A Benchmark- and Competition-Based Approach to Software Engineering Research

Derek Bronish
The Ohio State University
Department of Computer Science and Engineering
2015 Neil Ave.
Columbus, OH 43210
bronish@cse.ohio-state.edu

Jason Kirschenbaum
The Ohio State University
Department of Computer Science and Engineering
2015 Neil Ave.
Columbus, OH 43210
kirschen@cse.ohio-state.edu

Aditi Tagore
The Ohio State University
Department of Computer Science and Engineering
2015 Neil Ave.
Columbus, OH 43210
tagore@cse.ohio-state.edu

ABSTRACT

Many fields of study within computer science have benefited from the adoption of community-wide benchmarks and competitions. Software engineering has yet to fully embrace this approach. Case studies of existing uses of these techniques are presented, and a hypothetical application to software engineering based on research presented in last year's FSE are elaborated.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General

General Terms

Human Factors

1. INTRODUCTION

Software engineering is a fundamentally fragmented field of study. Models of distributed computation, empirical analyses of enterprise processes, language theory, verification, and debugging are just a sampling of the diverse topics that fall under this umbrella. This heterogeneity has naturally led to a compartmentalized research community. Often, different groups work on a similar problem in mutual unawareness of each other, or devise solutions so differently informed that comparison proves cumbersome.

We envision a future in which the community has adopted new ways to unify software engineering research. Specifically, we imagine that benchmarks and competition-based research would not only focus the efforts of groups working on essentially the same problem, but also would serve as guideposts for the community as a whole, steering research in fruitful directions that are always conscious of the state-of-the-art.

FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA.

We begin with a discussion of two related but distinct styles in which such research strategies can be employed, broadly classified as "competitive" and "collaborative." In support of the position that both these styles are tenable, we present brief case studies that elaborate how competitions and collaborative benchmarks have aided other fields within computer science. We also suggest an idea for a competition that could have been applied to non-unified research efforts published in last year's FSE conference.

2. COMPETITIONS AND COLLABORATIVE BENCHMARK PROBLEMS

In broad terms, this position paper calls for an approach to software engineering research that thrives on communication and cross-pollination among intellectually and geographically diverse research groups. In practice, such efforts tend to develop in two different ways. First, the groups may enter into competition to solve a particular problem. Often, such competitions are administrated at a research conference, and due to matters of academic integrity no actual hybridization of ideas occurs until after the solutions are submitted and the results are determined. Alternatively, inter-group research can be structured at the outset around "benchmark problems," and ideas can be exchanged throughout the process.

We do not wish here to advocate one of these two styles to the exclusion of the other. Often, incidental circumstances such as market forces will dictate the research style (e.g., private companies conducting research in parallel will be unlikely to adopt the collaborative style, but may be willing to participate in closed-source competitions). It is interesting to observe, however, that while the collaborative benchmark style seems to be much less popular, it has precipitated substantial innovations in cases where it has been adopted.

3. CASE STUDY: THE FOUNDATIONS OF MATHEMATICS MAILING LIST

The Foundations of Mathematics (FOM) mailing list [1] has been active since September 1997. It was created by a small group of active researchers in the foundations of mathematics as a means to obtain rapid feedback from academic peers. Typical postings include discussions of open issues in the field, aid in finding references to other work, critiques of

previously published papers, and solicitations for feedback on the essence of unpublished work.

The FOM is an open forum; all communications on the list are archived for anyone to read. In order to receive the FOM emails in real-time or in a digest, one must subscribe to the mailing list and include an explanation of one's interest in the subject. Posts to the list are vetted by a moderator and an editorial board. The purpose of the moderator is to reject posts that are not relevant to the list or do not meet basic quality standards. The editorial board aids the moderator when it is unclear whether or not a post is acceptable.

The collaborative interactions and editorial provisions are designed to ensure that messages on the FOM list comprise a high-quality, focused discussion, and hence that setting aside regular time to participate in the list is in the interest of the busy people who follow it. The FOM remains a highly active venue for collaboration in the area of the foundations of mathematics.

4. CASE STUDY: DARPA GRAND CHALLENGE

In many fields outside of computer science, competitions and benchmarks have promoted an explosion of activity, such as the three DARPA Grand Challenges [8] in 2004, 2005 and 2007. The original challenge was to create an autonomous vehicle that would be able to drive through rough terrain completely automatically. No team finished the course in 2004 within 10 hours, yet in 2007 the course was completed by many teams within 5 hours. The competition initially had no other traffic, but by 2007 it was a requirement that the autonomous vehicles be able to negotiate traffic; the state of the art was clearly advanced, in part, because of this competition.

5. CASE STUDY: HIGH-PERFORMANCE COMPUTING

In the high-performance computing community, there are a few well-known benchmarks which are accepted widely and are used to measure the performance of supercomputers.

The TOP500 [3] is an official list of the 500 most powerful supercomputers. This list has been compiled based on the LINPACK benchmark, which measures a system's floating point computing power based on its performance in solving a dense system of linear equations.

The High Performance Computing Challenge (HPCC) Benchmark suite [2] is a set of seven benchmarks that test various attributes of a high-performance system. The LINPACK benchmark used in the TOP500 is one of the benchmarks used in HPCC. The other benchmarks in this suite are:

- DGEMM: measures the floating point execution rate for double precision real matrix-matrix multiplication.
- STREAM: measures sustainable memory bandwidth.
- PTRANS: measures the rate of transfer for large arrays of data from multiprocessor memory.
- RandomAccess: measures the rate of random memory updates.
- FFTE: measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform.
- b_eff: measures latency and bandwidth of network communication using basic MPI routines.

The HPC Challenge Awards competition is held annually to measure the performance and the productivity of the computers based on four of the HPCC Benchmarks (LINPACK, RandomAccess, STREAM and FFTE).

The NAS Parallel suite, maintained by the NASA Advanced Supercomputing (NAS) division, is another widely recognized set of benchmarks for evaluating the performance of parallel supercomputers. The motivation for developing this set of benchmarks was to define them only algorithmically, leaving the details to the implementor. The current version consists of 11 benchmarks. Some examples are MultiGrid, which approximates the solution to a three-dimensional discrete Poisson equation, Integer Sort, which uses bucket-sort to sort integers, and FT, which solves a partial differential equation using a Fourier transform.

6. CASE STUDY: THEOREM PROVING

The automated theorem proving community has a rich tradition of competition as a mechanism to spur innovation. Two such competition benchmarks are SMT-Comp [5], which is for Satisfiability Modulo Theory provers, and CASC [13], which is for automated first-order logic provers.

Both competitions use rigorously defined standardized problem formats: SMT-Lib [11] and TPTP [12]. In each of these competitions, there are distinct categories of problems intended to comprehensively exercise the competing tools. For example, in SMT-Comp, some of the problems are randomly chosen, others are designed to test for certain capabilities, others are chosen from industrial applications, and some are crafted to expose issues in the logical setup. The CASC competition adds additional constraints on the tools, namely that they should not be able to prove certain known false conjectures. Moreover, each of the tools in either competition is run on a standard system with known hardware and specification for what a tool should do to be considered to solve a benchmark problem. For example, tools may not take more than a certain amount of time to solve a problem.

This competition-based approach seems to have had an appreciably positive impact on the quality of automated theorem proving tools. For example, the SMT-Comp competition documentation notes that every year the tools have increased in quality. The objective competition allows for separate ideas to be independently developed and transparently evaluated, to the extent the competitors are willing and able to reveal their secrets.

7. CASE STUDY: VERIFICATION BENCHMARKS

In the software verification research community, two of the authors of this position paper have contributed to the definition of a set of benchmark problems [14]. At least two solutions to the problems—by two disjoint research groups using qualitatively different strategies—have been submitted so far. Comparative evaluation of the merits of the two solutions is currently in the review process, and our hope is that this will encourage others to participate, to the benefit of all. Indeed, paper [14] has inspired the development and publication of another set of benchmarks [10]. The benchmarks in [14] were proposed as a method to improve common

understanding of diverse research efforts, while those of [10] included a proposed scoring system for a competition. So both the collaborative and competitive tacts are being explored in this community.

The proposal of these benchmarks, and the fact that other groups have taken up the challenge of implementing solutions, has led to a thriving exchange of ideas between groups working on software verification who previously would only encounter each others' work through the relatively slow means of refereed conference and journal papers. To further catalyze this speedier, more fruitful communication, the authors of this paper and colleagues are developing a web site to serve as a community resource and unify further verification research. A new, limited-functionality site with a similar focus has been set up within a European project (COST Action IC0701) [4].

8. APPLICATIONS TO SOFTWARE ENGINEERING RESEARCH

In last year's FSE proceedings, two papers [6, 9] proposed fundamentally divergent tactics for finding potential threats to sound reasoning about multithreaded programs: one involves runtime-checked assertions wrapping blocks of code intended to behave deterministically, and the other gives an algorithm for detecting harmful data races statically. Thus, the papers' respective authors propose qualitatively different solutions to a widely acknowledged problem. As a community, our duty is to compare, contrast, and evaluate these proposals, and to adjudicate precisely how best to advance the state of the art. Unfortunately, each such paper is a self-contained system, performing its own evaluation of its own ideas. This makes fair inter-paper comparison exceedingly difficult.

Consider instead a unified framework for pitting such systems against each other and evaluating their performance via widely accepted metrics and challenge problems. This would provide fair grounds for comparison among multiple solutions to a problem, in a sense that the current method of research in the software engineering community does not.

We emphasize that the community should be welcoming of such comparative, review-oriented publications, as they in themselves may contribute novel ideas. This is well understood in other fields of study not far from computer science, e.g., linguistics (see [7] for a famous example), but not sufficiently so in our field itself. A paper that simply reviews or recreates and then critiques the work of another group is likely to be rejected for not being "novel" enough, whereas we argue that such reviews are exactly how other sciences remain well-regulated and progressive.

9. CONCLUSION

This proposed methodology of competition- and benchmark-based research is of course not a panacea. For instance, it is less well-suited to comparing and evaluating *unimplemented* theoretical proposals than it is to situations where automation is possible. Regardless, it seems to be a promising strategy that has served other communities well, and is insufficiently appreciated in Software Engineering.

10. REFERENCES

[1] FOM mailing list. http://www.cs.nyu.edu/mailman/listinfo/fom, Last accessed August 2010.

[2] HPC Challenge homepage. http://icl.cs.utk.edu/hpcc, Last accessed June 2010.

[3] Top 500 homepage. http://top500.org, Last accessed June 2010.

[4] VerifyThis: A collection of verification benchmarks. http://verifythis.cost-ic0701.org, Last accessed September 2010.

[5] C. Barrett, L. de Moura, and A. Stump. SMT-COMP: Satisfiability Modulo Theories Competition. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification*, pages 20–23. Springer, 2005.

[6] J. Burnim and K. Sen. Asserting and checking determinism for multithreaded programs. In *ESEC/FSE '09: Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 3–12, New York, NY, USA, 2009. ACM.

[7] N. Chomsky. A review of B.F. Skinner's Verbal Behavior. *Language*, 35(1):26–58, 1959.

[8] K. Iagnemma and M. Buehler. Editorial for journal of field robotics—special issue on the DARPA grand challenge: Editorial. *J. Robot. Syst.*, 23(9):655–656, 2006.

[9] V. Kahlon, N. Sinha, E. Kruus, and Y. Zhang. Static data race detection for concurrent programs with asynchronous calls. In *ESEC/FSE '09: Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 13–22, New York, NY, USA, 2009. ACM.

[10] K. R. M. Leino and M. Moskal. VACID-0: Verification of ample correctness of invariants of data-structures, edition 0. In *Verified Software: Theories, Tools and Experiments, VS-Tools and Experiments Workshop*, 2010.

[11] C. T. S. Ranise. The SMT-LIB standard: Version 1.2. Technical report, Department of Computer Science, University of Iowa, 2006.

[12] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[13] G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.

[14] B. W. Weide, M. Sitaraman, H. K. Harton, B. M. Adcock, P. Bucci, D. Bronish, W. D. Heym, J. Kirschenbaum, and D. Frazier. Incremental benchmarks for software verification tools and techniques. In *Verified Software: Theories, Tools, and Experiments (VSTTE)*, pages 84–98, 2008.