Reproducibility as a Technical Specification

Tom Crick
Department of Computing & Information Systems
Cardiff Metropolitan University, UK
tcrick@cardiffmet.ac.uk

Benjamin A. Hall
MRC Cancer Unit
University of Cambridge, UK
bh418@mrc-cu.cam.ac.uk

Samin Ishtiaq
Microsoft Research
Cambridge, UK
samin.ishtiaq@microsoft.com

***Abstract*—Reproducibility of computationally-derived scientific discoveries should be a certainty. As the product of several person-years' worth of effort, results – whether disseminated through academic journals, conferences or exploited through commercial ventures – should at some level be expected to be repeatable by other researchers. While this stance may appear to be obvious and trivial, a variety of factors often stand in the way of making it commonplace. Whilst there has been detailed cross-disciplinary discussions of the various social, cultural and ideological drivers and (potential) solutions, one factor which has had less focus is the concept of reproducibility as a *technical* challenge. Specifically, that the definition of an *unambiguous* and *measurable* standard of reproducibility would offer a significant benefit to the wider computational science community.**

In this paper, we propose a technical specification for a service for reproducibility, presenting cyberinfrastructure and associated workflow for a service which would enable such a specification to be verified and validated. In addition to addressing a pressing need for the scientific community, we further speculate on the potential contribution to the wider software development community of services which automate *de novo* compilation and testing of code from source. We illustrate our proposed specification and workflow by using the *BioModelAnalyzer* tool as a running example.

Keywords*—*Reproducibility, Artifact evaluation, Cyberinfrastructure, Research software, Benchmarks, Verification, Validation

I. Introduction

This paper aims to start a discussion in the wider computational science community about the reproducibility of its algorithms, models, tools and benchmarks. There is a significant opportunity for this (broad and diverse) community — of whom we are proud members — to identify and address the technical and socio-cultural issues surrounding reproducibility in both their specific research domain as well as more broadly for computational science; a desirable outcome would be a clear specification to encourage, enable and ultimately enforce reproducibility. Enumerating a standard for reproducibility would have a clear benefit for researchers as well as the wider community as a whole.

Over the past decade, we have seen a step-change in how science and engineering is done. Experiments, simulations, models, benchmarks, even proofs cannot be done without leveraging software and computation. A 2012 report by the Royal Society summarised that computational techniques have "*moved on from assisting scientists in doing science, to transforming both how science is done and what science is done*" [1]. Thus, the reproduction and replication of reported scientific results is a widely discussed topic within the scientific community [2]–[6], even spawning legislation in the USA [7]. Whilst the increasing number of high-profile retractions of scientific studies, from climate science to bioscience, has drawn the focus of many commentators, automated systems, which allow easy reproduction of results, offer the potential to improve the efficiency of scientific exploration and drive the adoption of new techniques. Nevertheless, this is a wider socio-technical problem that pervades the scientific community, with estimates that as much as 50% of published studies, even those in top-tier academic journals, cannot be repeated with the same conclusions by an industrial lab [8]. Furthermore, just publishing (linked and open) scientific data is not enough to ensure the required reusability [9]. There are numerous non-technical impediments to making software maintainable and re-useable. The pressure to "make the discovery" and publish quickly disincentivises careful software curation, with only recent imperatives from funding bodies and governments trying to change this position. Releasing code prematurely was often seen to give your competitors an advantage, but we should be shining light into these "black boxes" [5]. In essence: better software, better research [10].

We can thus exploit a fundamental advantage of the wider impact of computer science and computational techniques to research: the unique ability to share the raw outputs of their work as software and datafiles. New experiments, simulations, models, benchmarks, even proofs cannot be done without software. And this software does not consist of simple hack-together, use-once, throw-away scripts; scientific software repositories contain thousands, perhaps millions, of lines of code and they increasingly need to be actively supported and maintained. More importantly, with reproducibility being a fundamental tenet of science, they should be re-useable. However, if we closely analyse the scientific literature related to software tools it often does not appear to be adhering to these rules [11]. How many of them are reproducible? How many explain their experimental methodologies, in particular the basis for their benchmarking? In particular, can we (re)build the code [12][1]? We, the authors, are perhaps as guilty as anyone in the past: we have published papers [13], [14] with in-depth benchmarks and promises of code to be released online in the near future.

However, we recognise that in many cases we are "preaching to the converted" and do not need to sell the idea of reproducibility to the computational science research community too much: we are used to writing papers about our algorithms, models and tools; and reproducing others' work (or having

[1] In turn, stimulating further discussions in this space: http://cs.brown.edu/~sk/Memos/Examining-Reproducibility/

our own work reproduced) is encapsulated in the "Benchmark Tables" and presentation of empirical results that authors and referees in this community both think are essential to any paper. The idea of reproducibility is gaining momentum across the wider scientific research community, for example in computer science [15], [16], engineering [17], life sciences [18], biomedical sciences [19], climate science [20], ecology [21], epidemiology [22], psychology [23], econometrics [24] and the social sciences [25], as well as the wide utility of creating and sharing reusable scientific workflows and web services [26]–[28]. While there has been a revolution in the sharing and dissemination of published papers (*open access*) and the subsequent discussions relating to the sharing of protocols and materials (*open science*) [1], the ability of a researcher to *(a)* keep track of the increasing number of research outputs in their domain [29] and *(b)* take these published results and data and reimplement the described workflow remains difficult [30]–[33].

There has been promising work in this area, particularly around benchmarking and cyberinfrastructure [34]–[37], along with a number of manifestos and community initiatives to encourage and support reproducible research, such as the Recomputation Manifesto [38][2] and cTuning [39], as well as curated recommendations on where to publish research software[3]. We have previously encapsulated some of the technical barriers to reproducing work across computing and the computational sciences, particular in terms of the sharing of algorithms, models and benchmark sets [40]–[42], as well as drawing attention to some of the wider socio-cultural issues [43].

For example, although reproducibility is not a new concept to the computer science research community [44], more recently a number of high-profile computer science conferences, including PLDI, POPL, SIGMOD, CGO, SPLASH and ECAI, have explicitly acknowledged the importance of reproducibility[4] (and repeatability, recomputability and the multitude of 'Rs' that underpin e-research), as well as promoting community-driven reviewing and validation [45]. For many, this takes the form of the author providing *artefacts* — an accessible tool for reproducing results — for the reviewers to evaluate. Journals such as Science, Nature, PLoS and from Royal Society Publishing explicitly require that source code and data is made available online under some form of open source license. While these initiative are great, they are often optional, seem piecemeal, and do little to easily enable verification or validation of scientific results at a later stage. Even within the same field, there are different ideas of what defines reproducibility.

In contrast to more traditional research outputs, the development of algorithms has unique advantages. Algorithms are tested through their implementation in code, and as such they can be accurately communicated either by sharing the implementation and a pseudo-code description of the algorithm behaviour. Furthermore, because of their discrete nature it is expected that they aways return the same output for the same input. It follows that these features are more easily tested than the outputs of other scientific disciplines. To demonstrate the reproducibility of a newly developed algorithm, one must *(a)* be able to create an operational artefact from the code, and *(b)* show that published models (i.e. benchmarks) show the same behaviours as those reported. This specification for reproducibility can in principle be tested for every publication in the field.

Therefore, this paper invites computational researchers to embrace a new methodology (and culture) for disseminating research. We propose an initial specification for what a reproducibility service for key conferences and journals in a field should look like. We present the requirements of the prototype, and a suggested plan for introducing the service specifically to a high-profile conference. In discussing the service, we highlight key implementation issues relating to security and general applicability which will need mitigating or resolving before widespread acceptance by the research community. The benefits of a reproducibility testing service are clear. In a sense, reproducibility here is an coming together of the standard practice of *testing* and ongoing work done in many continuous integration systems (notably, automated build services). Three features are prominent in our aims for reproducibility here: compilation and testing in a new machine, a continuous integration strategy for code commits and the ability to add and remove benchmarks to the test set.

We use a running example based upon *BioModelAnalyzer*[5], a tool for the development and analysis (simulation, model checking) of a specific class of formal models for biology, which has been described by the authors over a series of VMCAI and CAV papers [46]–[48]. The tool specifically allows users to test for model *stability*; that is, a bespoke algorithm proves that for all initial states a model always ends in a single, unique fixpoint. We have chosen this example due to our familiarity with the tool, and to highlight historical examples where a reproducibility service would have supported both toolchain development and algorithm discovery.

II. A Specification for Reproducible Computational Science

A service for reproducibility is intended to play three important roles. It should:

(i) Demonstrate that the source of an algorithm or tool can be compiled, run and behave as described, without manual intervention from the author;
(ii) Allow new benchmarks to be added, by users other than the developer, to widen the testing and identify potential bugs;
(iii) Store and link specific artefacts with their linked publications or other publicly-accessible datasets.

A. There Are Two Types of People

To address these needs, we propose that the service should follow the workflow presented in Figure 1. Two main classes of user are defined; developers, who generate code; and modellers, who generate new benchmarks. An individual might in practice play either or both of these roles; here the roles serve to define ways in which people interact with the system. A developer writes new code, which is periodically pushed to

[2] http://www.recomputation.org/

[3] http://www.software.ac.uk/resources/guides/which-journals-should-i-publish-my-software

[4] See http://www.artifact-eval.org/ and http://ctuning.org/reproducibility

[5] Available to use online: http://biomodelanalyzer.research.microsoft.com/

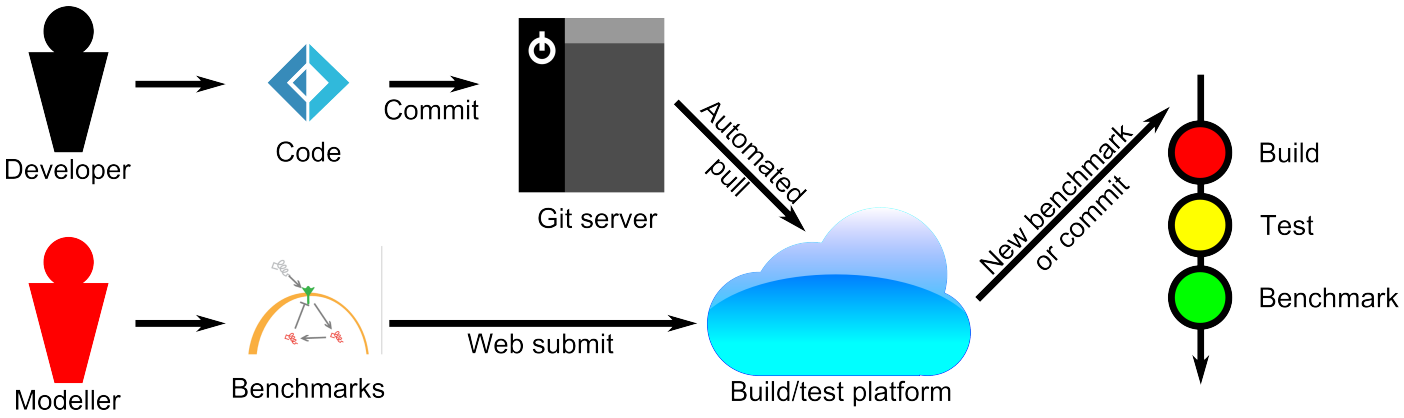

Figure 1. Proposed reproducibility service workflow

a public repository such as GitHub. Through integration with the repository, the server responds to new code by undergoing a process of pulling the code from the repository, downloading required dependencies, and compiling the code. If this stage fails, the developer is informed and the workflow ends. If the code is successfully compiled, two stages of testing are performed.

The first stage (labelled *Test* in Figure 1), involves running a series of basic tests defined by the developer. This is intended as a sanity check to ensure that basic features of the code have not been broken by the updated code, and failure to pass these tests is reported and ends the workflow. If this completes successfully, the second stage (labelled *Benchmark* in Figure 1) runs and a series of models are tested for a known property, and the results recorded. These results can then be stored in a database, with a note of the commit ID, and available through a web interface for future audit and analysis.

The service must fit easily into the developer workflow; as noted in Section III we expect that there will be some costs to the users in terms of the time required to ensure that the code compiles and runs on the service. To minimise this, the service needs to connect to standard code repositories, automatically detecting and responding to new versions of the code and updates to dependencies, running tests for every new code commit.

B. de Novo *Build Environments*

To address the socio-cultural issues discussed earlier, a service such as this must require minimal developer intervention. This serves multiple purposes – through automation for example, the service can be enabled to compile new code and test new benchmarks trivially. This also forces the developer to make publicly available their local workarounds (i.e. hacks and workflow "glue"). As such, this requires the developer to make the project dependencies clearly available, and enables future changes in the dependencies (such as a library update) to be tested automatically too.

Throughout the lifetime of the *BioModelAnalyzer* tool, development has been shared between a number of developers, each working on different aspects of the tool. Work in algorithm development focuses on adding new features to a command line tool with few dependencies, aiding rapid development. In contrast, the graphical model construction and testing environment has typically been done by a single or pair of individuals. This necessarily required a number of dependencies, reflecting the use of Azure (Microsoft's cloud computing platform) and Silverlight (a framework for rich Internet applications).

In an early stage of development it was found that only a single machine was capable of deploying the web service. This arose as the developer responsible for writing and deploying the user interface had run a series of commands necessary to run the mixture of 32- and 64- bit components on Azure. These commands needed only be run once, and went undocumented, thus needing to be rediscovered later when other team members attempted to deploy. These problems would be identified trivially through the proposed service; such undocumented commands would lead to all tests failing until explicitly added to the build process.

C. *Tool refinement*

In contrast to the developer role, a modeller supplies benchmarks for a piece of code to test against. These do not require that the latest version of the code is recompiled, but on submission the models are tested and added to the local repository of models for analysis.

In the case of *BioModelAnalyzer*, throughout the development of the tool, many refinements have been made to different implementations. Some of these were subtle, and were identified by unit tests; for example rounding mechanisms were switched between floors and rounds following a scientific discussion. More complex changes however broke behaviours which were not tested in our available benchmark set. One example was in the treatments of nodes without inputs; "biological intuition" suggested that such nodes should have an alternative default function from other nodes. Here, the ability of users to submit new benchmarks would aid identification of these breaking changes, by extending the test sets and simplifying the process of adding to the test sets, and forcing the question of what changes are appropriate (and how to update old models to keep correct behaviour).

D. *Identification of Algorithmic Weaknesses*

After a model is submitted, it is tested on every new piece of code pushed to the server and the changes in the behaviour

can be noted and linked to specific code commits. Whilst the developer's role has a transparent value (in providing an implementation of an algorithm), the value of the modeller may be less immediately clear. The modeller submits a broad range of tests which may highlight material flaws (i.e. bugs) in the implementation, or the algorithm. More than this however, the modeller may generate models which identify weaknesses of either an algorithm or a specific implementation.

One example from the authors experience is the series of models with "timed-switches" described in [48]. There, we presented a new algorithm for proving stability in a new class of models. Whilst the paper focused on discussing the algorithm, identifying the new class of models was complex. Models with long cycles and the new class of models ("non-trivially stable models") both can take substantial time to search for cycles, and these models could only be proved stable using a combination of simulation (to identify the fixpoint) and LTL queries (to prove that there existed no paths beyond a certain length which did not include the fixpoint) [49].

E. Algorithm-Model Axes

The proposed service would allow both algorithm and model type of tests to be included explicitly, and models to be routinely added to each algorithm. Models which time-out with one but are successfully proved can be logged and identified for future study. The features which define them could then be more easily researched, and new algorithms developed to address the specific features of the model. It could further be used to demonstrate the speed improvement arising from new algorithms.

F. `make depend`

Dependencies for a given implementation need explicit testing. Due to the highly variable and sometimes complex nature of dependencies, we see this as an optional part of the workflow, as developers may chose to supply certain dependencies as binary files in the code compilation process. For completeness however, we note that such a system could also respond to updates in external dependencies by triggering compilation and testing in the same manner as defined for a new code commit. This would aid developers in identifying code breaking changes introduced by third parties.

G. "I'm First!"

Another issue is around performance comparisons of benchmarks: how can we estimate, compare and evaluate raw performance in the cloud? Testing new algorithms on benchmarks is in the first instance about pass/fail, but very soon the focus is on raw performance. Benchmark tables are about out-performing other algorithms, other tools. But, if the whole verification workflow is running on the cloud, then acquiring and evaluating raw performance numbers is not immediately feasible. There is no cloud equivalent of `top` or `time` that gives user–resource statistics. There is too much infrastructure interference/dependence — with VMs spinning up, being torn down, migrating, the bus being used by other VMs, etc — to obtain faithful numbers for the user/process/VM itself. Projects such as Recomputation.org[6] have been focusing on using virtual machines in the cloud to freeze, and later unfreeze, computational experiments; while this approach is not the complete solution, it is certainly a move in the right direction [50]. Nevertheless, the project's primary aim is to validate recomputation [38], with performance a secondary consideration [51], thus making this a key avenue for further investigation.

[6] http://recomputation.org/

H. Running Arbitrary Code

There are clearly significant potential security concerns around providing open cyberinfrastructure that pulls, compiles and runs arbitrary code as part of an autonomous continuous integration framework; we need to consider precisely how this infrastructure would interact with other open services, as well as privileges it would require to run as an autonomous cloud service. Nevertheless, there are existing models of sandboxing and privilege restriction from elastic cloud computation providers that could be further developed and applied.

III. A Reproducibility Workflow for a Computational Research Community

Following the proposal of such a system, the question becomes: *how do we encourage widespread uptake, or even standardisation?* Such a service may appear non-trivial, given the large numbers of tools and workflows that could potentially require to be supported by the service. Furthermore, after such a service has been implemented, how do we ensure it is *useful* and *usable* for researchers. To address this, we propose the following workflow that could be adapted and used for conferences in computational science community:

1) **Pre-conference:** clear signposting for authors; it should be advertised and promoted in the call for papers to highlight this is a step-change in how we address reproducibility. Call for artefact reviewers with a range of specialisms, with a named chair of the review team.
2) **Explicit criteria for authors:** *make this as easy as possible for us to evaluate/execute your artefact!*. We would aim to articulate the review criteria, but the primary aim is: *can I evaluate/execute this artefact and get the same results that are presented in the paper?*
3) **Submission:** when papers are submitted, they have to nominate whether they want their paper to go through artefact review (at the start, this may not be compulsory, but this will change over a period of time – effecting cultural change and this would then become a necessary condition and a formal stage in the reviewing workflow), along with required tools, libraries and (ideally) computational requirements.
4) **Reviewing:** in the first instance, it may be seen as an extra (voluntary) step to the normal reviewing process: e.g. *This submission is voluntary and will not influence the final decision regarding the papers.*. Independent of the scientific merit of the paper, the results will be verified. To encourage this, there may be a prize, as well as ranked ordering and profiled listed in conference proceedings/final publication.
5) **Artefact evaluation:** artefact evaluation process runs concurrent to the standard paper review process.

6) **Reporting:** traffic lights system (potentially with ranked list) to indicate the level of reproducibility of the submitted artefact.
7) **Community curation:** over a number of conference cycles, we would have a community curated repository/database of previous artefacts, which would provide exemplars, comparisons and emerging best practice.

The key question for different research communities then becomes: *how to initiate and initialise this change?* Such a requirement creates a set of new costs to researchers, both in terms of time spent ensuring that their tools work on the centralised system (in addition to their local implementation), but also potentially in terms of equipment (in terms of running the system). Such costs may be easier to bear for some researchers compared to others, especially those with large research groups who can more easily distribute the tasks, and it is important that the service does not present a barrier to early career researchers and those with efficient budgets (this type of cost analysis is not unique to reproducibility efforts – it has been estimated that a shift to becoming exclusively open access for a journal may lead to a ten-fold increase in computer science publication costs [52]). Another advantage of such a service however is that it sets a clear minimum standard for reproducibility in computational research. It does not impose a specific set of licences (a limited licence for testing would be all that is necessary). By its nature, it can be uniformly applied to all users in a conference, preventing the "weaponisation of reproducibility" that has been highlighted as a potential weakness[7].

IV. Conclusions

The benefits to the community from a cultural change to foster and favour reproducibility are clear and as such we should aim through the development and adoption of open cyberinfrastructure, software toolchains and workflow to mitigate these costs. Furthermore, we can reasonably expect the needs of the community to evolve over time, and initial implementations of the platform may require refinement in response to user feedback. As such, if the community is to move to requiring reproducibility, it seems most reasonable that this is staggered over a number of years to allow for both of these elements to develop, until eventually all researchers are required to use the service. This plan balances competing needs within the community, and would reduce the disruption for uptake by gradually introducing it to researchers.

- **Year *t*:** Offer the service as an optional extra in the testing phase, allowing users to demonstrate the reliability of their code which could be taken into account in the review process.
- **Year *t+1*:** All authors must use the reproducibility service, but results are not used in the review process. The results of the test are used to refine the service and pick out any unaddressed issues
- **Year *t+2*:** All authors are required to use the service, and the results are explicitly used to assess reproducibility in the review process.

[7] http://simplystatistics.org/2015/03/13/de-weaponizing-reproducibility/

This open discussion, understanding and acceptance of what reproducibility means for the wider computational research community is clearly important. It is imperative that we see it as worthwhile and address it, but we now have to move beyond manifestos, pledges and top tips: as researchers, we all need to publicly acknowledge that this is worthwhile and put concrete measures in place to address it, or stop going on about it.

References

[1] Royal Society, "Science as an open enterprise," 2012, available from: https://royalsociety.org/policy/projects/science-public-enterprise/report/.

[2] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000.

[3] N. Barnes, "Publish your computer code: it is good enough," *Nature*, vol. 467, no. 753, 2010.

[4] J. P. Mesirov, "Accessible Reproducible Research," *Science*, vol. 327, no. 5964, pp. 415–416, 2010.

[5] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz, "Shining Light into Black Boxes," *Science*, vol. 336, no. 6078, pp. 159–160, 2012.

[6] L. N. Joppa, G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'Hara, D. Gavaghan, and S. Emmott, "Troubling Trends in Scientific Software Use," *Science*, vol. 340, no. 6134, pp. 814–815, 2013.

[7] US House of Representatives, "Secret Science Reform Act," https://www.congress.gov/bill/113th-congress/house-bill/4012, November 2014, H.R.4012, 113th Congress.

[8] L. Osherovich, "Hedging against academic risk," *Science-Business eXchange*, vol. 4, no. 15, 2011.

[9] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, J. Ainsworth, J. Bhagata, P. Couch, D. Cruickshank, M. Delderfield, I. Dunlop, M. Gamble, D. Michaelides, S. Owen, D. Newman, S. Sufi, and C. Goble, "Why linked data is not enough for scientists," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 599–611, 2013.

[10] C. Goble, "Better Software, Better Research," *IEEE Internet Computing*, vol. 18, no. 5, pp. 4–8, 2014.

[11] Editorial, "Devil in the details," *Nature*, vol. 470, no. 7334, pp. 305–306, 2011.

[12] C. Collberg, T. Proebsting, G. Moraila, A. Shankaran, Z. Shi, and A. M. Warren, "Measuring Reproducibility in Computer Systems Research," Department of Computer Science, University of Arizona, Tech. Rep., 2014.

[13] T. Crick, M. De Vos, M. Brain, and J. Fitch, "Generating Optimal Code using Answer Set Programming," in *Proceedings of 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, ser. Lecture Notes in Computer Science, vol. 5753. Springer, 2009, pp. 554–559.

[14] J. Berdine, B. Cook, and S. Ishtiaq, "SLAyer: Memory Safety for Systems-Level Code," in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV 2011)*, ser. Lecture Notes in Computer Science, vol. 6806. Springer, 2011, pp. 178–183.

[15] J. Vitek and T. Kalibera, "Repeatability, Reproducibility, and Rigor in Systems Research," in *Proceedings of the 9th ACM International Conference on Embedded Software (EMSOFT'11)*. ACM Press, 2011, pp. 33–38.

[16] C. Collberg, T. Proebsting, and A. M. Warren, "Repeatability and Benefaction in Computer Systems Research: A Study and a Modest Proposal," Department of Computer Science, University of Arizona, Tech. Rep. TR 14-04, 2014.

[17] J. Ledet, A. Teran-Somohano, Z. Butcher, L. Yilmaz, A. E. Smith, H. Oğuztüzün, O. Dayibaş, and B. Görü, "Toward model-driven engineering principles and practices for model replicability and experiment reproducibility," in *Proceedings of the 2014 Symposium on Theory of Modeling & Simulation (DEVS'14)*. Society for Computer Simulation International, 2014, pp. 1–8.

[18] N. D. Rollins, C. M. Barton, S. Bergin, M. A. Janssen, and A. Lee, "A Computational Model Library for publishing model documentation and code," *Environmental Modelling & Software*, vol. 61, pp. 59–64, 2014.

[19] Y. Huang and R. Gottardo, "Comparability and reproducibility of biomedical data," *Science*, vol. 14, no. 4, pp. 391–401, 2013.

[20] B. D. Santer, T. M. L. Wigley, and K. E. Taylor, "The Reproducibility of Observational Estimates of Surface and Atmospheric Temperature Change," *Science*, vol. 334, no. 6060, pp. 1232–1233, 2011.

[21] J. C. Thiele and V. Grimm, "Replicating and breaking models: good for you and good for ecology," *Oikos*, March 2015, http://onlinelibrary.wiley.com/doi/10.1111/oik.02170/abstract.

[22] R. D. Peng, F. Dominici, and S. L. Zeger, "Reproducible Epidemiologic Research," *American Journal of Epidemiology*, vol. 163, no. 9, pp. 783–789, 2006.

[23] C. D. Chambers, E. Feredoes, S. D. Muthukumaraswamy, and P. J. Etchells, "Instead of "playing the game" it is time to change the rules: Registered Reports at AIMS Neuroscience and beyond," *AIMS Neuroscience*, vol. 1, no. 1, pp. 4–17, 2014.

[24] R. Koenker and A. Zeileis, "On reproducible econometric research," *Journal of Applied Econometrics*, vol. 24, no. 5, pp. 833–847, 2009.

[25] R. Conte, N. Gilbert, G. Bonelli, C. Cioffi-Revilla, G. Deffuant, J. Kertesz, V. Loreto, S. Moat, J.-P. Nadal, A. Sanchez, A. Nowak, A. Flache, M. S. Miguel, and D. Helbing, "Manifesto of Computational Social Science," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 325–346, 2012.

[26] S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. ACM Press, 2008, pp. 1345–1350.

[27] T. Crick, P. Dunning, H. Kim, and J. Padget, "Engineering Design Optimization using Services and Workflows," *Philosophical Transactions of the Royal Society A*, vol. 367, no. 1898, pp. 2741–2751, 2009.

[28] S. Olabarriaga, G. Pierantoni, G. Taffoni, E. Sciacca, M. Jaghoori, V. Korkhov, G. Castelli, C. Vuerli, U. Becciani, E. Carley, and B. Bentley, "Scientific Workflow Management – For Whom?" in *Proceedings of 10th IEEE International Conference on e-Science (e-Science 2014)*. IEEE Press, 2014, pp. 298–305.

[29] P. Della Briotta Parolo, R. Kumar Pan, R. Ghosh, B. A. Huberman, K. Kaski, and S. Fortunato, "Attention decay in science," March 2015, available from: http://arxiv.org/abs/1503.01881.

[30] R. D. Peng, "Reproducible Research in Computational Science," *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011.

[31] 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 Computer Science*, vol. 4, pp. 648–657, 2011.

[32] G. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten Simple Rules for Reproducible Computational Research," *PLoS Computational Biology*, vol. 9, no. 10, 2013.

[33] G. Wilson, D. A. Aruliah, C. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson, "Best Practices for Scientific Computing," *PLoS Biology*, vol. 12, no. 1, 2014.

[34] S. Sim, S. Easterbrook, and R. Holt, "Using benchmarking to advance research: a challenge to software engineering," in *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*. IEEE Press, 2003, pp. 74–83.

[35] F. Chirigati, M. Troyer, D. Shasha, and J. Freire, "A Computational Reproducibility Benchmark," *IEEE Data Engineering Bulletin*, vol. 36, no. 4, pp. 54–59, 2013.

[36] V. Stodden and S. Miguez, "Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research," *Journal of Open Research Software*, vol. 2, no. 1, pp. 1–6, 2014.

[37] V. Stodden, S. Miguez, and J. Seiler, "ResearchCompendia.org: Cyberinfrastructure for Reproducibility and Collaboration in Computational Science," *Computing in Science & Engineering*, vol. 17, no. 12, 2015.

[38] I. P. Gent, "The Recomputation Manifesto," April 2013, available from: http://arxiv.org/abs/1304.3674.

[39] G. Fursin, R. Miceli, A. Lokhmotov, M. Gerndt, M. Baboulin, A. D. Malony, Z. Chamski, D. Novillo, and D. Del Vento, "Collective mind: Towards practical and collaborative auto-tuning," *Scientific Programming*, vol. 22, no. 4, pp. 309–329, 2014.

[40] T. Crick, B. A. Hall, and S. Ishtiaq, ""Can I Implement Your Algorithm?": A Model for Reproducible Research Software," in *Proceedings of 2nd International Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2)*, 2014.

[41] T. Crick, B. A. Hall, S. Ishtiaq, and K. Takeda, ""Share and Enjoy": Publishing Useful and Usable Scientific Models," in *Proceedings of 1st International Workshop on Recomputability*, 2014.

[42] T. Crick, S. Ishtiaq, and B. A. Hall, "Towards "Reproducibility-as-a-Service"," March 2015, available from: http://arxiv.org/abs/1503.02388.

[43] N. P. C. Hong, T. Crick, I. P. Gent, and L. Kotthoff, "Top Tips to Make Your Research Irreproducible," April 2015, available from: http://arxiv.org/abs/1504.00062.

[44] K. Price, "Anything you can do, I can do better (No you can't)..." *Computer Vision, Graphics, and Image Processing*, vol. 36, no. 2-3, pp. 387–391, 2000.

[45] G. Fursin and C. Dubach, "Community-Driven Reviewing and Validation of Publications," in *Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering (TRUST'14)*. ACM Press, 2014, pp. 1–4.

[46] B. Cook, J. Fisher, E. Krepska, and N. Piterman, "Proving stabilization of biological systems," in *Proceedings of the 12th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'11)*, 2011.

[47] D. Benque, S. Bourton, C. Cockerton, B. Cook, J. Fisher, S. Ishtiaq, N. Piterman, A. Taylor, and M. Y. Vardi, "BMA: Visual Tool for Modeling and Analyzing Biological Networks," in *Proceedings of the 24th International Conference on Computer Aided Verification (CAV 2012)*, 2012.

[48] B. Cook, J. Fisher, B. A. Hall, S. Ishtiaq, G. Juniwal, and N. Piterman, "Finding instability in biological models," in *Proceedings of the 26th International Conference on Computer Aided Verification (CAV 2014)*, 2014.

[49] K. Claessen, J. Fisher, S. Ishtiaq, N. Piterman, and Q. Wang, "Model-Checking Signal Transduction Networks through Decreasing Reachability Sets," in *Proceedings of the 25th International Conference on Computer Aided Verification (CAV 2013)*, 2013.

[50] M. R. Bareford, L. R. de Silva, I. P. Gent, B. M. Gorman, M. Hajiarabderkani, T. Henderson, L. Hutton, A. Konovalov, L. Kotthoff, C. McCreesh, M. A. Nacenta, R. R. Paul, K. E. J. Petrie, A. Razaq, D. Reijsbergen, and K. Takeda, "Case Studies and Challenges in Reproducibility in the Computational Sciences," April 2014, available from: http://arxiv.org/abs/1408.2123.

[51] I. P. Gent and L. Kotthoff, "Recomputation.org: Experience of Its First Year and Lessons Learned," in *Proceedings of 1st International Workshop on Recomputability*, 2014.

[52] M. Y. Vardi, "Openism, IPism, Fundamentalism, and Pragmatism," *Communications of the ACM*, vol. 57, no. 8, 2014.