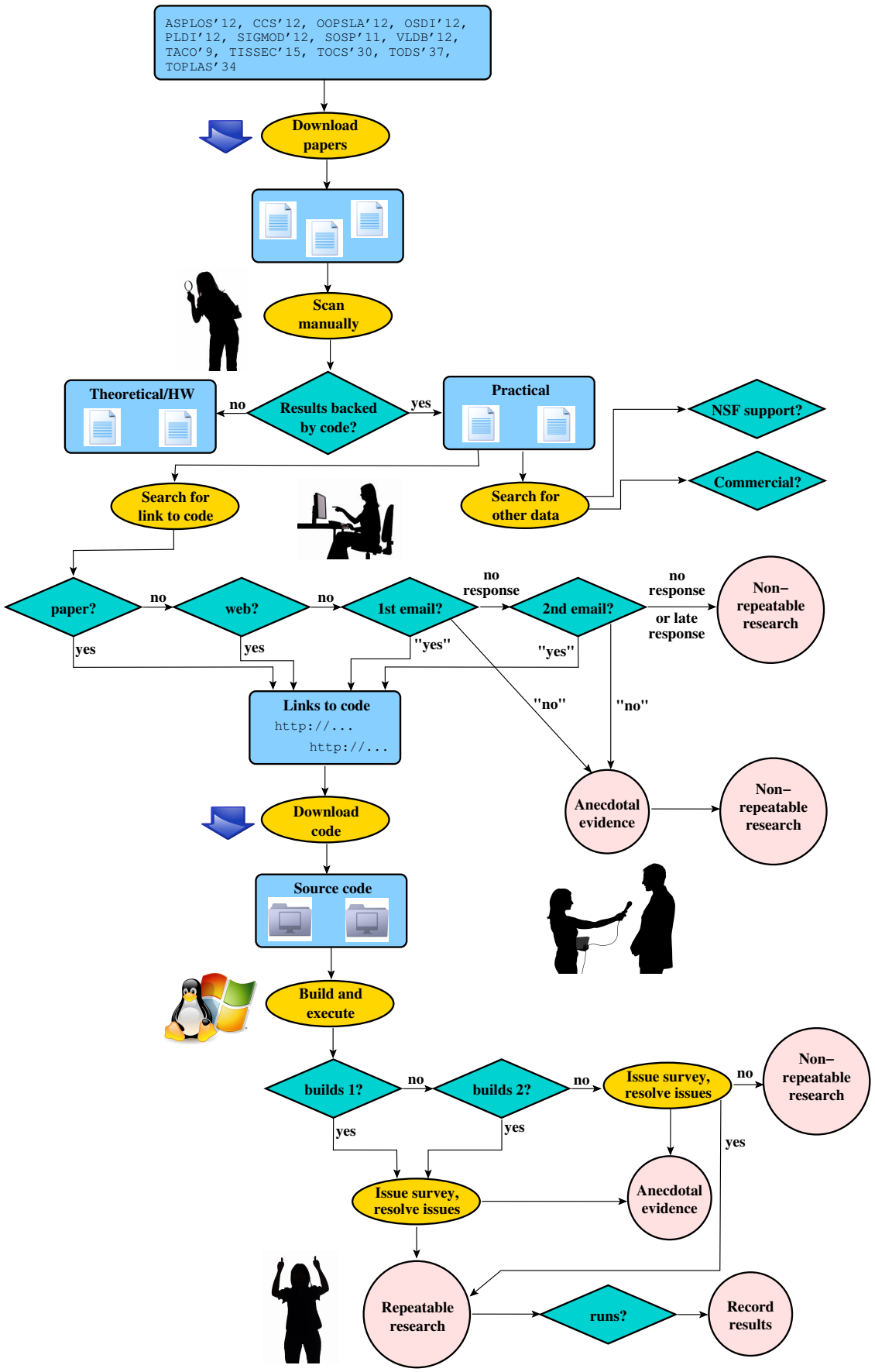


Figure 4: Process by which the study was performed.

Author workload. Since the study is asking authors to provide their artifacts under (somewhat) false pretenses, it is unethical to require them to expend an inordinate amount of time on tasks which, to them, have no value.

Deceitfulness. While it is essential to hide the existence of the study while it is ongoing, it is at the same time unethical to outright lie to authors about why their artifact is being requested.

Privacy vs. repeatability. A study that advocates the sharing of research artifacts cannot hide its own artifacts from the public. However, since this study deals not only with artifacts, but also the researchers who created them, it is important to take the privacy needs of the authors into account when deciding what to make public.

Student repercussions. Since we recognize that this study has the potential for becoming controversial, it is essential to isolate the student research assistants who carry out much of the work from any possible negative consequences.

In addition to issues of bias and ethics, the study also had to contend with very real resource constraints. A study such as this requires extensive use of human labor and we were constrained both with respect to monetary and human resources.

To avoid author bias and minimize author workload, we sent a single request for code, and one remainder if the first request received no response. Neither request revealed that it was sent as part of a repeatability study. To avoid finding ourselves in a situation where we had to make a choice between being untruthful and revealing the existence of the study, we consciously did not enter into a conversation with the authors, such as making requests for missing files or asking for help when a build failed. In addition to severely taxing our resources, such conversations would inevitably lead to the authors asking what we wanted to do with their code; a question we must answer truthfully, and this would have lead to the investigation being outed.[11]

We furthermore hypothesized that we would get different response rates depending on *who* sent email requests for code: a faculty member, a graduate student, an undergraduate student, a citizen outside academia, etc. We briefly considered sending the emails from a fake identity, such as a graduate student for whom we had created a department home page and LinkedIn and Facebook accounts, but rejected this idea as being unethical. In the end, all emails were signed by Collberg using an email address similar to, but not identical to, his own (`ccollberg@gmail.com` vs. `collberg@gmail.com`), and identified him as being from the University of Arizona. We do not know if any author bias resulted from these choices.

Avoiding longitudinal bias turned out to be difficult. Ideally, to minimize such bias, tasks should be assigned randomly to investigators. However, given our limited resources we chose not to do so, but rather gave each student the tasks they were best equipped to carry out. Clearly the experience and skill level of the students who carried out the investigation, and how tasks were assigned to them, affected our results.

To avoid investigator bias, we made a conscious effort of keeping the potential for such bias in the forefront of our minds and, whenever appropriate, deciding in favor of the authors ("their code builds").

Before commencing the first four phases of the study, we communicated (verbally) with a representative of the University of Arizona's Institutional Review Board (IRB). We were informed that our study did not need further action. Before sending out the author survey (see Section 3.6) we again investigated whether IRB approval would be necessary. We filled out the University of

[11] Even with this proviso we received at least one query asking what we planned to do with the code we requested. We answered vaguely, but truthfully.

```
ARTICLE:ANALYSIS_BY[name]
ARTICLE:ANALYSIS_TIME[minutes]
ARTICLE:COMMENT[string]
ARTICLE:COMMERCIAL_EFFORT[none,part,full]
ARTICLE:GRANT_SUPPORT[none or string]
ARTICLE:NSF-SUPPORT[none or number]
ARTICLE:IMPLEMENTATION_EXISTS[unknown,hardware,yes,no]
ARTICLE:LINK[url]
ARTICLE:NSF-SUPPORT[none or number]
ARTICLE:PAGE_COUNT[pages] 21
ARTICLE:STATUS[not_finished,finished]
ARTICLE:TYPE[conference,journal,poster,abstract]
AUTHOR:EMAIL_REAL[list of strings]
AUTHOR:NAMES[list of first_last]
BIBTEX:LABEL[string]
BIBTEX:LINK[url]
BUILD:ANALYSIS_BY[name]
BUILD:ANALYSIS_TIME[minutes]
BUILD:BENEFIT_OF_DOUBT[unknown,yes,no]
BUILD:COMMENT[string]
BUILD:ERROR_COMMENT[not_needed,comment]
BUILD:STATUS[one of {unknown,needed,not_needed,started,finished} and list of {downloaded,compiles,runs}]
BUILD:VERIFY_BY[name]
BUILD:VERIFY_COMMENT[string]
BUILD:VERIFY_STATUS[needed,not_needed,started,finished]
BUILD:VERIFY_TIME[minutes]
EMAIL1:CODE_AVAILABLE[yes,no,no_response]
EMAIL1:REMARK[comment]
EMAIL:STATUS[unknown,not_needed,not_found or list of {needed,request_1,response_1,sent_thank_you}]
PI:COMMENT_CC[string]
PI:COMMENT_TP[string]
TOOL:NAME[string]
TOOL:ARTICLE_LINK[unknown,none,url,broken and url]
TOOL:GOOGLE_LINK[unknown,none,url,broken and url]
TOOL:EMAIL_LINK[unknown,none,sent_no_url,url,broken and url]
TOOL:DATA_LINK[unknown,none,url,broken and url]
VERIFY:ANALYSIS_BY[name]
VERIFY:STATUS[unknown,needed,not_needed,started,finished]
VERIFY:COMMENT[string]
```

Figure 5: Textual form for the data collected for a paper.

Arizona's Office for Responsible Conduct of Research *Human Subjects Protection Program Form F309* which was then reviewed by a colleague in another department. It was concluded that IRB approval would not be necessary.

We decided to publish all the raw data from the study on our web site, and for each paper include the title, author names, results, and author feedback. Without this information there would be no way for the community to check our results. Email responses that were included in the technical report were anonymized, however, since the respondents had a reasonable expectation of privacy when responding to our messages.

To protect the student research assistants as much as possible, their contributions were anonymized (i.e. their names were removed from data files such as shown in Figure 5) prior to uploading them to our web site. Furthermore, they never communicated with authors under their own names.

3.2 Phase 1: Paper Analysis

In an initial step, each paper from the selected conferences and journals were downloaded from the ACM Digital Library or directly from the conference web site. The papers were next scanned manually by a team of undergraduate and graduate research assistants to determine if the results reported on were in any way backed by source code. Purely theoretical papers or papers that relied on hardware not available to us were not considered further. Keynote addresses, panels, and demonstrations where also excluded from further analysis. The resulting set of papers consisted

Figure 6: Editor to enter data during the study.

```
From: Christian Collberg <ccollberg@gmail.com>
To: first-or-corresponding-author
Cc: remaining-authors
Subject: Your conference-name paper

Dear Dr. first-or-corresponding-author,

I've been looking at your conference-name paper
   paper-title
and would like to try out the implementation. However,
I haven't been able to find it online. Would you please
let me know how I can obtain the source code so that I
can try to build and run it?

Thank you very much for your help!

Christian Collberg
ccollberg@gmail.com
Department of Computer Science
University of Arizona
```

Figure 7: First email.

mostly of full research articles (9 pages or more in length), but a few were shorter (2-4 pages), which included posters and abstracts from a doctoral symposium.

An attempt was next made to find links to any source code from the article body and bibliography. If no link was found, or if the paper contained a dead link, a web search was carried out via Google and popular code repository sites such as GitHub, Google Code, and SourceForge. Search queries combined the tool name and fields such as institute name, author name, and other keywords. If the institution or the authors had public web pages related to the project, those pages were considered as well.

In order to ensure accurate results, each paper was reviewed by two research assistants. In addition, random papers were further selected to be reviewed by a professor, to monitor the quality of the analysis and reviews. To further facilitate data entry and reduce the risk of erroneous data, a special data entry editor was developed, shown in Figure 6.

3.3 Phase 2: Email Requests

If no link to source code was found online or in the article itself, an email was sent to the authors to request access to the code. The emails had the structure shown in Figure 7. In order to avoid messages being flagged as spam or making the recipient suspicious as to the motive behind the emails, papers were removed from consideration so that the resulting set had no overlapping author lists. If no response was received within a few weeks, a reminder email was sent, shown in Figure 8.

Our emails were sent out in two batches, around June 19, 2013, and October 9, 2013. We did not accept responses after December 9.

In the following cases we marked a paper as "code not available:"

```
From: Christian Collberg <ccollberg@gmail.com>
To: first-or-corresponding-author
Cc: remaining-authors
Subject: RE: Your conference-name paper

Hi!

Sorry for bothering you again!

Can you please let me know if this implementation is available,
and from where I can obtain the code?

Thanks!
```

Figure 8: Reminder email.

- when the authors only made partial code available;
- when the authors only made binary releases available;
- when the authors promised they would "send code soon," but we didn't hear anything further;
- when the authors asked us to sign a license or non-disclosure agreement;
- when the authors requested author credit for any follow-up work; and
- when the authors sent us code more than two months after the original email request.

3.4 Phase 3: Fast Build

Each paper for which source code was found was assigned to one of the student research assistants. The code was usually tested under the newest Ubuntu environment, unless specified otherwise by instructions accompanying the source code. Java-based code was tested under Windows 7. Each build was given 30 minutes of programmer time (excluding download and install time) to complete. If, at the end of the 30 minutes, unresolved make or compilation errors remained, the system was marked as "did not compile." If the system compiled but did not run, it was classified as "compiles but does not run." Finally, if a build executed without errors, it was marked as "runs." No attempt was made to verify whether the system produced results consistent with the claims made in the original paper.

Given that the papers selected for analysis cover a wide variety of topics and circumstances, some papers did not fit nicely into the build process and required additional attention.

For example, one paper provided an entirely online implementation that executes with some errors. We marked that paper as "compiles but does not run." Another paper provided an API as a tool, and requires C++ to call and use. This was deemed to be out of the scope of the research. There were other papers that were out of scope as well. For example, one tool required installing a custom-provided operating system on an Android device to run, and another required downloading and modifying images for Android systems to run. In all such cases, we gave the authors the benefit of the doubt, and marked the paper was "compiles and runs."

Although often unclear, we assumed that the code we found was the version that produced the results in the paper.

A small number of systems had online implementations and thus did not need to be compiled. We marked these as "builds."

Overall we found these kind of occurrences to be rare and as such did not significantly impact the results of the study.

3.5 Phase 4: Extensive Build

If a code failed to build during Phase 3, we made a second attempt without setting an artificial time limit. Here, we made use of one undergraduate Computer Science student, one undergraduate Electrical and Computer Engineering student (mostly in an administrative role), and one postdoctoral researcher in Computer Science. We furthermore held weekly meetings where failed builds were discussed, communal debugging was engaged in, and further approaches were suggested. Only when all investigators agreed that they could see no further avenues of attack did we cease the effort, and the program was marked as "does not build."

3.6 Phase 5: Author Survey

After Phase 4 was completed we sent emails to all authors who were part of the survey asking them to help verify the data we had gathered. Authors of papers that were removed in Phase 2 (to ensure none-overlapping author lists) were excluded. We were particularly interested in whether

- we had classified the papers correctly,
- we had found the correct code,
- we had found the version of the code that corresponded to the results in the papers, and
- the authors themselves thought the code ought to build.

Figure 9 shows the email request that was sent out. Appendix C shows the survey questions. Section 4.4 discusses the feedback we received from the survey.

In response to the survey results we embarked on the final stage of the study. There were three major issues that we had to resolve:

1. In cases where we had misclassified a paper as "not backed by code" and the authors gave us access to code, we downloaded it and tried to build it, using the same process as before. Similarly, if our web searches had turned up the wrong code, and the authors provided a correct link, we downloaded and built this code.
2. In cases where it was clear from a survey response that there had been a misunderstanding between us and the authors, we communicated with them in order to clear up the problem.
3. In cases where the authors provided solutions to build issues we had had we *did not* attempt to build the code again. Rather, we recorded the fact that the authors said that the code would build.

Figure 10 shows the email we sent to authors to request code in reply to their survey response.

From: Christian Collberg <ccollberg@gmail.com>
To: **author**
Subject: Repeatability Study: Your **paper-venue** Paper

Dear author,

Over the last year we have conducted a study into the extent to which
Computer Systems researchers share their research artifacts, and
whether published code builds. Your **paper-venue** paper,

paper-title

was part of this study. In fact, you may have received an email
from us asking for code related to your paper, or you might
have come across our website
http://reproducibility.cs.arizona.edu
or even read our (very preliminary) technical report
http://reproducibility.cs.arizona.edu/v1/tr.pdf.

In our study we looked for code related to papers published in ACM
conferences and journals, and, if found, tried to build it. For your
paper, we found that
* the paper is
  1. **PRACTICAL (i.e. backed by code)**
  2. **THEORETICAL (not backed by code)**
  3. **HARDWARE (requires special hardware to reproduce)**
* 1. the code could be found online at **link-to-code**
  2. we received the code in an email from you
  3. no code could be located
* we were
  1. **able**
  2. **unable**

  to build the code.

The complete data we collected for your paper can be found here:
http://reproducibility.cs.arizona.edu/v2/#**this-paper**
The following document explains the data schema:
http://reproducibility.cs.arizona.edu/format.txt

To make sure that our results are as accurate as possible, we would
appreciate it if you would answer a very short online questionnaire
related to the data we collected about your paper:
https://www.surveymoz.com/s/124740RHYUP?x=**unique-ID**

Please complete this survey by September 21, 2014.

Note that this is not an anonymous survey. Rather, your response
to the questionnaire can be attributed to you. Also note that if your
paper has multiple authors, each author will get a copy of this email,
with a unique code that ties their response to them. Thus, each author
can provide us with a different response.

We very much appreciate your help,
Christian Collberg
Todd Proebsting

Figure 9: Survey email.

```
From: Christian Collberg <ccollberg@gmail.com>
To: all-authors
Subject: RE: Repeatability Study: Your paper-venue Paper

Dear authors!

Thank you for responding to our survey!

The main purpose of the survey was to be able to correct any
mistakes in our repeatability study. In the case of your paper,
    1. we failed to follow up on your original email
       response, and so your paper was erroneously
       classified as "code-does-not-exist".
    2. you point out that we found the wrong code!
    3. it was tagged as "not-backed-by-code" which, as
       you point out, was a misclassification.

We would be very grateful if you could send us the code that
was used to create the results in your paper, so we can attempt
to build and run it, and include the results in our study.

Thank you very much for your assistance,
Christian and Todd
```

Figure 10: Email to resolve issues in response to survey results.

3.7 Detailed Workflow

Three undergraduate and one graduate research assistant were tasked with downloading and scanning papers for links to code. For each paper the information in Figure 5 was collected. The research assistants went through the following workflow:

1. Get pdf and basic information:
   (a) Look for grant support. Search for the word "grant", usually at beginning or end of the paper, possibly in the footnote or acknowledgement sections.
   (b) Extract the email addresses of authors.
   (c) Determine if any authors come from a commercial entity. Look at authors' email address and grant/financial support.
   (d) Add a link to article.
2. Look through Abstract/Introduction:
   (a) Skim the text to get an idea of what the paper is about.
   (b) Find out what a possible tool might be called.
   (c) Determine if the paper is backed by an implementation (code).
   (d) Search the introduction for a link to code.
3. Search for "git," "code.google," "http," and "ftp" for links to code:
   (a) If you find a link in the citations search back for the citation number to see what it is.

4. Skim section headings and text for links to code:
    (a) Sections named “Implementation” or “Experiments” could have information on the tool.
5. Do a web search:
    (a) Search the name of the tool.
    (b) Search with relevant keywords.
    (c) Search each authors’ homepage and check if they link to the software.
    (d) Try to figure out what lab/research group the project belongs to and check the lab’s website.
    (e) If a slide deck related to the paper is found look for links in them (probably at end).
6. Send initial emails:
    (a) An initial email needs to be sent if: the paper is backed by code, and no link was found.
    (b) Exclude papers so that no more than one email is sent to any author.
    (c) Send a reminder email if a response to the first one is not received.
7. Build code 1:
    (a) Follow link and download code.
    (b) Attempt to build. Install any missing libraries. Fix any broken search paths.
    (c) Attempt to run.
    (d) Abort after 30 minutes.
8. Build code 2:
    (a) Attempt to build again.
    (b) Ensure the right versions of operating system, compilers, libraries, and frameworks have been installed.
    (c) Attempt to run.
    (d) If no success, consult with others on the project to get ideas on how to proceed, repeat from (a).
    (e) Abort when no more progress can be made.
9. Send survey emails:
    (a) Create a unique code for each paper/author-pair.
    (b) Send request email to each author, with a link to the survey containing the unique code.
    (c) Send a reminder email after a week to the authors of those papers for which no survey response has been received.
    (d) Collect the results.
    (e) Resolve issues. In some cases this will require emailing the authors for clarification, downloading code that was missed, and attempting to build that code, using the same process as before.

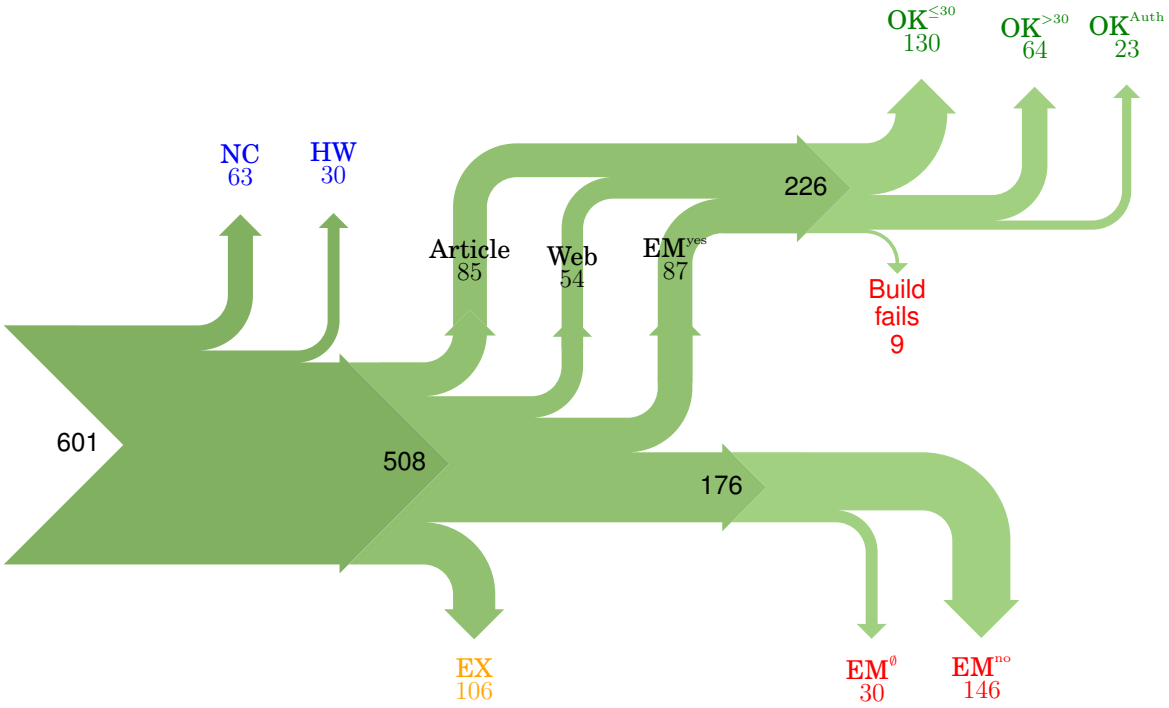


Figure 11: Study result. Blue numbers represent papers that were excluded from consideration, green numbers papers that are weakly repeatable, red numbers papers that are non-weakly repeatable, and orange numbers represent papers that were excluded (due to our restriction of sending at most one email to each author).

10. Notes:

    (a) If a link was found through a web search go back and check the paper again to make sure it was not there.

    (b) It can be complicated to determine when there is a larger project of which the current paper is a subset. In that case the paper may refer to the larger project as though it were a separate subject when in fact their current code is included with it.

4 Results

Table 2, Figure 11, and Appendix B show the results of the study. Table 4 lists the abbreviations we use.

Table 2 shows that out of an initial 601 papers, we excluded 30 because they required esoteric hardware, 63 because the results presented were not backed by code, and 106 in order to avoid sending multiple email requests to the same author, resulting in a total of 402 papers whose results were backed by code. Out of these, we found 85 codes through links in the paper itself, 54 codes through web searches, and 87 codes through email requests. For the remaining 176 papers backed by code we either got a negative response to our email requests, or no response within two months.

Our results show that for 32.3% of the papers backed by code we were able to obtain the code and, within ≤ 30 minutes, also build it (*weak repeatability A*); for 48.3% of the papers we managed to build the code, but it may have required extra effort (*weak repeatability B*); and for 54.0% of the papers either *we* managed to build the code or the authors stated the code would build with reasonable effort (*weak repeatability C*).

Table 2: Summary of the results of the study.

Group	#	Classification				Code location					Build Result				Weak Repeatability (%)		
		HW	NC	EX	BC	Article	Web	EM^{yes}	EM^{no}	$EM^{\emptyset}$	$OK^{\leq 30}$	$OK^{>30}$	OK^{Auth}	Fails	A	B	C
ASPLOS'12	36	4	2	7	23	2	0	6	14	1	4	3	1	0	17.4	30.4	34.8
CCS'12	75	5	14	19	37	4	4	15	12	2	16	5	2	0	43.2	56.8	62.2
OOPSLA'12	73	0	12	5	56	29	3	10	13	1	21	17	2	2	37.5	67.9	71.4
OSDI'12	24	0	0	7	17	4	4	2	7	0	7	3	0	0	41.2	58.8	58.8
PLDI'12	48	0	1	7	40	9	6	10	14	1	9	13	3	0	22.5	55.0	62.5
SIGMOD'12	46	1	0	19	26	3	6	8	8	1	11	3	3	0	42.3	53.8	65.4
SOSP'11	27	0	1	6	20	3	4	3	9	1	3	3	2	2	15.0	30.0	40.0
TACO'9	60	18	3	2	37	7	2	6	17	5	8	2	2	3	21.6	27.0	32.4
TISSEC'15	13	1	6	0	6	3	0	1	2	0	2	0	2	0	33.3	33.3	66.7
TOCS'30	13	1	0	0	12	3	2	0	5	2	2	3	0	0	16.7	41.7	41.7
TODS'37	29	0	12	2	15	1	3	3	3	5	6	1	0	0	40.0	46.7	46.7
TOPLAS'34	16	0	5	2	9	7	1	1	0	0	4	4	0	1	44.4	88.9	88.9
VLDB'12	141	0	7	30	104	10	19	22	42	11	37	7	6	1	35.6	42.3	48.1
Total	601	30	63	106	402	85	54	87	146	30	130	64	23	9	32.3	48.3	54.0
NSF	252	11	15	57	169	36	25	43	54	11	55	31	14	4	32.5	50.9	59.2
No NSF	349	19	48	49	233	49	29	44	92	19	75	33	9	5	32.2	46.4	50.2
Academic	409	20	47	64	278	66	43	69	81	19	102	51	20	5	36.7	55.0	62.2
Industrial	44	2	8	6	28	6	0	2	17	3	4	2	2	0	14.3	21.4	28.6
Joint	148	8	8	36	96	13	11	16	48	8	24	11	1	4	25.0	36.5	37.5
Conferences	470	10	37	100	323	64	46	76	119	18	108	54	19	5	33.4	50.2	56.0
Journal	131	20	26	6	79	21	8	11	27	12	22	10	4	4	27.8	40.5	45.6

Table 3: Build error summary.

error	count	percentage
distribution is missing files	12	19.4%
prerequisite failed to build	2	3.2%
unavailable environment	3	4.8%
other errors	15	24.2%
incomplete documentation	11	17.7%
missing third party package	8	12.9%
build:error comment[not needed,comment]	2	3.2%
runtime error	9	14.5%
Total	62	

internal compiler error: Compilation terminated with the message "Internal Compiler Error."

unavailable environment: The build failed due to an unavailable version of a particular piece of software, compiler etc.

missing third party package: A particular piece of software, package or tool required to run the build was not found.

distribution file is missing: The author did not make available files that should be part of the build, such as a missing header file or a missing file with input data to run the code.

incomplete documentation: Documentation necessary to build the software is incomplete or missing.

runtime error: The code compiles and fails to run, due, for example, to a segmentation fault or a runtime exception.

prerequisite failed to build: A pre-requisite for a tool could not be successfully built by the team in under 30 minutes.

other errors: The error could not be categorized.

The types of build errors encountered can be found in Table 3.

4.1 Does NSF Funding Affect Sharing?

The National Science Foundation's (NSF) Grant Policy Manual[12] states that

> **b.** Investigators are expected to share with other researchers, at no more than incremental cost and within a reasonable time, the primary data, samples, physical collections and other supporting materials created or gathered in the course of

[12] `http://www.nsf.gov/pubs/manuals/gpm05_131/`

Table 4: Abbreviations used in Table 2, Figure 11, and Appendix B.

Notation	Number of papers...
BC	where the results are backed by code
NC	excluded due to results not being backed by code
HW	excluded due to replication requiring special hardware
EX	excluded due to overlapping author lists
EM^{yes}	where the author provides code after being emailed
EM^{no}	where the author responds to email that code cannot be provided
$\text{EM}^{\emptyset}$	where the author does not respond to email requests within 2 months
$\text{OK}^{\leq 30}$	where code is available and we succeed to build the system in ≤ 30 minutes
$\text{OK}^{>30}$	where code is available and we succeed to build the system in > 30 minutes
OK^{Auth}	where code is available, we fail to build, but the author says the code builds with reasonable effort
Fails	where code is available, we fail to build, and the author says the code may have problems building

work under NSF grants. Grantees are expected to encourage and facilitate such sharing. [...]

c. Investigators and grantees are encouraged to share software and inventions created under the grant or otherwise make them or their products widely available and usable.

d. NSF normally allows grantees to retain principal legal rights to intellectual property developed under NSF grants to provide incentives for development and dissemination of inventions, software and publications that can enhance their usefulness, accessibility and upkeep. Such incentives do not, however, reduce the responsibility that investigators and organizations have as members of the scientific and engineering community, to make results, data and collections available to other researchers.

Even so, we see no significant difference in weak repeatability rates between those authors who are supported by the NSF and those who are not.

4.2 Does Industry Involvement Affect Sharing?

Unsurprisingly, papers with only authors from industry have a low rate of weak repeatability, and papers with only authors from academia have a higher than average rate. The reasons why joint papers also have a very low rate is not immediately obvious: the industrial partner might have imposed intellectual property restrictions, the research could be the result of a student's summer internship, etc.

Does the Right Version Exist? From the responses we received from authors, we noticed that published code does not always correspond to the version that was used to produce the results in the corresponding paper. To see how common this is, in our author survey we asked "Is your published code identical to the version that you ran to get the results in the paper (ignoring inconsequential

bug fixes)?" It is encouraging to see that out of the 177 responses to this question, 83.1% answered "yes", 12.4% answered "No, but it is possible to make that version available", and only 4.5% answered "No, and it is not possible to make that version available."

4.3 Reasons Why Code Cannot be Shared

The email responses we received were generally pleasant, apologetic if code could not be provided, and accomodating. Below are a few representative examples of emails from authors who turned down our request.

4.3.1 Versioning Problems

Version problems appear to be a major obstacle to repeatable research. In most cases we could not determine if the code we downloaded corresponded to the version in the paper, and often, neither could the author:

> Thanks for your interest in the implementation of our paper. The good news is that I was able to find some code. I am just hoping that it is a stable working version of the code, and matches the implementation we finally used for the paper. Unfortunately, I have lost some data when my laptop was stolen last year. The bad news is that the code is not commented and/or clean. So, I cannot really guarantee that you will enjoy playing with it.

> Attached is the ⟨system⟩ source code of our algorithm. I'm not very sure whether it is the final version of the code used in our paper, but it should be at least 99% close. Hope it will help.

4.3.2 Code Will be Available Soon

Many responses we received indicated that the code corresponding to the published paper was too bad to be released, that new code was being worked on, and once done, we could get access to the cleaned up system:

> Thank you for your interest in our work. Unfortunately the current system is not mature enough at the moment, so it's not yet publicly available. We are actively working on a number of extensions and things are somewhat volatile. However, once things stabilize we plan to release it to outside users. At that point, we would be happy to send you a copy.

While the authors should be commended for eventually making their code available, it should be pointed out that for *repeatability*, such delayed releases are unacceptable: it will never be possible for a reviewer or reader of the paper to verify the results that are actually presented in the paper.

4.3.3 No Intention to Release

A few authors acknowledged that they never had the intention to make the code available:

> I am afraid that the source code was never released. The code was never intended to be released so is not in any shape for general use.

4.3.4 Programmer Left

In some cases, the one person who understood the system left:

> ⟨STUDENT⟩ was a graduate student in our program but he left a while back so I am responding instead. For the paper we used a prototype that included many moving pieces that only ⟨STUDENT⟩ knew how to operate and we did not have the time to integrate them in a ready-to-share implementation before he left. Still, I hope you can build on the ideas/technique of the paper. Regards,

4.3.5 Bad Backup Practices

In one case, the system physically ceased to exist:

> Thanks for your interests in this paper. Unfortunately, the server in which my implementation was stored had a disk crash in April and three disks crashed simultaneously. While the help desk made significant effort to save the data, my entire implementation for this paper was not found. I do have reports generated by the tool for ⟨SYSTEM⟩. If you are interested in these reports, I can send them to you. Sorry for that.

4.3.6 Commercial Code

Research done by employees of commercial entities were often not able to release their code

> Since this work has been done at ⟨COMPANY⟩ we don't open-source code unless there is a compelling business reason to do so. So unfortunately I don't think we'll be able to share it with you.

but sometimes the author added a helpful suggestion:

> The code owned by ⟨COMPANY 1⟩, and AFAIK the code is not open-source. Your best bet is to reimplement :(Sorry.

4.3.7 Proprietary Academic Code

Even university researchers at universites had licensing issues

> Thank you for the interest in our ⟨CONFERENCE⟩ contribution.
> Unfortunately, the ⟨SYSTEM⟩ sources are not meant to be opensource
> (the code is partially property of ⟨UNIVERSITY 1⟩,
> ⟨UNIVERSITY 2⟩ and ⟨UNIVERSITY 3⟩.)
>
> In particular, while there is interest, I do not have the liberty of
> making available the source code at my current institution
> (⟨UNIVERSITY 4⟩).
>
> If this will change I will let you know, albeit I do not think there
> is an intention to make the ⟨SYSTEM⟩ sources opensource in the
> near future.

or put restrictions on the release of the code:

> Thanks for your interest.
>
> We haven't yet open-sourced the software, but we are making a collaboration
> release available to academic partners.
>
> If you're interested in obtaining the code, we only ask for a description of
> the research project that the code will be used in (which may lead to some
> joint research), and we also have a software license agreement that the
> University would need to sign.

4.3.8 Industrial Lab Issues

Industrial labs walk a thin line between producing publishable research and research which is relevant for their parent company:

> The short answer is we can't share what did for this paper. We appreciate that this is not in the academic tradition, but this is a hazard in an industrial lab.
>
> The slightly longer answer is that the implementation is set of discrete pieces rather than a whole system. There are legal hazards around what we have:
>
> - Some code of the implementation is custom modifications to the ⟨SYSTEM⟩ source tree and in places that current policy would absolutely forbid sharing. As researchers, there is some tension between implementing stuff on the companies platform in the hope it might influence the product groups and implementing on simpler systems which would turn the product groups off. Ultimately the product groups sponsor our employment.
>
> - There is also code that contains IP covered under NDA with the hardware manufacturer.
>
> There would not be much left on the table in this instance that we could get past legal review and have value to the outside world.
>
> If you were interested in experimenting with a system with ⟨HARDWARE⟩, ⟨COMPANY1⟩ are shipping dev boards and have a custom Linux kernel with support for their product. ⟨COMPANY2⟩ might also be shipping boards now, but I've not heard anything in this regard.

4.3.9 Unavailable Subsystems

Some systems were built on top of other systems which were either not publically available

> We implemented and tested our sparse analysis technique on top of a commercialized static analysis tool. So, the current implementation is not open to public. Sorry for this.

or obsolete:

> Currently, we have no plans to make the scheduler's source code publicly available. This is mainly because ⟨ANCIENT OS⟩ as such does not exist anymore, and as a result installing this OS on new hardware is a bit troublesome; not to mention how tedious it is to prepare a new machine for OS development under a discontinued version of ⟨ANCIENT OS⟩ such as the one we used (official package repositories no longer exist). Although some researchers asked us to disclose the source code, we strongly believe that the efforts to make the code available and fully documented now are not worth it since few people would manage it to get it to work on new hardware.

4.3.10 Multiple Reasons

Sometimes, there were multiple reasons why not to release the code, including licensing, bad code, and reliance on other systems:

> Sorry we haven't released our code because it is too hard to maintain. A large part of our code contains/relies on the code from other tools (probably old versions), such as ⟨LONG LIST OF SYSTEMS⟩. It would require huge amount of effort to make our code work with the latest versions of these tools. We are not using the code at ⟨COMPANY 1⟩ or ⟨COMPANY 2⟩ so we didn't spend time on maintaining it. Besides, we cannot release our code with the tools in one package since they are protected by licenses.

4.3.11 Intellectual Property

Some authors were worried about how their code might be used:

> ... I hope the above will be proved of help. If these suggestions are not helpful in your efforts, we could provide our implementation but please kindly note that:
> - We won't be able to provide further support in a timely manner. Of course, installation instructions and initialization parameters will accompany the implementation
> - We would like to be notified in case the provided implementation will be utilized to perform (and possibly publish) comparisons with other developed techniques. If this is indeed the case, please note that we would prefer to look into the extracted results and be able to comment on the comparison results before their publication.
>
> The reason for the second provision is that, based on earlier (bad) experience, we would like to make sure that our implementation is not used in situations that it was not meant for. For instance, in this particular case, our code has not been optimized for speed, so it makes no sense to use it for performance measurements over streaming data. In other words, we would like to have some say on how our code is used, especially if the results are to be used in a paper submission.

4.3.12 Research vs. Sharing

Producing artifacts which are solid enough to be shared is clearly labor intensive, and one researcher explained how he had to make a draconian choice:

> Thank you for your email to ⟨STUDENT⟩, expressing interest in our work.
> Since giving you ⟨system⟩ would necessitate us also giving you our complex
> Simulation infrastructure, which has been developed over a lot of
> years by a large number of my former and current PhD students, he knew
> that I would want it forwarded it to me for answering.
>
> Our simulator is very complex, and continues to become more complex
> as more PhD students add more pieces to it. The result is that without
> a lot of hand holding by senior PhD students, even our own junior PhD
> students would find it unusable. In the past when we attempted to share
> it, we found ourselves spending more time getting outsiders up to speed
> than on our own research. So I finally had to establish the policy that
> we will not provide the source code outside the group.
>
> On the other hand, ⟨STUDENT⟩ would be very happy to answer any questions you
> have about ⟨SYSTEM⟩ if you decide to implement it on your own simulation
> infrastructure. In fact, both ⟨STUDENT⟩ and I would welcome that opportunity.
> For us, in fact, it would show that the ideas in ⟨SYSTEM⟩ can be validated on
> another Simulator developed by an outside group. Feel free to take ⟨STUDENT⟩
> up on this offer if you wish.
>
> I wish you success in your undertakings and hope that ⟨STUDENT⟩'s help
> understanding ⟨SYSTEM⟩ will be a win-win for us both.

4.3.13 Security and Privacy

Unlike those working in other fields, Computer Security researchers have to contend with the possible negative consequences of making their code public:

> Thanks for your interest. However, we have an agreement with the ⟨BUSINESS-
> ENTITY⟩ company,
> and we cannot release the code because of the potential privacy risks to the general
> public.

4.3.14 Design Issues

Design and implementation choices can affect whether the resulting program is useful to others:

> The code used to implement the ⟨CONFERENCE⟩ paper is
> complete, but hardly usable by anyone other than the authors. This is
> due in large part due to our decision to use ⟨OBSCURE LANGUAGE VARIANT⟩ for the
> input language. The error messages which are produced by the compiler
> are useless to anyone not fluent in both ⟨LANGUAGE⟩, ⟨SYSTEM⟩, and the
> compiler architecture.

4.3.15 Too Busy to Help

Some people were too busy to help out understanding the code:

```
Most importantly, I do not have the bandwidth to help anyone come up to
speed on this stuff.
```

4.4 Feedback on the Study

We received feedback on the study both from authors (as part of their response to the survey) and from other researchers who heard about the study through other channels. The feedback falls roughly in three categories:

- Criticism raised against the methodology of the study ("you should have...");
- Criticism raised against individual results ("anyone competent should be able to build this code"); and
- Appreciation of the importance and difficulty of code sharing and repeatability.

We reproduce some representative feedback we received below. It should be noted that, in contrast to the responses we received to our email requests for code (see Section 4.3), the feedback we show here came from authors who were aware of the existence of our study.

4.4.1 Methodological Issues: "You Should have been More transparent!"

Some authors were unhappy that our email requests for code did not explicitly state the purpose of the request:

```
Since you have used us as subjects of your study without our permission, I feel that
I must help you improve your study.
```

However, as this author points out, had our emails been more transparent, the results might have been biased:

```
However, I have some reservations regarding the way you approached the authors in
your first email.  I believe the authors had the right to know upfront the actual
reason you were asking the code of their system, so as to decide whether they wanted
to invest the time needed for packaging and submitting their code, especially since
you intended to make your findings publicly available.
```

4.4.2 Methodological Issues: "You Should have used a Different Measure of Repeatability!"

In this study we measure only the availability of research artifacts, specifically source code, and whether such shared code will build. This is clearly a very limited precursor to a "real" study of repeatability (which would require verifying the published results) and reproducibility (which would require, at the very least, a reimplementation and additional experiments). Several authors were critical of such a limited investigation. We are inviting our colleagues to conduct more indepth studies, and we make all materials collected during our study available as a starting point.

> A 30-minute verification that an author's code compiles, runs, and prints the expected output is interesting but is not a convincing test of scientific reproducibility.
> A better test of reproducibility of a publication's results would be to independently implement and measure the algorithm or system described. This would be unlikely to replicate any bugs or systematic measurement errors, increasing the meaningfulness of a positive reproducibility result. Negative results would also be more likely to provide technical insight and foster scientific discussion.

> There are serious flaws in your methodology that you have failed to recognize. Also, you should carefully reconsider your view of reproducing results in other fields (such as physics, astronomy, etc.). In the tech report, your view reads as somewhat naive, and it misinforms your recommendations for computer science. If you are interested in discussing further, please let me know.

> Our work is useful for researchers working with ⟨TOOL1⟩, ⟨TOOL2⟩ etc. Researchers need to understand how to build/use these tools before they look at our work. We strongly suggest that your study applies at least the same standards of evaluation as those used by the ECOOP Artifact Evaluation process.

> Would have been nice if the students attempting the build were also saving the detailed log file of the build and included it when corresponding with the authors.

> A repeatability study takes more than clicking on links and running make scripts or the resulting executables. See for example Casey Klein et al at POPL 2012 for a repeatability study that went into depth. When we ran this study, we spent almost two years with one PhD student per case study.

> I do not like the way this study was conducted and found the follow-up work from Brown University more palatable, but I have not followed all the ripples from the initial "results" closely.

4.4.3 Methodological Issues: "You Should have Asked for Assistance!"

In our study, we decided not to enter into a conversation with the authors, asking for assistance, for example, when builds failed. Thus, weak repeatability A and B measure the extent to which code builds out-of-the-box. This is important in several scenarios such as when a reviewer of a conference paper examines a submitted research artifact, or when the lead developer is no longer available for consultation. Our measurement of weak repeatability C, however, captures (an approximation of) the situation where the authors believe their code builds (even if we failed to do so) and are responsive to requests for assistance.

> This process would have gone much more smoothly had the authors interacted with us. While the second round of results do say that our code builds with some modifications, a quick not from the authors would have allowed us to fix the issue and enabled the source to build cleanly.

> Your ‘validation process’ is, frankly, absurd and therefore provides useless results. In our case - the version we posted had unnecessary dependency (file) in makefile, all you had to do is follow the error message and remove it and all would work fine- or ask us by mail - you did neither.
> Besides, you only checked that projects compile; what exactly does it mean that the project compiles? to really check reproducibility you must validate it by running and comparing results and functionality. Your ‘test‘ would pass a an "hello, world!" program... absurd. You wasted more time for us in understanding your useless ‘evaluation‘ and writing such response as this, than you invested in evaluating our project. A waste of time.

> If you’re having trouble building, why not correspond with us? This is not commercial software; we are academics with limited resources so our testing is similarly limited. Most of these issues can probably be ironed out with a brief conversation.

> Or ask us by mail.. but you didn’t. and obviously you had no problem finding our mail. Absurd. We used code of others many times, often dealing with much much more serious issues without giving it a second thought.

4.4.4 Methodological Issues: “You Should have used the Latest Version!”

Some authors pointed out that newer versions of their system were available and should be used instead:

> As with the last time you folks inspected our software, you appear to be using an old build. Since April 18, 2014 (nearly 5 months ago), our software has used ⟨BUILD SYSTEM⟩ for our build system. Your notes indicate that you tried (and failed) to use the old Makefile-based build system.

> In fact, we have since fixed the issue. If the authors download the new code, it will build without modification.

In order to test repeatabilty, however, the version of the code that was used to produce the results in the published paper should be used. We therefore evaluated the code that we found during the time of our study, not any versions which may have been made available at a later date.

4.4.5 Methodological Issues: “You Should have used the Binary Instead!”

Some authors pointed to the availabilty of a binary version of their system:

> If you’re interested in the original version of ⟨SYSTEM⟩ that was the subject of the paper, you can sidestep your build problems. We distribute a binary JAR here, which is the link that you used: ⟨URL⟩.

While a pre-compiled version of a project can be helpful, we consider being able to read the code, and connect it back to algorithms in the paper, essential for a full repeatability or reproducibility examination. Except for a few online implementations, we therefore only accepted papers for which source code was made available.

4.4.6 Methodological Issues: "You Should Sign this License Agreement!"

A few authors were upset that we marked papers for which code was only available after signing a license or non disclosure agreement as "code not available:"

> You should be more diligent in doing these kind of studies: "code not available" is plain untrue for our work and as pointed out in my email to you on June 25, 2014 this is an unfair labelling given that we make the code available to anyone willing to sign an NDA. Your label makes it look like as if we did not respond or are unwilling to share the code which is simply untrue.

Many conferences (such as OSDI[13]) will not consider papers accompanied by nondisclosure agreement forms. Similarly, for a conference that requires, or admits, the submission of research artifiacts along with a paper submission, restrictive license agreements will be problematic. We therefore marked papers where the authors asked us to sign any sort of license agreement as "code not available."

4.4.7 Critique Against Individual Results

Several authors were miffed at our inability to build their code, and pointed out that "any competent programmer" should be able to work around the problems we encountered.

> The errors encountered are not linked to the project or the code added for this paper but to the simple compilation of the GCC compiler (which is included in the sources of ⟨SYSTEM⟩).
> Building GCC (which is a very standard tool for development on all platforms) cannot be considered too complicated in any reasonable result reproduction effort.

> I think the build error would be obvious to anyone that's familiar with building C/C++ programs, and I don't think the conclusion of a build failure for this is legitimate.
> From the build notes: ⟨URL⟩ The first error a few lines in is: ⟨...⟩fatal error: google/protobuf/message.h: No such file or directory compilation terminated.
> So Google's protocol buffers library wasn't found: either it wasn't installed or wasn't at the expected location.
> See also how ⟨PERSON⟩ was able to build the project: ⟨link to https://github.com/shriram/repro-in-cs/...⟩

> you recieved: g++: error: ControlThread.cpp: No such file or directory
> this was due to a minor bug - this file was simply not needed. All you had to do - and what any competant programmer imho would do - is delete this dependency from the make-file and see if it compiles (and it does, and we also fixed right away).

We also received positive feedback when we were able to successfully build a system that even the author agreed was complex:

[13] https://www.usenix.org/conference/osdi14/call-for-papers

> ⟨SYSTEM⟩ has a ridiculous set of dependencies, and is highly complex to build. I'm relieved to know you succeeded in getting it to run, albeit with some additional effort that evidently was not described in the directions.
> I believe my response to this question would best be described as, "I believe the code you downloaded can be build with significant effort." Well done.

> Our experimental setup needs several involved steps and not all of them are automated currently. It requires using several tools and systems that are not widely (or freely) available such as ⟨TOOL1⟩, an ⟨PLATFORM⟩ system for compiling and disassembling ⟨OPERATING SYSTEM⟩ binaries, and a cluster of machines to perform fault injection campaigns. This makes it complicated for us to package it for release

4.4.8 Reproducibility and Repeatability: "It's Important!"

Some authors provided thoughtful feedback on the importance of repeatability and reproducibility, and expressed an interest in our study.

> I would first like to mention that I find interesting the data you collected during your weak repeatability initiative. Moreover, I think it is of great importance to have repeatable experiments in order to advance the research in our field and to concentrate on the right problems.

> I would like to make clear that *we* believe you did us a service by pursuing this question. Like you, I have been on the receiving end of stonewalling, silence, and maybe even lies when trying to get hold of code from others. Your emails and Appendix completely resonated with me. We have been pointing this out to people, and indeed I sense others share it. Eg, there have been some twitter threads just about the emails.

> Great effort! We need more studies like this, please keep up the good work.

> It's a wonderful idea to analyse the repeatability of software and hardware based papers. But I guess it is a huge amount of work. I sincerely congratulate the team for investing this effort.

> I think it's fantastic that you have undertaken this comprehensive study.

> I found your study on repeatability in Computer System Research a very useful initiative. I hope it will make more researchers aware of what is needed so that their results can be reproduced with minimal effort.

> Very worthwhile study. Thanks for doing this. I know you caught a lot of flak for the initial results, but I really appreciate the followup and care that you've taken.

> This survey is a monumental effort and ultimately of great service to our community, thank you!

> This is a significant project that deserves broad support across the computing community. There are often perfectly legitimate obstacles to code publishing but these should become the exception.

> Your study on repeatability in computer science research is quite interesting. Looking forward to see how it will impact the publication process and culture of this area of engineering.

4.4.9 Reproducibility and Repeatability: "It's Difficult!"

Some authors provided thoughtful commented on how difficult code sharing can be to achieve in practice. The problems expressed range from resource constraints when working in academia, to licensing issues when working in industry, to problems motivating authors to make their code available when there is no tangible benefit for doing so.

> I came across your study entitled "Measuring Reproducibility in Computer Systems Research" and I found it fascinating. I also particularly enjoyed the Anecdote 2, which was both hilarious and dramatic.
> With several colleagues, I also tried to improve reproducibility in science (mainly in ⟨OTHER FIELD⟩ in our case) by creating ⟨WEB SITE FOR UPLOADING RESEARCH ARTIFACTS⟩. But to tell the truth, it has been quite frustrating to see the low adoption rate among ⟨RESEARCHERS⟩. As you put it in your study: ''the if you build it they will come paradigm does not always work''. I could not agree more with your statement.
> Your work on reproducibility in CS is a source of inspiration and motivation for me and my team and I thank you for that.
> All the best with your research.

> Publishing or giving out code is easy. The hard part is to find enough resource to clean it up, make it portable, document it, and support it especially when the main goal of a project is not to build a widely used open source system but to demo ideas of a paper. Anyone with large system experience could imagine that there will be many system specfic hacks, shortcuts, and workarounds in the implementation of distributed system with many moving parts. For example, we may hard code a storage path or make some assumptions about access control configuration of the underlying systems. Such things will definitely not impact the validity of our results. And they can be smoothed out from the system given enough engineering resouce. Before that happen, we had to do substantial support work to help people deploy and run our system in the past several cases.

Cool, repeatability is extremely important and strongly in favor even though I work in the industrial complex. As an industrial engineer, sometimes we can share stuff and sometimes we can't. There are no recognition points internally for making code available or supporting it. Our background is probably no different from most academics, but we don't have the same level freedom and are more-regularly assaulted by performance reviews. We also have an existing portfolio of IP and there are lawyers who fear releasing code/ideas that pertain broadly to that area may be invalidating company rights to defend that IP. It is a deal with the devil - the company pays us to generate valuable IP, have to appreciate they want first dibs and to protect it.
In your TR you comment on lack of collaboration between industrial research and academia and pin it on industrial IP. The problem is often the other way around, by collaborating with academia their IP departments often come after us with completely unrealistic agreement proposals - the assumption is big company X participated in this research and they have deep pockets, the research must be worth millions today. Entering these negotiations is a huge time and cost sink and sours the process. It is a very naive approach as only 1 in N projects ever reach product in anyway and a tonne of additional work goes into making this happen. We try to put interns on projects that do not directly relate to their PhD research to avoid this issue - and we encourage writing up and publication but not future work after the intern has left. We do not place any restrictions on the interns being able to talk about what they've worked on or start their own research projects in similar areas after the event.
In an industrial lab, publication is seen as the easiest activity researchers/engineers participate in. Research that is good enough for tier 1 publication may not fit in with the product portfolio or may not be implementable to the satisfaction of the product teams. Product teams are much more discerning/dismissive than paper reviewers. The progress scale is roughly (1) tier 1 publication, (2) getting interest in from a product team, (3) getting your code in their depot, (4) getting commitment and resources to ship the research, (5) shipping.

Note that publishing code is also a rather thankless task. Likely outcomes include: (a) Nobody will look at it and you will have wasted your time. (b) Anyone who looks at it will see that it perfectly matches the paper's specification (except for being MUCH harder to read) and won't have learned anything. (c) Someone will try to use it, run into some kind of trouble, and bother you with annoying questions. (d) Someone will catch some trivial bug, approximation, or limitation and use it to cast doubt on your results. (e) Some professor will convince a naive first-year student to use your code as a starting point for their experiments and you will have contributed to that poor student's misery and anguish.

Publishing code is often difficult or even impossible for researchers working in private industry. It can be done in some cases, but it always comes with quite some legal effort.

> We aimed in our paper to provide enough information to allow research groups to reproduce our results, i.e., to reimplement the concepts in our analysis. We believe that we accomplished this.
> We made the code available as a courtesy on request, and were prepared to spend time and energy enabling interested researchers to replicate and extend our results, i.e., to use our code base. We did not aiming to provide a distribution of our code that works robustly out of the box. While we may aim to do this with future research projects, it was not an aim of the current work.

4.4.10 Reasons for not Being Able to Share

Similarly to what we saw in Section 4.3, the survey provided additional reasons why sharing sometimes is impossible.

> Unfortunately, the author who has done most of the coding for this paper (⟨AUTHOR⟩) has passed away and the code is no longer maintained.

> As indicated on my webpage, my implementation builds off of another author's code, who has since claimed IP rights and refuses to let me publicly release it. Therefore, I just released the portion of the implementation described in the paper that I wrote, and to run the code, one would need to re-implement the code written by the other author.

> We are involved in a commercialization effort along this line. So, we are not planning to release code at this point.

5 Recommendations

As we have seen, there are many ways to improve the state of repeatability and reproducibility in Computer Systems research. We could simply, as some conferences have done, require that along with every paper submitted for publication the authors should attach the corresponding code, perhaps in the form of a virtual machine image. Or, we could build special tools that help authors produce code that can run reliably and with repeatable results, regardless of the execution environment. Or, we could build web sites that allow authors to make their code available to colleagues in perpetuity.

Unfortunately, based on the study presented in this paper, it is our belief that it is currently not realistic to expect Computer Systems researchers always to make their code available to others for scrutiny. The reasons for the inability to share code are manyfold: the code may not be clean enough for public distribution; the authors may be considering commercialization of their work and would prefer to keep the code out of the hands of competitors; (part of) the code may not be publishable because of licensing restrictions; the main author may no longer be available to package up and distribute the code; or the researchers may be too busy to make their code available.

Given that sharing will not always be possible, our proposal is quite modest. Rather than forcing authors to make their systems available, we propose instead that *every article be required to specify the level of repeatability a reader or reviewer should expect.* We will term such a statement a **sharing specification**. The sharing specification should be provided in the article header (along with

Table 5: Data needed for sharing contracts.

location	`email address and/or web site (RFC 3986 URI)`
resource	`kind (code, data, media, docs, ...)`
	`availability (no access, access)`
	`expense (free, non-free)`
	`distribution form (source, binary, service)`
	`expiration date (ISO 8601 Date)`
	`license (string)`
	`comment (string)`
support	`level (L1, L2, L3)`
	`expense (free, non-free)`
	`expiration date (ISO 8601 Date)`

$$
\begin{array}{rcll}
\rho & \in & \{\texttt{code}, \texttt{data}, \texttt{media}, \texttt{docs}, \ldots\} & \textit{(resources)} \\
\alpha & \in & \{\texttt{no access}, \texttt{access}\} & \textit{(availability)} \\
\phi & \in & \{\texttt{source}, \texttt{binary}, \texttt{service}\} & \textit{(distribution form)} \\
\xi & \in & \{\texttt{free}, \texttt{non-free}\} & \textit{(expense)} \\
\tau & \in & \textit{ISO 8601 Date} & \textit{(expiration date)} \\
\lambda & \in & \textit{RFC 3986 URI} & \textit{(uri)} \\
\pi & \in & \{\texttt{L1}, \texttt{L2}, \texttt{L3}\} & \textit{(support level)} \\
\sigma & \in & \textit{string} & \textit{(license-string)} \\
\textit{res} & ::= & \rho(, \rho)^* : \alpha[, \xi][, \phi][, \tau][, \sigma] \ ; & \textit{(sharing)} \\
\textit{loc} & ::= & \lambda \ ; & \textit{(location)} \\
\textit{sup} & ::= & \texttt{support} : \pi, \xi[, \tau] \ ; & \textit{(technical support)} \\
s & ::= & \textbf{Sharing } \textit{loc}^+ \ \textit{res}^* \ \textit{sup}^* & \textit{(contract)}
\end{array}
$$

Figure 12: Grammar generating sharing specifications.

keywords and classification terms) *at the time of submission* (allowing reviewers to take repeatability into account when deciding on whether to recommend acceptance or rejection) as well as in the camera ready version (allowing readers to know if, and how, to access the relevant code, data, and other resources). The specification can be construed as a *contract* between an author and their reader/reviewer in which (a) the author commits to making available certain resources that were used in the research leading up to the paper and (b) the reader/reviewer commits to take these resources into account when evaluating the contributions made in the paper.

5.1 Syntax of Sharing Specifications

In Table 5 we show the data that need to be part of a sharing contract: the external **resources** that back up the results in the paper, the **locations** where these resources can be found or ways to contact the authors, and the level of **technical support** the authors will provide. Figure 12 shows a grammar for one possible sharing specification. Figure 13 gives an example of a paper header with a sharing specification generated from this grammar.

Resources can include code, data, media, etc. For each resource the specification states if it will be made accessible to the community, if there is a cost involved in obtaining the resource, an optional deadline after which the resource may no longer be available, an optional licensing scheme,

```
Keywords
    Mobile, Ubiquitous Computing, Sensors

ACM Classification Keywords
    H5.m. Information interfaces and presentation:
    Miscellaneous

General Terms
    Design Document, Work-in-Progress

Sharing
    mailto:alice@example.com; http://mobile-project.example.com;
    code:access,free,source,Apache 2.0;
    support:L1,free,perpetuity; support:L2,non-free,2014-12-31;
```

Figure 13: Example of paper heading. The sharing section specifies that the paper is backed by code but no other resources, that the code will be provided for free in source form, that support will be provided until the end of 2014, and that the code is distributed under the Apache license.

---

and whether the resource is available in source or binary form or is accessed as a service. Here, "service" represent any dissemination method that gives a user indirect rather than direct access to the resource. Examples include programs executing as a web service, media streamed from a server, or authors guaranteeing that they will run the code by hand, on input data provided by the user, and then provide them with the resulting output. An absence of an expiration date indicates the resource will be available in perpetuity.

The `support` field specifies the level of technical support the authors will provide, for how long, and whether support is free or not. There are three levels of support: `L1` indicates the authors will resolve issues with downloading, installing, and executing the program; `L2` indicates the authors will maintain the program, fixing bugs, and upgrade as necessary as language specifications and operating systems evolve; `L3` indicates a willingness on part of the authors to port the program to new environments, improve algorithms and performance, add new features, etc. Multiple support levels can be specified. An absence of the `support` field indicates no support is offered. An absence of an expiration date indicates that support will be available in perpetuity.

For each resource the specification can specify the license under which a resource is provided. An absense of the license field indicates that the resource is provided under an "anything goes" license. It should be noted that *sharing*, as defined in this paper, is different from *licensing*. The sharing specification in Figure 12 represents a commitment an behalf of the author of a paper to make resources available to the wider community for scrutiny. A *license*, on the other hand, describes the actions allowed on these resources, such as modification, re-distribution, reverse engineering, etc. Under some circumstances it is necessary to specify both licensing and sharing. For example, if code is distributed in binary form only and the license prohibits reverse engineering, the community's ability to verify that the actions performed by the software is consistent with what is described in the publication is diminished. Similarly, repeatability is hampered by code that makes use of libraries whose license prohibits re-distribution.

5.2 Example Sharing Specifications

The sharing specification in Figure 12 is designed to handle most common situations, but obviously represents ta tradeoff between expressiveness and conciseness. Since our hope is that in the future every paper will contain such a specification, we do not want it to take up too much space nor do we want writing it to be too onerous. In most cases, authors will find that a very terse specification will suffice. For a purely theoretical paper, i.e. one not backed by any external resources, the specification would be simply

```
Sharing
    bob@cs.example.edu
```

i.e. it defaults to providing "corresponding author" information. In the common situation where a paper is backed by code and data, the authors expect to put them both up on their web site, never taking them down, not providing any support, and not worrying about licensing, the specification would be

```
Sharing
    http://project.example.edu; code,data:access,free,source;
```

In a situation where the authors are employed by a commercial organization and, as a result, the code is proprietary and cannot be shared, the benchmarks used are freely shared, and limited support in how to download, install, and run the benchmarks will be given until the beginning of 2015, the specification would be

```
Sharing
    http://project.example.com; code: no access;
    data:access,free,source; support:L1,free,2015-01-01;
```

There may certainly be complex situations that cannot be expressed by the grammar in Figure 12, but these can be handled by the authors providing additional information in a footnote.

5.3 Lessons Learned

From the work that has lead up to this paper, in particular the responses to our email queries, we have learned the following valuable lessons:

1. **Unless you have compelling reasons not to, plan to release the code.** It is the right thing to do, and if you start with this mind-set from the beginning of the project, the amount of extra work will likely be negligible.

2. **Students will leave, plan for it.** When building the system keep in mind that the code should outlive both you and the student.

3. **Create permanent email addresses.** You and your students will most likely be changing jobs a few times during your career. While some schools will keep old email addresses around, or forward email, you cannot count on it. Create email addresses that you know will be permanent throughout your working life and use them in all professional correspondence.

4. **Create project websites.** These are more likely to remain functional over time than email addresses. Put the URL in the paper. Be prepared to upload code and test data to your web sites at the same time as you upload the paper describing your system to a conference site for review.

5. **Use a source code control system.** Whenever you submit or publish a paper, set a label on the corresponding code version so that you can easily recreate it.

6. **Backup your code.**

7. **Resolve licensing issues.** If you anticipate problems start the licensing process early, so that you are able to release the code at the same time as the paper is submitted for publication.

8. **Keep your promises.** If your grant application states that you will be sharing code with the community, plan for keeping that promise.

9. **Plan for longevity.** Projects may live on for a long time, with many students building on the code. Plan for this at the onset of the project, by setting up the appropriate directory structures, plug-in architectures, etc., which will allow the project to grow.

10. **Avoid cool but unusual designs.** Unless you have a compelling reason to do otherwise, stay with standard operating systems, programming languages, and tool chains.

11. **Don't rely on the permanence of external software.** Don't expect a particular version of a library to be available in perpetuity, and don't expect it to be available at a particular url. Include, whenever possible, all dependent code in your distribution.

12. **Plan for repeatable releases.** Use the same techniques that Release Managers in industry use to ensure consistent and repeatable builds. For example, check in your entire tool chain (compilers, linkers, libraries, etc.) into your source code control system, and start any release by building the tool chain from scratch. There is an extra startup cost when beginning a new project, but this will be paid off over time. It will, for example, make it easier for new students to join the project.

6 Discussion

As scientists and engineers we are predisposed to tackle what are, fundamentally, societal, psychological, or economic problems with technological fixes. When we become aware that many scientific results cannot be easily replicated, we design tools to help researchers store the sequence of events that make up the recipe for their studies; when we notice that research artifacts are often languishing in obscurity on researchers' hard drives, we create web repositories where they can be stored in perpetuity; and, when we suspect that scientific reproducibility might be a serious problem, we immediately launch into a scientific study (such as this one) to determine just how prevalent the phenomenon might be.

While there is certainly much room for follow-up studies to this one and technological advances to help researchers share their artifacts, we must acknowledge that the root of the problem is sociological, not technological: we do not produce solid ready-to-share artifacts or attempt to replicate the work of our peers because there is little professional glory to be gained from doing so. In [20], Brian Nosek writes, "Because of strong incentives for innovation and weak incentives for confirmation, direct replication is rarely practiced or published" and "Innovative findings produce rewards of publication, employment, and tenure; replicated findings produce a shrug."

Thus, the real solution to the problem lies not in sharing specifications, or web sites, or licenses, or tools for scientific provenance, but in finding new reward structures that encourage us to produce solid artifacts, to share these artifacts, and to validate the conclusions drawn from the artifacts

Table 6: Summary of results from various studies of repeatability and reproducibility.

Reference	What/Who was studied	What was measured	Results
Kovacevic [17]	15 papers published in the *IEEE Transactions on Image Processing*.	How well algorithms were explained and whether code and data were available.	All algorithms had proofs, 0% had code available, 33% had data available.
Vandewalle et al. [29]	All the 134 papers published in *IEEE Transactions on Image Processing* in 2004.	Reproducibility as measured by 2-3 reviewers.	9% of the papers had code available online and 33% had data.
Stodden [26]	Survey responses from 134 registrants affiliated with American universities at NIPS conference.	Proportion of registrants comfortable with sharing post-publication code on the web vs. proportion publishing some code on their web site.	74% are comfortable with sharing, 30% have code on web site.
Table 1 (page 9 of this report)	Artifact evaluation outcomes for the 268 papers that were accepted in the seven conferences for which we have complete information.	Submitted and accepted artifacts as a percentage of accepted papers.	43.3% submitted, 29.5% accepted.
Klein et al. [15]	Formal modeling and mechanized reasoning of nine papers published in ICFP 2009.	Proportion of papers in which no mistakes were found.	0%.
This study	Examination of 402 Computer Systems papers backed by code.	Proportion of papers with shared code, and the extent to which shared code builds successfully in 30 minutes, with extra effort, or by the authors.	56.2% shared, 32.3% builds in 30 minutes, 48.3% builds with extra effort, 54.0% builds by author.

published by our peers. Unfortunately, in this regard we remain pessimistic; we do not see a near future where such fundamental changes are enacted.

We conclude by discussing the results of our study in the context of other studies of reproducibility and repeatability in Computer Science, and how our sharing specification compare to previously proposed technological fixes.

6.1 Putting it all Together

Any study on repeatability and reproducibility has to be examined in context of the process by which it was carried out. The results of the study reported on in this paper, for example, depend on the facts that

- authors were unaware that they were the subjects of a repeatability study;
- our build process was run mostly by undergraduate students with limited experience;
- no attempt was made to communicate with authors during the build process; and
- we measured *weak repeatability*, rather than repeatability or reproducibility.

To gain a deeper understanding of the state of reproducibility and repeatability, it is illuminating to examine the outcome of studies that were designed differently from ours. Table 6 summarizes the results of six repeatability/reproducibility studies.

The first issue that differentiates the six studies is the rate of sharing reported. In Vandewalle et al.'s [29] study, for example, 9% of published papers had code available, in Stodden's [26] study, 30% of respondents had code published on their web sites, and in the Artifact Evaluation process, 29.5% of accepted papers had acceptable artifacts submitted. In our study we found that out of the 402 Computer Systems papers that were backed by code, for 56.2% of them the authors were able to make that code available.

A second issue is how subjects in the six studies were selected. Stodden's [26] study, for example, excludes researchers working in industry and academics outside the United States. Our study makes no such exclusions since much research in Computer Systems is done in an industrial setting (31.9% of 601 papers in our study had at least one industrial author). All the studies in Table 6 furthermore target top publication venues. It is conceivable that sharing rates would be different in second and third tier venues.

A third issue is the possibility of self-selection bias. The respondents to Stodden's [26] survey were aware of the fact that they were part of a study on reproducibility, which may help to explain the relatively low response rate (23%). Similarly, the high success rate of the Artifact Evaluation process could be the result of authors self-selecting whether to submit an artifact, or not (while 68.1% of the submitted artifacts were deemed acceptable, only 43.3% of accepted papers had a submitted artifact). In contrast, the authors of the papers we examined were, initially, unaware of taking part in a repeatability study. We believe this was indispensable to avoid self-selection bias. Furthermore, the explanations for why code could not be shared (Section 4.3)—given voluntarily and without knowledge of how this information might be used—gives fascinating insight into the issues that stand in the way of universal sharing of research artifacts.

A fourth issue is how to count whether artifacts are shared or not. Stodden [26], for example, only counts whether authors publish *some* code on their web sites. Our study shows that not everyone makes all their code available on their web sites, and that personal contacts matter: for 38.5% of the 226 papers for which we obtained code, the code was acquired through email requests.

6.2 Proposals for Improving Sharing

Table 7 summarizes the different proposals that have been presented for improving the sharing of code and data. It is our belief that none of them, in isolation, will solve the problems of sharing of research artifacts, but that, taken together, they will allow us to make strides towards repeatable and reproducible research: public repositories are necessary in order to ensure long-term availability

Table 7: Summary of proposals for improving sharing of research artifacts.

Reference	Proposal	Description
[4–6, 11, 13, 16, 27]	Artifact repositories	Public repositories of research artifacts; links in papers to related data and computation allowing readers to verify results, examine data, and re-execute computations; virtual machine technology to store complete execution environments.
[18, 23]	Coding conventions	Conventions for organizing code and data to simplify repeatability of results; literate programming.
[19], `artifact-eval.org`	Peer-reviewed artifacts	Peer-reviewed code and data accompanying published papers.
[24, 25]	Artifact licenses	Licensing frameworks to encourage researchers to share their research artifacts without fear of losing attribution.
This proposal	Sharing specifications	Authors are required to be transparent with respect to the level of sharing/repeatability a reader or reviewer should expect.

of artifacts; coding conventions are necessary in order to facilitate the understanding of submitted code; peer-review of artifacts is necessary in order to ensure the consistency between a publication and its supporting artifacts; licensing frameworks designed specifically for research artifacts are necessary in order to ensure that researchers get proper attribution for their work; and sharing specifications are necessary in order to ensure that readers and reviewers have an understanding of the level of sharing that is supported.

We believe that, if generally adopted, sharing specification will not only be of informational value, but will also have a positive effect on the extent to code and data is shared: knowing that reviewers may take a dim view of a paper whose sharing specification states that code and data will not be available for review, may serve as an incentive for authors to, whenever possible, share.

Acknowledgments We thank Saumya Debray and Shriram Krishnamurthi for valuable input. We also thank all the students who worked on this project, as well as the authors who contributed their time to send us code, and provide feedback on the study.

References

[1] *ESEC/FSE '11: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, New York, NY, USA, 2011. ACM. 594114.

[2] *Proceedings of the 20th Static Analysis Symposium*. Springer Verlag, June 2013.

[3] *PLDI '14: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, New York, NY, USA, 2014. ACM.

[4] W. M. 0001, I. Rojas, A. Eberhart, P. Haase, and M. Schmidt. A-r-e: The author-review-execute environment. *Procedia CS*, 4:627–636, 2011.

[5] J. Austin, T. Jackson, M. Fletcher, M. Jessop, B. Liang, M. Weeks, L. Smith, C. Ingram, and P. Watson. Carmen: Code analysis, repository and modeling for e-neuroscience. *Procedia CS*, 4:768–777, 2011.

[6] G. R. Brammer, R. W. Crosby, S. Matthews, and T. L. Williams. Paper mch: Creating dynamic reproducible science. *Procedia CS*, 4:658–667, 2011.

[7] C. Collberg, T. Proebsting, G. Moraila, A. Shankaran, Z. Shi, and A. M. Warren. Measuring reproducibility in computer systems research. Technical Report TR 13-03, Department of Computer Science, University of Arizona, Dec. 2013.

[8] C. S. Collberg, A. Huntwork, E. Carter, and G. M. Townsend. Graph theoretic software watermarks: Implementation, analysis, and attacks. In J. J. Fridrich, editor, *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2004.

[9] C. A. Dorgelo and J. M. Galloway. Notices. *Federal Register*, 79(145), July 2014. `http://www.gpo.gov/fdsys/pkg/FR-2014-07-29/pdf/2014-17761.pdf`.

[10] S. Fomel and J. F. Claerbout. Guest editors' introduction: Reproducible research. *Computing in Science and Engineering*, 11(1):5–7, 2009.

[11] M. Gavish and D. Donoho. A universal identifier for computational results. *Procedia CS*, 4:637–647, 2011.

[12] R. Gentleman. Reproducible research: A bioinformatics case study. *Statistical Applications in Genetics and Molecular Biology*, 4(1):1034, 2005.

[13] P. V. Gorp and S. Mazanek. Share: a web portal for creating and sharing executable research papers. *Procedia CS*, 4:589–597, 2011.

[14] R. Jones, editor. *ECOOP 2014 - Object-Oriented Programming - 28th European Conference, Uppsala, Sweden, July 28 - August 1, 2014. Proceedings*, volume 8586 of *Lecture Notes in Computer Science*. Springer, 2014.

[15] 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 *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '12, pages 285–296, New York, NY, USA, 2012. ACM.

[16] D. Koop, E. Santos, P. Mates, H. T. Vo, P. Bonnet, B. Bauer, B. Surer, M. Troyer, D. N. Williams, J. E. Tohline, J. Freire, and C. T. Silva. A provenance-based infrastructure to support the life cycle of executable papers. *Procedia CS*, 4:648–657, 2011.

[17] J. Kovačević. How to encourage and publish reproducible research. In *IEEE International Conference on Acoustic Speech Signal Processing*, volume IV, pages 1273–1276, Apr. 2007.

[18] S. Li-Thiao-T. Literate program execution for reproducible research and executable papers. *Procedia CS*, 9:439–448, 2012.

[19] N. Limare. Running a reproducible research journal, with source code inside. In *ICERM Workshop*, 2012.

[20] B. A. Nosek. An open, large-scale, collaborative effort to estimate the reproducibility of psychological science. *Perspectives on Psychological Science*, 7(6):657–660, 2012.

[21] Y. L. S. R. on Data and C. Sharing. Reproducible research. *Computing in Science and Engineering*, 12(5):8–13, 2010.

[22] S. Perianayagam, G. R. Andrews, and J. H. Hartman. Rex: a toolset for reproducing software experiments. In *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*, pages 613–617. IEEE, 2010.

[23] M. Schwab, N. Karrenbach, and J. Claerbout. Making scientific computations reproducible. *Computing in Science Engineering*, 2(6):61–67, 2000.

[24] V. Stodden. Enabling Reproducible Research: Open Licensing for Scientific Innovation. *Social Science Research Network Working Paper Series*, Mar. 2009.

[25] V. Stodden. The legal framework for reproducible scientific research: Licensing and copyright. *Computing in Science and Engineering*, 11(1):35–40, 2009.

[26] V. Stodden. The Scientific Method in Practice: Reproducibility in the Computational Sciences. Technical Report Working Paper 4773-10, MIT Sloan School of Management, Feb. 2010.

[27] V. Stodden, C. Hurlin, and C. Perignon. Runmycode.org: A novel dissemination and collaboration platform for executing published computational results. In *eScience*, pages 1–8. IEEE Computer Society, 2012.

[28] V. Stodden, I. Mitchell, and R. LeVeque. Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science and Engineering*, 14(4):13–17, 2012.

[29] P. Vandewalle, J. Kovacevic, and M. Vetterli. Reproducible research in signal processing - what, why, and how. *IEEE Signal Processing Magazine*, 26(3):37–47, May 2009.

[30] J. Vitek and T. Kalibera. Repeatability, reproducibility, and rigor in systems research. In *Proceedings of the ninth ACM international conference on Embedded software*, EMSOFT '11, pages 33–38, New York, NY, USA, 2011. ACM.

A Anecdotes

A.1 Anecdote 1

In 2001 the first author (Collberg) asked some colleagues at a major corporate research lab for access to a system which, eventually, would be published in a highly-regarded workshop:

```
... Do you have software available for us to play with?
```

The response was negative:

```
As you may guess, implementations of our papers (even the one on
⟨TECHNIQUE⟩) may not be made available. At least the
⟨TECHNIQUE⟩ has a chance to be accessible via applications that
may use it in whatever form.

Sorry about that.
```

As a result, our students spent a year re-implementing their system, making "reasonable" inferences in the many cases when the paper was unclear, in order to repudiate the purported results. Our conclusions were eventually published [8]. However, because of the many implementation details that were glossed over in the original paper, it is highly likely that we did not faithfully reproduce the original system, and this diminishes the value of our results.

A.2 Anecdote 2

During 2012 the first author (Collberg) together with colleagues submitted a paper to a series of security conferences: *IEEE Security and Privacy*, *CCS*, and *USENIX Security*. In the paper we showed a new attack against a defense that had been published in a top-tier security conference.[14] The reviewers of all three conferences pointed to a number of problems with the paper and it was ultimately rejected. One problem, in particular, was a less-than-conclusive security evaluation:

```
It is unfortunate that ⟨MAIN STUDENT⟩ et al. did not make
their tool publicly available, and it is great that the authors in
this paper promise to make their tools available.

The evaluation has a "built here" flavor--both the implementation and
analysis parts were done by the authors without relying on
experimental standards or using other implementations or benchmarks in
a significant way. This to me is a major threat to validity.

Interesting area to explore and useful tools to build, but overall
somewhat less than conclusive, the deployment is not seriously
considered, and the evaluation is of an "internal" kind, lacking
external benchmarks or tools that are compared directly.

2. The security evaluation does not sufficiently justify/support the
claim (in the abstract) that their new tool is resistant to known
attacks against obfuscation techniques
```

The problem was that we did not have access to ⟨SYSTEM-R⟩ and were not able to pitch it against our own tool, ⟨SYSTEM-D⟩, in order to prove, conclusively, that we would overcome the techniques published in ⟨PAPER⟩. We thus had to resort to a hand-wavy security argument which the reviewers did not find convincing.

[14] In the following, we use ⟨CONFERENCE⟩ as the name of the conference in which this paper was published, ⟨PAPER⟩ as the title of the paper, ⟨SYSTEM-R⟩ as the name of the system described in the paper, and ⟨MAIN STUDENT⟩, ⟨SECOND STUDENT⟩, ⟨JUNIOR PROFESSOR⟩, ⟨PROFESSOR⟩ as the authors. ⟨UNIVERSITY⟩ is the name of the public university at which the authors were studying/employed, at a top-10 Computer Science Department. ⟨COMPANY⟩ is the name of the company where ⟨MAIN STUDENT⟩ is now employed. ⟨SYSTEM-D⟩ is the name of the system we built and tried to publish.

A.2.1 First Emails

On October 14, 2012 we had sent this email to the authors:

> Hi!
>
> I'm Christian Collberg from the University of Arizona. I was wondering if you have a copy of ⟨SYSTEM-R⟩ that we could try out? We have a new ⟨SYSTEM⟩ and I'd like to see how well your system handles our ⟨TECHNIQUES⟩.
>
> Thank you!
>
> Christian

Three of the four email addresses extracted from the author list in the paper failed, the fourth, to ⟨PROFESSOR⟩, appeared to work. We received no response.

On February 19, 2013, we tried again:

> Hi again!
>
> I'm Christian Collberg from the University of Arizona. I wrote to you earlier to see if I could have a copy of ⟨SYSTEM-R⟩ so that we could try it out, but never received a reply. We have a new ⟨SYSTEM⟩ and I'd like to see how well your system handles our ⟨TECHNIQUES⟩.

Again, we received no response from ⟨PROFESSOR⟩, and the remaining addresses still bounced, as expected.

A.2.2 Trying to Reimplement

By September we had gotten frustrated with the situation, and decided to go ahead and reimplement ⟨SYSTEM-R⟩ ourselves. We started reading the conference paper, the technical report on which it was based, and ⟨MAIN STUDENT⟩'s PhD thesis in earnest, and painstakingly tried to reproduce their implementation. We soon ran into some problems, and on September 11, 2013 we sent the following email:

Hi!

I'm attempting to reimplement ⟨SYSTEM-R⟩, and as I'm working my way through
the ⟨PAPER⟩ and tech-report, I have come across some issues that
I hope you would be able to help me with:

1) [...]

2) [...]

3) In algorithm 2, in the ⟨PAPER⟩, in the
   Initialization step, you write ⟨FORMULA INVOLVING σ⟩
   where σ isn't defined .... In the tech report you write for
   the same step ⟨DIFFERENT FORMULA INVOLVING σ⟩.

   Can you enlighten me as to what is going on in this step?

4) [...]

5) In ... Section ⟨SECTION⟩ you write ⟨FORMULA⟩
   ... [which] doesn't typecheck. Did you mean
   ⟨DIFFERENT FORMULA⟩
   ?

6) The function ⟨FUNCTION⟩ appears to be defined in Section ⟨SECTION⟩
   but never used.

Thank you for your help,

Christian Collberg

This time we had spent significant effort tracking down working email addresses. As it turns out, both students and the junior faculty member had left for industrial research positions, and, apparently, had not had their email forwarded. We found email addresses from ⟨SECOND STUDENT⟩ and ⟨JUNIOR PROFESSOR⟩ and on September 12, 2013, we received the following response from ⟨JUNIOR PROFESSOR⟩:

I'm going to redirect you to ⟨MAIN STUDENT⟩, as he was the [then-] grad
student who did the actual work. We haven't stayed in touch and I don't
have his current email address, but you can probably ping him over
LinkedIn: ⟨URL⟩

We tried to connect with ⟨MAIN STUDENT⟩ on LinkedIn, but he would not respond.

Later the same day, we received a second clarifying email from ⟨JUNIOR PROFESSOR⟩:

I unfortunately have few recollections of the work. I just went and pulled up a PDF of the online tech report, and it was a bit like seeing a new paper for the first time. I wasn't able to immediately see answers to your questions, and if I sat down to reread it all again, I'd probably just end up with the same questions you asked and no new answers.

⟨MAIN STUDENT⟩'s writing style had a tendency to be verbose, so we were continually pruning. That may have impacted the final result given things like missing sigma definitions. Apologies if we accidentally deleted some important bits.

Last I knew (which was several years ago), he was in ⟨RESEARCHER's⟩ group at ⟨COMPANY⟩. You might try reaching him via ⟨RESEARCHER⟩.

I'd also consider him to be a member of the ⟨UNIVERSITY⟩'s PTSD[a] club. He may be deliberately ignoring communication related to ⟨UNIVERSITY⟩.

[a] PostTraumatic Stress Disorder.

On September 19, 2013, we found the main student's corporate email address by strong-arming an acquaintance also working at ⟨COMPANY⟩, who was not supposed to hand it out. On September 16, 2013, satisfied that we finally had a working email address for the main author of the paper, we sent ⟨MAIN STUDENT⟩ a query for the code and a request to clear up the confusions in the paper. We received no reply.

A.2.3 Formal Request for Source Code

We next made a formal request to ⟨PROFESSOR⟩ under the open records act of the state in which ⟨UNIVERSITY⟩ is located:

```
Dear Prof. ⟨PROFESSOR⟩,

My name is Christian Collberg, and I'm a Professor
of Computer Science at the University of Arizona.

I have on several occasions emailed you and your
research group asking for code, data, and information
related to the ⟨SYSTEM-R⟩ project. I have received no response
from you.

I am therefore making a formal request under the
   ⟨STATE⟩ OPEN RECORDS ACT
   ⟨URL⟩
for
   ALL SOURCE CODE, NOTES, AND TEST DATA
   RELATED TO THE ⟨SYSTEM-R⟩ SYSTEM
as published in
   * ⟨PAPER⟩
   * ⟨TECHNICAL REPORT⟩
   * ⟨PHD THESIS⟩
and supported under federal grants

   * ⟨MILITARY AGENCY⟩ GRANT #...
   * ⟨NSF GRANT #...⟩
   * ⟨NSF GRANT #...⟩

Note that the 2005 NSF Grant Policy Manual
   http://www.nsf.gov/pubs/manuals/gpm05_131/gpm7.jsp#734
under which your grants were funded, states that
   b) Investigators are expected to share with other researchers, at no
      more than incremental cost and within a reasonable time, the primary
      data, samples, physical collections and other supporting materials
      created or gathered in the course of work under NSF grants. Grantees
      are expected to encourage and facilitate such sharing.
   c) Investigators and grantees are encouraged to share software and
      inventions created under the grant or otherwise make them or their
      products widely available and usable.
I am willing to pay applicable fees. These records are sought in
furtherance of scholarly research, and I am employed by an Educational,
Non-commercial Scientific Institution. Therefore, I ask that fees,
other than duplication fees, be waived.

Thank you,
Prof. Christian Collberg
⟨ADDRESS, EMAIL, PHONE⟩
CC: ⟨DEPARTMENT CHAIR⟩
```

No response was forthcoming within the 3 day limit imposed by the statute, so on September 23, 2013 we sent a request to ⟨UNIVERSITY⟩'s legal department:

```
Dear Sir/Madam,

My name is Christian Collberg, and I'm a Professor
of Computer Science at the University of Arizona.

Last week I made a  request to Prof. ⟨PROFESSOR⟩ of
the Computer Science Department at ⟨UNIVERSITY⟩.
Please see below. It has been over 3 days, and I have
yet to receive a response. I wonder if you could let me
know what further steps I can take.
```

On September 24, 2013 we received the following response:

```
Dear Dr. Colberg:

The Office of Legal Affairs at ⟨UNIVERSITY⟩ has
received your Open Records Act Request, dated September 16, 2013,
appearing below.  We understand your request to be seeking:

  "ALL SOURCE CODE, NOTES, AND TEST DATA
   RELATED TO THE ⟨SYSTEM-R⟩ SYSTEM"

The requested records have not been located.  We are still in the
process of checking with additional personnel within the ⟨UNIVERSITY⟩.
However, to the extent such records may exist, they will not be
produced pursuant to ⟨OPEN RECORDS STATUTE⟩.  Please let me know if you have any
questions or concerns.  We will let you know if there are any such
records in existence as soon as possible.  Thank you.
```

The provision of the ⟨OPEN RECORDS STATUTE⟩ the preceding email refers to states that:

```
  Any data, records, or information developed, collected, or
  received by or on behalf of faculty, staff, employees, or students
  of an institution of higher education or any public or private
  entity supporting or participating in the activities of an
  institution of higher education in the conduct of, or as a result
  of, study or research on medical, scientific, technical, scholarly,
  or artistic issues, whether sponsored by the institution alone or in
  conjunction with a governmental body or private entity, until such
  information is published, patented, otherwise publicly disseminated,
  or released to an agency whereupon the request must be made to the
  agency. This paragraph shall apply to, but shall not be limited to,
  information provided by participants in research, research notes and
  data, discoveries, research projects, methodologies, protocols, and
  creative works;
```

On September 24, 2013, we replied to the ⟨UNIVERSITY⟩ legal department with the following email:

I am confused by your statement

> "We are still in the process of checking with additional personnel within the ⟨UNIVERSITY⟩. However, to the extent such records may exist, they will not be produced pursuant to ⟨OPEN RECORDS STATUTE⟩."

⟨OPEN RECORDS STATUTE⟩ states that:

> Any data, records, or information developed, collected, or received by or on behalf of faculty, staff, employees, or students of an institution of higher education or any public or private entity supporting or participating in the activities of an institution of higher education in the conduct of, or as a result of, study or research on medical, scientific, technical, scholarly, or artistic issues, whether sponsored by the institution alone or in conjunction with a governmental body or private entity, until such information is published, patented, otherwise publicly disseminated, or released to an agency whereupon the request must be made to the agency. This paragraph shall apply to, but shall not be limited to, information provided by participants in research, research notes and data, discoveries, research projects, methodologies, protocols, and creative works;

Since the information we're requesting has been published the clause "until such information is published, patented, otherwise publicly disseminated" applies, and the information should be released. The clause "or released to an agency whereupon the request must be made to the agency" does not apply since what was released to ⟨MILITARY AGENCY⟩ appears to be the following technical report

> ⟨FINAL REPORT RELEASED TO MILITARY AGENCY⟩

which does not include the source code of the ⟨SYSTEM-R⟩ system, which we are requesting, and that information should be released.

I would appreciate it if you would clear up these issues for me. Also, please let me know what steps you are taking to locate the requested information and when you believe this investigation will have been concluded.

On September 26, 2013, we received this response from ⟨UNIVERSITY⟩'s legal department:

Good afternoon,

We have been unable to locate a confirmed instance of ⟨SYSTEM-R⟩ source code on any ⟨UNIVERSITY⟩ system. The requested records do not exist.

Please let me know if you have any additional questions or concerns.

A.2.4 Formal Request for Email Trail

We next sent a request for the emails between the authors, in order to trace the whereabouts of the source code of ⟨SYSTEM-R⟩:

> Pursuant to ⟨OPEN RECORDS STATUTE⟩, the Open Records Act,
> I request copies of all electronic mail between
> ⟨MAIN STUDENT⟩
> ⟨SECOND STUDENT⟩
> ⟨JUNIOR PROFESSOR⟩
> ⟨PROFESSOR⟩
> regarding the article
> ⟨PAPER⟩
> published in
> ⟨CONFERENCE⟩
> and the development of the
> ⟨SYSTEM-R⟩
> system described therein.
>
> In accordance with ⟨OPEN RECORDS STATUTE⟩ I request that the
> electronic databases to which the following email addresses
> are attached be searched:
> ⟨UNIVERSITY EMAIL ADDRESSES⟩.
> I also request that any private (non-university) email accounts containing
> work-related emails be searched.
>
> I am willing to pay applicable fees. These records are sought in
> furtherance of scholarly research, and I am employed by an Educational,
> Non-commercial Scientific Institution. Therefore, I ask that fees,
> other than duplication fees, be waived.
>
> CC:
> ⟨DEPARTMENT CHAIR⟩

We received this reply:

Good morning,

The Office of Legal Affairs at ⟨UNIVERSITY⟩ has
received your ⟨OPEN RECORDS STATUTE⟩ request appearing below. We understand
your request to be seeking:

all electronic mail between
⟨MAIN STUDENT⟩
⟨SECOND STUDENT⟩
⟨JUNIOR PROFESSOR⟩
⟨PROFESSOR⟩
regarding the article
⟨PAPER⟩
published in
⟨CONFERENCE⟩
of the
⟨SYSTEM-R⟩
System [source code] described therein

To the extent that any such records may exist, they will not be produced
pursuant to ⟨OPEN RECORDS STATUTE⟩

⟨PARALEGAL, OFFICE OF LEGAL AFFAIRS⟩

This message is intended to be used exclusively by the addressee(s).
Unauthorized disclosure or use of this information is strictly prohibited.
If you have received this communication in error, please permanently
dispose of the original message and notify me immediately by sending an
email to ⟨EMAIL ADDRESS⟩

Also, please note that most communications to or from ⟨UNIVERSITY⟩
employees are a public record and available to the public and the media
upon request under ⟨STATE⟩'s broad open records law. Therefore, this
e-mail communication and any response may be subject to public disclosure.

We responded:

Thanks for your quick response.

> To the extent that any such records may exist, they will not be produced
> pursuant to ⟨OPEN RECORDS STATUTE⟩.

As you are aware, ⟨OPEN RECORDS STATUTE⟩ state that:

(35) Data, records, or information of a proprietary nature produced or
collected by or for faculty or staff of state institutions of higher
learning, or other governmental agencies, in the conduct of, or as a
result of, study or research on commercial, scientific, technical, or
scholarly issues, whether sponsored by the institution alone or in
conjunction with a governmental body or private concern, where such
data, records, or information has not been publicly released,
published, copyrighted, or patented;

(36) Any data, records, or information developed, collected, or
received by or on behalf of faculty, staff, employees, or students of
an institution of higher education or any public or private entity
supporting or participating in the activities of an institution of
higher education in the conduct of, or as a result of, study or
research on medical, scientific, technical, scholarly, or artistic
issues, whether sponsored by the institution alone or in conjunction
with a governmental body or private entity, until such information is
published, patented, otherwise publicly disseminated, or released to
an agency whereupon the request must be made to the agency. This
paragraph shall apply to, but shall not be limited to, information
provided by participants in research, research notes and data,
discoveries, research projects, methodologies, protocols, and creative
works;

The clause "information has not been publicly released, published,
copyrighted, or patented" is relevant here, since the ⟨SYSTEM-R⟩ system
has been published. Hence, information about this system, such as
research notes, contained, as in this case, in emails, is public
record and subject to disclosure. The clause "released to an agency
whereupon the request must be made to the agency" does *not* apply
since, what was released to the NSF/⟨MILITARY AGENCY⟩ was research reports, and not
source code and research notes.

I must therefore reiterate my request for emails between the authors
related to the ⟨SYSTEM-R⟩ system to be released.

> Also, please note that most communications to or from ⟨UNIVERSITY⟩
> employees are a public record and available to the public and the media
> upon request under ⟨STATE⟩'s broad open records law. Therefore, this
> e-mail communication and any response may be subject to public disclosure.

:)

Best wishes,
Christian Collberg

On October 8, 2013, we received the final communication:

> Dr. Collberg:
>
> The Office of Legal Affairs at ⟨UNIVERSITY⟩ has
> received your renewed/reiterated Open Records Act request below, dated
> October 3, 2013. We understand your request to be seeking emails
> to/from ⟨PROFESSOR⟩, ⟨MAIN STUDENT⟩, ⟨SECOND STUDENT⟩, ⟨JUNIOR PROFESSOR⟩
> ‘‘related to the ⟨SYSTEM-R⟩ system’’.
>
> To the extent that responsive records may exist, we estimate a total
> cost of $2,263.66* to search for, retrieve, redact and produce such
> records. There will also be a charge of $.10 per copy for any paper
> copies that you would like to receive. Please note that we will not
> produce any records, or portions of records, that are exempted
> pursuant to ⟨OPEN RECORDS STATUTE⟩, including but not limited to ⟨subclause⟩.
>
> If you would like to proceed with your request, please provide your
> check payable to ‘‘⟨UNIVERSITY⟩’’ for $2,263.66.
> Please note that you will be charged the actual cost, regardless of
> whether it is more or less than the estimated amount.
>
> **8 hrs @ $34.04 search/retrieval
> 37 hrs @53.82 review/redact records
>
> ⟨PARALEGAL, OFFICE OF LEGAL AFFAIRS⟩

We did not take up the ⟨UNIVERSITY⟩ on their offer, since we believed the charges requested were not reasonable under the ⟨OPEN RECORDS STATUTE⟩, which states:

> (c)(1) An agency may impose a reasonable charge for the search,
> retrieval, redaction, and production or copying costs for the
> production of records pursuant to this article. An agency shall
> utilize the most economical means reasonably calculated to identify
> and produce responsive, non excluded documents. Where fees for
> certified copies or other copies or records are specifically
> authorized or otherwise prescribed by law, such specific fee shall
> apply when certified copies or other records to which a specific fee
> may apply are sought. In all other instances, the charge for the
> search, retrieval, or redaction of records shall not exceed the
> prorated hourly salary of the lowest paid full-time employee who, in
> the reasonable discretion of the custodian of the records, has the
> necessary skill and training to perform the request; provided,
> however, that no charge shall be made for the first quarter hour.

A.2.5 Request for the NSF grant application

We made a formal request to the NSF for the two applications of the grants that supported the research. In one, the Principal Investigator (⟨PROFESSOR⟩ in the discussion above) writes:

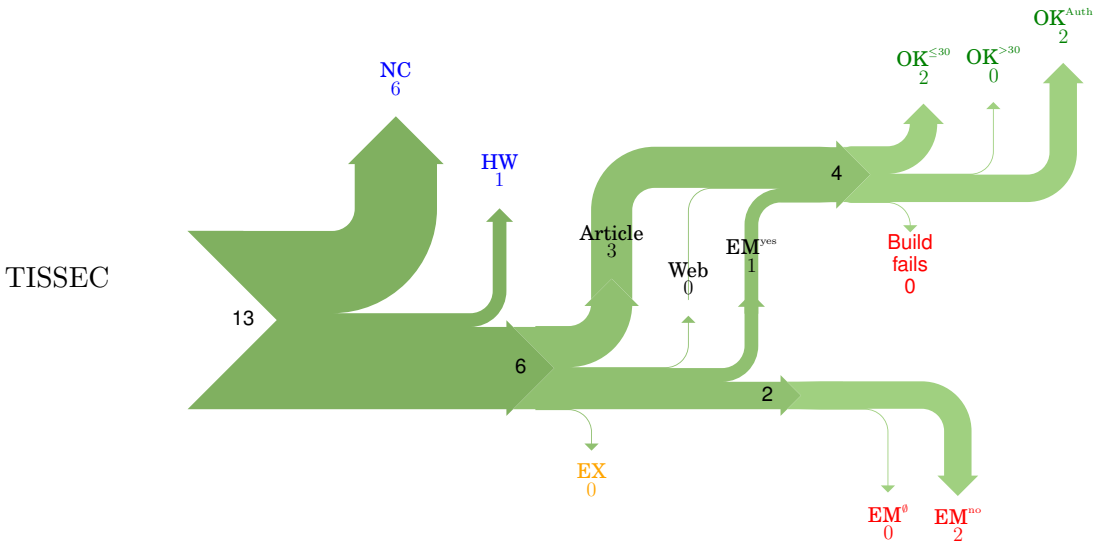
```
We wil also make our data and software available to the research
community when appropriate.
```

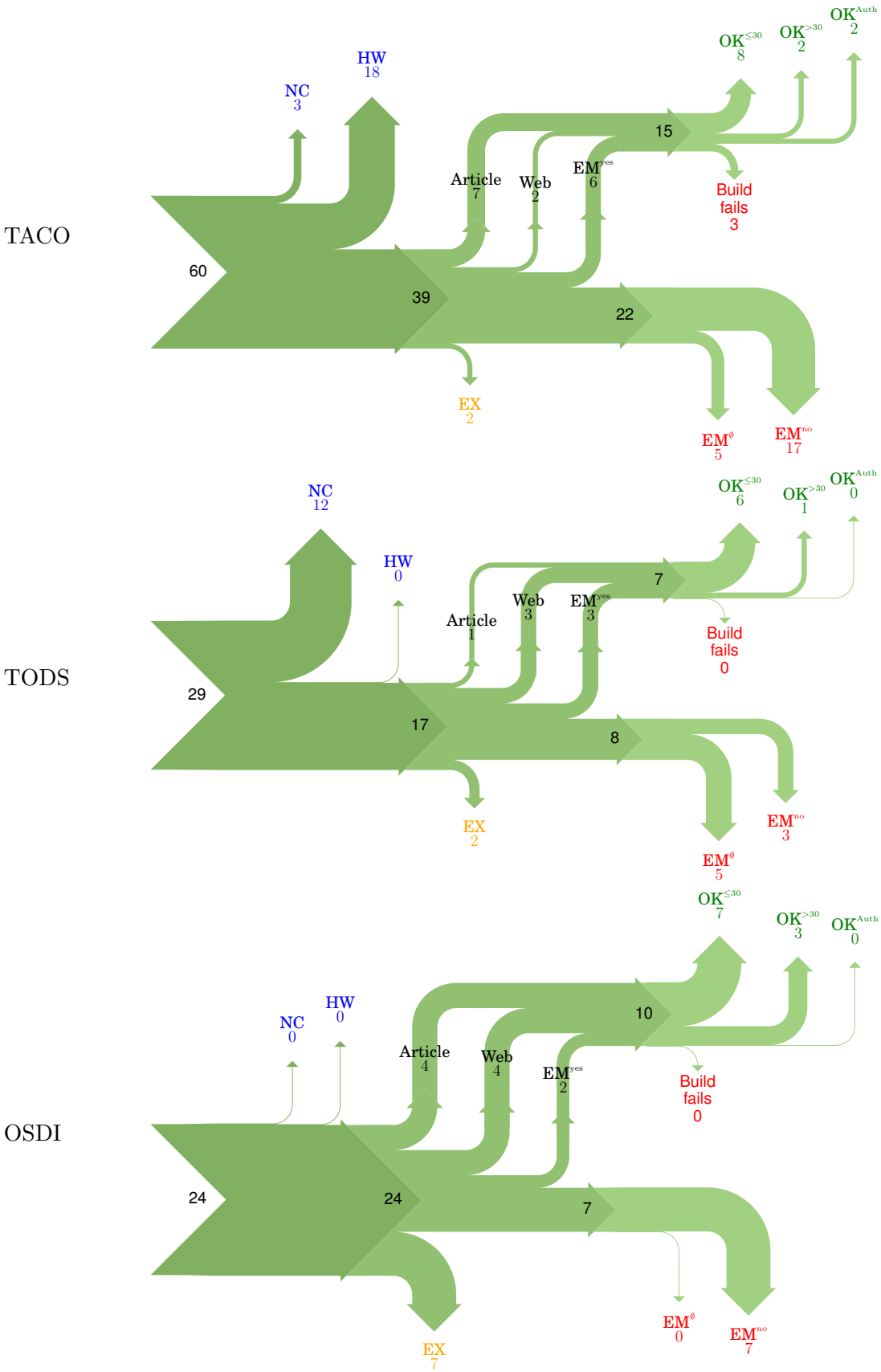
We never received any communication from ⟨PROFESSOR⟩, in particular no explanation as to why releasing the code might not have been "appropriate."

B Data for Individual Venues

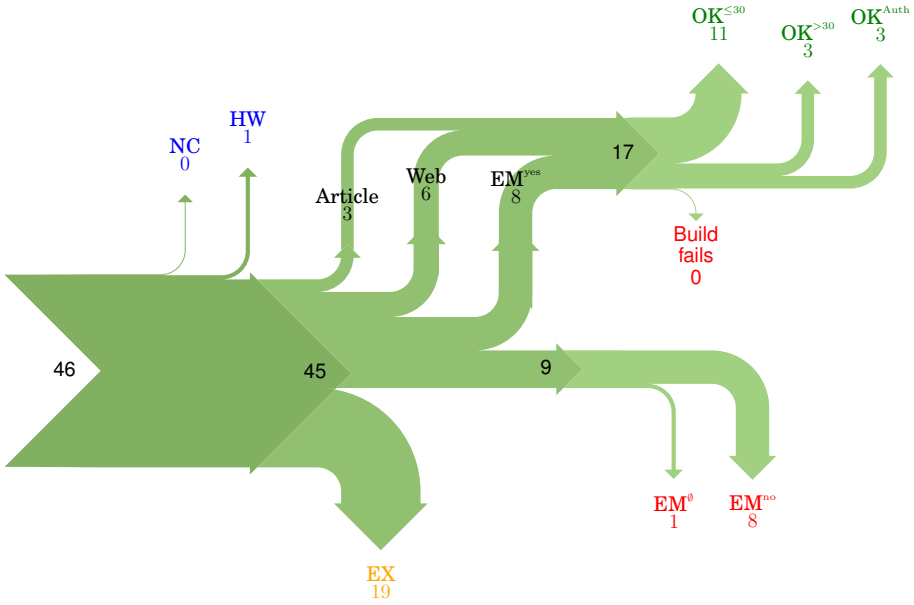
We here give the diagrams for individual venues. The abbreviations are the same as in Table **??**: HW=Requires special hardware; NC=Not backed by code; EX=Excluded due to overlapping author lists; BC=Backed by code; EM^{yes}=Author provides code after being emailed; EM^{no}=Author responds to email that code cannot be provided; $\text{EM}^{\emptyset}$=Author does not respond to email within reasonable time; $\text{OK}^{\leq 30}$=We succeed to build in $\leq$ 30 minutes; $\text{OK}^{>30}$=We succeed to build in $>$ 30 minutes; OK^{Auth}=We fail to build, but the author says the code builds.

Table 8: Results for individual publications.

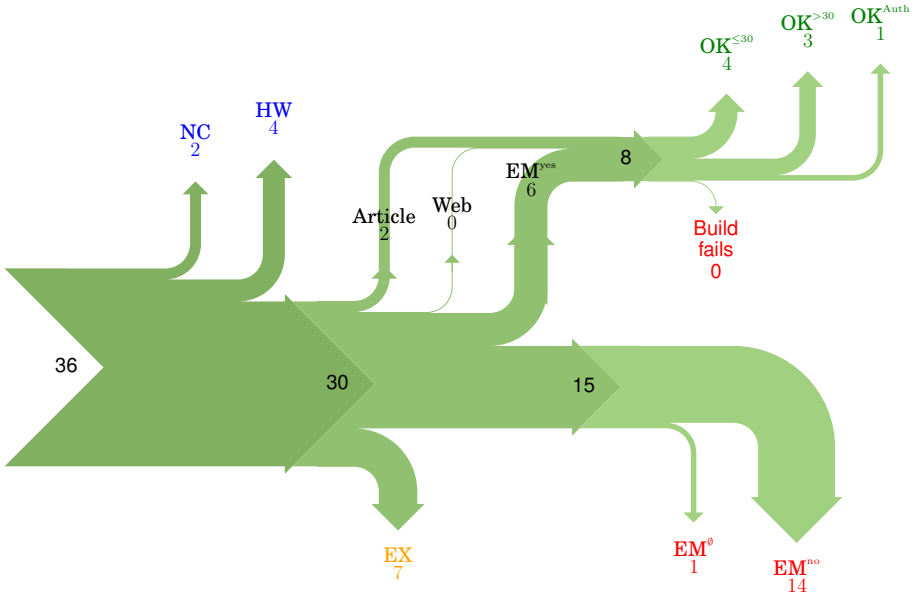


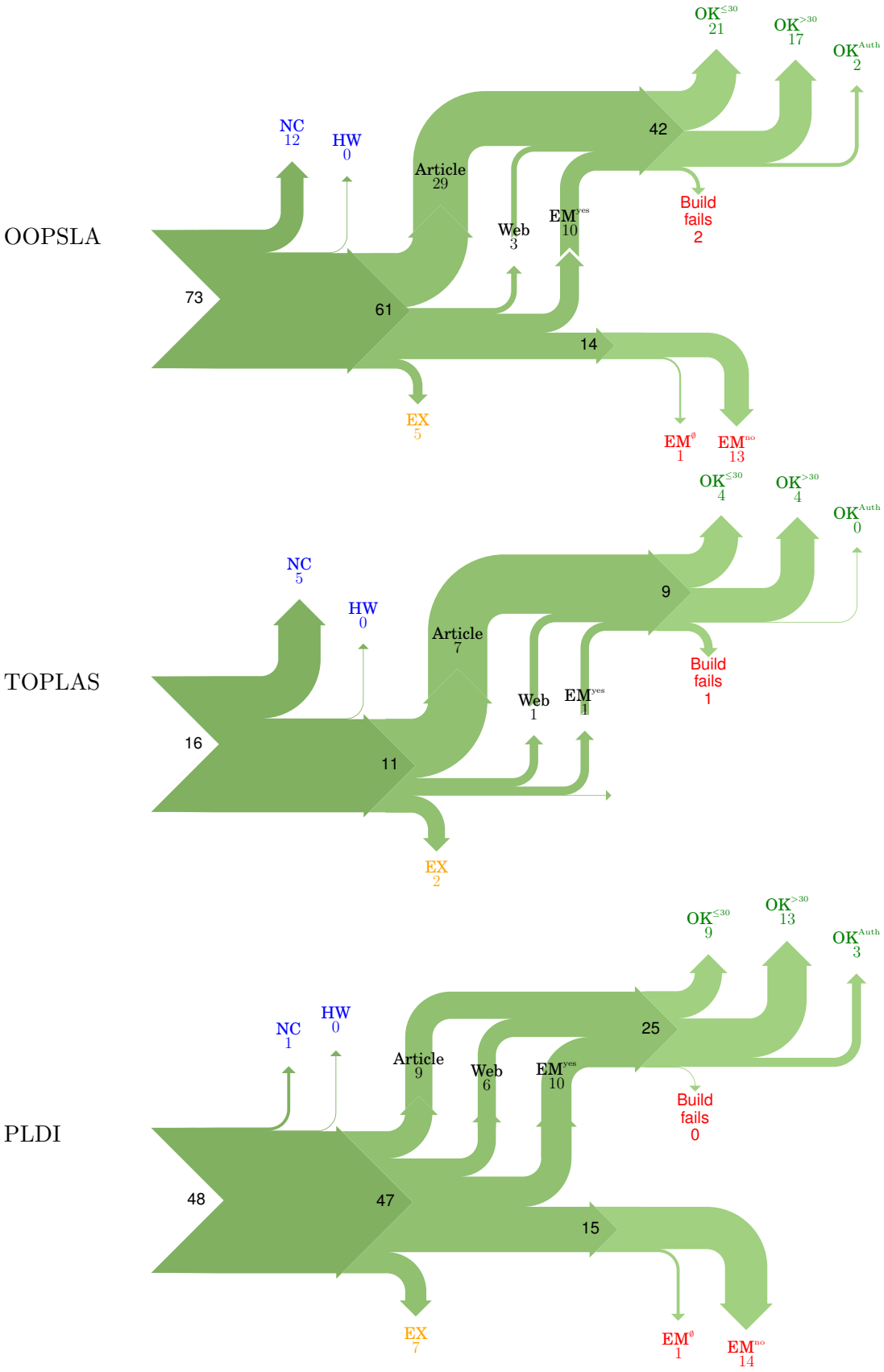

TACO
NC 3
HW 18
60
Article 7
Web 2
EM^yes 6
15
OK^≤30 8
OK^>30 2
OK^Auth 2
Build fails 3
39
22
EX 2
EM^∅ 5
EM^no 17
TODS
NC 12
HW 0
29
Article 1
Web 3
EM^yes 3
7
OK^≤30 6
OK^>30 1
OK^Auth 0
Build fails 0
17
8
EX 2
EM^no 3
EM^∅ 5
OSDI
NC 0
HW 0
24
Article 4
Web 4
EM^yes 2
10
OK^≤30 7
OK^>30 3
OK^Auth 0
Build fails 0
24
7
EX 7
EM^∅ 0
EM^no 7

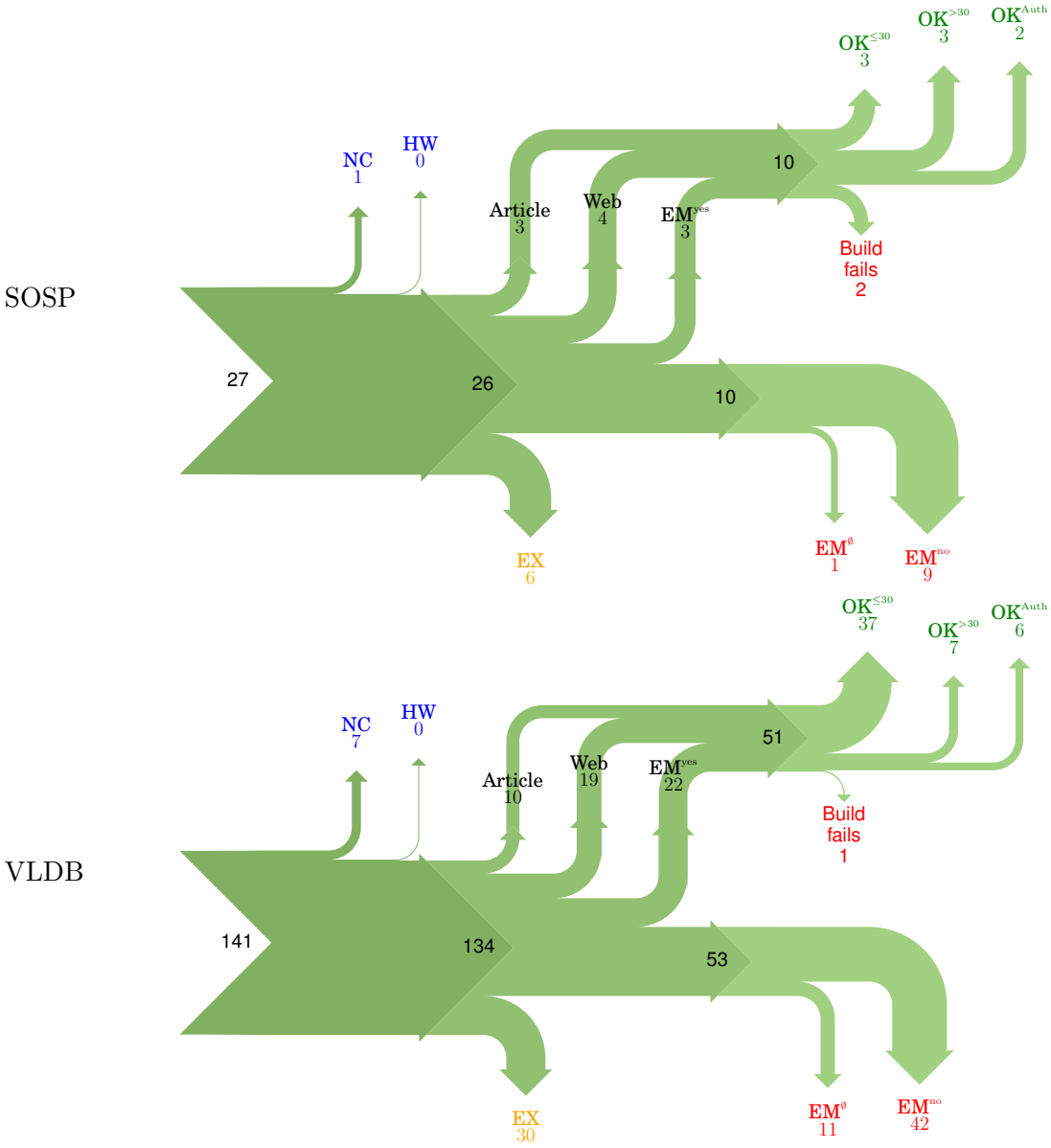
SIGMOD



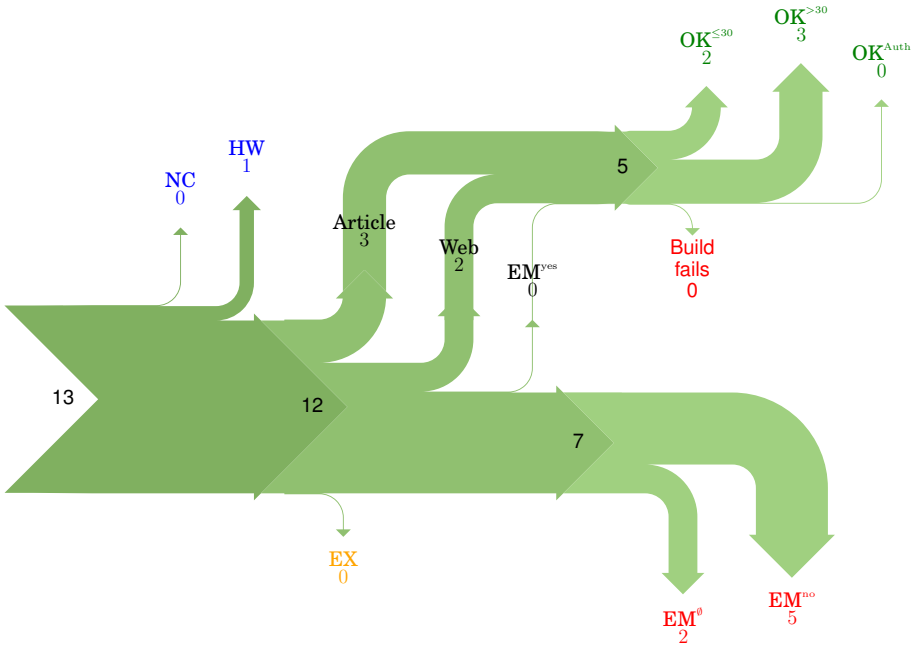
ASPLOS



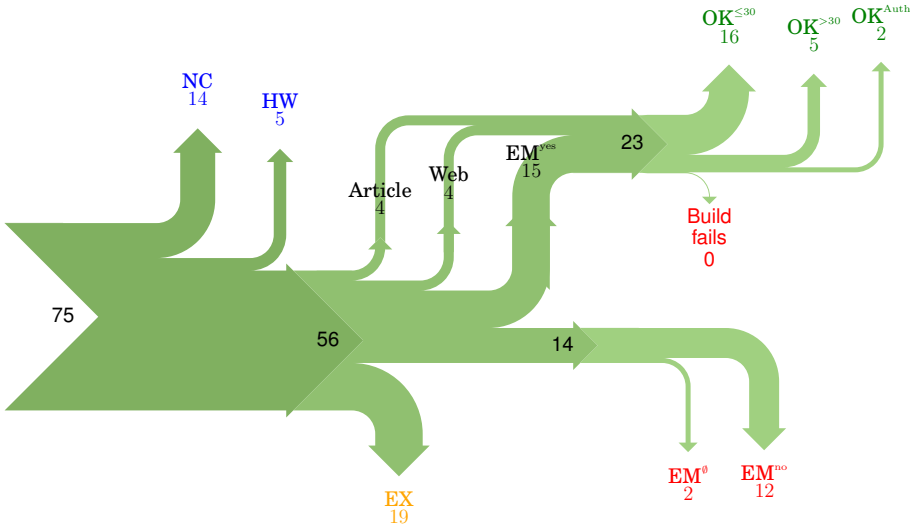

OOPSLA
NC 12
HW 0
Article 29
Web 3
EM^yes 10
OK^≤30 21
OK^>30 17
OK^Auth 2
Build fails 2
73
61
42
14
EX 5
EM^∅ 1
EM^no 13
TOPLAS
NC 5
HW 0
Article 7
Web 1
EM^yes 1
OK^≤30 4
OK^>30 4
OK^Auth 0
Build fails 1
16
11
9
EX 2
PLDI
NC 1
HW 0
Article 9
Web 6
EM^yes 10
OK^≤30 9
OK^>30 13
OK^Auth 3
Build fails 0
48
47
25
15
EX 7
EM^∅ 1
EM^no 14

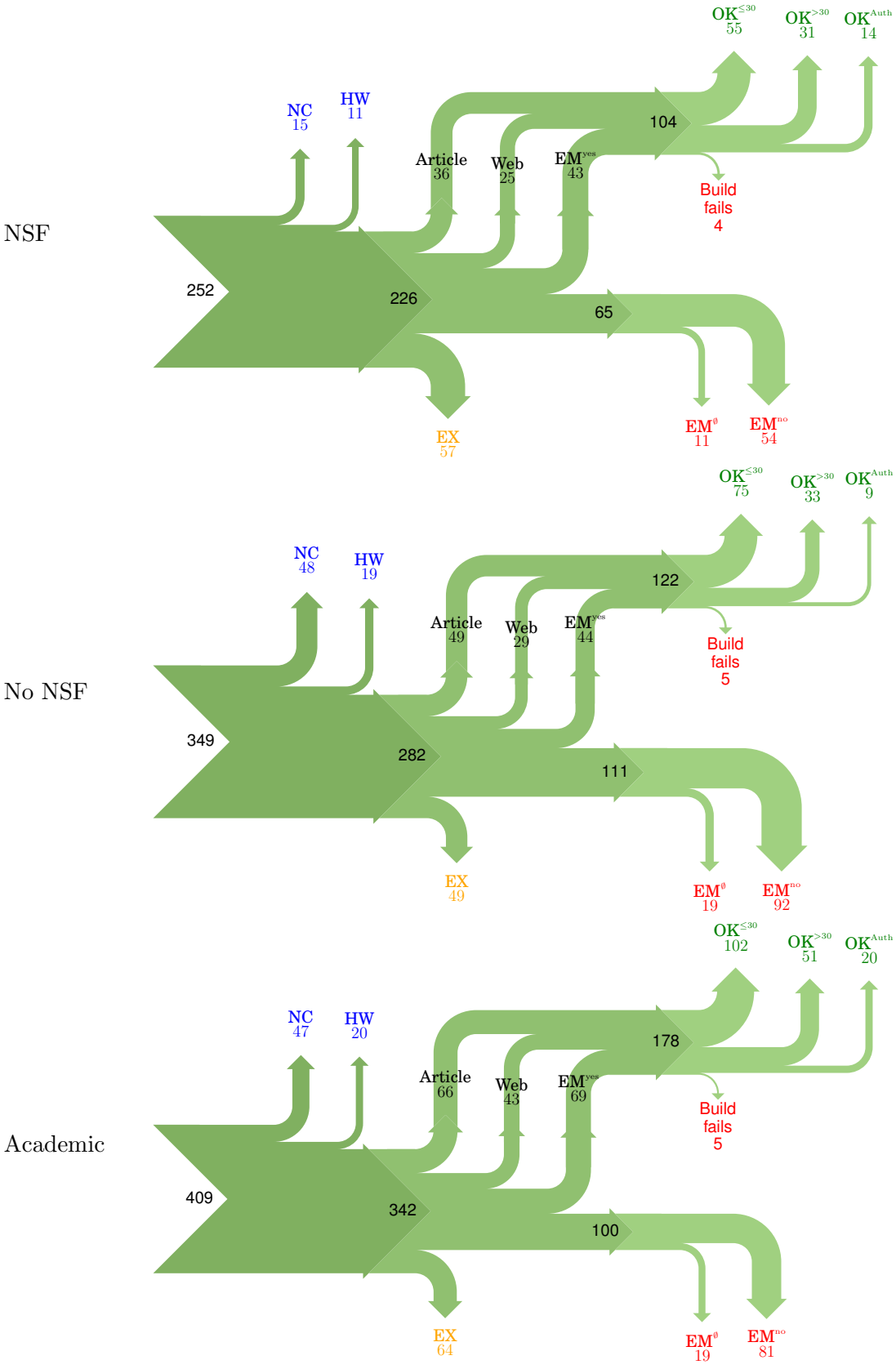

SOSP
NC 1
HW 0
Article 3
Web 4
EM^yes 3
10
OK^≤30 3
OK^>30 3
OK^Auth 2
Build fails 2
27
26
10
EX 6
EM^∅ 1
EM^no 9
VLDB
NC 7
HW 0
Article 10
Web 19
EM^yes 22
51
OK^≤30 37
OK^>30 7
OK^Auth 6
Build fails 1
141
134
53
EX 30
EM^∅ 11
EM^no 42

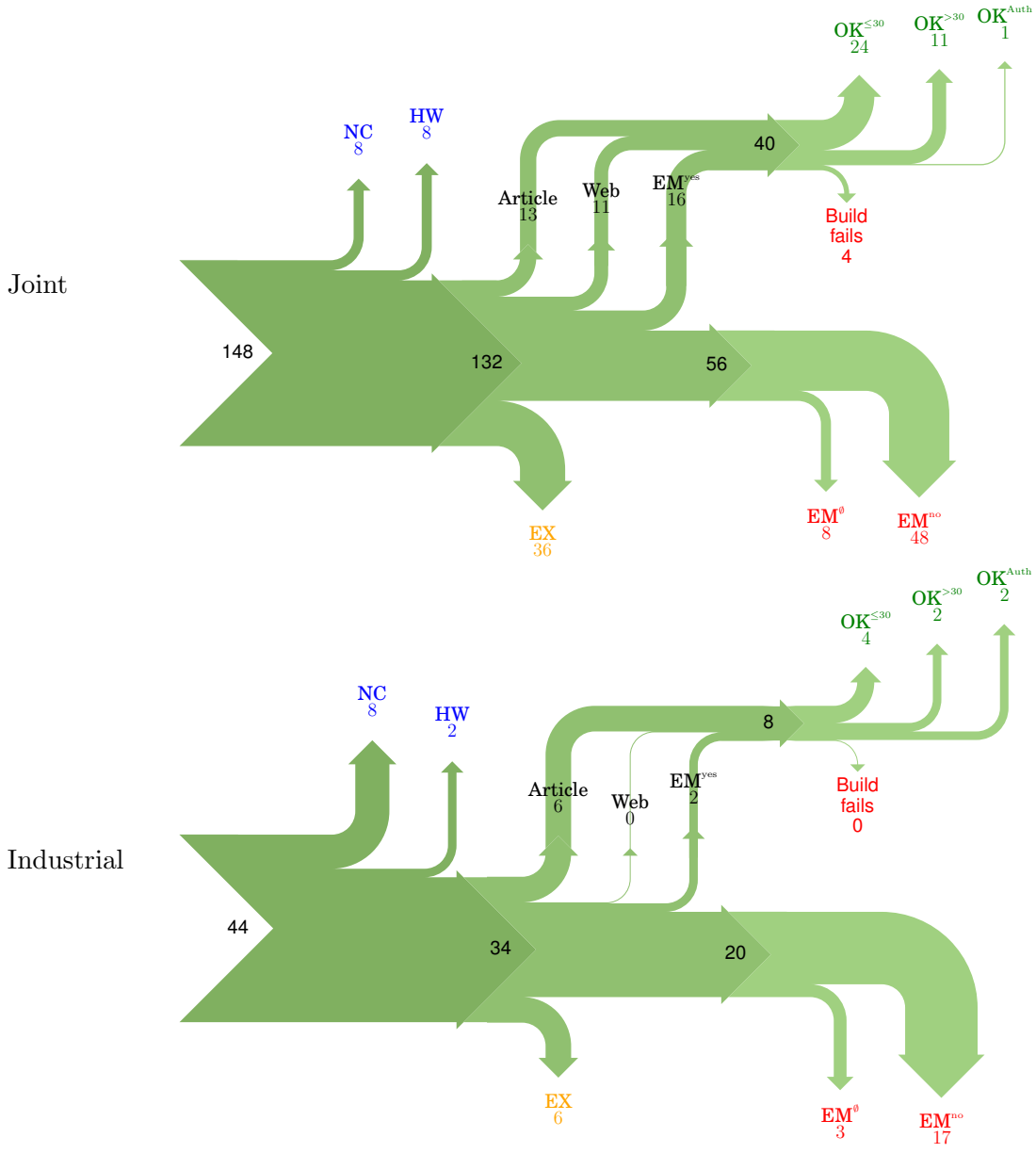
TOCS


NC
0
HW
1
Article
3
Web
2
EM^yes
0
13
12
7
5
EX
0
Build
fails
0
OK^≤30
2
OK^>30
3
OK^Auth
0
EM^∅
2
EM^no
5

CCS


NC
14
HW
5
Article
4
Web
4
EM^yes
15
75
56
23
14
EX
19
Build
fails
0
OK^≤30
16
OK^>30
5
OK^Auth
2
EM^∅
2
EM^no
12


NSF
NC 15
HW 11
252
Article 36
Web 25
EM^yes 43
226
104
65
OK^≤30 55
OK^>30 31
OK^Auth 14
Build fails 4
EX 57
EM^∅ 11
EM^no 54
No NSF
NC 48
HW 19
349
Article 49
Web 29
EM^yes 44
282
122
111
OK^≤30 75
OK^>30 33
OK^Auth 9
Build fails 5
EX 49
EM^∅ 19
EM^no 92
Academic
NC 47
HW 20
409
Article 66
Web 43
EM^yes 69
342
178
100
OK^≤30 102
OK^>30 51
OK^Auth 20
Build fails 5
EX 64
EM^∅ 19
EM^no 81



C Author Survey

Below, we show the survey questions that were sent to every author in order to help verify our results. The actual survey was taken on-line through `surveymoz.com`.

Study on Repeatability in Computer Systems Research

Dear author, Over the last year we have conducted a study into the extent to which Computer Systems researchers share their research artifacts, and whether published code builds. In fact, you may have received an email from us asking for the code related to your paper or you might have

come across our website `reproducibility.cs.arizona.edu` or even read our (very preliminary) technical report `http://reproducibility.cs.arizona.edu/tr.pdf`. In our study we looked for code related to papers published in ACM conferences and journals, and, if found, tried to build it. To make sure that our results are as accurate as possible, please answer the following questionnaire related to your paper.

> **Note that, unless otherwise specified, we will consider the answers you give to be public, i.e. we reserve the right to publish them on our website and in other publications. In the last section of this questionnaire there is an opportunity for you to provide us with confidential feedback.**
>
> **In this study, we're measuring what we call "weak repeatability." We say a project is weakly repeatable if is backed by code, this code is made available to the public, and it builds. In this context, THEORETICAL means that no source code was written for the project. PRACTICAL means that source code was written; this could include C/Java/... code to build an actual application, but also Bash/Python/... scripts used to process collected data, etc.. In other words, a PRACTICAL paper is one where the results in the paper in some way depend on the correctness of software written for the project; a THEORETICAL paper is one for which this is not the case. [This comment was added to the survey September 16, 21:24 UTC.]**[15]

We appreciate your help,
Christian Collberg
Todd Proebsting

Classifying the paper

We classified each paper as being PRACTICAL (the results in the paper are, at least partially, backed by code); THEORETICAL (the results in the paper are not backed by code); or HARDWARE (the results in the paper are based on special hardware - not available to us and most typical researchers).

1) Please indicate what you believe the correct classification should be. If your paper is best classified as HARDWARE or THEORETICAL, you are finished, and we will take you to the end of the survey.

- **PRACTICAL**
- **THEORETICAL**
- **HARDWARE**

If your classification is different from ours, please explain: ☐

Finding the code

We tried to find the code related to your paper through links embedded in the paper (if any), by looking at your website, by checking common code repositories such as GitHub, by doing a web search, or by sending you email.

[15] The initial responses we received to the survey indicated a confusion with respect to our use of the terms "THEORETICAL and "PRACTICAL." In our study, these mean "not backed by code" and "backed by code," respectively, but, they have very different connotation for many researchers. We therefore attached a clarification to the survey.

2) Have you published the code related to your paper, i.e. made it public by posting on your website, GitHub/Sourceforge/..., or making it available through email requests?

- **Yes**
- **No**

3) Is your published code identical to the version that you ran to get the results in the paper (ignoring inconsequential bug fixes)?

- **Yes**
- **No, but it is possible to make that version available**
- **No, and it is not possible to make that version available**

Comments: [ ]

4) Did we, in fact, find your published code? (The email linking to this survey contains the location where we found the code.)

- **Yes**
- **No**

If your answer is "no", please point us to the correct location: [ ]

Building the code

We made two attempts to build any code that we found. First, our students were given a maximum of 30 minutes to attempt to build the code. If that failed, a (mostly) different group would spend an "unbounded" amount of time to try to build, until they succeeded or we felt no more progress could be made. (The email that links to this survey contains the result of our effort.)

5) Which of the following best describes your code:

- **I believe the code you downloaded can be built with reasonable effort**
- **I believe the code you downloaded may have had problems that would prevent it from being easily built**
- **Other (please specify):** [ ]

Comments: [ ]

6) If we failed to build your code, but you believe your code should build, please have a look at our build notes (the link is in the email you just received) and give us a hint as to what we might have done wrong! [ ]

Comments

7) Provide any comments you'd like publicly attributed to you, including as part of our database of results regarding your work. I.e., we will link these to your row in the data table on our web site. We reserve the right to sanitize any html and remove any offensive language. [ ]

8) Provide any comments we may publish anonymously, but which will be excluded from our database of results regarding your work. I.e. we may quote these on our web site or use them in future publications or presentations, but we will anonymize them. [ ]