Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Software Technology 47 (2005) 233–244
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Replicating software engineering experiments: a poisoned chalice or the Holy Grail

James Miller*

Department of Electrical and Computer Engineering, Software Technology, Engineering and Measurement Research Centre (STEAM), University of Alberta, 2nd Floor, ECERF, 9107-116 Street, Edmonton, Alta, Canada T6G 2V4

Received 20 May 2004
Available online 5 October 2004

Abstract

Recently, software engineering has witnessed a great increase in the amount of work with an empirical component; however, this work has often little or no established empirical framework within the topic to draw upon. Frequently, researchers use frameworks from other disciplines in an attempt to alleviate this deficiency. A common underpinning in these frameworks is that experimental replication is available as the cornerstone of knowledge discovery within the discipline. This paper investigates the issues involved in accepting this premise as a fundamental building block with empirical software engineering and recommends extending the traditional view of replication to improve the effectiveness of this essential process within our domain.

Keywords: Software engineer; Real world settings; Empirical values

1. Introduction

Undertaking empirical work to understand the software engineering process is an increasingly common activity. However, due to the nature of most software engineering studies, drawing reliable conclusions from a single study is inherently dangerous. A single study will possess a large number of parameters, some controlled, some completely unconstrained. Any one of which can potentially cause a significant change in the result of the study; and hence their effect must be controlled, eliminated or understood, if reliable results are to be derived. The normal method of achieving this objective is to repeat the study, often while changing some of the parameters, to see if the original result is stable with regard to repetition and alteration of some of its components. In traditional scientific disciplines, this is known as replication or more accurately as partial replication. Replication comes in two basic forms: internal (the repetition is undertaken by the same researchers) or external (undertaken by other researchers)[1]. Although replication can be applied to any form of study, its effects are most powerful in controlled experiments or simulation studies; and hence this paper will limit itself to these types of studies. This power is derived from the ability to accurately repeat the study, due to the precise definition of variables of interest, and the comprehensive description of the experimental procedure and approach. In fact, unless explicitly stated, this paper will assume that the study under investigation is a controlled experiment, with human subjects. Results from these further studies either support or question the claims arising from the original study. When the results are similar, confidence is produced in both the validity of the experimental process and the original hypothesis. Without this confirming power, empirically based claims in software engineering should be regarded as having at best limited value and at worst with suspicion and mistrust.

* Tel./fax: +1 780 492 5580.
E-mail address: jm@ece.ualberta.ca.

[1] Unless otherwise stated, replication will mean external replication throughout this paper.

0950-5849/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2004.08.005

The potential problems with single software engineering empirical studies[2] are three-fold:

- Low statistical power.
- Large number of potential covariates with the treatment variable.
- Verification of the process and products of the study.

The normal way to tackle these limitations is by replication of the study.

1.1. Statistical power

Statistical Power is a fundamental consideration in any experimental design; unfortunately it is often ignored within a software engineering context. When the experimenter expects to perform parametric tests, it is simply a function of the significance criterion, the sample size and the expected effect size. Like the significance criterion, the level of statistical power should be set before undertaking the experimental procedure. This in turn describes the amount of risk of encountering a Type 2 error that is acceptable for a particular experiment. (See Cohen [9] for an excellent introduction to statistical power, and its application to subject-based experiments.) The fact that a phenomenon under study exists does not guarantee that any significant result will be produced. A high statistical power level means that a statistical test has a high probability of producing a statistically significant result when in fact the effect under study does exist. In other words, a Type 2 error is unlikely to be committed. Similarly, if an effect does not exist, the researcher has a solid statistical argument for accepting the null hypothesis—this is not the case, if the experiment has low statistical power.[3] Additionally, when conducting an experiment with low statistical power, often due to a small sample size with respect to the expected effect, many experimental decisions are based upon weak evidence. For example, the final decision to use parameter tests will often be based upon the result of a test of the parametric nature of the data. Unfortunately, in an experiment having a low statistical power, this test will often possess a low statistical power, and hence the decision carries an extremely high risk factor.

The calculation of the statistical power level is primarily dependent on the significance criterion (control of Type 1 errors), the sample size and the effect size. Hence, the researcher has only one parameter under their control—the sample size. Hence, if a researcher is attempting to produce a well-balanced experimental design only two paths exist[4]:

- Undertake a single experiment with a very large sample size- this is typically the approach taken in many medical sciences.
- Undertake an experiment with a small to medium sample size—typically found in the software engineering literature; and organise for the experiment to be widely replicated. Assuming that the results of the experiments can be shown to be consistent, the experimenter can informally consider that the sample sizes are additive; and hence the power of the series of experiments can be considered to be much greater than the single study.

The reader should not infer from the above statements that further replications, again by formal experimentation, are the only option open to them. As an alternative to repeated investigations using the same research method, the multi-method[5] [4] approach has gained much support in the worlds of psychology and sociology. These approaches help to address the perceived weaknesses of single studies by attacking problems with:

> "… An arsenal of methods that have non-overlapping weaknesses in addition to their complementary strengths" [4].

The interested reader should consult Brewer and Hunter [4] for a detailed description of the generic approach; and Daly [12] and Wood et al. [47] for discussions on the application of the approach to solving software engineering problems. This theme is expanded upon later in the article.

1.2. Potential covariates

Unfortunately, software engineering experiments suffer from having large numbers of factors, which may have an impact on the final result. These factors may be a direct covariate with the treatment variable or produce a secondary (or even tertiary) effect on the result. (See Campbell and Stanley [7] for an extensive treatment of this area.) Good experimental design attempts to limit or minimise these threats, but the number and scale of many of the threats are such that they can only be explored with large amounts of resources. Hence, the need for multiple researchers to explore these factors, this exploration is traditionally undertaken by replication of the original experiment. Additionally, as discovering a comprehensive hypothesis often requires an empirical study, many of the factors may only be hypothesised during the data analysis of the original experiment or alternatively remain undiscovered. For example, consider comparing two inspection processes, the performance of the processes may be dependent on the types of defects found in the document during the experiment, and hence the need to repeat the experiment

[2] These potential problems may pose less of a threat in other disciplines.

[3] Note this is a simplification of the exact case, as while power and Type 1 error rates are related, one is not a direct function of the other.

[4] Assuming the effect size is fixed and not massive.

[5] Also known as triangulation.

with a different distribution of defects. The type of defects found in a document may, in turn, be highly influenced by the type of document under inspection. Hence, more replications are required! Suspected covariates must be explored, if reliable conclusions are to be reached. Are defects in requirements and code documents likely to conform? If the researcher is not careful, the number of potential factors can exponentially explode. Somehow, a consensus must be reached between the need to limit the amount of work and the risk of a covariate remaining undetected. Even with this 'consensus' in place the amount of work is too great for an individual researcher to undertake, and hence the need for external replication. If these external replications are not undertaken the whole program of research risks drawing erroneous conclusions, the balancing act between effort and risk of accepting false conclusions is difficult, but must be undertaken! See Lindsay and Ehrenberg [30] for a discussion about generalising from replications.

1.3. Verification of the process and products of the study

This concern is in common with traditional scientific disciplines The basic question is: 'How can we be sure that an experiment is not methodologically flawed?' Every stage of an experiment from initial background research to the publishing of the results is prone to error. Verification of the methodology is principally by repeating the experiment and obtaining similar results. (Even in a well-designed software engineering experiment, this is a difficult, and risky, undertaking.) Hence, often researchers demand that the experimental results are reported in a way which allows the external replication (and hence potential verification) of the experiment. This has, potentially, massive implications for the reporting of experimental work. This is discussed later. An excellent example of this verification in operation was Scanlan's [42] replication of an experiment by Shneiderman et al. [43]. The original experiment by Shneiderman et al. strongly suggested that flowcharts were of limited use; and duplicated the information in the program itself. According to Scanlan, this work directly led to the decline of flowcharts as a mechanism for representing algorithms. Scanlan, however, identified several fatal flaws in the original experiment; for example that time was not a measurable dependent variable (subjects were allowed as much time as they required). Scanlan re-designed the experiment 'correcting' this deficiency and claims that his study shows that: 'significantly less time is required to comprehend algorithms represented as flowcharts (when compared with source code only)'. Readers are encouraged to review the two original papers, as they represent an excellent case study in the application of replication in software engineering experiments.

Software engineering is not alone in wrongly accepting information from single studies, see Broad and Wade [5] for examples from psychology. In their book, Broad and Wade attempt to show a picture of scientific endeavour as a strictly logical objective process, which is rigorously verified by peer review and examination. This is the ideal rather than everyday reality. They review several cases where it is believed that scientific claims were fictitious. Scientific replication and re-examination of the original hypothesises have subsequently rebuffed, or cast doubt on the validity of, these claims. Perhaps the best known case involves Pons and Fleischmann's 'cold fusion' experiments [1,8]. Their experiments have caused much controversy in the physics world, and opinion is divided about the validity of work. As with much experimentation, decisions about the 'value' of the work are not straightforward, and are certainly not black and white. To quote a leading physicist [1], perhaps the view of the physics world can be summed up as:

> "Most people would say [Pons and Fleischmann] either do not know what they are doing, they are frauds, or they are right."

The basic problem, regardless of the subject, is that the cost of replication is high; and what recognition do the replicators receive for their endeavours? In the field of cold fusion, the lure of discovering bountiful amounts of cheap energy may well be sufficient, but what can a software engineering researcher expect? Hence, if the field wants to foster a spirit of encouraging replicating then recognition of the valuable role of replicators must be recognised. Journals and conferences willingness to publish replications can provide assistance in encouraging a more widespread pursuit of experimental verification by replication. Nevertheless, much greater recognition is still urgently required!

2. What is replication?

While this may only seem an excellent question for philosophers to ponder, for practical researchers a precise definition is also extremely important and certainly varies from topic to topic. In traditional hard sciences, the definition is all encompassing—an exact replication. Given that the world changes and that reporting and measurement are finite, an exact replication cannot exist. In their book [5], Broad and Wade sight the Simpson-Traction replication as an example of a (nearly) exact replication; no comparable example exists within software engineering. Hence, it is more technically correct to describe replication attempts as partial replications. In traditional sciences, it is often the case that the replicator is attempting to minimise the variation between their replication and the original experiment; with the one proviso that to derive personal benefit from the replication, the investigator is seeking to improve on the original; and hence extend the state of the art via the replication.

What does this imply for replication within software engineering? Via the inability to duplicate exactly

and the desire to improve the experiment, the traditional sciences have already moved away from the model of replication as an exact repeat. In software engineering these 'reasons' for variation tend to be further exaggerated. For example, many experiments use human subjects often in a programming capacity, several studies (e.g. [6,11]) have shown that wide variations exist between the performance of subjects, even within 'peer' groupings. Hence, almost by definition, replication in software engineering is fated to be further removed from exact duplication than in the traditional sciences. Is this important? The short answer is no. What is important is that the replication examines the same (or a generalised or specialised version of the same) hypothesis. After all the purpose of the exercise is to find out more about the outcome, to a certain extent the intermediary details are unimportant. The replicator in software engineering has to strike a personal balance between two competing forces. On one hand, the desire to repeat the original experiment; this route allows for the re-use of existing (from the original experiment) materials, experimental designs, etc. Additionally, this route allows the replicator to access directly the 'quality' of the original experiment by 'following in the original experimenter's footprints'. The other hand suggests that by re-using the materials, etc. from the original experiment an opportunity is lost to generalise the hypothesis by investigating it in another scenario, or viewpoint, etc. Additionally, this re-use of materials, processes and procedures sets up correlations between the experiments, limiting the statistical independence of the two studies and hence reducing any confirmatory evidence that is gained by undertaking the second experiment. It also fails to examine additional values from potential covariates—for example, if we are investigating a static defect finding technique and the original experiment uses source code within the experiment, then the replication should use a specification or design document rather than another piece of source code (or even worse—the same source code). This approach provides us with an excellent insight into the applicability of the technique, assuming that the change does not invalidate the hypothesis under investigation.

Exploring this trade-off (between pseudo-exact and purposely inexact replications) is the main focus of this paper; the paper's conjuncture is that the field is failing to appreciate this spectrum of alternatives. Instead, the topic's norm is to accept a hard science-oriented interpretation of the spectrum-strive for as near an exact replication as possible. While this may have rewards in the short term (minimise the effort associated with replication), the paper argues that the topic is failing to reap a significant reward from this approach, and that in the longer term the approach will tend to minimise the knowledge discovery gains from empirical studies and their associated replications.

3. Designing and reporting experiments for replication

When designing and reporting an experiment it is essential that the experimenter should consider what is required by a potential replication. This requirement is also found in other fields. Unfortunately, information from these other fields is not encouraging. For example, reporting guidelines are well established for describing computational experiments with mathematical software [22]. Despite these guidelines being established over 20 years ago, recent articles have suggested that they have had little or no impact; and that the field is still failing to produce reports, which even conform to minimal standards of reproducibility reporting [19,23]. So what is the basic problem? Although it is difficult to be certain, the most likely candidate is—time. It is expensive to reproduce all the information required to fully describe an experiment. Additionally, software engineering, in general, faces even greater problems in capturing all the ingredients given the large numbers of variables that can influence its experiments. Hence, although, desirable this aspiration may not be feasible in the modern world of urgent demands. Having said this, a few 'brave' software engineering researchers have attempted to produce 'replication packages' to increase the reproducibility of their work. The package by Kamsties and Lott [25] represents an excellent example of such a package. Although the package provides much useful information to allow a replication, it is difficult to see how such a package could ever be complete in the software engineering context.

Although guidelines themselves are insufficient to ensure high standards of reproducibility reporting, they are an essential component in attempting to promote good practice in this area. The field is fortunate that two specific proposals exist, which could be used as reporting guidelines. The first set of guidelines was produced by Basili et al. [2]; this scheme categorises the experimental process into four phases: Definition, Planning, Operations and Interpretation. These phases are in turn refined in greater detail, for example definition is decomposed into the definitional elements: motivation, object, purpose, perspective, domain and scope. In their paper, Basili et al. demonstrate their schema's ability to succinctly summarise experiments. This schema is perhaps better suited for this purpose than for the purpose of ensuring reporting reproducibility, as the lowest level of decomposition of the phases is still rather high-level when compared with the precise, often low-level, decisions an experimental designer must consider. Although not derived from the work by Basili et al., the second schema (introduced by Lott and Rombach [31]) has many similarities, including the use of Goal-Question-Metric (GQM) to derive the subsequent schema. Specifically, Lott and Rombach used GQM to derive the design of their defect detection experiment, the resulting schema being a sub-component of the overall design. Hence, sub-components of their

Table 1
Overview of Lott and Rombach's schema

1.	Goals, hypothesis and theories
	A. Aspects of goal
	B. Hypotheses
	C. Theories
2.	Experimental plan
	A. Experimental design
	B. Treatments[a]
	C. Objects[b]
	D. Subjects
	E. Data collection and validation
	F. Data analysis
3.	Experimental procedures
	A. Training activities
	B. Conducting the experiment
	C. Feedback to subjects
4.	Results
	A. Data
	B. Interpretations

[a]These are explicitly defect detection treatments in the original paper.
[b]Again in the original paper, these are explicitly defect detection objects.

schema contain explicit references to their experiment,[6] but the generalisation of these components is extremely straightforward. Table 1 is an outline of the top-level of the schema; the interested reader is referred to their paper for the remaining details and discussion of the schema deployment. As can be seen, the basic four phase structure has been retained, but the sub-sections now contain greater detail and direction. As with any schema, criticisms can be levelled such as the data analysis section assuming that traditional statistical approaches will be utilised. Nevertheless, it is believed that it provides an excellent starting point in improving the reporting standard of software engineering experiments!

However, while the schema provides an excellent template for reporting experimental results from the original experiment, it only implicitly addresses the replication trade-off or spectrum. The danger exists that researchers may view the schema as a list of topics to consider when thinking about a replication—therefore failing to actively consider many alternatives within the spectrum of possible replication strategies. Researchers need to be aware that a schema for reporting a single experiment and a 'framework of replication' are different; and while the former can be a subset of the latter, the inverse situation is not viable.

4. Comparing results from replications[7]

Regardless of the mechanism used to conduct the replication, the replication only has substantial meaning, and interest, if it is analysed in conjunction with the original experiment.[8] Remember what we are trying to achieve with the replication, is to build up a substantial body of knowledge about a hypothesis. Additionally we are often interested in assessing the validity of the original approach.

Unfortunately, the second objective has a fundamental problem—who validates the validator!

Collins [10] in his excellent introduction to scientific replication describes the problem as:

> "The problem is that, since experimentation is a matter of skilful practice, it can never be clear whether a second experiment has been done sufficiently well to count as a check on the results of the first. Some further test is needed to test the quality of the experiment—and so forth"

Fortunately, this paradox can be resolved! If we currently ignore the internal details of the two experiments, the basic validation we are interested in is the final hypotheses. If we can show that the two experiments agree about their conclusions—then we can consider them supportive, and mutually validating. Of course, if both experiments are flawed and still agree…

Investigating our first objective (analysing both experiments) can test this question, if these two experiments can be shown to be supportive, then we can aggregate their results and expand our knowledge (of the hypothesis). If the two experiments are not supportive, then the replicator should explore the experiments for an explanation of the difference between the two results. The researcher has two alternatives: a relatively formalised statistical approach using some form of meta-analytical analysis or a more informal argument-based approach. While the formulised approach may seem the most obvious choice—the conjecture that Software Engineering experiments are suitable to this type of analysis needs to be explored.

4.1. Applying meta-analytical procedures to software engineering experiments

Despite having existed for nearly a century, the use of meta-analysis in software engineering experiments is still very uncommon. Meta-analysis provides us with a simple quantitative framework for comparing and combining experiments. Meta-analysis works at the hypothesis level, normally investigating the question: 'what is the impact of variable A on variable B?' The interested reader should see Wolf [46] for a general introduction to the field of meta-analysis. The following discussion will use a fixed-effects model, these models have the expectation, that no variance exists between the experimental results, and hence that differences are solely due to subject variability. The alternative is to use a random effects model; here inference is based upon the assumption that the studies are a random sample of the population. Unfortunately no empirical basis for preferring either model exists. But, most experts agree that the choice of model is of secondary concern; and that if the experiments are convergent then both approaches will

[6] Specifically sections 2B and 2C.
[7] This obviously includes the original experiment.
[8] Assuming no other replications exist.

produce identical results [21]. Further within these frameworks, alternative formulations exist. The interested reader should consult Hedges and Olkin [21] for a comprehensive treatment of fixed and random models; DerSimonian and Laird [14] for random models using the method of moments; and Dumouchel [15] for an introduction to using Bayesian methods for meta-analysis (both fixed and random).

When comparing two studies, meta-analysis provides a framework to determine if the two studies produce significantly different results. Although it can be applied to the results of significance testing, the use of effect size[9] estimates is strongly preferred in meta-analysis. In fact, the use of results of significance testing should only be used when the information for the evaluation of effect sizes is not available. (See Kramer and Rosenthal [28] for an extensive overview of effect sizes and their estimation.) Hence, what we are 'comparing' are the two estimates of the degree to which the phenomenon (under study) is present within the population.

As with most estimations, several formulations of effect sizes exist. The decision about which estimation to use is unimportant, and one can translate between most of the various estimates using simple equations. The two effect sizes can then be compared to attempt to answer the question—have they arisen from the same phenomenon? If the answer is (probably) no, then the analysts must either recast the analysis (by introducing one or more new variables into the hypothesis) or accept that the two experiments are incompatible, and hence search for an explanation. Under this situation, the two experiments cannot be said to provide mutual validation or confirmation. Alternatively, the comparison may suggest that the studies are compatible; in this case the analyst can continue to combine the two studies, producing a potentially more robust conclusion. If the two studies are combined, we are able to infer an increase in statistical power due to the effective increase in the number of subjects.

Section 4.1.1 provides a brief case study of analysing two software engineering experiments. Details of the meta-analysis procedure are derived from Rosenthal [40].

4.1.1. Case study

In this section, an original experiment and its subsequent replication will be outlined as an example of applying meta-analysis to two software engineering experiments. The two experiments in question are by Korson [26] and Daly et al. [13].

4.1.1.1. Review of Korson's experiment. Korson designed a series of four experiments each testing some aspect of maintenance. His first experiment was designed to test if a modular program used to implement information hiding, which localises changes required by a modification, is faster to modify than a non-modular but otherwise equivalent version of the same program. This first experiment was the experiment replicated by Daly et al.

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) in either the modular version or the monolithic version. The changes required could be classified as perfective maintenance, i.e. changes made to enhance performance, cost effectiveness, efficiency, and maintainability of a program. Korson hypothesised that the time taken to make the perfective maintenance changes would be significantly faster for the modular version. The experiment was conducted with 16 subjects, with eight subjects maintaining the modular program and eight subjects maintaining the monolithic program.

The modification process of the experiment proper contained four phases:

Phase 1 *think*: on paper, the modifications are coded noting deletions, additions and changes to the original source code.
Phase 2 *edit*: at the computer, the original program is edited to reflect the modifications made on paper.
Phase 3 *syntax*: the syntax errors are interactively removed from the modifications.
Phase 4 *logic*: logic errors are interactively debugged until the program passes a standard test.

Korson summarised his results for his first experiment as follows:

> "The study provides strong evidence that a modular program is faster to modify than a non-modular, but otherwise equivalent version of the same program, when the following conditions hold: modularity has been used to implement 'information hiding' which localises changes required by a modification."

In more detail, the modular group had a mean maintenance time of 19.3 min with a standard deviation of 8.1 min; and the monolithic group had a mean maintenance time of 85.9 min with a standard deviation of 47.8 min. Subsequently, Korson used a Wilcoxon Rank Sum test to reject the null hypothesis (information hiding has no effect on maintainability); and reports that the probability that these results are produced by chance is $p < 0.001$. Daly et al. demonstrated that parametric testing is robust with respect to the Korson data, and hence

[9] When estimated for meta-analytic purposes, effect sizes are often derived from inferential statistics (F or t values, for example) along with the associated degrees of freedom.

we will assume that it can be regarded as parametric in the subsequent meta-analysis.

4.1.1.2. Review of Daly's replication. Daly et al. chose to follow closely the experimental procedure described by Korson. Hence, apart from some minor recipe improving,[10] the experiment was as described above. In terms of results, the replication failed to reject the null hypothesis ($p < 0.4$). The detailed results are: the modular group (with eight subjects) had a mean maintenance time of 48.0 min with a standard deviation of 25.4 min; and the monolithic group (with nine subjects[11]) had a mean maintenance time of 59.1 min with a standard deviation of 27.0 min.

In conclusion, Daly et al. primarily decided to strive for a pseudo-exact replication, thus maximising its ability with regard to the *'Verification of the process and products of the (original) study"*while minimising its ability with regard to exploring*"Potential Covariates'*. An interesting question to consider is-is this strategy optimal?

4.1.1.3. Meta-analysis of the two experiments—comparative analysis. Given that both studies report the raw data obtained during the experiment, we have a very open choice of effect size estimator. This section will use Hedges' g, to briefly illustrate the procedure, which is simply defined as the difference between the means of the two groups (modular, monolithic) under investigation normalised by the pooled standard deviations of the two groups.[12]

Hence for the Korson experiment[13]: $g_a = 1.37$, and for the Daly et al. experiment[14]: $g_b = 0.30$.

As with all effect size estimates, this estimate suffers from a small bias. In the case of Hedges' g, the bias can be simply corrected for, by applying[15]

$$g^* = c(m)g$$

where $c(m)$ is

$$c(m) = 1 - \frac{3}{4m - 1}$$

where m is the degrees of freedom computed from both groups.

Hence the bias effect size estimates become: $g_a = 1.30$; $g_b = 0.29$.

Now for each of the studies to be compared, we calculate the quality $1/w$, which is the estimate of the variance of g. w can be obtained as follows

$$w = \frac{2(n_1 n_2)(n_1 + n_2 - 2)}{(n_1 + n_2)[t^2 + 2(n_1 + n_2 - 2)]}$$

where n_1, n_2 are the sample sizes of each group (modular, monolithic) and t is the familiar t statistic[16] from the experimental results.

For the two experiments: $w_a = 2.60$, $w_b = 4.13$

Finally, the comparison can be obtained from

$$\frac{g_a - g_b}{\sqrt{\frac{1}{w_a} + \frac{1}{w_b}}}$$

This equation is in fact a simple Z test, and in our case produces a result of $p < 0.1$. Since, this example is using the traditional Type 1 error rate value of 0.05, we are unable to reject the null hypothesis.

4.1.1.4. Discussion of comparative results. Hence, first impressions may be that we can accept the null hypothesis, and conclude that the two experiments are compatible. Unfortunately, because of the small number of experiments, a high risk exists that moderator, covariate or confounding variables may not be revealed. When the sample size in a meta-analysis is viewed as the number of studies, the power to detect a given variable relationship, during the meta-analysis, may be low [24]. Additionally, the moderator variables may be confounded with each other [18]. Therefore, are we really in a position to accept the null hypothesis? In general, a strong theory of the relationship between the variables is the 'best defence' of the results. However, the researcher's default position should be caution in the interpretation of the results as suggesting a causal relationship without further results.

Although the authors of the replication did not conduct a numerical meta-analysis, they decided that the results of the two experiments were incompatible. Hence, subsequently, they undertook an inductive analysis of the secondary information collected during the experiment in an attempt to find an explanation for the difference. They concluded that the allocation of subjects to group (via random allocation) was the principal factor in the difference, as the replication results were heavily impacted by the monolithic group possessing a significantly higher 'ability' (at the maintenance task) than the modular group. Therefore, they recommended that any subsequent replication of this study which deploys an allocation strategy should incorporate a mechanism to balance the two groups with regard to their ability.

Additionally the above calculations assume that the two experiments are (statistically) independent—but is this likely? Remember that the replication has re-used (nearly) all of the materials, procedures, etc. from the original

[10] Improvement of the experimental design by the replicators.

[11] Difference in group sizes is due to the experiment using a university class as its source of subjects.

[12] All quantities are reported to two decimal places.

[13] In this section, the subscript a refers to the Korson experiment.

[14] In this section, the subscript b refers to the Daly et al. experiment.

[15] In future equations, the superscript has been omitted, for the sake of readability.

[16] For comparison to the tabled value during hypothesis testing.

experiment. Hence, a large risk exists that correlations have been set up between the original experiment and the replication as the possibility exists that some aspects of the experiment will act as a (partial) covariate with the treatment variable. Unfortunately, we have little understanding of the relationship between the various components found in software engineering experiments. (Other fields report similar concerns [36,41].[17]) And hence, if we want to ensure independence then components really need to be a random selection from the total space of suitable components. If we take a less pure viewpoint then the question becomes—which dependencies are important in which situations? Again, we are in a poor position to answer this question. Probably the question has no straightforward answers, and the design of the replication (with regard to independence) must be left to the skill of the experimental designer. In addition, the pursuit of independence, almost by definition, requires the replicator to introduce new components into the experiment. This introduction helps the experiment, by broadening its scope (and hence generality), providing additional insight into potential covariates, etc.

The alternative viewpoint is 'is independence important?' Without it, meta-analysis has no foundation, and the field should really return to traditional reviews. As everyone knows, these suffer from their dependence on the subjective judgements, preferences, and biases of the reviewers and from the ill-defined procedures for undertaking the review process. Are we ready to discard the relatively clean quantitative approach of meta-analysis[18] in favour of qualitative review? Additionally, although re-using components is cheap for the replicator, is it cost-effective in the long run? As indicated above, Daly et al.re-used nearly all of Korson's components. Hence, it was cheap for them to undertake, and provided excellent verification of the original experiment. However, it did nothing with regard to investigating potential covariates; and due to dependencies, it reduced its ability to effectively increase the statistical power of the 'combined' investigations. In the long term, this author believes that these latter concerns are of much greater importance than the former. Hence, the potential replicator is strongly encouraged not to simply re-use the original experiment, but to explore the same hypothesis from a different viewpoint.

4.1.1.5. Meta-analysis of the two experiments—additive analysis. Returning to the meta-analysis, if we assume that the two studies are compatible; then we can proceed to combine the results from the studies. This is relatively straightforward, but no unique mechanism exists for achieving the combined estimate. Assuming that we are still using effect sizes, the investigator has, at least, four options:

- Simple arithmetic average.
- Average weighted by some estimate of research quality.
- Average weighted by size (degrees of freedom).[19]
- Average weighted by variance.[20]

Fortunately, all options can be covered by one simple equation

$$\frac{w_a g_a + w_b g_b}{w_a + w_b}$$

where w is the weighted component.

In our case study, Daly et al. stated that they believe that the variance of the modular group is heavily influenced by the program to be maintained. Hence, for our example we will use size as the weighting factor. This gives a combined effect size estimate of: 0.78. Most textbooks would agree that this could be interpreted as a large effect size, and hence it would have a significant impact on the maintainability of these two different types of code. But given the extremely weak evidence that the two experiments are compatible, quoting this number of evidence of a casual effect would be difficult to justify.

Perhaps it is tempting to consider this replication scenario as a unique disappointment. But unfortunately, it represents the opposite case, and is in fact the most supportive case in the literature of mutually confirming subject-based formal experiments. Previous literature [20,33] reports that the series of experiments analysed within them failed to pass the formalised comparative test. And hence, we have no real positive examples of replication in practice within Software Engineering to draw upon.

One point which should be clear throughout the discussion is that the interpretation of the results is not straightforward. In fact, despite its quantitative nature all the results require significant interpretation. This is common in most fields of statistics, but often the researcher becomes 'mesmerised' by the raw numbers. The interpretation of statistical results is a complex and inexact process, which has caused so much confusion and misinterpretation that it has given rise to its own research field-ethnostatistics. Care and caution are always required when interpreting experimental results!

4.2. Alternative: an informal approach

The evaluation of the scenario-based reading (and inspecting) approach has perhaps seen the greatest number of subject-based evaluations of any given topic within software engineering. Given the highly abstract nature of the genesis of this direction [37], we have witnessed a number of variations, or alternative implementation strategies, including [3,16,40,44]. In turn these have either

[17] In an early book Rosenthal [40] presents some weak evidence to the contrary. In a later book [41], he quotes the above references as current understanding, while ignoring his earlier work.

[18] Or similar quantitative processes, such as vote counting.

[19] Larger studies are more likely to be reliable.

[20] Small variance studies can be considered more precise.

Table 2
Comparison of results from experimental evaluations

Significant results	Non-significant results
[38]	[16]
[3]	[17]
[44]	[35]
[25]	[45]
[29]	
[38]	
[30]	
[40]	

included or spawned a number of related and independent evaluations [17,25,26,35,38] reported in refereed and highly respected journals.[21] While the experiments possess significant variations in the treatment of the topic and even within their detailed hypothesis, at their heart they all compare the genesis of the direction against the assumed industrial standard approach (Checklist-based). Table 2 presents their headline results with regard to this comparison; given the variation in the hypotheses, a traditional quantitative meta-analysis of these experiments are generally considered an unsound proposition.

Viewing Table 2, clearly no obvious result exists; the selected experiments are almost equally balanced between finding and not finding any differences between the approaches. What is missing is the detailed numerical treatment of the comparison-due to variations within the hypotheses—but was that really adding value to the comparison. Instead, these experiments have explored the initial hypotheses from a variety of angles and different instantiations. Providing us with a much more detailed and far ranging investigation of the underlying research question. Many other areas, would consider this variation of greater benefit than the numerical comparison. This type of alternative viewpoint on replication is further explored in Section 7.

5. Discussion

If we are failing to generate results that are easily replicated using traditional scientific disciplines, what are we doing wrong? Perhaps nothing! In Section 1, we outlined the principal problems with single software engineering studies, the same problems exist and are in fact amplified, if we look at the topic from a replication viewpoint. In Sections 3 and 4, we outlined excellent mechanisms for designing and reporting experiments—but clearly, given the complexity of our situation and the number of variables and difficultly (if not impossibility) of characterising them, we can never capture all of the 'facts' regardless of the quality of the framework. And even, if we could capture them, given the nature of the subjects used within the studies, we are still not in a position to derive consistently repeatable results. For example let us consider a programming experiment. Now, consider the productivity of programmers; Brooks [6] reports differences from 4 to 1 up to 25 to 1 across experienced programmers with equivalent backgrounds; also, Curtis [11] reports 25 or 30 to 1 differences in performance among programmers. If we accept that these figures are reasonable approximations to reality, consider two programming experiments which just happen to draw subjects from different ends of the spectrum, i.e. one draws a homogeneous group and one draws a highly heterogeneous group. Why would we expect their results to be statistically comparable? The work practices, thought processes and natural capabilities of these two groups are likely to introduce large quantities of noise into the statistical analysis of the hypothesis under investigation. This is then further compounded by the low statistical power that many experiments face and hey presto—we have a pretty good recipe for producing incompatible results from a statistical hypothesis viewpoint.

In common with many software engineers, I was educated in a Science Faculty and hence my natural tendency is to return to traditional scientific methods and disciplines. But other alternatives to traditional replication exist and perhaps the field should start considering that these alternatives may prove to be more fruitful in many replication situations. McGrath et al. [32] discuss the options available to researchers when exploring or validating research hypotheses. They identify eight distinguishable research strategies: laboratory experiments, experimental simulations, field experiments, field studies, computer simulations, formal theory, sample surveys and judgment tasks. They describe the relative merits of the eight strategies and correctly point out that these strategies have very different properties when applied to research hypotheses. They note that a researcher when conducting a hypothesis exploration is really seeking to maximize three mutually-conflicting goals during their design of the evaluation, specifically:

1. Precision in the control and measurement of the variables related to the activity of interest;
2. Generalisability of the results with respect to entire population; and
3. Existential realism of the context within which those behaviours are observed.

Unfortunately, as noted above, no single approach can succeed in meeting all of the above goals and only a diverse set of research strategies can maximise the probability of meeting all of the objectives.

Commonly empirical software engineering research undertakes a laboratory experiment followed by

[21] Other evaluations of this idea exist, the journal articles were selected due to their perceived high-quality in line with standard meta-analysis protocols. See Ref. [46] for a more extensive discussion of empirical data from Software Inspection experiments.

a replicating laboratory experiment which often attempts to duplicate exactly the initial investigation. While this approach maximises the 'Precision in the control and measurement of the variables related to the activity of interest' to often minimises the effect of the experimental process with regard to the other two objectives. In addition, the traditional single strategy formulation is unlikely to explore and illuminate the complexity and diversity of any realistic software engineering research hypothesis due to its single, and often highly constrained, focus. In addition, as noted above, we are currently failing to reap the reward from our 'single strategy formulation' replication approach as we are struggling to produce compatible results (from a statistical evaluation viewpoint) despite placing all of our resources into controlling the measurement activities. Hence, should we abandon scientific-oriented replication approaches? No—but it is perhaps time to recognise that this is only one option amongst many.

6. An alternative replication framework

What is required is a diverse set of treatments that allow us to select the most appropriate design characteristics for the replication (outlines in Section 5) while establishing a firm basis for minimising the problems associated with single software engineering empirical studies (outlined in Section 1). To solve this conundrum, it is believed that replications, or to be more accurate—further investigations upon a research hypothesis, should be investigations within the following high-level framework.

A replication framework that reflects best practice in the repeated evaluation of research hypotheses is dependent upon not only the selection of research strategies but also upon the 'context of use' of the hypothesis under investigation. Our framework identifies four key dimensions of replication effectiveness and a replication, or series of replications, should seek to balance the requirements of these four dimensions.

6.1. Existenial realism

Often software engineering experiments differ from the real world in that some aspects of the real world have been left out of the experimental situation while other aspects that do not exist in the real world may have been added. Large gaps between the experimental situation and the real world introduce a risk that what appears to be a significant result during the experiment will not be of genuine interest and application in real world settings. The same type of issue exists in many other software engineering settings; for example, does unit testing find meaningful failures? Or are the faults found during this process of little consequence to the end-user?

Existential realism can be further sub-divided into four areas:

- Subject or user gaps: differences between the subjects' or users' motivation to perform in an experimental setting and their day-to-day work.
- Task gaps: difficulties in generalising from the tasks that can be observed or monitored in the experimental setting and the entire set of tasks that exist in real world settings. These issues are further compounded by the fact that the tasks in the experimental setting are likely to be well defined whereas tasks in the real world settings are often ill-defined.
- Artefact gaps: differences between using a single system in (for example) a laboratory setting and using a large, complex computing environment in the real world. This issue is compounded by the additional observation that typically experimental practices focus of short-term use compared to long-termed use in the real world.
- Situational gaps: differences in job, social, and cultural contexts, etc…

While clearly any single individual experiment cannot successfully explore this diverse dimension, it is essential that if an experimental hypothesis is to be turned a real world fact, by empirical investigation, that a portfolio of experiments (the initial investigations plus its subsequent replications) carefully consider this aspect if we are truly going to produce meaningful results. Specifically, during the course of a portfolio of replications, we should seek to minimise each of the gaps at least once.

6.2. Robust

Experimental results are robust when they produce relatively stable results across a range of minor variations in the experimental setting. The robustness of an experimental finding depends heavily upon a number of issues that are potentially within the control of the investigator: type of experiment, number of experimental subjects, the degree of resolution at which the results are reported, etc. However, an experiment is also commonly heavily impacted by a large number of issues that are uncontrollable and often immeasurable and tend to vary from one subject-based experiment to another.

This dimension basically presents the status quo of current replication practice within software engineering. And while we are struggling to currently produce robust results, the problem lies not with the objective, but with our choice and utilisation of implementation strategies. For example, see Miller [35] for a discussion of the problems associated with using statistical significance testing within software technology experiments.

6.3. Impact of of findings

This is related to the idea of clinical significance in other disciplines. The impact of an empirical evaluation or series of evaluations is its ability to bring about genuine change in the evaluated system or procedure or process. That is, the impact is concerned with the persuasive-ness of the results of the experiment rather than its direct findings. Will it convince practitioners in real world settings to change, adapt or adopt these new practices based upon these findings? This dimension has been poorly addressed by current experimental practices—an interesting ratio would be the number of significant experimental results produced to the number of significant experimental results used in real world settings. Although no hard data exists, the likely ratio is likely to be rather unflattering to the field.

On a technical level, clinical significance is often equated to the estimated effect size of the experimental results. On a usage level, we require studies which explicitly target 'technology transfer' to real world settings. If a series of studies are not convincing individuals and organisations to adopt successfully demonstrated technology then it is not fulfilling its purpose. Success in this dimension should be measured by the 'adoption' rate of the technology; and a series of experimental evaluations of a research hypothesis should be seeking to expand this rate as they progress.

6.4. Resources

One of the greatest pressures on experimental inquiries is the costs associated with their undertaking In fact, one could argue that the field is 'shaped' by the serious limitations of resources experienced by researchers rather than by best experimental practice. While this situation has huge ramifications it is by no means all negative. While the number of potential software experimental investigations is almost limitless many of these will provide no real benefit. software engineers regularly analyse processes, procedures and products in terms of their costs and potential benefits—commonly comparatively against competing alternatives. But have regularly failed to apply this discipline to their experimental work.

Unfortunately, this argument is likely to be highly complex as the potential benefits are likely only to be realised at the end of the set of empirical investigations. Hence, investigators may be forced to envisage the total process to provide estimations—clearly this has many similarities with project planning, costing and management. Hence, in line with current industrial practice, with regard to unproven technologies and ideas, low-cost starting points should perhaps be the experimentalists starting point, with additional resources being added as the weight of supportive evidence increases.

7. Conclusions

Empirical evaluation within software engineering is far from straightforward. One of the principal tools in the evaluator's armoury is the controlled experiment. Unfortunately, this tool is highly deficient, when we are seeking to produce reliable and generisable results. Many of the deficiencies can be countered by external replication of the original study, especially if the replication concentrates on confirming and extending the original hypothesis rather than verifying the original experimental design, implementation and execution. Researchers conducting replications should investigate the compatibility of their results with the original. If they are compatible, then the results can be seen as additive, and our confidence is increased in the hypothesis. If they are not compatible, then the researcher should seek out reasons for the discrepancy, and this discrepancy should be used by further researchers as a cornerstone of their experimental design when investigating this hypothesis.

Perhaps, the greatest challenge facing the field (in common with most other fields) is the struggle to recognise the need for replications and hence the need to find mechanisms where the replication of original experiments is seen as an attractive proposition. This paper believes that one of the keys to this is the viewpoint that currently replications are only about proving robustness of pre-existing results. In fact, the need and role for replications or repeated re-examinations of research hypotheses, within software engineering is much more pronounced and demanding—to prove genuinely meaningful results for real world settings—the field needs to move to a portfolio of empirical studies (on a single research hypothesis) being the norm (as outlined in our high-level framework) rather than the currently unconvincing 'one-off', normally laboratory-based, studies that currently dominate the research literature. And unless the field finds mechanisms to increase the role and benefits associated with conducting repeated evaluations, especially by third parties, the topic will continue to suffer from a very low rate of technology transfer into real world settings. Without this, the field is likely to continue with its present low rate of replications and its current ills are likely to continue.

Acknowledgements

I would like to thank Drs Brooks, Daly, Roper and Wood for their help and support in constructing this paper.

References

[1] I. Amato, Pons and Fleischmann Redux, Science 260 (1993) 895.
[2] V.R. Basili, R.W. Selby, D.H. Hutchens, Experimentation in software engineering, IEEE Transactions in Software Engineering 12 (7) (1986) 733–743.

[3] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M. Zelkowitz, The empirical investigation of perspective-based reading, Empirical Software Engineering 1 (1996) 133–1264.
[4] J. Brewer, A. Hunter, Multi-method Research A Synthesis of Styles, Sage Publications, Beverly Hills, CA, 1989.
[5] W. Broad, N. Wade, Betrayers of the Truth, Oxford University Press, Oxford, 1986.
[6] R.E. Brooks, Studying programmer behaviour experimentally: the problems of proper methodology, Communications of the ACM 23 (4) (1980) 207–213.
[7] D.T. Campbell, J.C. Stanley, Experimental and Quasi-experimental Designs for Research, Houghton Mifflin Company, Boston, MA, 1963.
[8] F. Close, TOO HOT TO HANDLE: The Story of the Race for Cold Fussion, W.H. Allen Publishing, 1990.
[9] J. Cohen, Statistical Power Analysis for the Behavioural Sciences, Academic Press, London, 1977.
[10] H.M. Collins, CHANGING ORDER: Replication and Induction in Scientific Practice, SAGE Publications, Beverly Hills, CA, 1985.
[11] B. Curtis, Measurement and experimentation in software engineering, Proceedings of the IEEE 68 (9) (1990) 1144–1157.
[12] J. Daly, Replication and a Multi-Method Approach to Empirical Software Engineering research, PhD Thesis, Department of Computer Science, University of Strathclyde, 1996.
[13] J. Daly, A. Brooks, J. Miller, M. Roper, M. Wood, Verification of results in software maintenance through external replication, Proceedings of the IEEE International Conference on Software Maintenance, 1994 pp. 50–57.
[14] R. DerSimonian, N. Laird, Meta-analysis in Clinical Trials, Controlled Clinical Trials 7 (1986) 177–188.
[15] W. Dumouchel, Bayesian meta-analysis in: D. Berry (Ed.), Statistical Methodology in Pharmaceutical Science, Marcel-Dekker, New York, 1990, pp. 509–529.
[16] A. Dunsmore, M. Roper, M. Wood, The development and evaluation of three diverse techniques for OO code inspection, IEEE Transactions on Software Engineering 2003; 677–686.
[17] P. Fusaro, F. Lanubile, G. Visaggio, A replicated experiment to assess requirements, Inspection Techniques 2 (1997) 39–57.
[18] J.A. Hall, L. Tickle-Degnen, R. Rosenthal, F. Mosteller, Hypotheses and problems in research synthesis in: H.M. Cooper, L.V. Hedges (Eds.),, The Handbook of Research Synthesis, Sage Publications, Beverly Hills, CA, 1994, pp. 17–28.
[19] W.W. Hauck, S. Anderson, A survey regarding the reporting of simulation studies, The American Statistician 38 (3) (1984) 214–216.
[20] W. Hayes, Research synthesis in software engineering: a case for meta-analysis, Sixth IEEE International Symposium on Software Metrics, 1999.
[21] L.V. Hedges, I. Olkin, Statistical Methods for Meta-analysis, Academic Press, New York, 1985.
[22] D.C. Hoaglin, D.F. Andrews, The reporting of computation-based results in statistics, The American Statistician 29 (3) (1975) 122–126.
[23] J.N. Hooker, Needed: an empirical science of algorithms, Operations Research 42 (2) (1994) 201–212.
[24] J. Hunter, F.L. Schmidt, Cumulative research knowledge and social policy formulation: the critical role of meta-analysis, Psychology, Public Policy and Law 2 (1997) 324–347.
[25] E. Kamsties, C.M. Lott, An empirical evaluation of three defect detection techniques, International Software Engineering Network Technical Report, ISERN-95-02, 1995.
[26] T.D. Korson, An empirical study of the effects of modularity on program modifiability, PhD Thesis, College of Business Administration, Georgia State University, 1986.
[27] S. Kramer, R. Rosenthal, Effect sizes and significance levels in small sample research in: R. Hoyle (Ed.),, Statistical Strategies for Small Sample Research, Sage Publications, Beverly Hills, CA, 1999.
[28] O. Laitenberger, C. Atkinson, M. Schlich, K. El Eman, An experimental comparison of reading techniques for defect detection in UML design documents, The Journal of Systems and Software 53 (2000) 183–204.
[29] O. Laitenberger, K. El Eman, T.G. Harbich, An internally replicated Quasi-Experimental Comparison of Checklist and Perspective based Reading of Code Documents, IEEE Transactions on Software Engineering 2001; 387–421.
[30] R. Lindsay, A. Ehrenberg, The design of replicated studies, American Statistician 47 (3) (1993) 217–228.
[31] C.M. Lott, H.D. Rombach, Repeatable software engineering experiments for comparing defect-detection techniques, Journal of Empirical Software Engineering 1 (1996) 241–277.
[32] J. McGrath, J. Martin, J. Kulka, Judgment Calls in Research, Sage Publications, Beverly Hills, CA, 1982.
[33] J. Miller, Can results from software engineering experiments be safety combined?, Sixth IEEE International Symposium on Software Metrics, 1999.
[34] J. Miller, Statistical significance testing—a Panacea for software technology experiments?, Journal of Systems and Software 73 (9) (2004) 183–191.
[35] J. Miller, M. Wood, M. Roper, Further experiences with scenarios and checklists, Empirical Software Engineering 3 (1998) 37–64.
[36] M.J. Nye, Science in the Provinces, University of California Press, Los Angles, 1986.
[37] D.L. Parnas, D.M. Weiss, Active design reviews: principles and practices, Journal of Systems and Software 7 (4) (1987) 259–265.
[38] A. Porter, L.G. Votta, Comparing detection methods for software requirements specification: a replication using professional subjects, Empirical Software Engineering 3 (1998) 355–379.
[38] A. Porter, L.G. Votta, V.R. Basili, Comparing detection methods for software requirements inspections: a replicated experiment, IEEE Transactions on Software Engineering 21 (6) (1995) 563–575.
[40] R. Rosenthal, Meta-analytic Procedures for Social Research, Sage Publications, Beverly Hills, CA, 1984.
[41] R. Rosenthal, Replication in behavioural research in: J.W. Neuliep (Ed.),, Replication Research in the Social Sciences, Sage Publications, Beverly Hills, CA, 1991.
[42] D.A. Scanlan, Structured flowcharts outperform pseudocode: an experimental comparison, IEEE Software 6 (5) (1989) 28–36.
[43] B. Shneiderman, R. Mayer, D. McKay, P. Heller, Experimental investigations of the utility of detailed flowcharts in programming, Communications of the ACM 20 (6) (1977) 373–381.
[44] T. Thelin, P. Runeson, C. Wohlin, An experimental comparison of usage-based reading and checklist-based reading, IEEE Transactions of Software Engineering 29 (8) (2003) 687–704.
[45] C. Wohlin, H. Petersson, A. Aurum, Combining data from reading experiments in software inspections: a feasibility study in: N. Juristo, A.M. Moreno (Eds.),, in: Lecture Notes on Empirical Software Engineering (2003), pp. 85–132.
[46] F.M. Wolf, Meta-Analysis: Quantitative Methods for Research Synthesis, Sage Publications, Beverly Hills, CA, 1986.
[47] M. Wood, J. Daly, J. Miller, M. Roper, Multi-method research: an empirical investigation of object-oriented technology, Journal of Systems and Software 48 (1999) 13–26.