



Embracing the Engineering Side of Software Engineering

Lionel Briand



I HAVE NOW been a professional researcher in software engineering for roughly 20 years. Throughout that time, I've worked at universities and in research institutes and collaborated on research projects with 30-odd private companies and public institutions. Over the years, I have increasingly questioned and reflected on the impact and usefulness of my research work and, as a result, made it a priority to combine my research with a genuine involvement in actual engineering problems. This short piece aims to reflect on my experiences in performing industry-relevant software engineering research across several countries and institutions.

Not So Hot Anymore

I suppose a logical start for this article is to assess, albeit concisely, the current state of software engineering research. As software engineering is widely taught in many universities, due in large part to a strong demand for software engineers in industry, the number of software engineering academics is substantial. The *Journal of Systems and Software* ranks researchers every year, usually accounting for roughly 4,000 individuals actively publishing in major journals.

When I started my career, software engineering was definitely a hot topic in academia: funding was plentiful, and universities and research institutes were hiring in record numbers. This clearly isn't the case anymore. Public funding for software engineering research has at best stagnated, and in many countries, declined significantly.

Hiring for research positions is limited and falls far below the number of software engineering graduates seeking research careers. Industry attendance at scientific software engineering conferences is roughly 10 percent, including the scientists from corporate research centers. Adding insult to injury, in many academic and industry circles, software engineering research isn't even considered to be a real scientific discipline. I'll spare you the numerous unpleasant comments about the credibility and scientific underpinning of software engineering research that I've heard over the years.

This situation isn't due to the subject matter's lack of relevance. Software systems are pervasive in all industry sectors and have become increasingly complex and critical. The software engineering profession repeatedly tops job-ranking surveys. In many cases, most of a product's innovation lies in its software components—for an example, think of the automotive industry. In all my recent industry collaborations, I've observed that all the issues and challenges traditionally faced in software development are becoming more acute.

So how can we explain the paradox of being both highly relevant and increasingly underfunded and discredited?

Looking for Some Answers

Like other disciplines before us, because we're a young and still-maturing engineering field, we lack the credibility of more

continued on p. 93

continued from p. 96

established disciplines. After all, even the term *software engineering* was first coined in 1968.

But surely, there's more to it than that, and we, as a research community, must take some of the blame. Engineering is about the "application of scientific and mathematical principles to practical ends" (*American Heritage Dictionary*). In our case, the scientific disciplines of reference include not only computer science but also certain areas of discrete mathematics and operation research, statistics, psychology, and economics. Because software development is tightly coupled with economic considerations—with an overarching effect on all project decisions—and software development is still largely performed by humans, the latter two disciplines shouldn't come as a surprise.

How about the "practical ends" part of the engineering definition? Bertrand Meyer stated in one of his recent blog posts, "academic research has had its part, honorable but limited" (<http://bertrandmeyer.com/2010/04/>), referring to the impact of software engineering research on practice. Many others have made such comments over the years, and I tend to agree with them, based on my observations of software development practices.

The "impact project," launched by ACM SIGSOFT, aimed to demonstrate the (indirect) impact of software engineering research through a number of articles by research leaders. Although some impact can certainly be credited to research, I've talked to many of my engineering colleagues, and I've never heard of another engineering discipline trying to demonstrate its impact through such an initiative. This in itself is a symptom of a lack of impact as the benefits of engineering research should be self-evident. Of course, I'm not suggesting that all the research in other engineer-

ing disciplines bears fruit in the form of industrial applications—for every industrial success achieved through research, there are many failures, irrespective of the discipline. That said, software engineering research is not yet on par with other engineering disciplines in terms of industrial success stories. This is clearly visible to public funding agencies, which, after years of massive investments in software engineering research, have seen little return. It's also clearly perceptible from the many reactions that I've witnessed when discussing collaborations with practitioners.

Root Causes

There are several root causes for the limited impact of software engineering research. I'll cover what I believe are the main culprits. It's fair to say that the field has many highly educated and competent researchers. There's no reason to doubt the ability of individual researchers in this community.

However, does academia—where most researchers are employed—value research impact? We have to admit that in most computer science departments, to which most software engineering researchers belong, this isn't the case. People are typically evaluated based on their number of publications in high-quality venues and acquired funding. In contrast, in other engineering faculties, factors such as filed patents and industry collaborations and impact are more highly regarded. In fact, many engineering faculty members I've met, across several disciplines, see themselves primarily as inventors. It should therefore not be surprising that when under pressure, software engineering researchers focus on what they're rewarded for.

Another related issue is that the paradigm of research in engineering is somewhat different from that in natural sciences or applied mathematics. Engineering research must be problem-

driven, account for real-world requirements and constraints, address scalability and various human factors, and ensure that the end result hits the right trade-offs, for example, between quality and cost. This has significant implications as it isn't possible to follow such a paradigm without a thorough understanding of the challenges in practical settings and therefore without some form of collaboration with actual software development organizations. Some aspects of software engineering research are more theoretical in nature, but even these are initially based on formalizing the problem or solution for analysis. However, the largest proportion of software engineering papers needs to address these engineering research concerns, and such papers are much fewer than they should be in our research community.

One problem in promoting an engineering vision of software engineering research is the field's relative immaturity. Most institutions don't have software engineering departments; software engineering research tends to be part of the computer science departments, often in science faculties. Just imagine mechanical or civil engineering being part of a physics department. Would that work? No wonder many software engineering researchers find it difficult to perform high-impact research—they have to comply with computer science expectations in terms of research contributions.

A second problem is that a typical university department isn't an ideal environment for establishing tight collaborations with industry or public institutions. Furthermore, such collaborations can't involve just students and professors—they also require professional scientists and engineers in charge of tool development, knowledge transfer, and project management. This is why my current employer (University of Luxembourg) has created an interdisciplinary, cross-faculty center focusing on

system dependability. There are many requirements for such an initiative to be successful, which I can't discuss here due to space limitations.

Examples

Two brief examples illustrate my points, both negative and positive. First, over the past decade, a very large number of papers have been dedicated to debugging, for example, by ranking statements in programs. At the International Symposium on Software Testing and Analysis (ISSTA 2011), Chris Parnin and Alex Orso reported on a survey and study they performed on that subject. From 50 years of research on automated debugging techniques, they found that only five papers involved studies with real programmers.

The authors' experiment also showed that only low performers strictly followed the provided statement ranking, only one programmer out of 10 stopped when checking the buggy statement, automated support didn't speed up debugging, and programmers preferred an explanation rather than recommendations on fault locations. How could such a substantial research endeavor—one that spanned several decades—be misled to such an extent? The human factor was abstracted away from most of the research, and the research community had a rather superficial understanding of the problems facing practitioners while debugging. Nevertheless, literally dozens of published papers reported solutions that were a mismatch to the problem.

As a positive example, I use a project in which my colleagues and I focused on an application of model-based testing in close collaboration with an industry partner (most recently reported in a *Transactions on Software Engineering and Methodology* article by Hadi Hemmati and colleagues; <http://simula.no/publications/Simula.simula.120>). Our focus was on automating the testing of a video-conferencing system, with a particular focus on testing its robustness to network and hardware problems. We developed a sophisticated automation strategy that satisfied the requirements as specified. However, as our industry partners applied our tool, we learned that it tended to lead to too many test cases given the allocated test time. We had overlooked the need

ADVERTISER INFORMATION • JULY/AUGUST 2012

ADVERTISER

ABB AB
ICSE 2013
Shell
SPLC 2012

PAGE

15
74
7
Cover 2

Email: a.schissler@computer.org, d.schissler@computer.org
Phone: +1 508 394 4026; Fax: +1 508 394 1707

Southwest, California:
Mike Hughes
Email: mikehughes@computer.org
Phone: +1 805 529 6790

Advertising Personnel

Marian Anderson: Sr. Advertising Coordinator
Email: manderson@computer.org
Phone: +1 714 816 2139 | Fax: +1 714 821 4010

Sandy Brown: Sr. Business Development Mgr.
Email: sbrown@computer.org
Phone: +1 714 816 2144 | Fax: +1 714 821 4010

Southeast:
Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 585 7070; Fax: +1 973 585 7071

Advertising Sales Representatives (display)

Central, Northwest, Far East:
Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742; Fax: +1 888 886 8599

Northeast, Midwest, Europe, Middle East:
Ann & David Schissler

Advertising Sales Representatives (Classified Line)

Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 585 7070; Fax: +1 973 585 7071

Advertising Sales Representatives (Jobs Board)

Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 585 7070; Fax: +1 973 585 7071

to access a specific testing infrastructure to perform system testing (for example, to simulate IP network traffic). In other words, what was most important to them was the ability to adjust the amount of testing—regardless of the test strategy—to the test infrastructure’s available access time. We devised a solution based on similarity measurement of test cases and found an effective way to satisfy this requirement. But we never would have thought of focusing on that problem without interacting with this industry partner. This also shows how, to a large extent, context matters in our research discipline as it greatly affects a solution’s applicability. Researchers must therefore show due diligence in understanding relevant contextual factors, an activity that’s both time-consuming and essential in software engineering research.


A larger proportion of software engineering research should focus on solutions to real engineering problems. But this requires first and foremost a conscious effort by our research community to understand the problems and priorities of the many industry sectors that develop software. Such an understanding can come only through a closer collaboration; it requires a change of organization and

research paradigms in many academic institutions, as well as mechanisms to reward industrial impact and not just scientific publishing. Note that closer interactions between academia and industry don’t prevent high-risk, long-term research—rather, they ensure that such endeavors are, to the extent possible, rooted in a thorough understanding of the reality of software development practice.

Software engineering isn’t a branch of computer science; it’s an engineering discipline relying in part on computer science, in the same way that mechanical engineering relies on physics. One possibility, already implemented in some institutions, is to create “system engineering” departments, with various faculties contributing to system design and verification, including software at various levels, electronics, and mechanical areas.

As software engineering researchers, we also need to work on ourselves. In particular, we should stop seeing ourselves as computer scientists. Instead, we should place more value on the application and evaluation of new technologies in realistic contexts and on the combination of techniques from multiple disciplines to solve well-defined engineering problems. There are signs of progress in this regard. One noticeable change is that most conferences

now have “application” tracks (under various names), which are essentially engineering research tracks. Several major conferences also now have an interdisciplinary focus, including SSBSE (evolutionary computing and optimization) and ESEM (empirical studies and human factors).

Whether we successfully address the challenge of transforming ourselves into a true engineering research discipline will determine our impact and therefore the success of our profession in the future. We owe this to our students and the society at large, which is financing our research. 

Acknowledgments

I’m grateful to the following people for their reviews and valuable discussions: Victor Basili, Davide Falessi, Arnaud Gotlieb, Jacques Klein, Philippe Kruchten, Sandro Morasca, James Miller, Richard Torkar, Mehrdad Sabetzadeh, and Tao Xie.

LIONEL BRIAND is professor and FNR PEARL Chair at the University of Luxembourg’s Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT). Briand is an IEEE Fellow and recently received the IEEE Computer Society Harlan Mills award for his work on model-based testing and verification. Contact him at lionel.briand@uni.lu.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720-1314; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscription rates: IEEE Computer Society members get the lowest rate of US\$56 per year, which includes printed issues plus online access to all issues published since 1984. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: \$20 for members, \$193 for nonmembers (plus shipping and handling).

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the

copy; and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post the accepted version of IEEE-copyrighted material on their own webservers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy. An accepted manuscript is a version which has been revised by the author to incorporate review suggestions, but not the published version with copyediting, proofreading, and formatting added by IEEE. For more information, please go to: http://www.ieee.org/publications_standards/publications/rights/paperversionpolicy.html. Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2012 IEEE. All rights reserved.

Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.