DOI:10.1145/2812803

To encourage repeatable research, fund repeatability engineering and reward commitments to sharing research artifacts.

BY CHRISTIAN COLLBERG AND TODD A. PROEBSTING

Repeatability in Computer Systems Research

IN 2012, WHEN reading a paper from a recent premier computer security conference, we came to believe there is a clever way to defeat the analyses asserted in the paper, and, in order to show this we wrote to the authors (faculty and graduate students in a highly ranked U.S. computer science department) asking for access to their prototype system. We received no response. We thus decided to reimplement the algorithms in the paper but soon encountered obstacles, including a variable used but not defined; a function defined but never used; and a mathematical formula that did not typecheck. We asked the authors for clarification and received a single response: "I unfortunately have few recollections of the work ... "

We next made a formal request to the university for the source code under the broad Open Records Act (ORA) of the authors' home state. The university's legal department responded with: "We have been unable to locate a confirmed instance of [system's] source code on any [university] system."

Expecting a research project of this magnitude to be developed under source code control and properly backed up, we made a second ORA request, this time for the email messages among the authors, hoping to trace the whereabouts of the source code. The legal department first responded with: "... the records will not be produced pursuant to [ORA sub-clause]." When we pointed out reasons why this clause does not apply, the university relented but demanded $2,263.66 " ... to search for, retrieve, redact and produce such records." We declined the offer.

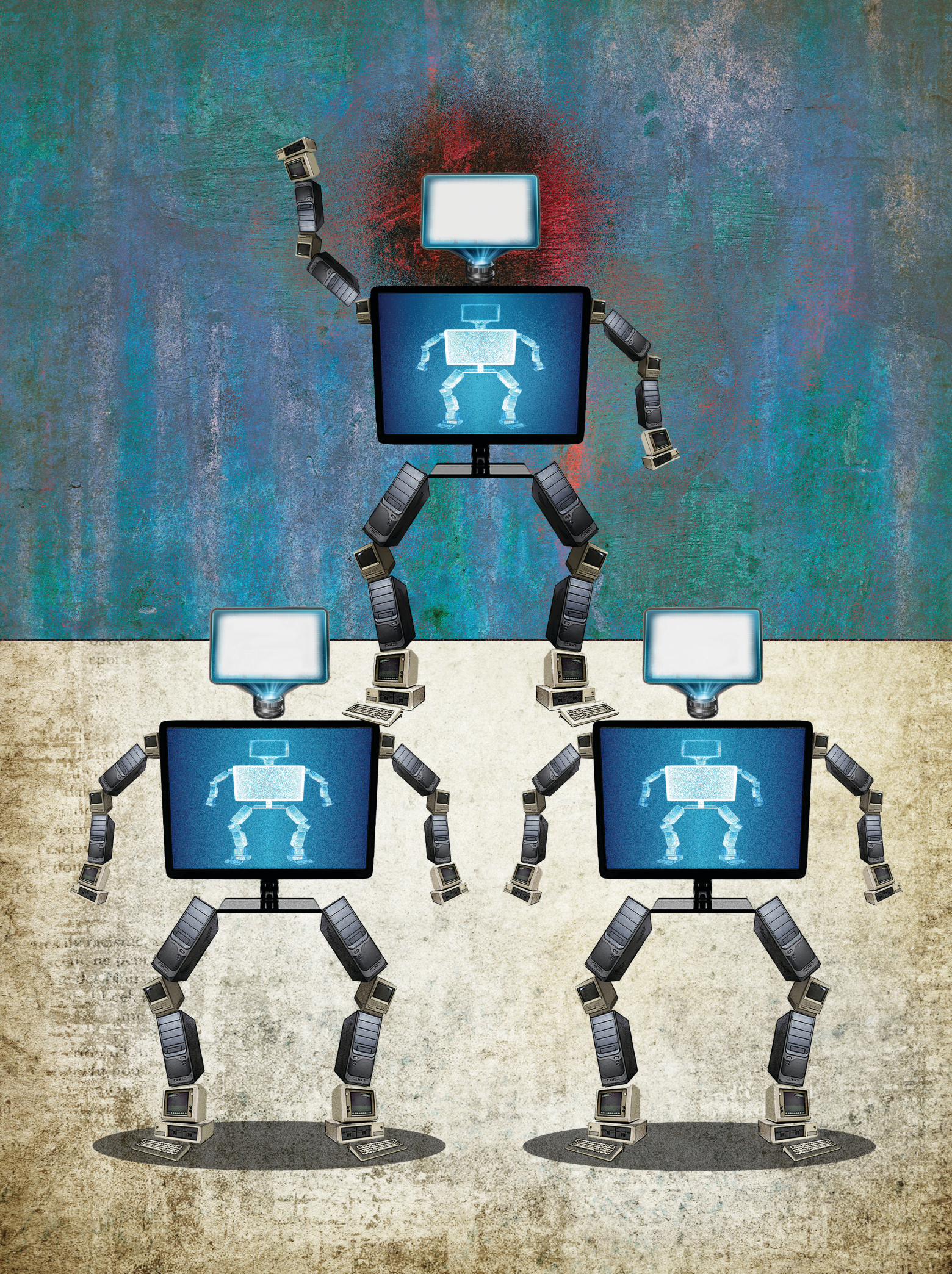
We instead made a Freedom of Information Act request to the National Science Foundation for the funded grant proposals that supported the research. In one, the principal investigator wrote, "We will also make our data and software available to the research community when appropriate." In the end, we concluded, without assistance from the authors to interpret the paper and with the university obstructing our quest for the source code of the prototype system, we would not be able to show the analyses put forth could be defeated.

Reproducibility, repeatability, benefaction. There are two main reasons to share research artifacts: repeatability and benefaction.[2,10,16,20] We say research is repeatable if we can re-run

» key insights

- **Published computer systems research is not always accompanied by the code that supports the research, which impedes peers' ability to repeat the experiments.**
- **Sharing research software presents many challenges, so funding agencies should provide support for the engineering resources necessary to enable repeatable research.**
- **To incentivize authors to share their research artifacts, publishers should require pre-publication declarations from authors specifying their commitment to sharing code and data.**

ILLUSTRATION BY ANTHONY FREDA



the researchers' experiment using the same method in the same environment and obtain the same results.[19] Sharing for repeatability is essential to ensure colleagues and reviewers can evaluate our results based on accurate and complete evidence. Sharing for benefaction allows colleagues to build on our results, better advancing scientific progress by avoiding needless replication of work.

Unlike repeatability, reproducibility does not necessarily require access to the original research artifacts. Rather, it is the independent confirmation of a scientific hypothesis,[19] done post-publication, by collecting different properties from different experiments run on different benchmarks, and using these properties to verify the claims made in the paper. Repeatability and reproducibility are cornerstones of the scientific process, necessary for avoiding dissemination of flawed results.

In light of our discouraging experiences with sharing research artifacts, we embarked on a study to examine the extent to which computer systems researchers share their code and data, reporting the results here. We also make recommendations as to how to improve such sharing, for the good of both repeatability and benefaction.

The study. Several hurdles must be cleared to replicate computer systems research. Correct versions of source code, input data, operating systems, compilers, and libraries must be available, and the code itself must build and run to completion. Moreover, if the research requires accurate measurements of resource consumption, the hardware platform must be replicated. Here, we use the most liberal definitions of repeatability: Do the authors make the source code used to create the results in their article available, and will it build? We will call this "weak repeatability."

Our study examined 601 papers from ACM conferences and journals, attempting to locate any source code that backed up published results. We examined the paper itself, performed Web searches, examined popular source-code repositories, and, when all else failed, emailed the authors. We also attempted to build the code but did not go so far as trying to verify the correctness of the published results.

Recommendations. Previous work on repeatability describes the steps that must be taken in order to produce research that is truly repeatable[11,12] or describes tools or websites that support publication of repeatable research.[4,6] Our recommendations are more modest. We recognize that, as a discipline, computer science is a long way away from producing research that is always, and completely, repeatable. But, in the interim, we can require authors to conscientiously inform their peers of their intent with respect to sharing their research artifacts. This information should be provided by the authors when submitting their work for publication; this would allow reviewers to take the expected level of repeatability into consideration in their recommendation to accept or reject. To this end, we make a recommendation for adding sharing contracts to publications—a statement by authors as to the level of repeatability readers can expect.

Table 1. Notation used in Table 2 and the figure.

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

Background

Three previous empirical studies explored computer science researchers' willingness to share code and data. Kovačević[5] rated 15 papers published in the *IEEE Transactions on Image Processing* and found that while all algorithms had proofs, none had code available, and 33% had data available. Vandewalle et al.[18] examined the 134 papers published in *IEEE Transactions on Image Processing* in 2004, finding "... code (9%) and data (33%) are available online only in a minority of the cases ..." Stodden[15] reported while 74% of the registrants at the Neural Information Processing Systems (machine-learning) conference said they were willing to share post-publication code and 67% post-publication data, only " ... 30% of respondents shared some code and 20% shared some data on their own websites." The most common reasons for not sharing code were "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." Stodden[14] has since proposed "The Open Research License," which, if universally adopted, would incentivize researchers to share by ensuring " ... each scientist is attributed for only the work he or she has created."[13]

Public repositories can help authors make their research artifacts available in perpetuity. Unfortunately, the "if you build it they will come" paradigm does not always work; for example, on the RunMyCode[17] and ResearchCompendia Web portals,[a] only 143 and 236 artifacts, respectively, had been registered as of January 2016.

One attractive proposition for researchers to ensure repeatability is to bundle code, data, operating system,

a http://RunMyCode.org and http://researchcompendia.org

and libraries into a virtual machine image.[4,9] However, this comes with its own problems, including how to perform accurate performance measurements; how to ensure the future existence of VM monitors that will run my VM image; and how to safely run an image that contains obsolete operating systems and applications to which security patches may have not been applied.

From 2011 until January 2016, 19 computer science conferences[b] participated in an "artifact evaluation process."[c] Submitting an artifact is voluntary, and the outcome of the evaluation does not influence whether or not a paper is accepted for publication; for example, of the 52 papers accepted by the 2014 Programming Language Design and Implementation (PLDI) conference, 20 authors submitted artifacts for evaluation, with 12 classified as "above threshold."[d] For PLDI 2015, this improved to 27 accepted artifacts out of 58 accepted papers, reflecting an encouraging trend.

Repeatability and reproducibility are cornerstones of the scientific process, necessary for avoiding dissemination of flawed results.

Study Process

Our study employed a team of undergraduate and graduate research assistants in computer science and engineering to locate and build source code corresponding to the papers from the latest incarnations of eight ACM conferences (ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, and VLDB'12) and five journals (TACO'12, TISSEC'12/13, TOCS'12, TODS'12, and TOPLAS'12).[e]

We inspected each paper and removed from further consideration any that reported on non-commodity hardware or whose results were not backed by code. For the remaining papers we searched for links to source code by looking over the paper itself, examining the authors' personal websites, and searching the Web and code repositories (such as GitHub, Google Code, and SourceForge). If still unsuccessful, we sent an email request to the authors, excluding some papers to avoid sending each author more than one request. We sent each request to all authors for whom we could determine an address and reminder email messages to those who did not respond.

In the following cases we marked a paper as "code not available" when we found only partial code or binary releases; when the authors promised they would "send code soon" but we heard nothing further; when we were asked to sign a license or non-disclosure agreement; when the authors requested credit for any follow-up work; or when we received code more than two months after the original email request.

We next made two attempts to build each system. This often required editing `makefiles` and finding and installing specific operating system and compiler versions, and external libraries. We first gave a research assistant a 30-minute time limit, and, if that failed, we gave another assistant "unlimited time" to attempt the build.[f]

Upon completing the build process we conducted an online survey of all authors to help verify the data we had gathered. We resolved cases where we had misclassified a paper, where our Web searches had turned up the wrong code, or where there had been a misunderstanding between us and the authors. We also asked the authors if the version of the code corresponding to the results in their papers was available and (in cases where we had failed to build their code) if they thought the code ought to build. The survey also let the authors comment on our study.

Study Results

We define three measures of weak repeatability—weak repeatability A, B, and C—with notation we outline in Table 1:

$$\frac{OK^{\leq 30}}{BC} \quad \text{(A)}$$

$$\frac{OK^{\leq 30} + OK^{>30}}{BC} \quad \text{(B)}$$

$$\frac{OK^{\leq 30} + OK^{>30} + OK^{Auth}}{BC} \quad \text{(C)}$$

b http://evaluate.inf.usi.ch/artifacts
c http://www.artifact-eval.org
d http://pldi14-aec.cs.brown.edu
e See Collberg et al.[1] for a description of the process through which the study was carried out.
f A group of independent researchers set out to verify our build results through a crowdsourced effort; http://cs.brown.edu/~sk/Memos/Examining-Reproducibility

Weak repeatability A models scenarios where limited time is available to examine a research artifact, and when communicating with the author is not an option (such as when reviewing an artifact submitted alongside a conference paper). Weak repeatability B models situations where ample time is available to resolve issues, but the lead developer is not available for consultation. The latter turns out to be quite common. We saw situations where the student responsible for development had graduated, the main developer had passed away, the authors' email addresses no longer worked, and the authors were too busy to provide assistance. Weak repeatability C measures the extent to which we were able to build the code or the authors believed their code builds with reasonable effort. This model approximates a situation where ample time is available to examine the code and the authors are responsive to requests for assistance.

The results of our study are listed in Table 2 and outlined in the figure here, showing repeatability rates of A=32.3%, B=48.3%, and C=54.0%. Here, C is limited by the response rate to our author survey, 59.5%.

Does public funding affect sharing? The National Science Foundation Grant Proposal Guide[7] says, "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." However, we did not find significant differences in the weak repeatability rates of NSF-funded vs. non-NSF-funded research.[g]

Does industry involvement affect sharing? Not surprisingly, papers with authors only from industry have a low rate of repeatability, and papers with authors only from academic institutions have a higher-than-average rate. The reasons joint papers also have a lower-than-average rate of code sharing is not immediately obvious; for instance, the industrial partner might have imposed intellectual-property restrictions on the collaboration, or the research could be the result of a student's summer internship.

We noticed published code does not always correspond to the version used to produce the results in the corresponding paper.

Does the right version exist? From the responses we received from authors, we noticed published code does not always correspond to the version 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 you ran to get the results in the paper (ignoring inconsequential bug fixes)?" It was 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."

Why is code not shared? The email responses we received were generally pleasant, accommodating, and apologetic if code could not be provided. In the following paragraphs, we explore several representative examples of email responses from authors who turned down our request.

In order for research to be truly repeatable, the correct version of all artifacts must be available, which is not always the case; for example, one respondent said, "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."

Authors often told us once their code was cleaned up we could have access to their system, in one case saying, "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." Eventually making (reworked) code available may be helpful for benefaction, but for repeatability, such delayed releases are ineffectual; it will never be possible for a reviewer or reader to verify the results presented in the paper.

Several authors acknowledged they never had the intention to make the code available, in one case saying, "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."

In some cases, the one person who understood the system had left, with one respondent saying, "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

g We tracked only whether papers mentioned NSF support; it is possible some might have had other sources of public funding.

time to integrate them in a ready-to-share implementation before he left."

Lack of proper backup procedures was also a problem, with one respondent saying, "Unfortunately, the server in which my implementation was stored had a disk crash in April and three disks crashed simultaneously ... my entire implementation for this paper was not found ... Sorry for that."

Researchers employed by commercial entities were often not able to release their code, with one respondent saying, "The code owned by [company], and AFAIK the code is not open-source." This author added this helpful suggestion: "Your best bet is to reimplement :(Sorry."

Even academic researchers had licensing issues, with one respondent saying, "Unfortunately, the [system] sources are not meant to be opensource [sic] (the code is partially property of [three universities])." Some universities put restrictions on the release of the code, with one respondent saying, " ... 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."

Some systems were built on top of other systems that were not publicly available, with one respondent saying, "We implemented and tested our ... technique on top of a commercialized static analysis tool. So, the current implementation is not open to public. Sorry for this." And, some systems were built

Summary of the study's results. Blue numbers represent papers we excluded from the study, green numbers papers we determined to be weakly repeatable, red numbers papers we determined to be non-repeatable, and orange numbers represent papers for which we could not conclusively determine repeatability (due to our restriction of sending at most one email request per author).

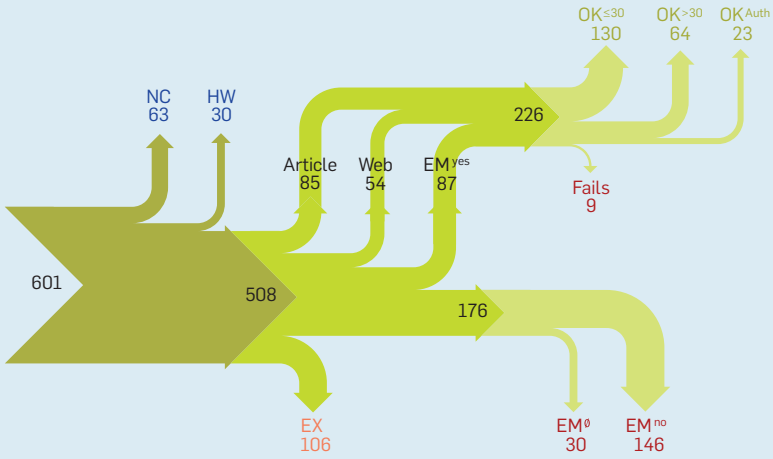


Table 2. Detailed results of the study.

Group	#	Classification				Code Location					Build Result				Weak Repeatability (%)		
		HW	NC	EX	BC	Article	Web	EM yes	EM no	EM ø	OK ≤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'12	60	18	3	2	37	7	2	6	17	5	8	2	2	3	21.6	27.0	32.4
TISSEC'12/13	13	1	6	0	6	3	0	1	2	0	2	0	2	0	33.3	33.3	66.7
TOCS'12	13	1	0	0	12	3	2	0	5	2	2	3	0	0	16.7	41.7	41.7
TODS'12	29	0	12	2	15	1	3	3	3	5	6	1	0	0	40.0	46.7	46.7
TOPLAS'12	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
Joint	148	8	8	36	96	13	11	16	48	8	24	11	1	4	25.0	36.5	37.5
Industrial	44	2	8	6	28	6	0	2	17	3	4	2	2	0	14.3	21.4	28.6
Conferences	470	10	37	100	323	64	46	76	119	18	108	54	19	5	33.4	50.2	56.0
Journals	131	20	26	6	79	21	8	11	27	12	22	10	4	4	27.8	40.5	45.6

on top of obsolete systems, with one respondent saying, "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 ... few people would manage to get it to work on new hardware."

Some authors were worried about how their code might be used, with one respondent saying, "We would like to be notified in case the provided implementation will be utilized to perform (and possibly publish) comparisons with other developed techniques ... 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."

Producing artifacts solid enough to be shared is clearly labor intensive, with one researcher explaining how he had to make a draconian choice, saying, "[Our system] continues to become more complex as more Ph.D. students add more pieces to it ... 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."

Unlike researchers in other fields, computer security researchers must contend with the possible negative consequences of making their code public, with one respondent saying, "... 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."

Some authors used unusual languages and tools that make it difficult for others to benefit from their code, with one respondent saying, "The code ... is complete, but hardly usable by anyone other than the authors ... due to our decision to use [obscure language variant] for the input language."

Recommendations

To improve the state of repeatability in computer science research we could simply require, along with every paper submitted for publication, the authors attach the corresponding code, perhaps in the form of a virtual machine image. Unfortunately, based on our study, it is unrealistic to expect computer science researchers to always make their code available to others. There are several reasons for this: the code may not be clean enough for public distribution; they may be considering commercialization; (part of) the code may have licensing restrictions; they may be too busy to answer questions about their system; or they may worry about not receiving proper attribution for any follow-up work.

We thus make a much more modest proposal that would require only minor changes to how public funding agencies and academic publishers operate:

Fund repeatability engineering. Funding agencies should encourage researchers to request additional funds for "repeatability engineering," including hiring programming staff to document and maintain code, do release management, and assist other research groups wanting to repeat published experiments. In the same way funding agencies conduct financial audits to ensure costs claimed by grantees are allowed, they should also conduct random audits to ensure research artifacts are shared in accordance with what was promised in the grant application; and

Require sharing contract. Publishers of conference proceedings and journals should require every article include a sharing contract specifying the level of repeatability to which its authors will commit.

While the first point will have the effect of shifting some funding from pure research to engineering and oversight, both are important because they ensure research results continue to benefit the academic community—and the public funding it—past the project end date. Here, we expand on the second point.

Sharing contracts. The sharing contract should be provided by the authors when a paper is submitted for publication (allowing reviewers to consider the expected level of repeatability of the work), as well as in the published version (allowing readers to locate research artifacts). The contract commits the author to making available certain resources that were used in the research leading up to the paper and committing the reader/reviewer to take these resources into account when evaluating the contributions made by the paper.

Table 3 lists the data that should be part of a sharing contract, including 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.

Resources can include code, data, and media. For each resource, the contract must state if it is accessible and at what cost, a deadline after which it might no longer be available, and whether it is available in source or binary form or accessible as a service. Code accessed as a service could be running as a Web service or be executed by the authors themselves on input data provided by the reader. We include an optional comment field to handle unusual situations.

Sharing is different from licensing. A sharing contract represents a com-

Table 3. Data needed for sharing contracts.

Location	►	Email address and/or website
Resource	►	Types (code, data, media, documentation ...)
	►	Availability (no access, access, NDA access ...)
	►	Expense (free, non-free, free for academics ...)
	►	Distribution form (source, binary, service ...)
	►	Expiration date
	►	License
	►	Comment
Support	►	Kinds (resolve installation issues, fix bugs, upgrade to new language and operating system versions, port to new environments, improve performance, add features ...)
	►	Expense (free, non-free, free for academics ...)
	►	Expiration date

mitment on behalf of the author 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, redistribution, and reverse engineering). Since copyright bars reuse without permission of the author(s), both licensing and sharing specifications are necessary; for example, if a license prohibits reverse engineering, the community's ability to verify the actions performed by the software are consistent with what is described in the publication is diminished. Likewise, benefaction is hampered by code that makes use of libraries whose license prohibits redistribution.

The contract must also specify the level of technical support the authors commit to provide, for how long they will provide it, and whether that support is free; Table 3 includes a non-exhaustive list of possible types of support.

In some situations authors will want to make their artifacts available under more than one sharing contract, where each contract is targeted at a different audience (such as academic and commercial).

Example of a sharing contract. Publishers must design a concrete syntax for sharing contracts that handles most common situations, balancing expressiveness and conciseness. For illustrative purposes, here is an example contract for the research we have presented, giving free access to source code and data in perpetuity and rudimentary support for free until at least the end of 2016:

Sharing
http://repeatability.cs.arizona.edu;
mailto:collberg@gmail.com;

code: access, free, source;

data: access, free, source, "sanitized";

support: installation, bug fixes, free, 2016-12-31;

In this research, we must sanitize email exchanges before sharing them. We express this in the comment field.

Discussion

While there is certainly much room for novel tools for scientific provenance, licensing frameworks that reassure researchers they will be properly attributed, and repositories that can store research artifacts in perpetuity, we must acknowledge the root of the scientific-repeatability problem is sociological, not technological; when we do not produce solid prêt-à-partager artifacts or attempt to replicate the work of our peers it is because there is little professional glory to be gained from doing so. Nosek[8] wrote, "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."

The real solution to the problem of researchers not sharing their research artifacts lies in finding new reward structures that encourage them to produce solid artifacts, share these artifacts, and validate the conclusions drawn from the artifacts published by their peers.[3] Unfortunately, in this regard we remain pessimistic, not seeing a near future where such fundamental changes are enacted.

In the near term, we thus propose two easily implementable strategies for improving this state of affairs: a shift in public funding to repeatability engineering and adopting sharing specifications. With required sharing contracts, authors—knowing reviewers are likely to take a dim view of a paper that says, up front, its results are not repeatable—will thus be incentivized to produce solid computational artifacts. Adjusting funding-agency regulations to encourage engineering for repeatability will provide them with the resources to do so.

Acknowledgments

We would like to thank Saumya Debray, Shriram Krishnamurthi, Alex Warren, and the anonymous reviewers for valuable input. C

References

1. Collberg, C., Proebsting, T., and Warren, A.M. *Repeatability and Benefaction in Computer Systems Research: A Study and a Modest Proposal.* Technical Report TR 14-04. Department of Computer Science, University of Arizona, Tucson, AZ, Dec. 2014; http://repeatability.cs.arizona.edu/v2/RepeatabilityTR.pdf
2. Feitelson, D.G. From repeatability to reproducibility and corroboration. *SIGOPS Operating Systems Review 49*, 1 (Jan. 2015), 3–11.
3. Friedman, B. and Schneider, F.B. *Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion.* Computing Research Association Best Practices Memo, Feb. 2015; http://archive2.cra.org/uploads/documents/resources/bpmemos/BP_Memo.pdf
4. Gorp, P.V. and Mazanek, S. Share: A Web portal for creating and sharing executable research papers. *Procedia Computer Science 4* (2011), 589–597.
5. Kovačević, J. How to encourage and publish reproducible research. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume IV* (Honolulu, HI, Apr. 15–20). IEEE Computer Society, 2007, 1273–1276.
6. Li-Thiao-Té, S. Literate program execution for reproducible research and executable papers. *Procedia Computer Science 9* (2012), 439–448.
7. National Science Foundation. *Grant Policy Manual 05-131.* Arlington, VA, July 2005; http://www.nsf.gov/pubs/manuals/gpm05_131
8. Nosek, B.A. An open, large-scale, collaborative effort to estimate the reproducibility of psychological science. *Perspectives on Psychological Science 7*, 6 (Nov. 2012), 657–660.
9. Perianayagam, S., Andrews, G.R., and Hartman, J.H. Rex: A toolset for reproducing software experiments. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine* (Hong Kong, Dec. 18–21). IEEE Computer Society, 2010, 613–617.
10. Rozier, K.Y. and Rozier, E.W.D. Reproducibility, correctness, and buildability: The three principles for ethical public dissemination of computer science and engineering research. In *Proceedings of the IEEE International Symposium on Ethics in Science, Technology, and Engineering* (Chicago, IL, May 23–24). IEEE Computer Society, 2014, 1–13.
11. Sandve, G.K., Nekrutenko, A., Taylor, J., and Hovig, E. Ten simple rules for reproducible computational research. *PLoS Computational Biology 9*, 10 (Oct. 24, 2013).
12. Schwab, M., Karrenbach, N., and Claerbout, J. Making scientific computations reproducible. *Computing in Science Engineering 2*, 6 (Nov. 2000), 61–67.
13. Stodden, V. Enabling reproducible research: Licensing for scientific innovation. *International Journal of Communications Law & Policy 13* (Winter 2009), 22–46.
14. Stodden, V. The legal framework for reproducible scientific research: Licensing and copyright. *IEEE Computing in Science and Engineering 11*, 1 (Jan.-Feb. 2009), 35–40.
15. Stodden, V. *The Scientific Method in Practice: Reproducibility in the Computational Sciences.* Technical Report Working Paper 4773-10. MIT Sloan School of Management, Cambridge, MA, Feb. 2010; http://web.stanford.edu/~vcs/papers/SMPRCS2010.pdf
16. Stodden, V., Borwein, J., and Bailey, D.H. 'Setting the default to reproducible' in computational science research. *SIAM News 46*, 5 (June 2013).
17. Stodden, V., Hurlin, C., and Perignon, C. RunMyCode.org: A novel dissemination and collaboration platform for executing published computational results. In *Proceedings of the Eighth IEEE International Conference on E-Science* (Chicago, IL, Sept. 15). IEEE Computer Society, 2012, 1–8.
18. Vandewalle, P., Kovačević, J., and Vetterli, M. Reproducible research in signal processing: What, why, and how. *IEEE Signal Processing Magazine 26*, 3 (May 2009), 37–47.
19. Vitek, J. and Kalibera, T. Repeatability, reproducibility, and rigor in systems research. In *Proceedings of the 11th ACM International Conference on Embedded Software* (Taipei, Taiwan, Oct. 9–14). ACM Press, New York, 2011, 33–38.
20. Yale Law School Roundtable on Data and Code Sharing. Reproducible research: Addressing the need for data and code sharing in computational science. *Computing in Science and Engineering 12*, 5 (Sept./Oct. 2010), 8–13.

Christian Collberg (collberg@gmail.com) is a professor of computer science in the Department of Computer Science at the University of Arizona, Tucson, AZ.

Todd A. Proebsting (proebsting@email.arizona.edu) is a professor of computer science in the Department of Computer Science at the University of Arizona, Tucson, AZ.

© 2016 ACM 0001-0782/16/03 $15.00