Chapter 14
Replication's Role in Software Engineering

A. Brooks, M. Roper, M. Wood, J. Daly, and J. Miller

Abstract We provide motivation for researchers to replicate experiments in software engineering. The ideology of replication is discussed. We address the question: Is an experiment worth repeating? The current lack of replication studies is highlighted. We make clear that exact replication is unattainable and we draw on our first experience of performing an external replication. To categorise various kinds of replication, we propose a simple extension to Basili et al.'s framework for experimentation in software engineering. We present guidance as to the level of reported detail required to enable others perform a replication. Our conclusion is that there is only one route for empirical software engineering to follow: to make available laboratory packages of experimental materials to facilitate internal and external replications, especially the latter, which have greater confirming power.

1. Introduction

Experimental design is difficult and the experimental process can be error prone. As a consequence, all experimental results should be reproducible by an external agency. By other researchers successfully repeating an experiment, confidence is built in the procedure and the result. Without the confirming power of external replications, a result should be at best regarded as of limited importance and at worst with suspicion and mistrust.

We distinguish two main forms of replication: internal and external. Internal replication is undertaken by the original experimenters (or teams that contain members of the original experimental team): they repeat their own experiment. External replication is undertaken by independent researchers and is a critical verification step. We are not concerned here with replication as it applies to an individual experimental design.

The section that immediately follows provides motivation for researchers to replicate experiments in software engineering. There then follows sections on the theory of replication and replication in practice. As subsections of the latter, we discuss criteria for deciding whether an experiment is worth repeating, the frequency of replication studies, the unattainability of an exact replication, and

© Springer 2008

our first experience of performing an external replication. In the section that then follows, to categorise various kinds of replication, we present a simple extension to Basili et al.'s (1986) framework for experimentation in software engineering. The penultimate section presents guidance as to the level of reported detail required to enable others perform a replication. In the final section, we conclude that there is only one route for empirical software engineering to follow: to make available laboratory packages of experimental materials to facilitate internal and external replications, especially the latter, which have greater confirming power.

2. Replication: The Motivation

No one doubts the need for software engineers to work from principles and guidelines in which the professional community has high confidence, all the more so if the application is safety critical. High levels of confidence are only attained when independent researchers successfully replicate an experiment. Without the confirming power of external replication, many principles and guidelines in software engineering should be treated with caution.

Much is to be gained, therefore, by critical examination of previous experiments, by identifying experiments that are worthy of replication, and by replicating these experiments externally.

Huxley (1965) has noted,

> And in science, as in common life, our confidence in a law is in exact proportion to the absence of variation in the result of our experimental verifications.

So the greater the number of experimental verifications the better, at least until such time as additional verifications carry no further power of confirmation. Moreover, given the human component and the rich variety of software and hardware technologies, it surely is beholden on the community to perform many, many, such verifications. Only under exceptional circumstances should one-shot studies involving subjects be relied upon. For example, when the following criteria are all met: (1) a large number of subjects were used, (2) the effect present is so large, the use of statistical tests to convince the reader that an effect exists are unnecessary, and (3) peer review has not found any criticism with the work. Even then of course the effect cannot be extrapolated to just any context. Thus, we strongly agree with Curtis (1980) when he says,

> …results are far more impressive when they emerge from a program of research rather than from one-shot studies.

Much is said and written about quality control in software development (e.g. Card (1990)). It is ironic, to say the least, that the quality control mechanism of replication, especially external replication, is so little practiced amongst those doing the science behind the engineering. There is an additional irony: because of the current state of software development practice, N-version programming has been suggested as a fault recovery mechanism (see, for example Kelly et al. (1991)). We know so little about doing it right, we end up replicating system functionality across several programs.

Concerning a particular flawed study in psychology which was accepted as being valid for a long time, Broad and Wade (1986) wrote,

> Why did nobody helping to raise generations of undergraduates…replicate the study?

Such a question could equally as well be addressed to many educators of software engineering students regarding numerous studies whose results are communicated often quite uncritically to students. We should all be motivated to carry out replications or at least give support to those who do.

3. Replication: The Ideology

Subjecting theory to experimental test is a crucial scientific activity. Popper (1968), however, explains that researchers must be sure of their results before reporting them, stating,

> We do not take even our own observations quite seriously, or accept them as scientific observation, until we have repeated and tested them.

Coupled with this advice, modern scientific ideology now also demands that experimental results are replicable by an external agency. For example, as Lewis et al. (1991) rightly claim,

> The use of precise, repeatable experiments is the hallmark of a mature scientific or engineering discipline.

Furthermore, Goldstein and Goldstein (1978) take this one step further, stating,

> We now take for granted that any observation, any determination of a 'fact', even if made by a reputable and competent scientist, might be doubted. It may be necessary to repeat an observation to confirm or reject it. Science is thus limited to what we might call 'public' facts. Anybody must be able to check them; experimental observations must be repeatable.

Not only must the researcher make his work repeatable, however, some even regard it as being beholden on the scientific community to execute replications just to verify the experimental results, as we ourselves do. For example, Huxley (1965) has stated,

> In scientific inquiry it becomes a matter of duty to expose a supposed law to every possible kind of verification…

Broad and Wade (1986), in their description of the scientific ideology, consider replication to be the third check in verifying scientific claims, the first two being the peer review system that awards research grants and the journal refereeing that takes place prior to publication. They also describe the ideal of reporting experiments as follows,

> A scientist who claims a new discovery must do so in such a way that others can verify the claim. Thus in describing an experiment a researcher will list the type of equipment used and the procedure followed, much like a chef's recipe. The more important the new discovery, the sooner researchers will try to replicate it in their own laboratories.

Replication is also concerned with the way the original hypothesis is expressed. As Smith (1983) has stated,

> Replication does two things: first, it tests the linguistic formulation of the hypothesis; second, it tests the sufficiency of the explicit conditions for the occurrence of the phenomena.

For example, an original hypothesis may be linguistically expressed to almost encourage conclusions to be expressed with the wrong meaning. Henry and Humphrey (1990) state their hypothesis as follows: "the hypothesis of this study is that systems designed and implemented in an object-oriented manner are easier to maintain than those designed and implemented using structured techniques." In order to test this, their subjects were asked to make modifications to an object-oriented system and a functionally equivalent procedure-oriented system. After their data analysis, Henry and Humphrey concluded that the "experiment supports the hypothesis that subjects produce more maintainable code with an object-oriented language than with a procedure-oriented language," which turns around the meaning of the original hypothesis: the idea was not for subjects to produce code to be tested for maintainability, but rather to test the maintainability of two different systems by having subjects perform maintenance tasks on them.

Another important example is that criteria for subject participation in a software engineering experiment may be insufficiently specific and, as a result, the replication yields different results due to variability unaccounted for between the subjects.

4. Replication: In Practice

4.1. Determining Worthy Experiments

Even if an empirical study was found to be replicable in terms of the availability of experimental artifacts, there can be, and usually are, several other reasons why one should first be wary of devoting the resources necessary to performing a replication study. The background may not be properly researched and the empirical study may be addressing the wrong issue. Inappropriate methods may be used; for example, when people are involved, very strictly controlled laboratory experiments may be less useful than more qualitative or ethnographic forms of experimentation. Errors of commission or omission may be made or experimental variables may be incorrectly classified. For example, Scanlan (1989) criticises Shneiderman et al. (1977) for not making use of time as a measurable dependent variable (the subjects were all given as much time as they required) and claims as a result that "any significant difference may have been washed out." From his experimental result, however, Shneiderman et al. called into question the utility of detailed flowcharts, stating "we conjecture that detailed flowcharts are merely a redundant presentation of the information contained in the programming language statements." The experimental flaw identified by Scanlan can be classified as an error of omission, and one which, according to Scanlan, has seen "the decline of flowcharts as a way to represent algorithms." Scanlan then went on to design a new experiment to test the same hypothesis using time as a dependent measure and claimed "my experiment shows that significantly less time is required to comprehend algorithms represented as flowcharts."

Missing details may prevent the reader from forming their own view of the worth of the data, for example, error estimates may not be provided for some or all of the critical measures or raw data may be crudely summarised when it could have been presented in full. Statistical procedures may be misapplied. Alternative interpretations may not be presented: when people are involved it is more than likely that more than one interpretation can be placed on the data. We agree with Collins (1985) who regards an experiment to have been incompetently performed if some alternative explanation for the data has been overlooked. For example, in a comparative study of C and C++ development times involving only four subjects, Moreau and Dominick (1990) concluded that there was a significant difference in favour of C++. One of the four subjects, however, took very much longer on the third C++ task. The experimenters simply attributed this to a debugging difficulty, i.e. they appeared not to have checked that use of C++ itself was the real cause of the problem. Failure to discuss alternative interpretations of data can prevent a reviewer performing a meaningful meta-analysis of the research area. (Brooks and Vezza (1989) is an example of a paper providing the reader with alternative interpretations.)

Should the report of an experiment pass a detailed critical reading of its design, execution, analysis and interpretation, then it can be deemed worthy enough to replicate.

4.2. Frequency of Replication Studies

In schools, colleges, and universities, replication studies are performed daily. But such studies are usually scaled-down versions of an original experiment, are performed by students in the act of learning, and have no confirming power. As Collins (1985) notes,

> As more becomes known about an area however, the confirmatory power of similar-looking experiments becomes less. This is why the experiments performed every day in schools and universities as part of the scientific training of students have no confirming power; in no way are they tests of the results they are supposed to reveal.

Those employed in research rarely perform replication studies. Again, as Collins (1985) notes,

For the vast majority of science, replicability is an axiom rather than a matter of practice.

Broad and Wade (1986) also draw attention to the lack of replication work by stating,

How much erroneous…science might be turned up if replication were regularly practiced, if self-policing were a more than imaginary mechanism?

Broad and Wade (1986) reckon that the Simpson–Traction replication is,

…probably one of the very few occasions in the history of science in which the philosopher's ideal of replicability has been attained.

In 1961, Simpson had Traction watched while Traction unsuccessfully tried to repeat a biochemistry experiment concerned with protein synthesis.

Of course, since Broad and Wade's remark was made, there has been the saga of cold fusion. Many laboratories around the world tried to repeat the cold fusion experiment by Pons and Fleischmann – see Close (1990) or Amato (1993). Ordinarily, no scientist would have dreamt of trying to replicate a poorly reported experiment. The lure of cheap, relatively pollution free energy in abundance, was an exceptional motivation.

Historically the frequency of external replication work in software engineering research has been low. For example, no mention of external replication studies were made in Sharpe et al.'s (1991) investigation of the characteristics of empirical software maintenance studies between 1980 and 1989, nor in Roper's (1992) selected annotated bibliography of software testing.

More recently, even with the advent of a specialist journal such as the Empirical Software Engineering journal, the frequency of external replication work remains low, with fewer than 15 publications specifically addressing replication since the inception of the journal in 1996. A systematic survey of controlled experiments in software engineering between 1993 and 2002 by Sjoberg et al. (2005) found only twenty studies claiming to be replications of which only nine were external replications. Interestingly, six of these nine external replications are said to have failed to confirm the results of the original experiment.

This relative lack of output is likely because of the effort and resources needed to conduct an experiment, the lack of availability of laboratory packages of experimental materials, and last, but perhaps not least, the lack of glamour associated with replicating the work of others.

4.3. The Unattainability of Exact Replication

Care must be taken, however, to clarify what is meant by replication. The Universe is forever changing. Human observers and subjects are unique (Brooks (1980) and Curtis (1980) report on empirically discovered programming ability differences ranging from 4–1 to 25–1). There is no end to the number of measurements that can be made to describe the experimental setting. The art of experimental science is in making neither errors of commission or omission. Accuracy of observations can always be improved upon until such time as the Uncertainty Principle becomes important. Strictly speaking, it is more correct to talk of partial replication and the goal of performing as near exact replication as possible. Exact replication is unattainable.

According to Broad and Wade, exact replication is an impractical undertaking because the recipe of methods is incompletely reported, because to do so is very resource intensive, and because credit in science is won by performing original work. They do, however, draw attention to the important activity of improving upon experiments. They state,

> Scientists repeat the experiments of their rivals and colleagues, by and large, as ambitious cooks repeat recipes - for the purpose of improving them. All will be adaptations or improvements or extensions. It is in this recipe-improvement process, of course, that an experiment is corroborated.

With respect to poor statistical power levels caused by too few subjects, Baroudi and Orlikowski (1989) qualify this and note,

> Where a study fails to reject a null hypothesis due to low power, conclusions about the phenomenon are not possible. Replications of the study, with greater power, may resolve the indeterminacy.

Statistical power is the probability that a particular experiment will detect an effect between the control group (e.g. no use of inheritance) and the treatment group (e.g. use of inheritance). Calculations of statistical power probabilities depend on how many subjects take part, the size of any effect, and the p-value used in statistical tests (often 0.05). If the effect size is not large, and too few subjects are used, statistical power may be much less than 0.8 (a typical recommended level). The effect may go undetected. A replication with twice the number of subjects may boost the power level beyond 0.8 so that there is now a good chance of detecting the effect – at least eight out of ten experiments will detect the effect. In pioneering experimental work, it can be difficult knowing what effect size to expect, and it becomes the duty of the investigator to use as many subjects as is practically possible.

4.4. An Example: Our Replication of Korson's Experiment

Korson (1986) and Korson and Vaishnavi (1986) designed a series of four experiments each testing some aspect of maintenance. The experiment which was of greatest interest to us (Experiment 1) was designed to test if a modular program used to implement information hiding, which localizes changes required by a modification, is faster to modify than a non-modular but otherwise equivalent version of the same program. The non-modular (or monolithic) program was created by replacing every procedure and function call in the modular version with the body of that procedure or function. Programmers were asked to make functionally equivalent changes to an inventory, point of sale program – either the modular version (approximately 1,000 lines long) or the monolithic version (approximately 1,400 lines long). Both programs were written in Turbo Pascal. The changes required could be classified as perfective maintenance as defined by Lientz and Swanson (1980) i.e. changes made to enhance performance, cost effectiveness, efficiency, and maintainability of a program. Korson reckoned that the time taken to make the perfective maintenance changes would be significantly faster for the modular version. This is exactly what he found. On average, subjects working with a modular program took 19.3 min to make the required changes as opposed to the 85.9 min taken by subjects working with a monolithic version of the program. With a factor of 4 between the timings, and with the details provided in Korson's thesis, we were confident that we could successfully externally replicate Korson's first experiment.

Our external replication (Daly et al., 1994b), however, shocked us. On average, our subjects working with the modular program took 48 min to make the required changes as opposed to the 59.1 min taken with the monolithic version of the program. The factor between the timings was 1.3 rather than 4 and the difference was not found to be statistically significant.

To determine possible reasons for our failure to verify Korson's results, we resorted to an inductive analysis. A database of all our experimental findings was built and data-mining performed.

A suggested relationship was found between the total times taken for the experiment and a pretest that was part of subjects' initial orientation. All nine of the monolithic subjects appeared in the top twelve places when ranked by pretest timings. We had unwittingly assigned more able subjects to the monolithic program and less able subjects to the modular program. Subject assignment had simply been at random, whereas in retrospect it should have also been based on an ability measure such as that given by the pretest timings. The ability effect interpretation is the béte noir of performance studies with subjects and researchers must be vigilant regarding the lack of homogeneity of subjects across experimental conditions.

Our inductive analysis also revealed quite different approaches taken to program understanding by our subjects. Some subjects were observed tracing flows of execution to develop a deep understanding. We had evidence that the four slowest modular subjects all tried to understand the code more than was strictly necessary to satisfy the maintenance request. Others worked very pragmatically and focused simply on the editing actions that were required. We call this pragmatic maintenance. Our two fastest finishers with the monolithic program explained in a debriefing questionnaire that they had no real understanding of the code.

Our inductive analysis revealed at least two good reasons as to why we did not verify Korson's results and taught us many valuable lessons about conducting experimental research with human subjects. We were motivated to develop an experiment that would be easily replicable, and which would show once and for all that modular code is superior to monolithic code, but it was clear to us that it was more important to understand the nature of pragmatic maintenance. How do software maintainers in industry go about their work? Is pragmatic maintenance a good or bad thing?

5. A Simple Extension to Basili et al.'s Framework

As stated earlier, we are not concerned here with replication as it applies to an individual experimental design.

What we mean by internal replication is when researchers repeat their own experiments. For example, Korson (1986) and Korson and Vaishnavi (1986) claimed to have succeeded in providing internal replicability and stated,

> …the study has demonstrated that a carefully designed empirical study using programmers can lead to replicable, unambiguous conclusions.

Internal replications involving an evolutionary series of experiments have some confirmatory power. In many areas of science, internal replications, carried out either by design, or as part of a program of research, or because the sensitivity of the results required improving, are relatively commonplace.

By external replication we mean published experiments carried out by researchers who are independent of those who originally carried out the empirical work. Greater confirmatory power inevitably comes with external replications.

Exact replication is unattainable, so it is important to consider and categorise the differences.

First, researchers must consider the experimental method. Should a similar or alternative method be used? A basic finding replicated over several different methods carries greater weight. As Brewer and Hunter (1989) have stated,

> The employment of multiple research methods adds to the strength of the evidence.

Does a keystroke analysis of a software engineering task yield the same conclusions as observing users' performance on the task? Are the conclusions the same as those obtained from a questionnaire survey of users who have performed the task?

As a first step, the existing method could be improved. For example, the replication might add a debriefing session with subjects after the formal experiment is over if no such debriefings too place during the original experiment. Such debriefings can provide many useful insights into the processes involved. This type of improvement does not compromise the integrity of the replication.

Second, researchers must consider the task. Should a similar or alternative task be used? A basic finding replicated over several different tasks carries greater weight. As Curtis (1980) has stated,

> When a basic finding…can be replicated over several different tasks…it becomes more convincing.

Does a complex refactoring task yield the same conclusions as a simple refactoring task?

Or should the task be improved by, for example, making it more realistic? For example, rather than refactor a small program of a few hundred lines, refactor widely used open source software of many tens of thousands of lines of code.

Third, researchers must consider the subjects. For example, should a similar or alternative group of subjects be used? A basic finding replicated over several different categories of subjects carries greater weight. Does working with undergraduates produce the same conclusions as working with postgraduates? Are the conclusions the same as those obtained working with professional software engineers?

Or should the group of subjects be improved by, for example, by using more subjects or more stringent criteria for participation?

A comprehensive framework for experimentation in software engineering was established by Basili et al. (1986). The four main phases of the framework are: definition, planning, operation, and interpretation.

In the definition phase, a study is characterized by six elements: motivation, object, purpose, perspective, domain, and scope. For example: A motivation might be to understand the benefits of inheritance. The object might be the maintenance process. The purpose might be to evaluate. The perspective might be that of the software maintainer. The domain might be the individual programmer working on a program. The scope might be several programmers working on several programs, which captures the notion of internal replication within an individual experimental design.

In the planning phase, a study is characterised by design, criteria, and measurement. For example: A 2×3 factorial design might be used if we have several observations from two types of programmers (inexperienced and experienced) across three types of programs (no existing inheritance, inheritance of depth three used, inheritance of depth five used). Criteria might be the cost of implementing a maintenance request. Measurement might be the time taken to fulfill the request, as well as programmers' views on the ease or difficulty of making the code changes.

In the operation phase, a study is characterised by three elements: preparation, execution, and analysis. For example: In preparation, a pilot study might be performed to check that implementing the maintenance request does not take an excessive amount of time. In execution, start and end times might be recorded and programmers' views taken in debriefing sessions. In analysis, a 2×3 analysis of variance might be applied and statistical results compared with programmers' views.

In the interpretation phase, a study is characterised by three elements: interpretation context, extrapolation, and impact. For example: The context might include the results of other published work on the maintenance of object-oriented programs. Extrapolation might suggest that the results from the laboratory study are generalizable to industry settings because professional programmers were employed in the study. Impact might involve applying the results in an industrial context. Basili et al. also point to another possible impact: that of replicating the experiment. They, however, do not explicitly distinguish between replication by the original experimenters (internal replication) and replication by independent researchers (external replication). We propose their framework should be extended to distinguish between internal and external replication and its various forms where method, task, and subjects can each be either similar, alternative, or improved. So, for example: Under impact in the interpretation phase, the original experimenters might declare their intention to (internally) replicate the experiment with an alternative group of subjects or they might declare that the experiment needs now to be externally replicated. Under motivation in the definition phase, independent researchers might declare a motivation to verify findings by externally replicating a study but with an improved method.

We believe it unnecessary at this stage to work with more detailed categorizations of replication. We note that Sjoberg et al. (2005) chose to categorise replications simply as close or differentiated. By close replications they mean that as far as possible the known conditions of the original experiment are retained. By differentiated replications they mean variations are present in key aspects of the experimental conditions such as the kind of subjects used.

Of course, if too many alternatives are used, or if the scale of any recipe-improving is too substantial, it becomes debatable whether the study counts as a replication. Initially, the power of confirmation will be high with external replication studies but there will come a point when a result is so well established that the replication ceases to have research value and the experiment should be moved from the research laboratory into the teaching laboratory.

Across the vector of (method, task, and subjects), we categorize our Korson (Daly et al., 1994b) replication as an example of (improved, similar, similar). The method is categorized as improved because we debriefed our subjects.

6. Reporting for Replications

Once an experiment has been performed, analyzed and the time comes for writing the findings, the researcher must provide as much detail surrounding the empirical work as possible in order to allow others to replicate. Jedlitschka and Pfahl (2005) have reviewed reporting guidelines for controlled experiments in software engineering, as is described elsewhere in this book, and present a proposal for a standard. As a minimum, their guidelines on the reporting of experimental design, analysis, and interpretation should be followed.

Unfortunately, numerous empirical studies in the software engineering literature are lacking in that the experimental methods are poorly reported so that it is impossible to perform an external replication study. For example, instructions and task materials given to subjects may not be given in full, or may otherwise be unobtainable. Various authors in the past have criticised poor reporting, for example Basili et al. (1986) and MacDonell (1991).

In our Korson replication (Daly et al., 1994a), we found problems with several details which prevented the fullest possible analysis and interpretation of both Korson's results and ours. Reporting inadequacies with the Korson experiment were:

1. The experimenter employed monitors to time his subjects, and sort out problems which might arise with hardware failure and the like. It was not reported, however, whether these monitors controlled when a subject was ready to move from one experimental phase to the next, or simply just noted each phase time. Such information would have prevented speculation about monitor variability across the two studies.
2. Subject selection criteria was subjective in that almost any computer science student who had completed a practical Pascal programming course could have met it. For example, one criterion was "an amount of programming experience." This should have been more objective by stating the minimum experience required, for example at least 2 years programming experience at college level. This may have reduced subject variability.
3. Expert times for testing the program were not published. There were three separate ways to test the program, one way taking much longer than the other two. A comparison of results is required in order to explain variability that might have arisen.
4. Pretest results were not published. This would have made important reading as all subjects performed the same task; this would have allowed a direct comparison with our subjects' times, and hence a direct comparison of the ability of our subjects to the original subjects. When timings such as these are collected they should always be published.
5. It was not made clear what was verbally communicated to the subjects prior to the experiment: was additional information given to them, were any points in the instructions highlighted, or was nothing said?

Of these reporting inadequacies, only the one regarding subjection selection is explicitly addressed by the guidelines proposed in Jedlitschka and Pfahl (2005). This illustrates the difficulties in conveying all necessary information required for external replication.

The original researcher, Korson, however, went much further than many researchers in reporting experimental details, and he must be commended for that. In his thesis he published his code for the experiments (both the pretest and the experimental code), and the instructions for both the pretest and experiment. He published individual subject timings rather than just averages, along with the statistical tests and their results. So, the original researcher has presented the major issues surrounding his experiment, but has unfortunately omitted details preventing the fullest possible interpretation of his work and the external replication.

We believe it is impractical to convey all the information necessary for external replication in a journal or conference paper. Experimental artifacts under consideration such as designs, code, instructions, questionnaires, and the raw data, would typically add too many pages as appendices. Such information is best conveyed over the internet as a downloadable laboratory package along with any underlying technical report or thesis. With a laboratory package in place, original researchers can more easily conduct internal replications, independent researchers more easily conduct external replications, and meta-analysts more easily combine raw data. Work by Basili et al. (1999) is exemplary in this regard, with the availability of laboratory packages (http://www.cs.umd.edu/projects/SoftEng/ESEG/downloads.html) stimulating a small family of internal and external replications and a consequent improved understanding of perspective-based reading. Without a laboratory package in some form, an experiment is unlikely ever to be verified through internal or external replication. Given the scale of effort and resources required to conduct an experiment, not to facilitate reuse of the experimental artifacts, by providing a laboratory package, seems folly.

We agree with Basili et al. (1999) that somewhere in the laboratory package, validity threats should be detailed so that these may be addressed in future replication attempts. There is no advantage in performing a close replication – similar, similar, similar – of an experiment where a serious validity threat is present. Making an improvement to address a serious threat will yield a better experiment and results.

We also recommend that any laboratory package should report even seemingly minor details, for example, verbal instructions made at the beginning of an experiment, to enable others perform an external replication. There may be times, however, when the only way reporting inadequacies are actually discovered is by replicating an experiment and analysing the results.

7. Conclusions

Basili et al. (1986) established a comprehensive experimental framework for software engineering in which replication is recognised in the scope of an individual experiment and as an impact on future work. We have proposed a simple extension to this framework to explicitly recognise internal and external replication and its various forms: similar, alternative, improved, across method, task, and subjects. This extension applies to the motivation and impact subsections of the framework.

Routinely we are told Tool X or Technique Y is a panacea to many of software engineering's problems, but where is the accompanying empirical evidence that can stand scrutiny, that has been verified by an independent research team? We conclude that there exists only one route for empirical software engineering to follow: to make available laboratory packages of experimental materials to facilitate internal and external replications, especially the latter, which have greater confirming power. The work of the replicator should be seen as glamorous not gruesome. By verifying results, so experiments can be subsequently crafted which software engineering students can repeat as laboratory exercises. If results are not verified, we need not be too despondent. As with our replication of Korson's experiment, it is very likely that the real issue requiring investigation comes to the fore. And those involved in conducting the replication will have improved their investigation skills enormously.

References

I Amato. Pons and fleischmann redux? *Science*, 260:895, 1993.

JJ Baroudi and WJ Orlikowski. The problem of statistical power in MIS research. *MIS Quarterly*, 13:87–106, 1989.

VR Basili, RW Selby, and DH Hutchens. Experimentation in software engineering. *IEEE Transactions in Software Engineering*, 12(7):733–743, 1986.

VR Basili, F Shull, and F Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, 1999.

J Brewer and A Hunter. *Multimethod Research: A Synthesis of Styles*. SAGE Publications, Newbury Park, CA, 1989.

W Broad and N Wade. *Betrayers of the Truth*, page 17 and 81. Oxford University Press, New York, 1986.

RE Brooks. Studying programmer behavior experimentally: the problems of proper methodology. *Communications of the ACM*, 23(4):207–213, 1980.

A Brooks and P Vezza. Inductive analysis applied to the evaluation of a CAL tutorial. *Interacting with Computers, the Interdisciplinary Journal of Human-Computer Interaction*, 1(2):159–170, 1989.

DN Card. Software quality engineering. *Information and Software Technology*, 32(1):3–10, 1990.

F Close. *Too Hot to Handle The Story of the Race for Cold Fusion*. W H Allen Publishing, London, 1990.

HM Collins. *Changing Order Replication and Induction in Scientific Practice*, pages 19, 35, 43. SAGE Publications, London, 1985.

B Curtis. Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68(9):1144–1157, 1980.

J Daly, A Brooks, J Miller, M Roper, and M Wood. An external replication of korson's experiment. Research report EFoCS-4-94, Department of Computer Science, University of Strathclyde, Glasgow, 1994a.

J Daly, A Brooks, J Miller, M Roper, and M Wood. Verification of results in software maintenance through external replication. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 50–57. IEEE, Los Alamitos, CA, 1994b. ICSM'94.

M Goldstein and Inge F Goldstein. HOW WE KNOW *An Exploration of the Scientific Process*, page 207. Plenum Press, New York and London, 1978.

SM Henry and M Humphrey. A controlled experiment to evaluate maintainability of object-oriented software. In *Proceedings of the IEEE Conference on Software Maintenance*, pages 258–265, 1990.

TH Huxley. We are all scientists. In H Shapley, S Rapport, and H Wright, editors, *The New treasury of Science*, page 14. Collins, London and Glasgow, 1965.

A Jedlitschka and D Pfahl. Reporting Guidelines for Controlled Experiments in Software Engineering. Verification of results in software maintenance through external replication. In *International Symposium on Empirical Software Engineering*, pages 95–104. IEEE, Los Alamitos, CA, 2005. ISESE 2005.

JPJ Kelly, TI McVittie, and WI Yamamoto. Implementing design diversity to achieve fault tolerance. *IEEE Software*, 8(4):61–71, 1991.

TD Korson. *An Empirical Study of the Effects of Modularity on Program Modifiability. PhD thesis, College of Business Administration*, Georgia State University, 1986.

TD Korson and VK Vaishnavi. An empirical study of the effects of modularity on program modifiability. In E Soloway and Iyengar S S, editors, *Empirical Studies of Programmers: First Workshop*, pages 168–186. Ablex Publishing Corporation, Norwood, NJ, 1986. A Volume in the Ablex Human/Computer Interaction Series.

J Lewis, S Henry, D Kafura, and R Schulman. An empirical study of the object-oriented paradigm and software reuse. *OOPSLA*, 184–196, 1991.

B Lientz and E Swanson. *Software Maintenance Management*. Addison-Wesley, Reading, MA, 1st edition, 1980.

SG MacDonnell. Rigor in software complexity measurement experimentation. *Journal of Systems and Software*, 16:141–149, 1991.

DR Moreau and WD Dominick. A programming environment evaluation methodology for object-oriented systems: part ii – test case application. *Journal of Object-Oriented Programming*, 3(3):23–32, 1990.

KR Popper. *The Logic of Scientific Discovery*. Hutchinson, London, revised edition, 1968.

M Roper. Software testing: a selected annotated bibliography. *Software Testing, Verification and Reliability*, 2:113–132, 1992.

DA Scanlan. Structured flowcharts outperform pseudocode: an experimental comparison. *IEEE Software*, 6(5):28–36, September 1989.

S Sharpe, DA Haworth, and D Hale. Characteristics of empirical software maintenance studies: 1980–1989. *Journal of Software Maintenance: Research and Practice*, 3:1–15, 1991.

B Shneiderman, R Mayer, D McKay, and P Heller. Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, 20(6):373–381, 1977.

DIK Sjoberg, JE Hannay, O Hansen, VB Kampenes, A Karahasanovíc, N-K Liborg, and AC Rekdal. A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering*, 31(9):733–752, 2005.

GP Smith. The problems of reduction and replication in the practice of the scientific method. *Annals of the New York Academy of Sciences*, 406:1–4, 1983.

University of Maryland Experimental Software Engineering Group. Lab packages. http://www.cs.umd.edu/projects/SoftEng/ESEG/downloads.html