Learning through Application

Victor R. Basili

University of Maryland and Fraunhofer Center - Maryland

The good news is that empirical studies have finally become recognized as an important component of the software engineering discipline. One sees more and more empirical studies and experiments in the literature. The bad news is that these studies are often used as confirming or not confirming the effectiveness of some method, technique, or tool, rather than as part of the process of discovery process. The experiment is an add-on, we do the study after the concept is considered complete, rather than applying the method, technique or tool and learning from the application about how to evolve the concept. This is the basis of the scientific method; theories are tested and evolved over time. In the software engineering discipline, where the theories and models are still in the formative stages and processes are applied by humans a part of a creative process, observing the application or performing exploratory studies become an important step in the evolution of the discipline.

Software engineering has several characteristics that that distinguish it from other disciplines. Software is developed, in the creative, intellectual sense, rather than produced in the manufacturing sense. Software processes are development processes not production processes. They are not replicated over and over again. This aspect of the discipline is probably the most important. This affects greatly how we learn. We need to always be on the lookout for the effect of context variables. Since it a human-based discipline there will always be variation in study results and we will never be able to control or even identify all the context variables. It creates a need for continual experimentation as we must understand how to modify and tailor processes for people.

The context issue is more than just the people; all software is not the same and all software development environments are different. One consequence of this is that process is a variable, goals are variable, etc. That is, we need to select the right processes for the right goals for the environment we are analyzing. So, before we decide how to study something we need to know something about the environment and the characteristics of the thing we are about to build.

Also, the software engineering discipline is still quite immature in the sense that there is a lack of models that allow us to reason about the process, the product and their relationships. This compounds the non-visible nature of software. It intensifies the need to learn from the application of the ideas in different situations and the requirement to abstract from what we see.

Add to this the fact that developing models of our experiences for future use (reuse) requires additional resources in the form of money, organization support, processes, people, etc. Building models, taking measurements, experimenting to find the most effective technologies, and feeding back information for corporate learning, cost both time and money. These activities are not a by-product of software development. If these activities are not explicitly supported, independent of the product development, they will not occur and we will not make quality improvements in the development process.

All this makes good experimentation difficult and expensive. Experiments can only be confirmatory in the small and are subject to problems in understanding scale-up, the integration of one process with another, the understanding of the effect of context variables, etc.

I believe we need to focus more attention on informal exploratory studies that provide insights, coupled, when appropriate, with more formal empirical studies to test out pieces of the whole that can be added to the tapestry that helps make the discipline clear. I believe that the study of the software engineering discipline is exploratory and evolutionary. It follows the scientific method but because of its nature, real experiments are not always possible or useful.

I like to say that the study of software engineering is a laboratory science; and it is a big science. So the laboratory is quite grand and we need methods that support the exploratory nature of this big science. The discipline cannot be understood only by analysis. We need to learn from application whether relationships hold, how they vary, what the limits of various technologies are, so we can know how to configure process to develop software better.

We need to take advantage of all opportunities to explore various ideas in practice, e.g., test its feasibility, find out if humans can apply it, understand what skills are required to apply it, test its interaction with other concepts. Based upon that knowledge, we need to refine and tailor it to the application environment in which we are studying it so it can be easily transferred into practice. So, we need to try out our ideas in practice and evolve them, even before we can build models. We are an exploratory science – we are more dependent on empirical application of methods and techniques than many disciplines and we need to share the results.

Time is an important aspect in the application of the scientific method to software engineering; there needs to be many applications of a process, in different environments, with each application providing a better understanding of the concepts and their interaction. Over time all context variables need to be considered and many of them don’t even pop up until we have seen the application of the approach in practice a by different people in different sites.