Report from Dagstuhl Seminar 16111

Rethinking Experimental Methods in Computing

Edited by

Daniel Delling[1], Camil Demetrescu[2], David S. Johnson[3], and Jan Vitek[4]

1 **Apple Inc., Cupertino, US,** daniel.delling@live.com
2 **Sapienza University of Rome, IT,** demetres@dis.uniroma1.it
3 **Columbia University, New York, NY, US**
4 **Northeastern University, Boston, US,** j.vitek@neu.edu

Abstract

This report documents the talks and discussions at the Dagstuhl seminar 16111 "Rethinking Experimental Methods in Computing". The seminar brought together researchers from several computer science communities, including algorithm engineering, programming languages, information retrieval, high-performance computing, operations research, performance analysis, embedded systems, distributed systems, and software engineering. The main goals of the seminar were building a network of experimentalists and fostering a culture of sound quantitative experiments in computing. During the seminar, groups of participants have worked on distilling useful resources based on the collective experience gained in different communities and on planning actions to promote sound experimental methods and reproducibility efforts.

Seminar March 13–18, 2016 – http://www.dagstuhl.de/16111
1998 ACM Subject Classification D.2 Software Engineering, D.3 Programming Languages, E.1 Data Structures, F.2 Analysis of Algorithms and Problem Complexity, H.3 Information Storage and Retrieval, C.3 Special-Purpose and Application-based Systems, C.2 Computer-Communication Networks, C.4 Performance of Systems, B.8 Performance and Reliability
Keywords and phrases Algorithms, Benchmarks, Data sets, Experiments, Repeatability, Reproducibility, Software Artifacts, Statistics
Digital Object Identifier 10.4230/DagRep.6.3.24
Edited in cooperation with Emilio Coppa

1 Executive Summary

Emilio Coppa
Camil Demetrescu
Daniel Delling
Jan Vitek

This seminar is dedicated to the memory of our co-organiser and friend David Stifler Johnson, who played a major role in fostering a culture of experimental evaluation in computing and believed in the mission of this seminar. He will be deeply missed.

The pervasive application of computer programs in our modern society is raising fundamental questions about how software should be evaluated. Many communities in computer science and engineering rely on extensive experimental investigations to validate and gain insights on properties of algorithms, programs, or entire software suites spanning several layers of

complex code. However, as a discipline in its infancy, computer science still lags behind other long-standing fields such as natural sciences, which have been relying on the scientific method for centuries.

There are several threats and pitfalls in conducting rigorous experimental studies that are specific to computing disciplines. For example, experiments are often hard to repeat because code has not been released, it relies on stacks of proprietary or legacy software, or the computer architecture on which the original experiments were conducted is outdated. Moreover, the influence of side-effects stemming from hardware architectural features are often much higher than anticipated by the people conducting the experiments. The rise of multi-core architectures and large-scale computing infrastructures, and the ever growing adoption of concurrent and parallel programming models have made reproducibility issues even more critical. Another major problem is that many experimental works are poorly performed, making it difficult to draw any informative conclusions, misdirecting research, and curtailing creativity.

Figure 1 David Stifler Johnson, 1945–2016.

Surprisingly, in spite of all the common issues, there has been little or no cooperation on experimental methodologies between different computer science communities, who know very little of each others efforts. The goal of this seminar was to build stronger links and collaborations between computer science sub-communities around the pivotal concept of experimental analysis of software. Also, the seminar allowed exchange between communities their different views on experiments. The main target communities of this seminar were algorithm engineering, programming languages, operations research, and software engineering, but also people from other communities were invited to share their experiences. Our overall goal was to come up with a common foundation on how to evaluate software in general, and how to reproduce results. Since computer science is a leap behind natural sciences when it comes to experiments, the ultimate goal of the seminar was to make a step forward towards reducing this gap. The format of the seminar alternated talks intended for a broad audience, discussion panels, and working sessions in groups.

The organisers would like to thank the Dagstuhl team and all the participants for making the seminar a success. A warm acknowledgement goes to Amer Diwan, Sebastian Fischmeister, Catherine McGeoch, Matthias Hauswirth, Peter Sweeney, and Dorothea Wagner for their constant support and enthusiasm.

2 Table of Contents

3 Overview of Talks

3.1 Soundness of Experiments in Parallel Computing

Umut A. Acar (Carnegie Mellon University – Pittsburgh, US)

Recent advances in hardware such as the mainstream use of SMPs (multicore) computers, and large-scale data centers have brought parallelism back to the forefront of computing. Parallelism is now common in many different forms of hardware ranging from mobile phones to laptops and desktop computers. Unfortunately, writing performant parallel code can require low-level optimizations that make it extremely difficult to reproduce results and compare different approaches via standard empirical methodologies. In this talk, I will consider two specific problems–granularity control and locality–that can require such optimizations. As a solution, I propose the general idea of developing programming languages and systems that manage performance automatically for the programmer. This approach requires developing programming-language abstractions and algorithms for managing hardware resources in execution. As examples, I propose specific solutions to the two problems considered—granularity control and locality–and show some evidence that this approach can work well. (Joint work with Guy Blelloch, Arthur Charguéraud, Matthew Fluet, Stefan Muller, Ram Raghunathan, Mike Rainey)

3.2 How Did This Get Published? Pitfalls in Experimental Evaluation of Computing Systems

José Nelson Amaral (University of Alberta – Edmonton, CA)

This talk illustrates, through simple analogies, that summarizing either percentages or normalized quantities, such as speedups, using the arithmetic mean is always wrong when the quantities were normalized using different denominators. The talk then presents two alternative goals for summarization – latency or throughput – and presents suitable solutions for the summarization of several measurements in both cases.

3.3 What is the Value of the Data?

José Nelson Amaral (University of Alberta – Edmonton, CA)

In publications in Computing Science that report experimental results there is often lack of rigour and precision in the description of the experimental methodology and experimental setup, in the reporting and summarization of results and in the formulation of claims based on the experimental results. In this talk I put forward the notion that this state of affairs is, in a non-trivial portion, due to the low value of the experimental results. There are

several reasons for the low value placed on published results. Experimental evaluations are difficult to reproduce because the required code and specifications are seldom available. The results of a given evaluation are very much dependent on variations in hardware operation and in configurations of the software stack. Benchmarks used for evaluation often do not reflect the programs Thus often, when an organization is interested in a technique described in the literature that organization puts very little weight on the experimental evaluation. Thus, there is a cycle where the low value placed in the experimental results gives little incentive for more rigorous experimental evaluation, which in turns keeps the valuation of the experimental results low.

3.4 Experimental Methodology in Parallel and Streaming Analytics

David A. Bader (Georgia Institute of Technology – Atlanta, US)

Emerging real-world graph problems include: detecting community structure in large social networks; improving the resilience of the electric power grid; and detecting and preventing disease in human populations. Unlike traditional applications in computational science and engineering, solving these problems at scale often raises new challenges because of the sparsity and lack of locality in the data, the need for additional research on scalable algorithms and development of frameworks for solving these problems on high performance computers, and the need for improved models that also capture the noise and bias inherent in the torrential data streams. In this talk, the speaker will discuss experimental methodologies in massive data-intensive computing for applications in computational science and engineering.

3.5 The Importance of %: Why We Need to Think about Goals, Targets and Populations

Judith Bishop (Microsoft Research – Redmond, US)

Experiments frequently involve counting, and the absolute numbers are then measured over time (a graph that increases) or compared to other software that has similar characteristics (a bar chart). Sometimes, larger numbers are broken down into groups (a pie chart). Recently, there has been a push towards asking an additional question: what is the denominator? In other words, having measured a number, how does that relate to perfect? The result can be presented as a percentage. An example might be reporting the figure of 10,000 uses a month of a website. Immediately one can ask some more questions: geographically, is that for US students – well, a nice number – or for US consumers – not so good. If the denominator is known, then one can have more confidence regarding relative numbers. However, a chart showing usage of 10,000 and 15,000 by two companies, might still be missing by a million uses of a third company: the denominator reveals this hidden number. Nevertheless, it is extremely difficult to find denominators, even to estimate them. Emphasizing denominators is very important to industry where return on investment (ROI) is a factor in research. I shall give some examples and raise discussion on the issue. I'd be interested to know how the search for denominators fits into the other aims and expectations of the rest of the workshop.

3.6 Reproducibility in Computing: The Role of Professional Societies

Ronald F. Boisvert (NIST – Gaithersburg, US)

A scientific result is not fully established until it has been independently reproduced. Unfortunately, much published research is not independently verified. And, in the rare cases when a systematic effort has been made to do so, the results have not been encouraging. This threatens to undermine public confidence in the enterprise, and has led to calls for improvements to the process of reporting and reviewing scientific research in, for example, the biomedical sciences. The record in computing is not much better. Changing this state of affairs is not easy. Reproducing the work of others can be quite challenging, and does not garner the same credit as performing the original research. Professional societies have an important role in developing and promoting an open science ecosystem. As part of their role of arbiters and curators of the research literature, they can play a key role in changing the incentive structure to promote higher standards of reproducibility. In this talk I will describe some of the grassroots efforts being undertaken to improve the scientific process within the Association for Computing Machinery (ACM), the world's largest professional society in computing research. I will also describe steps within the ACM Publications Board being taken to support this.

3.7 Tools from Statistics, Machine Learning and Data Visualization for the Assessment of Heuristics for Optimization

Marco Chiarandini (University of Southern Denmark – Odense, DK)

The experimental analysis of algorithms within the engineering cycle may be conducted at different levels and with different goals. For example, we may be interested in describing behaviours for understanding and explaining algorithm execution or in modeling performance for prescribing the best algorithm configurations. Several tools from statistics and machine learning have been used to address these tasks. In the field of optimization heuristics, a machine learning approach seems, with good reasons, to be prevailing. New opportunities to make sense of data and improve presentation and reproducibility are offered by data-driven documents with interactive and dynamic graphics and by virutalized environments. In the talk, I will briefly go through these themes drawing examples from my experience as practitioner and reviewer.

3.8 Computing in the Cloud: Tools and Practices

Dmitry Duplyakin (University of Utah – Salt Lake City, US)

In this talk, I will outline the current work on supporting experiments on CloudLab, an NSF-funded large-scale cloud testbeds. I will share our motivation for using a configuration management system for "orchestrating" software components and discuss scenarios in which such systems can benefit experimenters. I will also highlight some of the proposed work, both on the infrastructure side and on techniques for characterizing the available computational resources and better understanding of scientific applications.

3.9 Network Testbeds and Repeatable Research

Eric Eide (University of Utah – Salt Lake City, US)

The Flux Research Group at the University of Utah has developed and operated public network testbeds, such as Emulab and CloudLab, for more than fifteen years. Beyond providing realistic and highly configurable environments for a variety of computer-based experiments, one of the goals of these testbeds is to encourage repeatable research. In this talk, I will review the history of testbed development at Utah with focus on features that support repeatable research. In addition, I will discuss issues and opportunities for network testbeds to better support repeatable research in the future.

3.10 Experimentation and Replication in Embedded and Real-Time Systems

Sebastian Fischmeister (University of Waterloo, CA)

Main reference A. Born de Oliveira, J.-C. Petkovich, T. Reidemeister, and S. Fischmeister, "DataMill: rigorous performance evaluation made easy", in Proc. of the 4th ACM/SPEC Intl. Conf. on Performance Engineering (ICPE '13), pp. 137–148, 2013.
URL http://dx.doi.org/10.1145/2479871.2479892

This talk will discuss the need for rigorous performance analysis and report results of a replication experiment conducted on the DataMill infrastructure. Based on the lessons learnt from building DataMill and running experiments, the talk then also discusses the challenges of performance analysis with specific focus on experimentation in embedded and heterogeneous computing systems.

3.11 The PRIMAD Model of Reproducibility: A Framework Model of Reproducibility (Result of Dagstuhl Seminar 16041)

Norbert Fuhr (Universität Duisburg-Essen, DE)

For describing different degrees of reproducibility, the participants of Dagstuhl Seminar 16041 started from a model referring to the components of an experiment: (R) the research goal, (M) the method proposed for achieving this goal, (I) the implementation of this method, (P) the platform on which the implementation is run, (D) the data (input + parameters) used in the experiment, and finally (A) the actor performing the experiment. When a researcher tries to reproduce an experiment, he should specify which components are changed, i.e. 'primed': R → R' repurpose for a new research goal, M → M': a new method, I → I': alternative implementation, P → P': different platform, D → D': other input/parameters. Finally, other important aspects of reproducibility are consistency of experimental results, and transparency, i.e. the ability to look into all necessary components to verify that the experiment does what it claims.

3.12 Lessons Learned from Shortest Path Algorithm Evaluation

Andrew V. Goldberg (Amazon.com, Inc. – Palo Alto, US)

This is a retrospective on experimental evaluation of shortest path algorithms. We discuss bridging the gap between theory in practice in the area that happened in 1990s. We discuss the general problem, with negative-length arcs and possible negative cycles, and the special cases: No negative cycles and no negative arcs.

3.13 The TIRA Experiment Platform

Matthias Hagen (Bauhaus-Universität Weimar, DE)

The TIRA experimentation platform supports organisers of shared tasks in computer science to accept the submission of executable software and allows for reproducible experimental settings. Through virtualization techniques, participants in a shared task can directly work as they usually would on their own machines. TIRA also hosts the datasets used in a shared task with the option of keeping test data in a secure execution environment that protects them from leaking to the participants. Experimental results are displayed on a dedicated web page. Later reproducing of results or comparing to the state of the art for a given task becomes as easy as clicking a button.

3.14 Artifact Evaluation: Approach and Experience from OOPSLA's first AEC

Matthias Hauswirth (University of Lugano, CH)

The programming languages community recently introduced so-called "artifact evaluation committees" to their conferences (PLDI, POPL, OOPSLA, ECOOP and others). Those AECs complement the traditional program committees by evaluating the artifacts underlying the papers. I will describe the idea behind AECs and will report on the experience of running the first two AECs at OOPSLA.

3.15 Incentives & Rewards

Matthias Hauswirth (University of Lugano, CH)

We use the term "incentivize" quite often when thinking about solving certain problems or steering the community in certain ways. One such incentive is the AEC badge authors can place on their papers. While I designed and used that badge at OOPSLA, I claim that incentives don't usually work as intended. In this talk I will provide a brief glimpse into the evidence (in terms of arguments and experimental results) against using incentives in our situation, and in most situations we care about.

3.16 Rigorous Benchmarking in Reasonable Time

Tomas Kalibera (Northeastern University – Boston, US)

I speculate that the quality of experiments performed in systems/PL is often so low because some researchers believe they would have to run excessive amounts of experiments of benchmarks for excessive amount of time. We show how to adapt the experiment scale to a particular problem/system to keep the experimentation time reasonable, while not giving up on the rigor. We present the method on the example of DaCapo and SPEC CPU benchmarks. It is not that every time a new source of performance dependency (e.g. unix environment size, symbol naming at compile time, page allocation at process startup) is found, we have to multiply the size of all experiments we ever run.

3.17 Data Analysis for Performance Modeling

Catherine C. McGeoch (Amherst College, US)

This talk will discuss the problem of building an empirical performance model for algorithms: that is, developing a function that describes the relationship between time performance and the main factors that affect performance, including input, algorithm, and platform parameters. Contrasting advice as to choice of statistical and data analysis tools for this task may be found, between proponents of the Scientific Method (hypothesis testing, confirmatory statistics), and what has been called The New Experimentalism (exploratory data analysis). I will briefly survey these two schools of thought and how they apply to algorithmic performance modeling. To illustrate these points I will talk about some empirical surprises that arise when developing a performance model for D-Wave quantum annealing systems.

3.18 Chaos in Computer Performance

J. Eliot B. Moss (University of Massachusetts – Amherst, US)

This talk will first give a brief argument as to how any cache-like mechanism, and indeed most state machines, are subject to chaotic behavior. Whether and how chaos shows up depends on a program's interaction with the mechanism, however, and I will show some examples from cache behavior from SPEC CPU 2000 benchmarks to give a sense of this and close with suggestions of directions for future experimental methodology.

3.19 Assessing the Performance of Heuristics in Multiobjective Optimization: an Overview

Luís Paquete (University of Coimbra, PT)

Multiobjective optimization problems are usually hard to solve. Heuristics are often used to find an reasonably good approximation to the set of optimal solutions. The image of such approximation in the objective space consists of a set of points that contains a particular structure. Since most of these heuristic approaches are of stochastic nature, the sets of points produced may differ from run to run. This raises an interesting challenge: how to characterize and compare heuristics based on the sets of points produced in a collection of runs? In this talk, we give an overview of the main methods for assessing the performance of heuristics, from a solution quality perspective, for this class of optimization problems.

3.20 Algorithm Engineering: An Attempt at a Definition

Peter Sanders (KIT – Karlsruher Institut für Technologie, DE)

This talk defines algorithm engineering as a general methodology for algorithmic research. The main process in this methodology is a cycle consisting of algorithm design, analysis, implementation and experimental evaluation that resembles Popper's scientific method. Important additional issues are realistic models, algorithm libraries, benchmarks with real-world problem instances, and a strong coupling to applications. Algorithm theory with its process of subsequent modeling, design, and analysis is not a competing approach to algorithmics but an important ingredient of algorithm engineering. At the end of the talk, we additionally discuss how algorithm engineering might help with interdisciplinary research in particular in grand challenge big data applications.

3.21 The Truth, the Whole Truth and Nothing but the Truth

Peter F. Sweeney (IBM TJ Watson Research Center – Yorktown Heights, US)

The EVALUATE 2011 workshop, co-located with PLDI, brought together members of the programming language community to discuss experimental evaluation. The outcome of this endeavor has resulted in "The Truth, the Whole Truth, and Nothing but the Truth: A Pragmatic Guide to Assessing Empirical Evaluations" that has been accepted for publication at TOPLAS. Specifically, an unsound claim can misdirect a field, encouraging the pursuit of unworthy ideas and the abandonment of promising ideas. An inadequate description of a claim can make it difficult to reason about the claim, for example to determine whether the claim is sound. Many practitioners will acknowledge the threat of unsound claims or inadequate descriptions of claims to their field. We believe that this situation is exacerbated and even encouraged by the lack of a systematic approach to exploring, exposing, and addressing the source of unsound claims and poor exposition. This paper proposes a framework that identifies three sins of reasoning that lead to unsound claims and two sins of exposition that lead to poorly described claims. Sins of exposition obfuscate the objective of determining whether or not a claim is sound, while sins of reasoning lead directly to unsound claims. Our framework provides practitioners with a principled way of critiquing the integrity of their own work and the work of others. We hope that this will help individuals conduct better science and encourage a cultural shift in our research community to identify and promulgate sound claims.

3.22 Experimenting with Humans vs. Experimenting with Machines

Walter F. Tichy (KIT – Karlsruher Institut für Technologie, DE)

Experiments in computing are done either with or without human participants. I will contrast the experimental protocols. Benchmarks can often substitute for human subjects. Analyzing recorded data does not normally qualify as a controlled experiment, as indpendent variables cannot be varied systematically; thus sich studies can show correlation (which can be used for prediction) but not causation. Exeriments with humans, as for instance in HCI or software engineering, are time-consuming, expensive, and high-risk. Often reviewers do not understand the difficulties of such experiemnts and reject them out of hand for minor imperfections.

3.23 I Think Nobody Wants to Do Bad Science!

Petr Tuma (Charles University – Prague, CZ)

This is a very short talk based on the initial seminar survey results, which aims to pose some questions not explicitly discussed in the survey responses. The questions revolve around our ability to cover all the technical intricacies of experiments, balancing the costs and benefits of performing robust experiments, and more generally finding ways of accepting the limits of experimental results.

3.24 Some remarks on data sharing and the replication of results

Dorothea Wagner (KIT – Karlsruher Institut für Technologie, DE)

During the last 5 to 10 years, national and international science and research funding organization started a discussion on principles of good scientific practice with regard to the replication of research results. According agreements clearly state that primary research data should be shared openly and promptly. In many scientific disciplines, we can observe intensive common efforts to support "open data in science". In this talk I want to raise the question if our community is doing enough in this respect. How can we support and intensify the collection of data (like benchmark data), and the reproducibility and comparability of experiments?

3.25 Experimenting with Innocent Humans

Roger Wattenhofer (ETH Zürich, CH)

A random sentence from a random paper: "We tested our software with 7 subjects". The 7 PhD students in your lab, probably... Anyway, a few years ago we started testing some of our software with thousands of innocent users. In short this talk will present a few lessons we learned. We will discuss the ethical parameters of such massive user studies.

4 Working groups

4.1 Educating the community

Umut A. Acar (Carnegie Mellon University – Pittsburgh, US), José Nelson Amaral (University of Alberta – Edmonton, CA), David A. Bader (Georgia Institute of Technology – Atlanta, US), Judith Bishop (Microsoft Research – Redmond, US), Ronald F. Boisvert (NIST – Gaithersburg, US), Marco Chiarandini (University of Southern Denmark – Odense, DK), Markus Chimani (Universität Osnabrück, DE), Daniel Delling (Apple Inc. – Cupertino, US), Camil Demetrescu (Sapienza University of Rome, IT), Amer Diwan (Google – San Francisco, US), Dmitry Duplyakin (University of Utah – Salt Lake City, US), Eric Eide (University of Utah – Salt Lake City, US), Erik Ernst (Google – Aarhus, DK), Sebastian Fischmeister (University of Waterloo, CA), Norbert Fuhr (Universität Duisburg-Essen, DE), Paolo G. Giarrusso (Universität Tübingen, DE), Andrew V. Goldberg (Amazon.com, Inc. – Palo Alto, US), Matthias Hagen (Bauhaus-Universität Weimar, DE), Matthias Hauswirth (University of Lugano, CH), Benjamin Hiller (Konrad-Zuse-Zentrum – Berlin, DE), Richard Jones (University of Kent – Canterbury, GB), Tomas Kalibera (Northeastern University – Boston, US), Marco Lübbecke (RWTH Aachen, DE), Catherine C. McGeoch (Amherst College, US), Kurt Mehlhorn (MPI für Informatik – Saarbrücken, DE), J. Eliot B. Moss (University of Massachusetts – Amherst, US), Ian Munro (University of Waterloo, CA), Petra Mutzel (TU Dortmund, DE), Luís Paquete (University of Coimbra, PT), Mauricio Resende (Amazon.com, Inc. – Seattle, US), Peter Sanders (KIT – Karlsruher Institut für Technologie, DE), Nodari Sitchinava (University of Hawaii at Manoa – Honolulu, US), Peter F. Sweeney (IBM TJ Watson Research Center – Yorktown Heights, US), Walter F. Tichy (KIT – Karlsruher Institut für Technologie, DE), Petr Tuma (Charles University – Prague, CZ), Dorothea Wagner (KIT – Karlsruher Institut für Technologie, DE), and Roger Wattenhofer (ETH Zürich, CH)

One of the key challenges discussed during the seminar was how to educate the community to care and recognise the pitfalls and the benefits of conducting sound experiments. After a common discussion, we formed three subgroups, which worked on: (i) guidelines for

authors and reviewers in writing and assessing experimental work, (ii) a Wikipedia page on experimental methods in computing, and (iii) resources on software experimental methods for students.

4.1.1 Great Papers: a reviewer's guide to evaluating experimental research in computer science

The main goal of this working group was distilling some key aspects that characterise good quantitative experimental work in computer science. In particular, the group identified six lines that authors and reviewers are encouraged to consider in writing and assessing experimental work in computing:

1. **Experimental context.** Great papers have: clearly specified goals, scope, and research questions matching the claims.
2. **Experimental design.** Great papers have: clear description of the methodology, which matches the stated goals of the experiments and encourages reproduction; well chosen baselines, competing approaches, and benchmarks; consistent comparison to previous experimental work; independent and control variables identified that are most important to the stated goals; attention paid to possible hidden variables.
3. **Conduct of the experiment.** Great papers have: a clearly stated procedure for collecting data; metrics and measurement procedures that match experimental goals; sufficient repetitions of trials in cases of non-deterministic or random outcomes.
4. **Analysis.** Great papers: use appropriate statistical procedures in terms of the data, its distribution, and the experimental goals; expose unusual distribution properties (including skew, bimodality, and outliers); acknowledge and attempt to explain anomalous or missing data.
5. **Presentation of results.** Great papers have: clear and insightful presentation that addresses the stated goals of the work; careful choice of aggregate and summary statistics; effective use of visualisation techniques; accompanying text that describes figures and tables.
6. **Interpretation of results.** Great papers have: claims that are clearly justified by the data and analysis; consideration of alternative causes for the observations; explicit separation between justified claims and generalisations beyond the scope of the experiment.

The group plans to extend the list with concrete examples and reach out to program chairs of conferences and journal editors to refine and tailor the list to specific sub-communities.

References

1 Preliminary Guidelines for Empirical Research in Software Engineering. `http://evaluate.inf.usi.ch/node/30`
2 How to Write a Scientific Evaluation Paper. `http://evaluate.inf.usi.ch/node/54`

4.1.2 Wikipedia page on Experimental methods in computing

A second goals of this working group was to start writing a Wikipedia page on *Experimental methods in computing*. An initial draft of this page can be found at `https://en.wikipedia.org/wiki/Draft:Experimental_methods_in_computing`.

4.1.3 How to produce sound quantitative research: information for students

This working group has also produced a summary document providing useful resources for new graduate students and young researchers. An on-going version of the document is available at http://tinyurl.com/j6cbghz.

4.2 Evangelism

Mauricio Resende (Amazon.com, Inc. – Seattle, US), David A. Bader (Georgia Institute of Technology – Atlanta, US), Ronald F. Boisvert (NIST – Gaithersburg, US), Catherine C. McGeoch (Amherst College, US), J. Eliot B. Moss (University of Massachusetts – Amherst, US), and Dorothea Wagner (KIT – Karlsruher Institut für Technologie, DE)

The main goal of this working group was to consider how best to publicize the work of Dagstuhl Seminar 16111 and how to facilitate uptake in the larger community. A list of the main ideas follows:

1. Contributions to *CRA Best Practices Memos*, *Informatics in Europe* or to a stand-alone collection. Announce presence with targeted emails. Topic ideas:
   - How to review an experimental paper in subfield *X*.
   - How to add artifact submission and evaluation to your conference or journal submission process.
   - Advice for tenure committees on reviewing experimental work in computer science.
   - How to integrate experimental projects in your classroom.
   - How to build and maintain a living benchmark repository.
   - How to run a DIMACS Challenge.
2. Ensure continuity of DIMACS Challenge series and evaluate if there should be challenges in other areas.
3. Talk to people who organize summer schools and people who might teach summer schools, about an experimental methodology course.
4. Contact conference and journal steering committees, to propose that they consider initiating procedures for artifact submission and evaluation.
5. Put together a short list of best papers in experimental methodology.
6. Talk to organisers about a follow-up of this meeting, at Dagstuhl or FCRC.
7. Write letters (position papers) to key players in CS research. Introduce experimental methodology/support for artifacts; explain why it is important; highlight key points; suggest that the recipient creates incentives and support mechanisms; say how to find out more information. The group suggested two white papers, addressing different topics for different (but overlapping) focus groups:
   a. Position paper on research methodology.
   b. Position paper on artifact evaluation and publishing.
8. Contact steering committees of appropriate meetings, to propose that they consider experimentalists when selecting plenary speakers.
9. Propose speakers to ACM speaker series or to Sigma Xi speaker series.

10. Start an online ask-the-experimentalist resource, e.g., in StackOverflow, ResearchGate, for questions about experimental methodology and statistics/data analysis. Identify people willing to monitor and answer the questions.
11. Find someone to write a regular column in a discipline-wide venue with topics in experimental methodology, or start a blog.
12. Teach a MOOC on experimental methods in computer science. It could be taught cooperatively (several professors) with standalone 'units' in different areas.
13. Suggest special issues on empirical methodology to appropriate journal Editors-in-Chief.

4.3 Replicability

Petr Tuma (Charles University – Prague, CZ), Umut A. Acar (Carnegie Mellon University – Pittsburgh, US), Judith Bishop (Microsoft Research – Redmond, US), Ronald F. Boisvert (NIST – Gaithersburg, US), Amer Diwan (Google – San Francisco, US), Dmitry Duplyakin (University of Utah – Salt Lake City, US), Eric Eide (University of Utah – Salt Lake City, US), Norbert Fuhr (Universität Duisburg-Essen, DE), Matthias Hagen (Bauhaus-Universität Weimar, DE), J. Eliot B. Moss (University of Massachusetts – Amherst, US), and Peter F. Sweeney (IBM TJ Watson Research Center – Yorktown Heights, US)

4.3.1 Guidelines

As a first activity, the group identified a set of guidelines to be given to a team doing a reproducibility study:

1. *What is a reproducibility study?*
   A reproducibility study attempts to confirm independently some (not necessarily all) important claim(s) made in a preceding paper; it may also attempt to provide more insight. Such a study should be carried out by a different team, using a different apparatus, in a different location. A different implementation of the same algorithm may be used, additional benchmarks or inputs, additional metrics and/or additional statistical analysis.
2. Motivation: *Why do a reproducibility study?*
   A reproducibility study serves to increase the confidence in reported scientific results by confirming (or refuting) the claims of the original study in changed settings. A reproducibility study may also contribute to the discipline methodologically.
3. *What issues should a reproducibility study address?*
   - Motivation: Why did you choose this work to reproduce? Argue why this is an interesting and important paper to reproduce.
   - What claim or claims (implicit or explicit) in the original paper are you reproducing?
   - What additional or broader claims (if any) are you making in this study?
   - What artifacts (software, hardware, data, etc.) did you use from the original paper? How did you audit their correctness?
   - What challenges did you face in doing this study and how did you surmount those challenges?
   - What interaction did you have with the original researchers? (Your evaluation should be arguably independent of the original study.)
   - What did you change in your experiment compared to the original evaluation?

- Do your claims support or refute the claims in the original paper? If they refute the claims, please explain the likely cause, if possible.
- What additional insights did you gain? (Remember, the goal is not to find fault but to increase insight!)
- What recommendations do you make (if any) in the light of your experience?

4. Experimental methodology. A reproducibility study should:
   - Deal with nondeterminism: multiple runs: look at distribution, statistics to aggregate; i.e. average stdev
   - Analyze uncertainty: systematic and random effects
   - Apply appropriate statistical methods on raw data
   - Investigate any failures to support the original claims. Are their hidden variables?
5. *How is a reproducibility study different from an artifact evaluation?*
   An artifact evaluation only tries to repeat the original results using apparatus and conditions as similar as possible to the original, within the time constraints of the review process for a scientific publication. Specifically, an artifact evaluation checks that the artifact is consistent with the original paper, complete, well documented and easy to reuse. A reproducibility study has broader goals, intentionally works with different apparatus (such as a different computer system, possibly different software, etc.), produces and checks results, and does not have similar time constraints.

4.3.2 A suggested format of a study

As a second contribution, the group discussed possible formats that may be suggested for a reproducibility study, which can be summarised as follows:

1. Synopsis of original idea: motivation for choosing this work.
2. Description of original evaluation.
3. Description of reproducibility evaluation:
   - What artifacts from original work have been used.
   - How experimental system has changed:
     a. New benchmarks.
     b. New metrics.
     c. New experimental setup.
     d. New analysis of raw data.
4. Results
   - Replicability success?
   - Reproducibility success on different dimensions of change.
   - Graphs.
5. Lessons learnt: positives and negatives; what was easy, what was hard.

4.3.3 Letter to journal editors

Following up on the activities of Section 4.2, the group drafted a letter for editors-in-chief of journals to propose the idea of devoting an issue to selected reproducibility results:

> *Dear Editor,*
>
> *How do we know that an idea in a paper is effective? Many papers include evaluations to determine the effectiveness of an idea; however, evaluations are difficult to do. For example, if an evaluation fails to consider some important variable, it may produce invalid results. Even if an evaluation is valid it is likely severely limited in scope (e.g., applies only to a benchmark suite or to a particular programming language) and thus of limited use.*

The traditional method for improving the confidence in an idea or to extend the generality of an idea is to do reproducibility studies: some group other than the original group tests the idea in a different context from the original evaluation (e.g., using different benchmarks or hardware). If the reproduction is successful (i.e., the results are compatible with the original results) we gain confidence in the effectiveness of the idea and also generalize the original results. If the reproduction is unsuccessful, we discover that the idea has limited applicability or even that the claim in the original paper is invalid. A great reproduction study provides deep insights into an important idea from prior work.

Unfortunately, Computer Science does not have a tradition of reproducibility studies. We would like to change that.

Our proposal is for several journals (each in a different area of Computer Science) to reserve and advertise one or more reserved slots for reproducibility papers once a year (we propose the first issue of each year, but can certainly adjust this based on the responses we get from the journals). This paper would be held to the same standards as any other paper in the journal however (i) submissions for this slot would have a deadline; (ii) the journal editors would prioritize the review of these submissions so that they are ready in time for the next slot; and (iii) the reviewers would be instructed to evaluate the paper as a reproduction paper. Of course, if there is no reproduction submission that meets the journal's standards, the journal would use the slot for a regular paper.

By having multiple top-tier journals publishing reproducibility papers, the journals will send a clear message that such papers are valued contributions to top publication venues. In doing so, the journals will promote the identification of effective ideas and thus improve upon the scientific rigor in Computer Science.

Our team (attendees of the Dagstuhl Rethinking Experimental Methods in Computing Seminar) will (i) help advertise these slots widely; and (ii) help identify authors of reproducibility studies (and of course authors can simply submit in response to our advertisement). If you need help finding reviewers for a reproduction study, we can help to identify reviewers. The final decision to accept the paper, of course, lies in the hands of the journal editors.

Participants

- Umut A. Acar
Carnegie Mellon University – Pittsburgh, US
- José Nelson Amaral
University of Alberta – Edmonton, CA
- David A. Bader
Georgia Institute of Technology – Atlanta, US
- Judith Bishop
Microsoft Res. – Redmond, US
- Ronald F. Boisvert
NIST – Gaithersburg, US
- Marco Chiarandini
University of Southern Denmark – Odense, DK
- Markus Chimani
Universität Osnabrück, DE
- Emilio Coppa
Sapienza University of Rome, IT
- Daniel Delling
Apple Inc. – Cupertino, US
- Camil Demetrescu
Sapienza University of Rome, IT
- Amer Diwan
Google – San Francisco, US
- Dmitry Duplyakin
University of Utah – Salt Lake City, US
- Eric Eide
University of Utah – Salt Lake City, US
- Erik Ernst
Google – Aarhus, DK
- Sebastian Fischmeister
University of Waterloo, CA
- Norbert Fuhr
Universität Duisburg-Essen, DE
- Paolo G. Giarrusso
Universität Tübingen, DE
- Andrew V. Goldberg
Amazon.com, Inc. – Palo Alto, US
- Matthias Hagen
Bauhaus-Universität Weimar, DE
- Matthias Hauswirth
University of Lugano, CH
- Benjamin Hiller
Konrad-Zuse-Zentrum – Berlin, DE
- Richard Jones
Univ. of Kent – Canterbury, GB
- Tomas Kalibera
Northeastern University – Boston, US
- Marco Lübbecke
RWTH Aachen, DE
- Catherine C. McGeoch
Amherst College, US
- Kurt Mehlhorn
MPI für Informatik – Saarbrücken, DE
- J. Eliot B. Moss
University of Massachusetts – Amherst, US
- Ian Munro
University of Waterloo, CA
- Petra Mutzel
TU Dortmund, DE
- Luís Paquete
University of Coimbra, PT
- Mauricio Resende
Amazon.com, Inc. – Seattle, US
- Peter Sanders
KIT – Karlsruher Institut für Technologie, DE
- Nodari Sitchinava
University of Hawaii at Manoa – Honolulu, US
- Peter F. Sweeney
IBM TJ Watson Research Center – Yorktown Heights, US
- Walter F. Tichy
KIT – Karlsruher Institut für Technologie, DE
- Petr Tuma
Charles University – Prague, CZ
- Dorothea Wagner
KIT – Karlsruher Institut für Technologie, DE
- Roger Wattenhofer
ETH Zürich, CH