Empir Software Eng (2008) 13:211–218
DOI 10.1007/s10664-008-9060-1

VIEWPOINT

The role of replications in Empirical Software Engineering

Forrest J. Shull · Jeffrey C. Carver · Sira Vegas · Natalia Juristo

Published online: 29 January 2008

Editor: Claes Wohlin

Abstract Replications play a key role in Empirical Software Engineering by allowing the community to build knowledge about which results or observations hold under which conditions. Therefore, not only can a replication that produces similar results as the original experiment be viewed as successful, but a replication that produce results different from those of the original experiment can also be viewed as successful. In this paper we identify two types of replications: *exact replications*, in which the procedures of an experiment are followed as closely as possible; and *conceptual replications*, in which the same research question is evaluated by using a different experimental procedure. The focus of this paper is on exact replications. We further explore them to identify two sub-categories: *dependent replications*, where researchers attempt to keep all the conditions of the experiment the same or very similar and *independent replications*, where researchers deliberately vary one or more major aspects of the conditions of the experiment. We then discuss the role played by each type of replication in terms of its goals, benefits, and limitations. Finally, we highlight the importance of producing adequate documentation for an experiment (original or replication) to allow for replication. A properly documented replication provides the details necessary to gain a sufficient understanding of the study being replicated without requiring the replicator to slavishly follow the given procedures.

F. J. Shull
Fraunhofer Center for Experimental Software Engineering, College Park, MD, USA
e-mail: fshull@fc-md.umd.edu

J. C. Carver (✉)
Mississippi State University, Mississippi State, MS, USA
e-mail: carver@cse.msstate.edu

S. Vegas · N. Juristo
Universidad Politécnica de Madrid, Madrid, Spain

S. Vegas
e-mail: svegas@fi.upm.es

N. Juristo
e-mail: natalia@fi.upm.es

Keywords Experimental replications · Experimental infrastructure · Lab package

1 Goals of Replication in Empirical Software Engineering

Experimental replications are performed for the same reasons that independent experiments are conducted, to better understand software engineering phenomena and how to improve the practice of software development. One important benefit of replications is that they help mature software engineering knowledge by addressing both internal and external validity problems. In terms of external validity, replications help researchers show that experimental results are not dependent on the specific conditions of the original study. In terms of internal validity, replications also help researchers show the range of conditions under which experimental results hold.

Generally speaking, a successful replication is one that helps the research community build knowledge about which results or observations hold under which conditions (Basili et al. 1999; Vegas et al. 2006). Thus, a replication that produces results that are similar to those of the original experiment on which it was based is just as useful to the community as a replication that produces results that are different from those of the original experiment. A replication produces different results is useful because it provides insight to help the community understand *why* the results were different. Therefore, the success of a replication must be judged relative to the knowledge it contributes to the body of knowledge (for example, identifying possible new variables that have an influence on the response variable).

Software engineering is not the only field for which the general experimental concept of replication is important. For example, within the field of Behavioral Research, replications are viewed as an important tool for advancing knowledge. In this vein, two types of replications have been defined. An *exact replication* is one in which the procedures of an experiment are followed as closely as possible to determine whether the same results can be obtained [we use the term "exact" here rather guardedly; Brooks et al. (2007) among other authors, have pointed out the impossibility of duplicating a study when human subjects and experimenters are involved]. Replications that reuse the original procedures (e.g. the study design and the experimental steps) but modify the subject pool or other experimental conditions (e.g. time, learning process, or artifacts) in order to gain more insight into the original results also fall into the category of exact replications. Conversely, a *conceptual replication* is one in which the same research question or hypothesis is evaluated by using a different experimental procedure, i.e. many or all of the variables described above are changed (Cozby 2007).

However, this taxonomy can be further refined. Within exact replications we can find two sub-categories: those replications in which researchers attempt to keep all the conditions of the experiment the same or very similar, and those replications in which researchers deliberately vary one or more major aspects of the conditions of the experiment to address a specific research question.

For software engineering researchers, two possible goals that have different implications for the replication are:

1. Testing that a given result or observation is reproducible (e.g. demonstrating that a technique which shows promising results in a controlled environment really results in an improvement when used under more realistic conditions). This approach is useful for gaining confidence in results of previous studies.

2. Understanding the sources of variability that influence a given result (e.g. understanding the types of projects for which a technique is effective). This approach is useful for understanding the scope of the results.

The main goal of a replication, therefore, does not need to be limited to statistical hypothesis testing and p-values. Such a goal may lead to additional data being added to the datasets with the hope of finding statistical significance to support the validity of a small or subtle effect. By simply adding data to the dataset in order to mine that data to uncover small effects, the researcher increases the likelihood that he or she will identify a subset of the data that seems to exhibit the phenomenon of interest, merely by chance (introducing what is known as data snooping bias). This approach of searching for small effects is also undesirable from another, more practical, point-of-view. The goal of software engineering research should be to impact software development practice, therefore, researchers should be looking for techniques that have measurable effects across different environments and have a clear and sizeable impact on development results. Even if experimental studies demonstrate that a technique provides a statistically significant improvement, organizations are unlikely to invest the resources necessary to adopt that technique and integrate it into their existing culture for such a small return on investment.

Before describing the benefits of various types of replications, it is important to discuss *who* should perform a replication. When a replication is studying the effects of a particular technique or process, there are two types of researchers, those with a vested interest in the results and those without a vested interest. Ultimately, replications run by researchers without a vested interest are likely to have less bias and potentially more value. But, it is also important to realize that before a researcher without a vested interest will be willing to conduct a replication, they need enough evidence to convince them that a study is actually worth replicating. At least initially, this evidence will most likely come from a researcher who has a vested interest in the outcome.

2 The Role of Independent, Dissimilar Replications

When the research goal is to show that a given effect is robust (e.g. that a new technique is really more effective than other approaches), the ideal case is for a completely independent set of researchers to replicate a published study using their own experimental design (which is different than the design of the original study). This type of replication is classified as a *conceptual replication*. In this case, if the results of the original study are reproduced using different experimental procedures, then the community has a very high degree of confidence that the result is real, significant, and independent of the specific study approach. The independence of the replicators from the original experimenters lends additional confidence that the original results were not the result of experimenter bias. In addition, if the context or environmental factors of the replication are also very different from those of the original study, then the replication can contribute some confidence that the effect is not limited to one particular setting. In this way, this type of replication can address external as well as internal validity.

However, this approach provides little insight when the results of the replication contradict those of the original study. In this case, neither the replicators nor the original researchers have a solid basis for hypothesizing about the cause of the discrepancy. It is impossible to discuss, for example, whether the original observation was erroneous or whether a key variable in the original study, which may have not been reported, was altered

in the replication creating a context in which the phenomenon under study simply could not exist (some examples of such sources of variation are that insufficient training was provided for subjects to learn a new technique, or the system to which the technique was applied did not have defects of the type targeted by a particular analysis technique).

Since studies in software engineering, especially those that are realistic, require significant cost and effort, conducting a replication that has the possibility of producing no useful results is simply too risky to be the norm. Moreover, designing and running conceptual replications from scratch is very costly. Therefore, not all researchers are able or willing to run such replications. Conversely, it is cheaper and more feasible to run replications that reuse more of the existing experimental design and materials.

3 The Role of Dependent or Similar Replications

As hinted at above, when the goal of the replication is to identify factors that influence whether an observed effect holds under different conditions (e.g., testing whether technique X is effective with large teams as well as small ones), then the replicator needs to understand and control the variations between the original study and the replication. In this case, an independent replication in which a large number of factors change is not likely to provide the desired insight into the influencing factors. In such a case, the large number of factors that change between the original study and the replication make it less likely that observed effects can be traced to any particular factor of interest.

Conversely, the use of a replication in which a small number of factors are varied to specifically test those factors, has a better chance of providing the type of information the researcher is interested in. This type of replication is referred to as a dependent replication because the replicators rely on the design of the original study as the basis for the design of the replication. The types of changes a replicator makes generally affect the context in which the study is conducted and not the procedure that the subjects follow. Therefore these dependent replications are a type of *exact replication* described earlie. Although a dependent replication provides less confirmatory power than an independent replication regarding the effect itself, using a dependent replication is necessary to understand all the variables that influence a result. In other words, both types of replication play an important role, but it seems that dependent replications must come before independent replications (because, while either verifying or contradicting the original results, more insight can always be gained).

In performing a dependent replication, researchers will generally make changes of the following types: (1) changing the population on which the phenomenon is tested (e.g. moving from students to professionals or changing the student population from one site to another) or (2) changing the artifact on which the technique or method is applied (e.g. executing an inspection technique on a realistic requirements document rather than a scaled down version or applying a testing techniques to a different program because the new subjects are not familiar with the language of the original program).

4 The Role of Lab Packages

Regardless of the goal of a replication, an exhaustive documentation of the original experiment is a valuable resource. Often, researchers find that the page limitations of journal and conference papers require some of the relevant details to be excluded from the report. A solution to this problem is the use of more detailed reports commonly called *lab packages*

(Brooks et al. 2007). The existence of a lab package does not imply that a "good replication" is one that follows the "recipe" provided. Rather, replicators have a responsibility to run the best study possible to address their research goal; not to slavishly follow the given procedure. Lab packages should be seen simply as providing the details necessary to gain a sufficient understanding of the study they are replicating. A replicator may decide that a *dependent replication* with minor modifications to the original study is the best plan of action. In this case, the lab package provides valuable information to support the replication. Conversely, the replicator may choose to use an *independent replication* to re-test the same hypothesis with a completely new experimental design. In this case, the lab package provides the replicator with enough details about how the original study was conducted to allow him or her to design a sufficiently different study. Of course, a replicator can also decide to run conceptual replications without using the lab package (using only the hypothesis published in the experiment report) in order to make the replication more independent.

Even while using a lab package, due to the nature of software engineering research, it is all too easy to introduce a slight change, consciously or unconsciously, into an experimental setup that diminishes or negates even normally very robust effects. Examples of such difficulties have been analyzed and reported (Basili et al. 1999; Shull et al. 2004; Vegas et al. 2006), and includes issues such as:

–When a process is the object of study, the training that the subjects receive has a large influence on their performance. However, even when using the same training materials, it is difficult for two completely independent instructors to deliver comparable training without some type of coordination. If the training is not comparable, then the validity of the study outcome is in jeopardy. This phenomenon is true even for mature processes like software inspection; the problem becomes much harder when the process being studied is quite new and has not yet been transferred to external researchers before (which is often the case with processes that are studied empirically).

–Replicators often need to modify the timing of the experimental procedures to fit the constraints of their environment. However, these modifications run a high risk of accidentally removing important guidelines or artificially constraining the length of particular activities and thereby impacting the outcome of the study.

To properly address and understand the variations introduced, we feel that there is no alternative other than to rigorously and sufficiently document each study in a lab package or technical report. This documentation assists other researchers both in performing the replication and determining whether the changes introduced in the experiment might have caused a different result or observation than occurred in the original study.

We must be clear that dependent replications that make use of lab package documentation have, by nature, less confirmatory power than completely independent replications. Because the dependent replications rely on the same underlying protocols as the original study, their results cannot be considered as truly independent of the original study. Moreover, they may propagate any accidental biases from the original study into the results of the replication (Miller 2000).

However, when the goal of a replication is either to isolate influencing factors, or to identify the scope of results by introducing small, controlled changes, then a dependent replication that is based largely on the original lab package with changes made to only a small number of carefully selected factors is most appropriate (in such a replication, if the original effect cannot be replicated and there is a level of confidence in the reasonableness of both experimental designs, then it can be hypothesized that the different outcome was the result of one of the few changes that were introduced into the replication).

Lab packages also have value for supporting replications that have the goal of confirming the robustness of an effect. Anecdotal evidence suggests that a researcher can start with an existing lab package a starting point and, even while introducing some variations, design and run a replicated experiment more cheaply than designing the entire replication from scratch (Lindvall et al. 2005). This ability is especially important since, as discussed before, even small variations can introduce large biases into the results—so learning from the experiences of the original experimenters can reduce the risk of introducing inadvertent biases in later studies of the same phenomenon.

In addition, lab packages can be useful for researchers who want to be introduced into the empirical software engineering field or by researchers who are not empiricists by nature (i.e., researchers who are willing to run replications, but have other goals that are more important to them). Often, these researchers are discouraged by the effort needed to design and run an experiment. The presence of a lab package can significantly reduce the effort of running experiments.

Finally, it is important to remember that replicators who have a vested interest and those without a vested interest can still successfully make use of a lab package. Simply using a lab package does not imply that the replicator, without a vested interest, has a preference as to the results. The lab package is simply a way to transmit all the necessary details of the experiment in order to be able to perform an exact replication.

For these reasons, we argue that replications with some connection to the original study, i.e. dependent replications, help evolve understanding and have a beneficial effect on future studies. As such, dependent replications should not be rejected by reviewers out of hand, but their value should be judged based on whether they help refine the understanding of important variables or the phenomenon itself. However, the dependencies between the replication and the original experiment should be discussed in publications as an important threat to validity.

References

Basili V, Shull F et al (1999) Building knowledge through families of experiments. IEEE Trans Softw Eng 25 (4):456–473

Brooks A, Roper M et al (2007) Replication's role in software engineering. In: Shull F, Singer J, Sjoberg DIK (eds) Guide to advanced empirical software engineering. Springer, London, pp 365–379

Cozby (2007) Methods in behavioral research. McGraw-Hill, New York

Lindvall M, Rus I et al (2005) An evolutionary testbed for software technology evolution. Innovations in Systems and Software Engineering—A NASA Journal 1(1):3–11

Miller J (2000) Applying meta-analytical procedures to software engineering experiments. J Syst Softw 54 (1):29–39

Shull F, Mendonca M et al (2004) Knowledge-sharing issues in experimental software engineering. Empirical Software Engineering—An International Journal 9(1):111–137

Vegas S, Juristo N et al (2006) Analysis of the influence of communication between researchers on an experiment replication. Proceedings of 5th International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, pp 28–37

Dr. Forrest Shull is a senior scientist at the Fraunhofer Center for Experimental Software Engineering in Maryland (FC-MD), where he serves as Director for the Measurement and Knowledge Management Division. He is project manager for projects with clients that have included Fujitsu, Motorola, NASA, and the US Department of Defense. He has also been lead researcher on grants from the National Science Foundation, Air Force Research Labs, and NASA's Office of Safety and Mission Assurance. He is Associate Editor in Chief of IEEE Software, specializing in empirical studies.

Dr. Jeffrey Carver is an Assistant Professor in the Computer Science and Engineering Department at Mississippi State University. He received his PhD from the University of Maryland in 2003. His research interests include Software Process Improvement, Software Quality, Software Inspections, Software Architecture, Software Engineering for Computational Science, and Software Engineering Education. His research has been funded by the United States Army Corps of Engineers, the United States Air Force, and the National Science Foundation. He is on the Editorial Board of the Software Quality Journal.

Dr. Sira Vegas is assistant professor of Software Engineering with the Computing School at Madrid's Technical University (UPM), Spain. She was a summer student at the European Centre for Nuclear Research (Geneva) in 1995. She was a regular visiting scholar of the Experimental Software Engineering Group at the University of Maryland from 1998 to 2000, and visiting scientist at the Fraunhofer Institute of Experimental Software Engineering in Germany in 2002. Dr. Vegas is the UPM's second representative at the ISERN. She has been program chair for the International Symposium on Empirical Software Engineering and Measurement (ESEM) held in 2007.

Dr. Natalia Juristo (http://grise.upm.es/miembros/natalia/) is full professor of software engineering with the Computing School at the Technical University of Madrid (UPM) in Spain. She has been the Director of the UPM MSc in Software Engineering for 10 years. Natalia has been fellow of the European Centre for Nuclear Research (CERN) in Switzerland, and staff of the European Space Agency (ESA) in Italy. During 1992 she was resident affiliate of the Software Engineering Institute at Carnegie Mellon University (USA). Natalia has served in several Program Committees ICSE, RE, ISESE and others. She has been Program Chair for SEKE97 and ISESE04 and General Chair for ESEM07, SNPD02 and SEKE01. Prof. Juristo has been Key Speaker for CSEET03 and ESELAW04. She has been member of several Editorial Boards, including IEEE Software and the Journal of Empirial Software Engineering. Dr. Juristo has been Guest Editor of special issues in several journals, including IEEE Software, the Journal of Software and Systems, Data and Knowledge Engineering and the International Journal of Software Engineering and Knowledge Engineering. She is senior member of IEEE-CS and has been leader of IEEE-CS SEOnline. Natalia has a B.S. and a Ph.D. in Computing from UPM.