The Role of Controlled Experiments in Software Engineering Research

Victor R. Basili

1 The Experimental Discipline in Software Engineering

Empirical studies play an important role in the evolution of the software engineering discipline. They allow us to build a body of knowledge in software engineering that has been supported by observation and empirical evidence. These studies allow us to test out theories, identify important variables and to build models that can be supported by empirical evidence.

But what kinds of empirical studies should we focus on?

There are a variety of study types that can be performed. In many cases the type of study will depend on the circumstances. Much of what we do in the software engineering domain is opportunistic and we are often limited by the situation available.

I will use the word study here to mean the act of discovering something unknown or of testing a hypothesis. Studies can be driven by hypotheses or simply a need to understand. They can use quantitative or qualitative analysis, or a mix of both. They can vary from controlled experiments to quasi-experiments or pre-experimental designs, to simply observations. These latter studies tend to involve more of a qualitative analysis component, including at least some form of interviewing.

The type of study can be an experiment or an observation. On one end of the spectrum, studies can be cause-effect, allowing us to understand the effect of the independent variable on the dependent variable. Sometimes we can do a correlational study, where we can at least recognize that two variables are related. However, sometimes the best we can do is a descriptive study in which we can observe a particular situation. The subject may vary in experience from novice to expert. The experimental setting can be in vivo or in vitro. The type of analysis can be quantitative or qualitative or a mix of both.

The goals of our studies vary from trying to understand the effects of processes and techniques on products to evaluating the product characteristics themselves, from predicting cost and schedule to trading off environmental constraints. For example, when introducing any form of process, method or tool, the organization needs to evaluate its feasibility, usability, and effectiveness in context. These goals help the researcher build knowledge about the discipline as well as help the practitioner understand how to build software systems better.

The varied reasons for study, the immaturity of the discipline in terms of well formulated models, and the opportunistic nature of the circumstances requires a palate of many kinds of studies, types of analyses, settings, experience levels, etc. Each has it benefits and drawbacks and provides insight into our body of knowledge about software engineering. One of our biggest problems is how to combine them to form a coherent body of knowledge.

V. Basili et al. (Eds.): Empirical Software Engineering Issues, LNCS 4336, pp. 33 – 37, 2007.

Approaches vary in cost, level of confidence in the results, insights gained, balance between quantitative/qualitative research, etc. When we are studying a single team the costs are reasonable, we can observe large projects in realistic (in vivo) settings. Our analysis is more qualitative. Examples here include case studies and interviews. If we can study a population over time, we can use quasi-experimental designs. If we can study multiple teams performing the same activity the costs are high, the projects tend to be small, but we can gain statistical confidence in the results. Examples here include fractional factorial designs and pretest /post test control group designs. Thus the ideal is to be able to combine types of study to build the body of knowledge [1].

Thus all forms of study help contribute to knowledge.

2 Use of Controlled Experiments in Empirical Software Engineering

Controlled experiments offer several specific benefits. They allow us to conduct well-defined, focused studies, with the potential for statistically significant results. They allow us to focus on specific variables, measures, and the relationships between them. They help us formulate hypotheses by forcing us to clearly state the question being studied and allow us to maximize the number of questions being asked. Such studies usually result in well defined dependent and independent variables and well-defined hypotheses. They result in the identification of key variables and good proxies for those variables. They allow us to measure the relationships among variables.

So, for example, to identify which of two techniques are best for identifying a certain class of defects, a controlled experiment forces the specification of the treatment techniques, creates training materials.

The very act of defining a controlled experiment forces specification and provides many insights. In running a controlled experiment we are forced to state clearly what questions the investigation is intended to address and how we will address them, even if the study is exploratory. We have to specify context. We have to address the issue of process conformance along with communicable technique definition when we are studying a technique. We can create a design that allows us to maximize the number of questions asked. We can analyze small variations in the variables, such as human ability, experience, learning effects, process conformance, domain understanding, technique definition, etc.

A controlled study provides good insights into why relationships and results do and do not occur. It forces us to analyze the threats to validity, leading to insights in the identification of where replications or alternate studies are needed and where variations might show different effects.

However, a single controlled experiment, outside the context of a larger set of studies, has little value. Controlled studies need to be replicated, varying conditions (to understand the effects in different context), designs (to make up for threats to validity in any one design and context), and allowing for the answering of new and evolving questions from earlier studies. We can combine small focused studies to build knowledge. Each study can be a small contribution to the knowledge tapestry.

In the tapestry of studies it is also important to integrate negative results. Negative results and repeated experiments are important and valuable.

Of course there are drawbacks to controlled experiments. They are expensive. They tend to deal with a microcosm of reality as each study is limited to a specific set of context variables and a specific design. As stated above, you need several such studies where you can vary the context and the design. There are often issues with small sample size, small artifacts analyzed, learning effect issues, unknown underlying distributions, and potentially huge variations in human behavior.

A major threat to validity is the generalization from in vitro to in vivo.

Although a good design has many of the benefits, it is not always easy to achieve. It is easy to miss important points. It is easy to contaminate subjects. TA good controlled experiment usually involves the collaboration of several experimenters and the more people you can get to review your design, the better.

In all forms of empirical study there are problems. Scaling up is difficult and the empirical methods change as the scale grows. It is hard to compare a new technique against the current technique because of the learning curve. You cannot expect perfection or decisive answers. However, insights about context variables alone are valuable.

3 Building a Body of Knowledge

We need to combine studies to build knowledge where each study is a small contribution to the knowledge tapestry. We can build up knowledge by "replication". Replication can take many forms. We can keep the same hypothesis, combining results, we can vary the context variables, e.g., subject experience. We can vary the experimental design, e.g., order of events and activities, study type, artifact size and type. This allows us to balance threats to validity and expand our knowledge.

We can expand our studies by varying the independent variable. For example, we can change the specification or procedure associated with the process, e.g., changing the specificity of the process or technique. We can select another technique from the class of techniques.

We can expand, evolve, change the hypotheses, adding new context variables or evolving the dependent variables.

4 Example Use of Controlled Experiments

We have used controlled experiments in three ways as part of a larger set of studies.

Studying the effects of reading [2, 3, 4, 5]: This is an example of studying the scale up a technique to larger problems. Here we began with a fractional factorial experiment to study the effects of a specific reading technique (reading by stepwise abstraction) versus two specific testing techniques (structural testing with 100% statement coverage and boundary value testing). Based upon the positive results for reading we ran a control group controlled experiment to study the effect of reading by stepwise abstraction on a team development with larger artifacts (Cleanroom study). Based again on positive results, we applied the technique on a real project at NASA Goddard Space flight Center using a quasi-experimental design (case study) to study the scale up of the effects of reading by stepwise abstraction. The next step was to

apply the Cleanroom technique to a series of projects. At each stage we were able to scale up to larger studies using different empirical methods and learn different things about the effects of reading techniques on defect detection and development costs. At each stage we gained the needed confidence in the approach to apply it to larger and larger live projects, perturbing actual developments at Goddard.

Studying different reading techniques [5, 6, 7]: This is an example of replicating a series of controlled experiments, varying the techniques, artifacts, and context. Based upon the studies at NASA using stepwise abstraction as the reading technique, we recognized the need for reading earlier documents than code as well as the need to focus the reading on particular perspectives and defect classes. We ran several studies using Perspective-Based reading. We varied the level of specificity of the techniques, we varied the artifacts being analyzed (requirements, object oriented design documents) and notation of the artifacts (English language, Software Cost Reduction (SCR), notation, UML) being read, and varied the context (students, professionals) including the cultural biases in applying the techniques (U.S, Germany, Norway, Brazil). This allowed us to study and compare the effects of changes to various independent, dependent, and context variables.

Building knowledge about the high end computing development domain [8, 9]: The High Productivity computing Systems project has the to goal of shrinking the time to solution for the development and execution of high end computing systems. In order to better understand software development for HPC systems we are developing and evolving models, hypotheses, and empirical evidence about the bounds and limits of various programming models (e.g., MPI, Open MP, UPC, CAF), the bottlenecks to development, the cost of performance improvement, and the best ways to identify and minimize defects. The idea is to provide some confidence in what works best under what conditions and for whom. The project involves the use of numerous empirical studies with novices and professionals all done in concert, rather than sequentially. Study types include: controlled experiments (grad students), observational studies (professionals, grad students), case studies (class projects, HPC projects in academia), and surveys and interviews (HPC experts). Results are stored in an experience base with chains of evidence connected to hypotheses on one end and implications for various stakeholders on the other end. This mix allows early use of incomplete results rather than the confidence built over a sequential set of studies, each attacking a particular issue.

No particular empirical method is stand alone, but controlled experiments play an important role in building the discipline of software engineering.

References

1. V. Basili, “The Experimental Paradigm in Software Engineering,” in Lecture Notes in Computer Science 706, Experimental Software Engineering Issues: Critical Assessment and Future Directives, H.D. Rombach, V. Basili, and R. Selby, eds., Proceedings of Dagstuhl-Workshop, September 1992, published by Springer-Verlag, 1993.
2. R. Selby, V. Basili, and T. Baker, “Cleanroom Software Development: An Empirical Evaluation,” IEEE Transactions on Software Engineering, vol. 13(9): 1027-1037, September 1987.

3. V. Basili and R. Selby, "Comparing the Effectiveness of Software Testing Strategies," IEEE Transactions on Software Engineering, vol. 13(12): 1278-1296, December 1987.
4. V. Basili and S. Green, "Software Process Evolution at the SEL," IEEE Software, vol. 11(4): 58-66, July 1994.
5. V. Basili, "Evolving and Packaging Reading Technologies," Journal of Systems and Software, vol. 38 (1): 3-12, July 1997.
6. V. Basili, S. Green, O. Laitenberger, F. Shull, S. Sorumgaard, and M. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading," Empirical Software Engineering: An International Journal, vol. 1(2): 133-164, October 1996
7. V. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments," IEEE Transactions on Software Engineering, vol. 25(4): 456-473, July 1999.
8. S. Asgari, V. Basili, J. Carver, L. Hochstein, J. Hollingsworth, F. Shull, J. Strecker and M. Zelkowitz, "A Family of Empirical Studies on High Performance Software Development," Proceedings of Supercomputing 2004 (SC 04). Pittsburgh, November 2004.
9. L. Hochstein, J. Carver, F. Shull, S. Asgari, V. Basili, J. Hollingsworth, M. Zelkowitz, "Parallel Programmer Productivity: A Case Study of Novice HPC Programmers," Proceedings of Supercomputing 2005 (SC 05), Seattle, WA, November 2005.