Software Design: Why It's Hard To Do Empirical Research

Shawn A. Butler
Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213

shawnb@cs.cmu.edu

Voice: 412-268-8101
Fax: 412-268-5576

Abstract

Two problems plague my thesis research, few data points and variability issues across organizations. Although essential, empirical validation of design methods is not easy. My problems are not unique in software engineering research, especially software design research. I seek help in the research methods of social scientists, organizational behaviorists, and design researchers and I believe that their research methods will improve the empirical validation techniques of other software engineer researchers.

Keywords: software design, empirical research, security architecture, software engineering research

1. Introduction

I am frustrated and disappointed at the number of weakly validated software engineering research results. I have found none that specifically address my research problems and only a few examples that generally apply. In particular, I have looked for accepted research techniques that can be used with relatively few data points and research methods that address variability issues across organizations. My problems are not unique in software engineering research. I believe that software engineering researchers are hindered because of limited access to industrial data, system complexity and variability, and inexperience with qualitative research methods. Social scientists and design researchers share some of these problems but have established validation techniques to deal with them. Although software engineering researchers face some unique challenges, I am hopeful that research methods from other disciplines can improve validation of my results.

My thesis proposes a model and methodology that guides software engineers in making design decisions. More specifically, the model and methodology assist software engineers in selecting security technologies to build security architectures. In the security community, the term "security architecture" commonly refers to the system security components and their integration into the system architecture. At the architectural level, extra-functional system attributes, such as scalability, maintainability, reliability, and security are common design issues. There have been few papers that prescribe design principles, and fewer still that validate these design principles. The fundamental question relevant to my research is: How do researchers validate the design methods that software

engineers use to develop their systems? Then, which design validation techniques can I use to show that my model and methodology support development of the security architecture? It is imperative that validated design principles be established, especially as the software engineering field struggles to codify a software engineering body of knowledge.

2. My thesis research

I consider empirical validation essential to showing that the model and method support software engineers. Models and methods, which assist inexperienced software engineers in selecting the appropriate security technologies for their systems, should perform as well as (or better than) experienced software engineers in the field. Ideally, the model and method should improve software engineering practice. A reasonable or logical argument cannot substitute for empirical evidence that shows the model and method improves software engineering practice. However, despite the importance of empirically validated results, it can be very difficult to get enough data to show statistically conclusive results

Limited access to resources has restricted the number of organizations I can evaluate for my thesis. The problem is that traditional statistical methods usually require more data points than I will have for analysis. Most statistical techniques require a larger sample size than is feasible and Analysis of Variance (ANOVA) techniques are not appropriate because my data is categorical. For example, a type of security technology is either selected or not. I would like to find other empirical methods that are persuasive despite the lack of statistically significant data. I am interested in how social scientists and design researchers conduct studies and analyze data with limited participation and few data points. I suspect this is a common experience among software engineering researchers. I hope to find research techniques that enable software engineers to conduct empirical research with a limited number of data points.

I am also interested in how other disciplines evaluate different organizations. In my thesis, I would like to compare the results of similar organizations, but I am concerned about the pitfalls of comparing data across organizations. For example, is it enough to compare electronic commerce systems, or are there organizational factors that would make the comparison invalid? Can I compare the results of an e-commerce system and an academic system? Other disciplines may not have specific answers to these questions, but I think they have experience in determining which organizational factors matter. These insights might be useful in my research.

3. System Security Research Impediments

The primary impediment to empirical validation in security research is access to resources. Information managers are reluctant to allow researchers access to their information systems and sensitive company data. Even accurate threat data is difficult to obtain because security experts speculate that many information system attacks go unreported. System security data is among the most sensitive data because companies are

concerned that security weaknesses might be exposed in publication. Promises of anonymity are always a requirement, but even with those promises, complete information is not always forthcoming. Published data must be sanitized so readers cannot guess the source of the information. I imagine that social scientists have similar problems for assuring anonymity, such as patient confidentiality. Limited access to resources results in fewer data points and potentially little conclusive evidence.

Even if confidentiality were assured, managers tend not to perceive the benefits that justify their resource commitment. Almost all empirical research requires some commitment of an organization's time or resources, and my research is no exception. Other than the government systems that I solicited, organizations wanted assurance that there would be a short term benefit to them. Altruistic arguments about how my results might benefit software engineers generally were not persuasive. I believe it is common practice for researchers to compensate participants for their inconvenience; however, this is not practical when working with several organizations. Organizational behaviorists might have similar problems when conducting research.

Finding examples of well designed security in systems, which can be used as benchmarks, is significantly difficult in my thesis research. Many software engineers select and integrate security technologies into their systems, but most of these engineers have little or no experience with security technologies. It is commonly thought that integrating security after the system is implemented is more difficult and expensive than designing for security earlier in the development cycle. In practice, engineers rely on checklists, consultants, or marketing information to determine which security technologies to implement. Their goal is to achieve an acceptable level of risk against anticipated threats, though this threat information is also not easily obtained. In the end, system security is often based on a haphazard collection of security technologies chosen after system implementation rather than the result of an engineering analysis.

4. Software Engineering Research

It is unfortunate how little software engineering research is empirically validated. During the course of my Ph.D. program, I have read numerous, interesting software engineering papers but few with convincing research results. Early papers tend to describe observations about software engineering experiences and lessons learned. Many software engineers can still relate to these seminal papers, it doesn't appear that software engineering research methods have progressed significantly since 1994, when a review of 400 articles[1] described the state of software engineering research. A cursory review of the last two years of a few software engineering journals revealed that most claims were not empirically validated. The fact, that the entire June 1998 issue of IEEE Transactions on Software Engineering was devoted to empirical software engineering research is indicative of how unusual field research is. I believe there are two reasons for this stagnation: 1) computer science research methods are not always adequate for empirical validation, and 2) software engineering field research is very difficult. As I have already stated, some of these difficulties apply to my research.

Although software engineering has grown as a discipline from computer science, the research methods used in computer science are not always adequate for software engineering research. Computer scientists have well established performance metrics that lead to objective evaluation. One can measure CPU cycles, instructions, bandwidth delay, etc. Software engineers use these metrics but also require gathering data from the field to answer questions that concern design decisions, economic issues and ill-defined or unknown forces that ultimately affect software quality. The inherent human element in software engineering often requires quantifying qualitative data. Data collection from software development organizations requires research techniques similar to those found in sociology, psychology, and design, rather than those techniques found in the computer science field. Software engineering researchers need to develop methods that draw on both the computer science field and these other disciplines.

The term "engineering" implies a degree of applicability to practice. There is a general expectation that software engineering research eventually results in better quality or more efficient software production. During the last ten years, developers have seen "silver bullet" software engineering techniques come and go. Today developers are more skeptical of techniques that don't clearly show software development cost reductions, either through improved software quality or more efficient development methods. Clearly, methods and software artifacts that add bureaucratic overhead to development projects without an offsetting benefit are less likely to be adopted. Measuring the results in the field is the best way to validate claims that the technique or method is a positive contribution to practice. Field validation would also have the additional benefit of showing the context, limitations and expected payoff of adoption, issues that are not always addressed in experience articles or the academic research community. Not all research needs to show relevance to practice, but research techniques that show the value of results in practice are imperative to eventual adoption.

Managers might be more likely to experiment with new software engineering techniques if preliminary data supported an improvement to practice. Most software engineering projects have little room for experimental techniques that could result in increased development costs. I spent several years as a systems engineer and project manager and I could rarely afford the risk of adopting a new method without strong evidence that it would not adversely impact system performance or schedule. Some published papers claim that their method is cost effective, efficient, or flexible, based on the evaluation of one or two small academic systems. More rigorous testing in academic environments would increase confidence in the results. Software engineering researchers studying the effectiveness of a new technique or methodology must collect preliminary data before conducting field studies. This is another opportunity to learn from other disciplines.

Another problem related to obtaining resources is finding systems that are similar enough for comparison. Organizations vary a great deal; therefore finding similar organizations willing to participate in my research has proved most challenging. Sometimes, it is desirable to show that software engineering tools apply to a class of systems. At other times, it is invaluable to show which factors influence successful, or

limit, tool adoption. Traditionally, software engineering researchers have used lines of code or system type as the basis for determining whether systems are similar, but it is unclear whether those are sufficient measures. As empirical research becomes more common in software engineering, we need to identify relevant variability factors. Just assessing which factors can vary among organizations and which ones can't is difficult because the software engineering research community does not have a long history of established empirical methods to deal with software development organizational variability. Social science researchers, particularly organizational behaviorists and anthropologists, must deal with variability among organizations. We should evaluate the lessons learned in these fields and determine how they might apply in software engineering research.

A unique problem that plagues software engineer researchers is the speed with which the field is changing. Like the computer industry in general, new software engineering ideas cycle quickly through the community. Social scientists have experience with studies that take many years before the results can be analyzed. Adoption of a new idea is slow, and only occurs after considerable and convincing empirical evidence shows the idea has merit. I believe that many software engineering studies would benefit from observations made during the lifetime of a project. The effectiveness of a new technique or methodology may not be known until the later stages of the software development cycle. Evaluation may have to wait until the maintenance phase. Large projects can take many years before a system is fielded. Sometimes, software artifacts can provide enough useful information for studies, but often, they lack information about decision rationale or limitations of the technique. Unfortunately, the field of software engineering is changing so rapidly that studies, which take several years to complete, may be obsolete by the time the results are published.

Other disciplines have explored the research problems that software engineering researchers deem to be unique. As the software engineering field matures, so will the research methods. Software engineers will turn to the social scientist and organization behaviorist to borrow research methods that are relevant to their empirical research studies. To date I have not resolved all the problems with my thesis validation. I have started to explore research methods in other disciplines because I want to ensure that my results are persuasive. More importantly, I want to collect enough evidence to support claims that the model and methodology improves practice.

[1] Lukowicz, P., Heinz, E., Prechelt, L., and Tichy, W. Experimental Evaluation in Computer Science: A Quantitative Study. Journal of Systems and Software, January 1995