Experimental Software Engineering: A Report on the State of the Art

Lawrence G. Votta
Software & Systems Research Lab
AT&T Bell Laboratories
Naperville, Illinois 60566
votta@research.att.com

Adam Porter
University of Maryland
Computer Science Department
College Park, Maryland 20742
aporter@cs.umd.edu

1 Introduction

Computer science and software engineering are relatively new disciplines when compared to other fields like physics and electrical engineering. And as with many new communities that are growing and defining themselves at the same time, there exist many growth spurts that strengthen parts of the communities interest and at the same time illuminate weaknesses in other parts. Periodically it is worthwhile to pause and reflect on the current state of the art. So, where are we with experimental work that is relevant to large software development?

We argue that there are now tremendous opportunities to improve our knowledge of software engineering for large software systems. We also believe that both professional software developers and researchers have important roles to play in getting that information! Professional developers need to reduce *cost* and *delivery* time and improve *quality* — these are excellent problems requiring creative solutions. Researchers need to work with developers to get data and encode it into knowledge whose validity can be tested.

Figure 1 depicts a model of how science, technology, and standard practice relate to each other. Thomas Allen developed this model to explain how the work of engineers and scientists relates to one another. What is important for us to realize here is that there is common ground for both developers and researchers to work together and extend the state of the art.

The goal of this session is to make the software engineering community aware of the opportunities that exist to pursue such an experimental approach. In the remainder of the essay, we describe an emerging model for empirical work and the language for discussing it. We then focus on the current state of experimental software engineering, the road blocks barring effective progress, and what developers and researchers can do to remove them.

2 Models of Empirical Work

Our goal is simple to state: we want a *credible* empirical basis for our knowledge of software engineering that is relevant to both large software developments and that is of value to both developers and researchers. At present, the existing models of empirical work found in the physical and social sciences are insufficient for the problems that we encounter. The physical sciences are inadequate due to the attendant social factors from both developers and sometimes the final customer. The social sciences are also inadequate because they can't economically get data by exploiting the non-social elements (that is, the technical) of an experimental context. We claim that a hybrid approach is necessary.

Instead of describing this hybrid approach completely, we bound it and specify some of the properties the model must have. For example, the dimensions that the model must consider are

- individual versus groups;
- student versus professional software developers; and
- *in vitro* versus *in vivo* studies (that is, a controlled environment versus 'the way it really happens').

Further, any model of empirical work must have *credibility* as its cornerstone.

The degree of credibility that empirical work possesses depends on its validity. *Internal validity* denotes the property of an empirical study where the result is consistent within its local context. *External validity* denotes the property where the result is generalizable to other contexts. One of the most common techniques used to establish the validity of an experiment's results is to repeat the experiment — in the local context to establish internal validity and in multiple contexts to establish external validity.

Finally, not all studies have the same level of credibility; nor do they yield the same depth or breadth of knowledge. For instance, anecdotal studies just record what happened in one specific context at one particular time. Case studies attempt to at least show a *correlation* between an independent variable(s) and

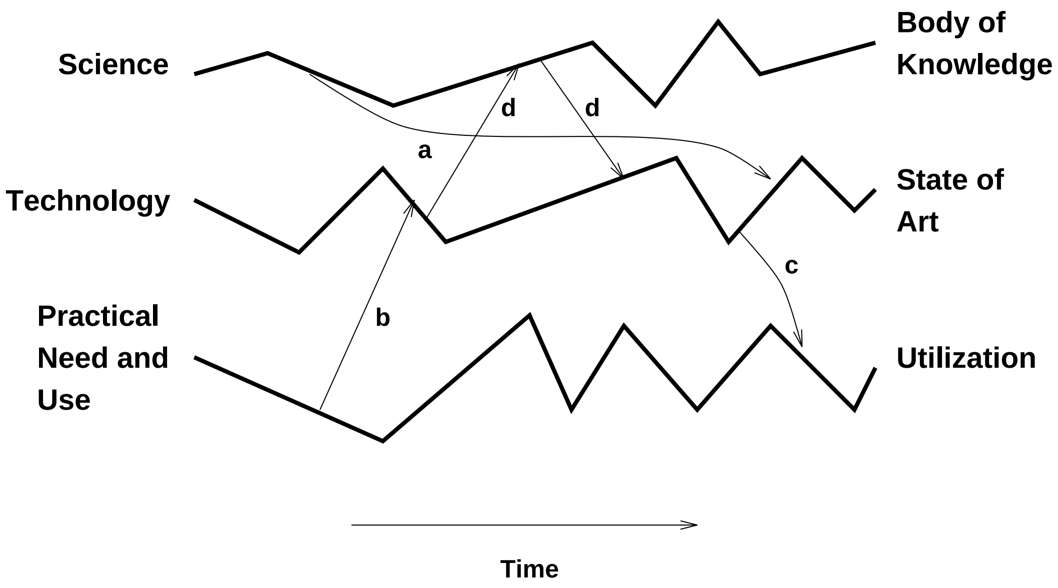

Figure 1: **Science, Technology, and the Utilization of Their Products.** The communication paths among the three streams of knowledge are shown for different transfers or needs. (a) The normal process of assimilating scientific results into technology. (b) Recognized need for a device, technique, or scientific understanding. (c) The normal process of adopting technology for use. (d) Technological need for understanding of physical phenomena and its response. (Reprinted from T. Allen, *Managing the Flow of Technology*, page 54).

dependent variable(s). True experiments attempt to show *causality*. To show causality between events A and B, in addition to showing correlation, one must also show that the correlation between A and B is not spurious, that A occurs before B (temporal precedence), and that there exists a constructive, testable theory that explains why A causes B. Figure 2 depicts the different types of studies and their relation to the above issues.

3 State of Affairs

The amount and quality of empirical studies of large software developments are sorely lacking. In particular, there are few studies reporting what people actually do when they do software development; rather, most papers dwell on what their authors thinks people should be doing. This is an important distinction. The July 1994 special issue of the *IEEE Software Magazine* devoted to measurement based process improvement has only one paper reporting how and what people do in large software development; the rest in one way or the other try to tell people what they should be doing – with mostly anecdotal evidence to support their positions.

But not all is lost, for many years now a wide variety of studies examining how individuals react to specific tools and techniques have been performed. This area is well established and has proven valuable to the entire computer science and software engineering community. Notwithstanding Bill Curtis' strong warning against using student subjects, the part of the community studying individual programmers has built good models of subject validity (when students and professional behavior would be the same or depart from each other in known, predictable ways).

4 Roadblocks

What are the blocks to forming a credible empirical study of large software developments? It is hard to know where to start! However, there are several things we need to change about our research paradigms and how we work as a community. The current blocks we see to making progress are listed below.

1. Repeated experiments are not valued as important research contributions.

2. Many theories are not testable. If there is no way to test a theory, then there is no way it can be compared against alternative theories.

3. There is poor synergy among computer science, software engineering, and software development enterprises.

4. Few failures are ever reported in the computer science and software engineering refereed or popular literature.

5. The proprietary mantle is invoked far too often and prevents the timely publication of important information about how certain tools or techniques worked, or did not work in their environment.

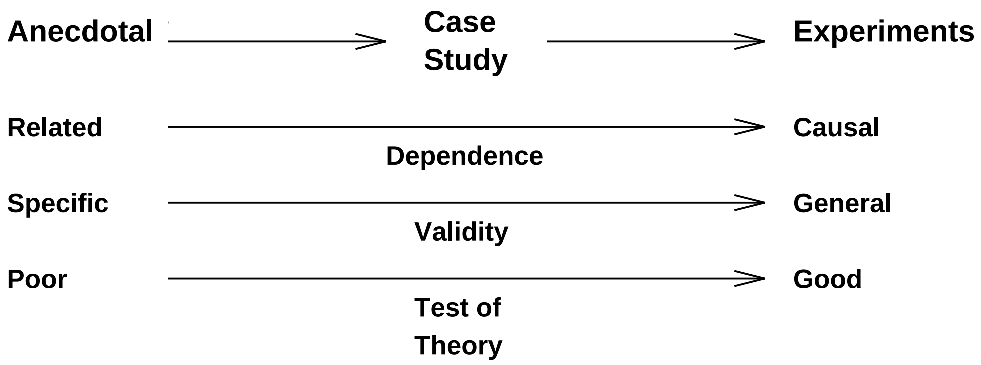

Figure 2: **Spectrum of Empirical Work** The plot depicts the range of empirical work along three important axes; dependence, validity, and test of theory. As you move from left to right these properties change as indicated.

5 Next Steps

The community of developers and researchers needs to take some serious steps towards improving the field's ability to support credible empirical work. We list five steps below that we believe will help set the community on the right path.

1. The community must accept the need to repeat experiments and to publish the results of the replicated experiments regardless of whether they agree or disagree with previous results.

2. When experiments disagree, the community needs to encourage the search for explanations.

3. The research community must reach a consensus about theories that are not testable empirically; this set of theories should not be allowed.

4. A model of the differences between student and professional software developers in organizational settings must be constructed so that the validity of student studies can be understood. This will serve as a major force in driving collaboration between industry and academia.

5. We must explore alternative, cheaper and safer ways of conducting experiments (for example, simulation, efficient data collection, and recalibration).

6. We must have more and better access to real project data — both successes and failures.

7. We must recognize that empirical work is important and is necessary for a successful discipline.

Several groups of researchers and developers have recognized the need for this cultural change in the computer science and software engineering community. One early example of this close collaboration is the NASA Goddard Software Engineering Laboratory. More recently, the *International Software Engineering Research Network*, ISERN (see World Wide Web page *http://uomo.informatik.uni-kl.de:2080/AG.html* for details), has been formed with a charter specifically chosen to show how many of the above steps can be done and to provide an infrastructure for developers and researchers to create a credible empirical field of study for large software developments.

In passing, we would like to thank Mark Bradac, Dewayne Perry, Nancy Staudenmayer, and Mary Zajac for many ideas, useful discussions, and support of this work.