The Past, Present, and Future of Experimental Software Engineering

Victor Robert Basili[1]

[1]Department of Computer Science
University of Maryland
4111 A.V. Williams Building
College Park - MD - 20742 - USA
basili @cs.umd.edu

Abstract

This paper gives a 40 year overview of the evolution of experimental software engineering, from the past to the future, from a personal perspective. My hypothesis is that my work followed the evolution of the field. I use my own experiences and thoughts as a barometer of how the field has changed and present some opinions about where we need to go.

Keywords: empirical software engineering, experimentation, context variables, replication, meta-analysis, big science.

1. INTRODUCTION

For ISESE 2007, Claes Wohlin and José Carlos Maldonado asked me to take a 40 year perspective on the evolution of experimental software engineering, from the past to the future. That is an arduous task. So I decided to simplify the problem for myself by making this a personal perspective. My hypothesis is that my work followed the evolution of the field. So, I will use my own work and thoughts as a barometer of how the field has changed since I have been working in the field for the past 30 years and have some opinions about where we need to go.

I will map the changes across several variables: the kinds of studies that were being performed, the set of methods used, the nature of publications and the issues with review of the work, the community of researchers, the status of replications and meta-analysis, and the role of context variables.

This article will be organized in sections, each section representing a phase. Section 2 will cover the early days (1974 - 1985), running isolated studies for a particular purpose. Section 3 will focus on 1986 – 1999, building software process and technique knowledge in one domain and one environment. Section 4 will deal with 2000 – 2005, expanding out across environments and limiting the technologies being studied, and Section 5 will focus on 2006 and beyond, building knowledge about a domain.

2. PHASE I: THE EARLY DAYS (1974 – 1985)

These were the early days when people were running isolated studies for a particular purpose, independently using case studies and controlled experiments as the means to analyze a particular question of interest. The focus was on learning about measurement in general and trying to identify an appropriate set of metrics. Many of us were learning about running an experimental study, and the need for baselines as a basis for evaluation. There were

attempts to run a small number of controlled experiments but they were done mostly in isolation, not as part of a larger study.

Two isolated studies I was involved in were the Iterative Enhancement product evaluation [1] and the methodology evaluation [2]. The former was a case study where we used quantitative observations over time, measuring the product, and comparing the product with itself, using prior versions as baselines. This was a single isolated study. The latter was a controlled experiment analyzing the effects of a collection of methods centered on chief programmer teams, including structured design and structured coding. The experimental method applied was a replicated study (controlled experiment) with three treatments: teams using the methods, teams not using the methods, and single programmers, all performing the same task. The study was again a single study in a single environment.

The break from the mold of isolated studies was the Software Engineering Laboratory (SEL) [3] at NASA Goddard Space Flight Center where we began to build baselines of various project variables (defects, effort, project metrics), identifying where methods might make a difference. The focus moved to collecting data from live projects, feedback on data collection and measures, and the storage and analysis of large amounts of data. The work involved multiple projects and multiple methods in a single environment and domain.

We learned the importance of understanding the environment (context variables), the need to build our own models to understand and characterize that environment, the interaction of many variables and the need to model them, (e.g., the environment, projects, processes, products) and that data collection has to be goal driven and well defined.

This early work stimulated the more general recognition that experimentation and measurement were an important aspect of software development and that the design of experiments is an important part of improvement (something Demming had been preaching in manufacturing for many years [7], that evaluation and feedback are necessary for learning, and that we need to experiment with technologies to reduce risk and tailor to the environment, make improvements.

With respect to our variables, we were running studies mostly characterizing knowledge via measurement in a single environment and single domain. The *publications* mostly consisted of project studies and reviews were mixed. The *community of researchers* was almost empty and consisted of model builders and some scattered set of individual experimentalists. The *set of methods* for experimentation study was mostly quantitative, nonparametric, using nominal and ordinal measurement. The *context variables* were taken as a given, not measured. There was no *replication* or *meta-analysis*.

For the 10th anniversary of TSE (1986), Rick Selby, Dave Hutchens, and I defined a framework for experimentation in software engineering and wrote a state of the field paper recognizing that most of the papers in the literature dealt with either experimental studies of programmers in the small doing controlled experiments or data collection on projects in the large [4].

3. PHASE II: TYING STUDIES TOGETHER (1986 – 1999)

During this time there were attempts to tie studies together. Controlled experiments, case studies, quasi-experiments, qualitative analysis became part of a larger tapestry, each useful in their own right but for varying purposes. Controlled experiments were of value for identifying specific variable relationships while case studies provided the opportunity for scale up. We learned that you could reduce risk by running smaller experiments off-line using the mix of studies to build confidence in a theory based upon multiple treatments. The major focus was on measuring the relationship between process and product. However, in our field, the kinds of studies performed and the topics studied are dependent on the opportunities available.

This work stimulated the realization that we need to package and integrate our experiences (build models). Experience needs to be evaluated, tailored, and packaged for reuse so software processes must be put in place to support the reuse of experience. But the experience packages are local to the environment in which they are observed. Generalization is difficult.

With respect to our variables, *studies* were performed to package knowledge and build models to improve software quality based upon experience in an environment. Project experiences were easier to *publish* then experiments, as the view was that they were usually flawed by some threat to validity. ISERN (started in 1993) identified a *community of researchers* and began to help them interact. There was enough research going on to create the Journal of Empirical Software Engineering (started in 1996). The *set of methods* being used were mostly quantitative, nonparametric, some qualitative, and nominal and

ordinal measurement. But *context variables* were taken as a given, not fully recognized as an important set of influencing variables. *Replication* involved building some studies that varied the context, threats to validity; building knowledge across studies about a particular technology.

4. PHASE III: EXPANDING STUDIES ACROSS DOMAINS AND ENVIRONMENTS (2000 – 2004)

During this period we began to see the expansion of studies across domains and environments. There were several examples of building knowledge for a limited number of techniques in different environments and domains, i.e., studying the effect of context on techniques. One specific example was the NSF sponsored Center for Empirically Based Software Engineering (CeBASE) [5, 6, 9].

CeBASE made it clear that there is a great deal to do more research before we can comfortably build an empirical research engine that can be applied universally to evaluate and provide support for the use of evaluated methods. This research engine requires that we define and improve methods to

- Formulate evolving hypotheses regarding software development decisions
- Collect empirical data and experiences
- Record influencing variables (context)
- Build models (lessons learned, heuristics, patterns, decision support frameworks, quantitative models and tools)
- Integrate models into a framework
- Testing hypotheses by application
- Package what has been learned so far so it can be used and evolved

At this point we better understood that context can change everything and is hard to identify. This means that experimentation in software engineering represents **big science**, involving many researchers, many environments, and many domains. We won't evolve the knowledge base without collaboration. This implies we need to shrink the focus because collecting experience across environments, domains, and technologies, is very difficult. We need to build testbeds to study and mature the techniques for practice. These testbeds need to be maintained and evolve; an expensive proposition.

The focus needs to be on specifying the effects of technologies, and experimentally identifying the effects, limits and bounds of techniques. So, technologists need to be more specific about what their technologies do and do not do and we need to evolve empirical evidence about various techniques, gaining new confidence over time by better understanding the effects of influencing variables. We need to concentrate on building a body of knowledge based upon empirical evidence.

With respect to our variables, *studies* were performed to evaluate techniques in multiple contexts and define the relationship between user needs and what's available. Journal and conference *publications* have come to expect some form of analysis from new methods, even if it is only a feasibility study. The community of researchers continues to grow; experimentalists are replicating each other's studies. There are numerous *repetitions* of a few experiments. The *set of methods* available is a rich palate of tools: a full mix of qualitative and quantitative methods, controlled and quasi-experiments, case studies, surveys, folklore gathering, structured interviews and reviews, etc. *Context variables* are being studied and characterized. There are attempts to build knowledge across studies.

5. PHASE IV: NOW AND THE FUTURE

The focus has to be bounded, limiting the context but not artificially if possible. Ideally we can build bodies of knowledge about specific domain. Then we can combine what has been learned from these domains to build larger bodies of knowledge across domains. For each domain, this involves folklore gathering, interviews, case studies, controlled experiments, experience bases, etc. An example of this is the work being performed by the development time working group of the DARPA High Productivity Computing Systems project where the domain is high-end computing [8]. There is a specific practical focus: improving time and cost of developing high end computing (HEC) codes. There is a specific research focus: developing theories, hypotheses, and guidelines that allow us to characterize, evaluate, predict and improve how a HEC environment (hardware, software, human) affects the development of high end computing codes. There is a large research team consisting of MIT Lincoln Labs, MIT, UCSD, UCSB, UMD, USC, FC-MD, UH, MSU, UNL, and SDSC. Work is proceeding by evolving a series of studies with novices and professionals using controlled experiments (grad students), observational studies (professionals, grad students), case studies (class projects, HPC projects in academia), surveys, and interviews (HPC experts).

Testbeds vary from classroom assignments (Array Compaction, the Game of Life, Parallel Sorting, LU Decomposition, …) to compact Applications (Combinations of Kernels, e.g., Embarrassingly Parallel, Coherence, Broadcast, Nearest Neighbor) to full scientific applications (nuclear simulation, climate modeling, …). There is the beginnings of an experience base focused on empirical evidence and as well as one focused on the sub-domain of high end computing defects. There are experimental packages containing experimentation supports from checklist for instructors and experts running studies to instrumentation downloads and data collection and analysis packages.

This full scale attack is only possible because the domain is limited, the team size is big, and there is a fair amount of support.

5.1. THE JOURNEY

To recapitulate, early work characterized the effects of various methods, (*all study variables fixed*). Then we built baselines of various project variables (defects, effort, product and project metrics) for a single domain and environment, identifying where methods might make a difference (*fixed context, varied techniques*). e.g., ground support software at NASA/GSFC (SEL). Then we expanded out across several domains, environments, focusing on building knowledge for a couple of techniques (*fixed the techniques to study context*), e.g., defect removal techniques, COTS-based development, and agile methods (CeBASE). Then we did experimental work to elicit and quantitatively define the software dependability needs of various stakeholders, identify the appropriateness and effectiveness of technologies to satisfy those needs under varying conditions before transferring them into practice, (*introduced testbeds (context) to study techniques*), e.g., increasing the ability of NASA to engineer highly dependable software systems via new technologies (HDCP). Now we are working on building knowledge in a particular domain, packaging that knowledge in an experience base so it can be used by others, demonstrating the effectiveness of various approaches and in what context they are effective (*fixed domain, studying techniques and context variables*), e.g., building a software domain experience base to help understand and increase the time and cost of developing high end computing (HEC) codes (HPCS).

Where are we and where are we going?

5.2. KINDS OF STUDIES

With regard to the study of techniques, we need to begin with feasibility studies. No technique should be published without trying it out. The feedback should be used for improvement. Techniques need to be experimentally tested to see where they can be improved. We need to evaluate the bounds and limits of each technique and see how techniques can be integrated and what their integration buys you.

We need to build knowledge about the domain, identify folklore, theories, do ethnographic studies, interviews, observations, build models using grounded theory, case studies, quasi-experiments, controlled experiments, and evolve models supported by evidence. We need to test models and hypotheses via experiments of all kinds.

5.3. COMMUNITY OF RESEARCHERS

We have been evolving a community that talks to each other. This year was the 14th ISERN workshop and the number of members has grown dramatically. The Empirical Software Engineering Journal is 11 years old and has a very good ISI Impact rating (.965). The International Symposium on Empirical Software Engineering (ISESE) is in its fifth year and has been merged with the Metrics Symposium to form ESEM. But we need more of a community that works with each other more. Collaboration is necessary for defining a research agenda. There has been a variety of proposals for experimental guidelines but no consensus. We haven't even solved the terminology problem; everyone uses a different terminology.

5.4. PUBLICATION

With regard to publications, the guidelines that exist are very long, especially for conference papers. So there is a need to break studies into small useful modules, possibly backed up by technical reports that deal with all guideline issues or are backed up by web site material. Journals are better than conferences as publication targets due to the feedback and dialog that is associated with the review process. The community needs to identify conference guidelines and find ways to use various publication forms to create an integrated whole.

Papers need to build on prior work. There is now a lot more literature around. Partly due to the history of isolated studies, we do not have a good enough culture of reading, referencing, and assimilating existing material. For example, we have been criticized for lots of studies about "inspections" that doesn't seem to recognize or integrate with the past work. This is partly because the "inspection" community of researchers hasn't made it clear that there are many reading techniques, like many testing techniques, that need to be developed, evolved, evaluated, etc. As a community we have not always distinguished the method (inspection) from the technique (reading) so why should anyone else – we need to be more scholarly.

5.5. CONTEXT VARIABLES

This is the biggest problem. There are too many influencing variables and we do not even know what they are or how to measure for them or the extent of their influence. They range from subject experience, e.g., professional vs. student to multi-dimensional categories such as environment, domain, class of SE technologies applied (how many variables are hidden in these?). If we are to build knowledge – we need to limit some of these categories, like focusing on specific domains, classes of technologies, or environments, expanding out slowly, unifying across the differences.

5.6. REPLICATIONS AND META-ANALYSIS

Building theories requires replication, varying the threats, varying the artifacts, and varying the population. These studies require coordination, collaboration, and independence. It takes a team to run an experiment; it is hard to do it alone. It involves multiple groups, multiple disciplines. Once a basis has been formed, it requires a certain level of independence in the studies.

5.7. CONVINCING A SOFTWARE DOMAIN COMMUNITY

As stated earlier, working with a specific domain is our best bet at studying the effectiveness of techniques and building a body of knowledge. Look at the focused work of Barry Boehm, government agencies and contractors, Nancy Leveson, aeronautical engineering, and Elaine Weyuker, telephony software. They made great progress because of their focus. So, do we work with software engineers or with engineers in a domain?

6. CONCLUDING REMARKS

Experimentation in Software Engineering is here to stay. Software engineering techniques need to be studied experimentally if software engineering is to be anything other than a theoretical discipline. Many technology developers are already doing feasibility studies, although they may not be called that. They are trying out their methods to see if they work. They are not experimentalists and may not want to be experimentalists. They may not want to do what it takes to perform such studies, and leaving the experimentalists to test the bounds and limits of their techniques. But no experimentalist wants to run an experiment on a technique that has not been shown feasible. We need find a balance between what is expected of the theoretician and the role of the experimentation.

So, what is the role of the experimental software engineering community? We need to develop the experimental research engine, perform studies, work with theoreticians, developers, and domain experts, e.g., the HPCS project.

Is there a future for experimentation in software engineering? We have matured a lot in terms of the questions we ask, the types of studies we perform, and the development of a community. Software Engineering is "**big science**"; and experimentation is a necessary ingredient of any big science.

ACKNOWLEDGEMENTS

The work alluded to and referenced here was developed by many people collaborating as teams at the University of Maryland and its partner organizations. Members of the team are too numerous to mention and have varied over time.

REFERENCES

[1] V. Basili, A. Turner, Iterative Enhancement: A Practical Technique for Software Development, *IEEE Transactions on Software Engineering*, vol. 1(4), December 1975.

[2] V. Basili, R. Reiter, Jr., A Controlled Experiment Quantitatively Comparing Software Development Approaches *IEEE Transactions on Software Engineering*, vol. 7(3): 299-320 (IEEE Computer Society Outstanding Paper Award), May 1981.

[3] V. Basili and M. Zelkowitz, Analyzing Medium Scale Software Development, *in Proceedings of the Third International Conference on Software Engineering*, May 1978.

[4] V. Basili, R. Selby, D. Hutchens, Experimentation in Software Engineering, *IEEE Transactions on Software Engineering* vol. 12(7): 733-743, July 1986.

[5] B. Boehm and V. Basili, Software Defect Reduction Top 10 List, *IEEE Computer*, vol. 34(1): 135-137, January 2001.

[6] V. Basili and B. Boehm, COTS-Based Systems Top 10 List, *IEEE Computer*, vol. 34(5): 91-93, May 2001.

[7] W. Edwards Deming, *Out of the Crisis* (Cambridge, Massachusetts: MIT Press, Center for Advanced Engineering Study, 1986).

[8] L. Hochstein, T. Nakamura, V.R. Basili, S. Asgari, M.V. Zelkowitz, J.K. Hollingsworth, F. Shull, J. Carver, M. Voelp, N. Zazworka, P. Johnson, Experiments to Understand HPC Time to Development, *Cyberinfrastructure Technology Watch Quarterly*, vol.2(4A): 24-32, November 2006.

[9] J. Maldonado, J. Carver, F. Shull, S. Fabbri, E.Dória, L.Martimiano, M. Mendonça, V. Basili, Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness, *Empirical Software Engineering: An International Journal*, vol. 11(1): March 2006.