On the Worthiness of Software Engineering Research

Antony Tang[1]

Swinburne University of Technology

Rick Kazman

University of Hawai'i

1. Introduction

The "Impact Project" tried to assess the impact software engineering (SE) research has had on software engineering practice [1]. Some of its findings indicated that (1) software engineering research has significantly affected SE practice; (2) lasting impacts require ongoing interactions between research and practice; (3) research impact might not be felt for at least 10 years. Whilst this study reflected the bright side of SE research, there is also a not-so-bright side.

Thousands of SE research papers are published in SE conferences and journals each year[2]. However, feedback from SE practitioners indicates that much of this research is not useful to them. In this article, we discuss the motivations behind SE research. We note issues that prevent research outputs from becoming more useful. We discuss the criteria for evaluating the worthiness of such research, from a software practice point of view. We hope that this article will start a frank discussion on the worthiness of SE research to practitioners and on directions for the future.

2. The Value of SE Research

Patterson et al. state that computation is synthetic in that many of the phenomena computer scientists study do not occur naturally [2]. Since anyone can invent something new, that alone does not establish a contribution or value. SE is an *engineering* discipline and an applied science. Hence the value of its research must be judged by benefits to practitioners. We use the Redwine-Riddle Maturation Model [3] to analyze two scenarios, and how value is passed between researchers and practitioners (see Figure 1). Value, in the Redwine-Riddle Model includes knowledge-based systems, metrics, programming, software principles, methodologies and techniques.

[1] Email addresses: atang@swin.edu.au (Antony Tang); kazman@hawaii.edu (Rick Kazman)

[2] We counted 85 conferences from http://web.engr.illinois.edu/~taoxie/seconferences.htm. A quick browse of DBLP (http://dblp.uni-trier.de/) yields many more conferences that can be classified as software engineering.

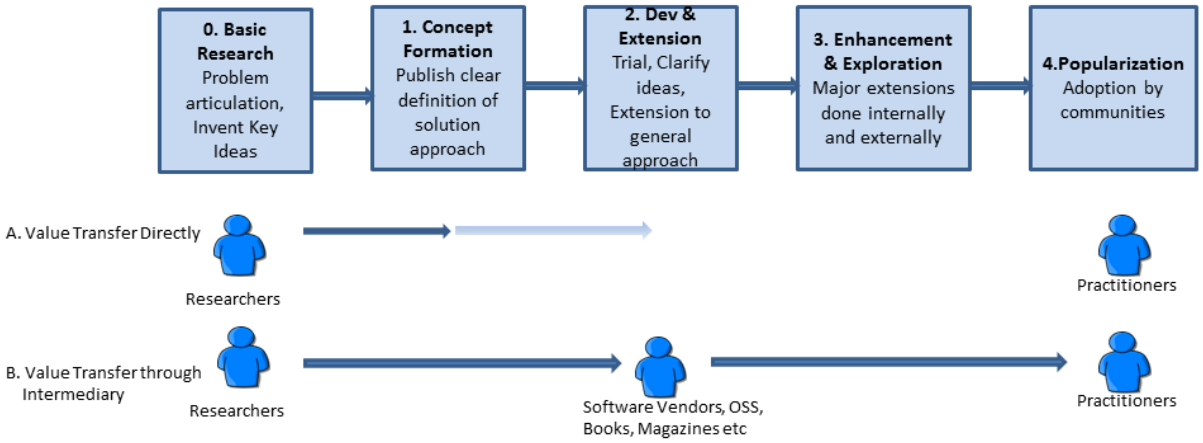

Figure 1 – Value Transfer between SE Researchers and Practitioners

This framework of software technology maturation [3] suggests that SE research should go through phases of (0) Basic Research; (1) Concept Formulation; (2) Development and Extension; (3) Enhancement and Exploration; (4) Popularization and Propagation. We map SE researchers and practitioners into this model to understand where and how research value is passed.

Successful SE research—that positively influences the community—is popularized and transitioned through working with intermediaries and practitioners, by creating tools, publications, training, workshops etc. (Scenario B in Figure 1). Several innovations over the past two decades have profoundly affected practice (Agile methods, cloud computing, relational/NoSQL databases, IDEs, the WWW, DevOps, SOA, open source, design patterns, etc.) and have had some of their roots in academia. However, much SE research is never communicated to or used by practitioners (Scenario A in Figure 1). For example, in a study on ADLs (architecture description languages), researchers found that most have no practical applications at all [4].

3. Gaps between SE Research and Practice

We conducted a survey of software practitioners; we recruited participants from a SATURN[3] workshop, industry software architecture courses and personal industry contacts and received 61 responses. We asked them if they read any "research papers" from SE conferences, or journals in the last 3 years. Out of these 61 respondents, only 19 (31%) answered in the affirmative. Amongst those 19, only 4 could name a true research journal or conference; the other 15 named trade publications. This survey shows that practitioners are disconnected from research publications. What are the implications? First, practitioners do not actively seek out academic research results and the majority of the research results that SE researchers produce are not communicated to practitioners; second, there are few venues where SE practitioners and researchers meet to exchange ideas. In a public forum, the first author asked why SE practitioners do not work with researchers. The answer was: "Professors are interested in publishing papers only. They don't understand real-life problems." Similarly, Dijkstra observed that software engineering often does not serve its supposed purposes [5].

[3] SATURN (SEI Architecture Technology User Network conference) is an annual practitioner-oriented conference focused on transitioning software architecture research into practice.

4. Why Is There a Gap between SE Research and Software Practice?

Research in the natural sciences relies on experiments, measurements and repeatability. However, many SE studies take place in complex, unique environments making testing, measurement, and repeatability difficult. Human factors add to the variability of the results, making it even harder to draw conclusions with confidence. We observe the following issues:

1. **SE Researchers Lack Understanding of Industry's True Needs** – many researchers who emerge from graduate programs have no exposure to commercial software development. They have little understanding of the issues faced by software developers in the real world.
2. **Unrealistic Assumptions** - SE researchers often fall into the trap of making implicit, unrealistic assumptions. Researchers often fail to formulate appropriate problems that need studying (Phases 1 and 2).
3. **No Knowledge of Existing Industry Solutions** – often SE researchers have little knowledge of existing solutions in the marketplace, and may be unaware of current industrial practices (Phases 2 and 3).
4. **SE Research Incentive Misalignment** – SE researchers are incentivized to publish ever-more papers and garner citations due to promotion and tenure concerns. The Guardian reported: "The blame for this sad situation [peer-reviewed publishing] lies with the people who have imposed a publish-or-perish culture, namely research funders and senior people in universities. To have `written' 800 papers is regarded as something to boast about rather than being rather shameful." [4] A similar comment was made by Vardi [6]: "if 10 papers are better than five, then surely 15 papers are better than 10".

As a result, SE research suffers from the following issues. **Irrelevance** – a research output has little relevance to software practice. **Disregarding practical constraints** – there are real-world constraints that could make the proposed research impractical. **No understanding of the application context** – some SE researchers invent solutions without considering whether they can be used in a real-world context. **No attempt to evaluate costs, benefits, and risks** – an engineering solution is useful if the benefit gained is significantly greater than the cost. There is little discussion in research of the full costs of a technique (including acquisition, training, and disruption to existing processes). There are seldom arguments made as to why the benefits are significantly greater than the costs and how this technique will reduce project risk. **No attempt to evaluate the usability** of the research. Often the only users of the technique are the researchers themselves, and they have made scant attempt to empirically evaluate the technique by practitioners.

5. SE Research Evaluation Criteria (5Cs)

We suggest using 5 criteria (5Cs) to help SE researchers align their research goals with practical needs, based on previous ideas such as [7]:

- **C1: Relevance** - *What is the relevance and significance of this research?* A researcher should specify the conditions under which the research would be meaningful, such as the conditions when the tools or methods could be used.

[4] Article from the Guardian - http://www.theguardian.com/science/2011/sep/05/publish-perish-peer-review-science

- **C2: Practicality** - *Does a method/technique solve or shed light on a practical software development issue of significance? Have people other than the inventor(s) been shown to successfully use the method/technique?* In this criterion, we suggest that a researcher needs to understand and justify the practicality of their research.
- **C3: Novelty** - *Does this research duplicate existing solutions?* Researchers need to justify that new and novel solutions are not minor variations of each other, or minor variants of what exists in commercial products.
- **C4: Beneficiaries** - *Who would benefit from this work, and how does the research have a positive impact on the beneficiaries?* SE researchers need to specify the key stakeholders who could benefit from their research.
- **C5: Costs and Benefits** - *What are the costs and benefits of applying a method/technique? Is it measureable?* The benefits of the research must significantly exceed the costs when being applied.

We realize that not all criteria are applicable to all research. For instance, some research aims at understanding software development phenomena, therefore cost-benefit criteria does not apply. And some SE research may be theoretical, and may not be *directly* applicable by practitioners.

6. Conclusion

There are gaps between SE research and practice. Our current culture does not focus on how SE research can aid *practice*. We need to shift this culture. As Vardi discussed [6], scholarly inflation is detrimental to computing research. SE researchers need to take a serious look at their own research, and evaluate the long-term value they are creating for the software development community. This is essential for the research community to thrive in the long term: we must provide *value* to software practice, and not be content with paper and citation counts. Here are a few simple steps that we can take:

- Get more experience in the software industry. Researchers can spend some of their time embedded in software development teams or have some years of software development experience before undertaking research roles.
- Put more emphasis on empirical research in collaborations with industry partners.
- Consider how to evaluate worthy research, using the 5Cs to assess impact on the community.
- Improve communication and knowledge transfer to software developers. Whilst we report our results in a scientific manner, we should also translate them into lay language and disseminate them to practitioners.

We welcome the Computing Research Association's advice to hiring units to focus on *quality* and *impact* [8]. But we fear that "paper counting" still prevails. There are challenges in changing the culture and this will not happen overnight. We hope that this article will germinate meaningful discussion and honest reflection to help us improve this situation.

7. References

[1] L. Osterweil, C. Ghezzi, J. Kramer, and A. Wolf, "Determining the impact of software engineering research on practice," *Computer,* 39-49, 2008.

[2] D. Patterson, L. Snyder, J. Ullman, "Best Practices Memo Evaluating Computer Scientists and Engineers For Promotion and Tenure," *Computing Research News, Computing Research Association,* vol. September 1999, A-B, 1999.

[3] S. Redwine, W. Riddle, "Software technology maturation," Proc. of the 8th International Conference on Software Engineering, 1985.

[4] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering,,* vol. 39, 869-891, 2013.

[5] E. W. Dijkstra, "On the cruelty of really teaching computing science," *Unpublished manuscript EWD,* 1036, 1988.

[6] M. Vardi, "Incentivizing quality and impact in computing research," *Comm. ACM,* vol. 58, 5, 2015.

[7] M. Shaw, "What makes good research in software engineering?," *International Journal on Software Tools for Technology Transfer,* vol. 4, pp. 1-7, 2002.

[8] B. Friedman, F. Schneider. (2015). *Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion*. Available: http://cra.org/resources/bp-view/best_practices_memo_evaluating_scholarship_in_hiring_tenure_and_promot/