

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

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

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

Dewayne Perry
Software & Systems Research Lab
AT&T Bell Laboratories
Murray hill, NJ 07974
dep@research.att.com

1 Introduction

Computer Science and software engineering are relatively new disciplines when compared to other fields like physics and electrical engineering. And as with most 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 relevant to large software development?

We argue that there are now tremendous opportunities for improving the knowledge of software engineering relevant to large software systems. Both professional software developers and researchers have important roles to play in getting that information! Professional developers need to improve *cost*, *quality*, and *delivery time* — these are excellent problems requiring creative solutions. Software engineering researchers need the empirical data from professional developers to focus their research and subsequently encode into knowledge whose validity can be understood.

Figure 1 depicts a model of how science, technology, and standard practice relate to each other [Thomas. J. Allen. *Managing the Flow of Technology*, MIT Press, Cambridge MA, 1977. page 54]. Thomas Allen developed this model to explain the relationship of the work of engineers and scientist to one another. The model depicts how the community work and is hence relevant to us. What is important for us to realize here is that there is common ground for both development professionals and researchers to work together and extend the state of the art.

The goal of this session is to make the software engineering community conscious of the opportunities that are possible in pursuing such an experimental approach. In the remainder of the essay, we describe an emerging model for empirical work and the language with which to talk about that work. We then focus on the current state in experimental software engineering, the road blocks to effective progress, and what professional software developers and researchers should do next.

2 Models of Empirical Work

Our goal is simple to state: we want a *credible* empirical basis for the software engineering relevant to both large and small software developments that is of value to both professional developers and researchers. At present, neither the physical sciences nor the social sciences models of empirical work are sufficient for the problems that we encounter. The physical sciences lack adequacy due to the attendant social factors from both professional developers and in some cases the final customer. The social sciences lack the economical use of the data from exploiting many of 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 of software developers;
- 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 as it’s cornerstone — *credibility*. The degree of credibility depends on the validity of the empirical result. *Internal validity* denotes the property of an empirical study where the result is consistent within it’s 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 are of the same level of credibility; nor do they yield the same depth or breadth of knowledge. For instance, anecdotal studies just record what happen in one specific context at one particular time. Case studies attempt to at least show a *correlation* between an independent variable(s) and dependent variable(s). Experiments attempt to show *causality*. To show causality between events A and

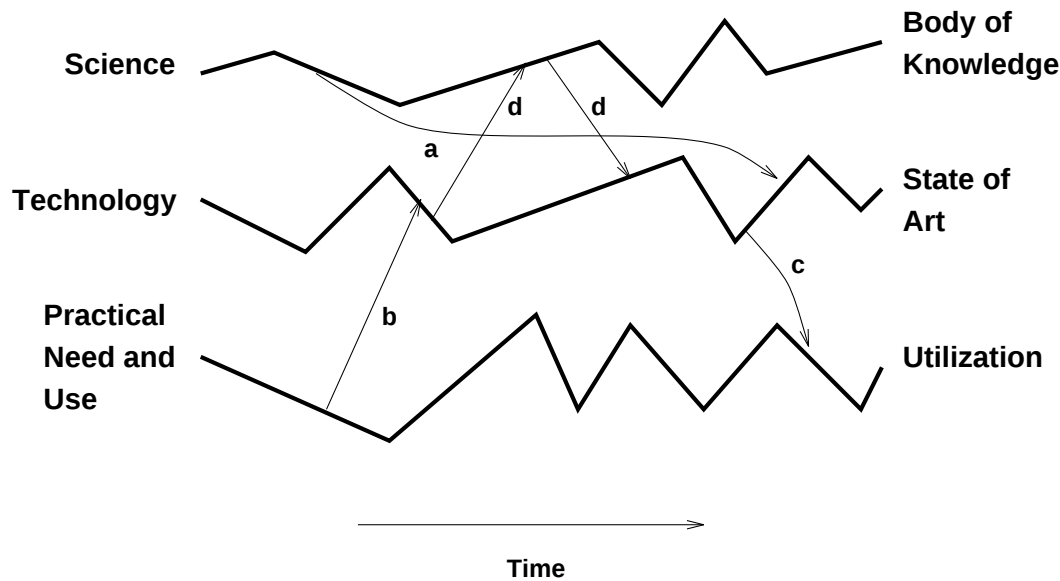


Figure 1: **Science, Technology, and the Utilization of Their Products.** The communication paths among the three streams of knowledge are shown for different transfers or needs. (a) The normal process of assimilation of scientific results into technology. (b) Recognized need for a device, technique,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

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 theory that explains why A causes B. Figure 2 depicts the different types of studies and their relation 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 do when they do software development; rather, most papers dwell on what 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.

But not all is lost, for many years now the studies of individuals and how they react to specific tools has been studied with many different techniques. This area is well established and has proven valuable to the entire computer science and software engineering community. Notwithstanding Bill Curtis' strong warning of the validity of using student subjects, the individual empirical studies community 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) [W. Curtis, "By the way, did you study any real programmers?" *Empirical Stud-*

ies of Programmers, 1986, pages 256-262].

4 Roadblocks

What are the impediments to forming a credible experimental 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 best work as a community. The current blocks I see to making progress are listed below.

1. Repeated experiments are not valued as important contributions to research.
2. Many theories are not testable. If there is no way to test a theory, there is no way to determine whether it is a valid one or not. We should, as the physicists have done, limit ourselves only to testable 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.

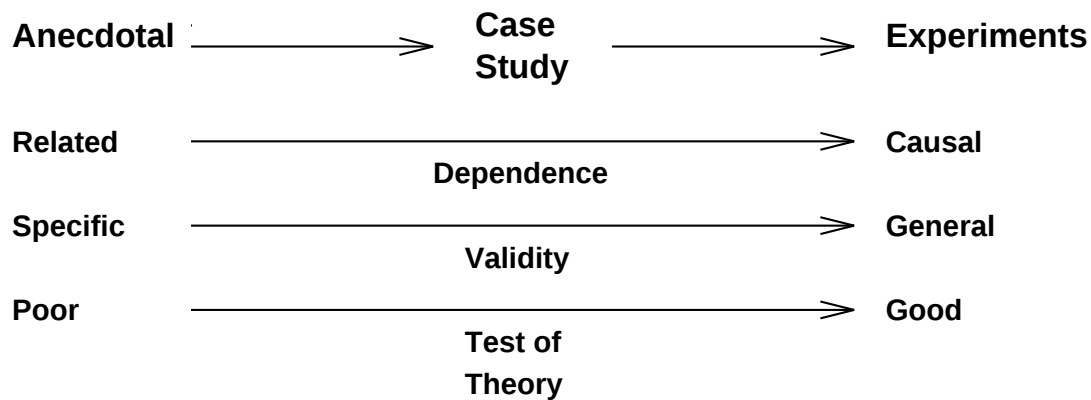


Figure 2: **Spectrum of Empirical Work** The plot depicts the range of empirical work along three important axis; dependence, validity, and test of theory. As you move from left to right these properties change as indicated.

5 Next Steps

The community of professional software developers and software engineering researchers need to take some serious steps forward to improve the field's ability to support credible empirical work. We list five steps below that we believe will 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. The research community must reach a consensus about theories that are not testable empirically; this set of theories should not be allowed.
3. 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.
4. We must explore alternative, cheaper and safer ways ways of conducting experiments (such as, for example, simulation, efficient data collection, and re-callibration).
5. We must have more and better access to real project data — both successes and failures.
6. We must recognize that empirical work is important and is necessary for a successful scientific and engineering discipline.

Several small communities of researchers and software development professionals have recognized the need for this cultural change in the computer science and software engineering community. The first example of this close collaboration is the NASA Goddard Software Engineering Laboratory [V. R.

Basili, "Software development: a paradigm for the future", *COMPSAC*, T59, September 1989, pages 471-485]. More recently, the *International Software Engineering Research Network*, ISERN (see World 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 professional software developers and researchers to create a credible empirical field of study for large software developments.