From Repeatability to Reproducibility and Corroboration

Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

ABSTRACT

Being able to repeat experiments is considered a hallmark of the scientific method, used to confirm or refute hypotheses and previously obtained results. But this can take many forms, from precise repetition using the original experimental artifacts, to conceptual reproduction of the main experimental idea using new artifacts. Furthermore, the conclusions from previous work can also be corroborated using a different experimental methodology altogether. In order to promote a better understanding and use of such methodologies we propose precise definitions for different terms, and suggest when and why each should be used.

1. INTRODUCTION

The title of this special issue, *Repeatability and Sharing of Experimental Artifacts*, emphasizes repeatability of experiments, and sharing of experimental artifacts as a means to achieve it. But there are many ways to redo a scientific experiment. In discussions about scientific methodology one more often hears of "replicability" or "reproducibility" rather than "repeatability". While the use of the different terms is not always consistent, it may be beneficial to try and distinguish between them.

The ideal behind this whole discussion is the notion that if some scientific fact is correct, and an experiment can demonstrate it, then it shouldn't matter who performs the experiment and how. But sometimes the devil is in the details. On a conceptual level, this involves the issue of whether knowledge is separate from the specific circumstances in which it was derived, and thus whether it generalizes to other (similar) situations. On a more practical level the question is what are the precise conditions under which the said knowledge is valid.

Regrettably the terminology used to discuss these issues is sometimes ill-defined (see e.g. [53] and references therein). In general discussions of the scientific method the term "reproducibility" is typically used. In social sciences (and specifically in psychology) the term "replication" is preferred. But some authors make a semantic distinction, for example using "replicability" to denote situations where the previous experiment is redone in exactly the same way, whereas in "reproducibility" the focus is on the result being verified and not on the method to achieve this result [20].

Given that various distinctions can be made, it seems advisable to use the availability of different terms to denote different things. This may help to promote our use and understanding of experimental methodology in two related ways. First, having crisp definitions will bring nuances to light, and enable a distinction between variants that had sometimes been bundled under a common name. Second, even if we disagree on what should be done and why, having precise definitions will at least ensure that we understand each other.

Term	*Essence*
Repetition	Rerun exactly what someone else has done using their original artifacts
Replication	Precisely replicate exactly what someone else has done, recreating their artifacts
Variation	Repeat or replicate exactly what someone else has done, but with some measured modification of a parameter
Reproduction	Recreate the spirit of what someone else has done, using your own artifacts
Corroboration	Obtain the same result as someone else, using other means and experimental procedures

Table 1: *Suggested terminology.*

Our suggested terminology contains five levels as explained in Table 1. The rest of the paper explains these terms and proposes a view of when and why each should be used. This is at least partly unique to computer science (and specifically computer systems) where sharing of artifacts can in principle include the whole system in the form of source code, and variability due to human subjects is often irrelevant. The terminology proposed here attempts to leverage the basic meanings of the commonly used terms in English, instead of trying to enlist new terms (e.g. [53, 55, 56]). To avoid confusion, we use "redoing" to refer to the general notion of doing an experiment again, without focus on one of the specific levels.

It should be noted that an orthogonal issue is the distinction between "internal" and "external" replication: Internal means that the same researchers perform the experiment again, while external means that the experiment is replicated by others [13]. While internal replication has its uses and is actually much more common, we will focus nearly exclusively on external replication in this paper.

2. CONTEXT

The natural sciences are based on the assumption of stability, where the world obeys certain laws, and science is the quest to elucidate these laws [53]. Thus the results of experiments performed in Beijing are expected to apply also in Montevideo, and experiments from the 19th century are expected to remain valid in the 21st cen-

tury. Deviations from such validity imply one of three things:

- An error was made and the supposed result is in fact wrong. For example it may have occurred by chance and does not reflect an underlying principle.
- An act of malicious fraud had occurred, and again the supposed result is in fact wrong.
- The result in question is valid only under certain conditions, and these limits to validity were not known before.

The reason to replicate or reproduce experimental evidence is therefore a desire for verification. We want to make sure that the results are correct, and perhaps map the exact conditions under which they are correct. As a by product, we also ascertain that their description is detailed and full.

A nice analogy is given by Schmidt [53]. Assume I claim to have invented a knife that can cut stones just like a regular knife cuts butter. If I claim to cut more and more stones of the same kind myself this has limited value in terms of verification. If I give you my knife and stones and let you do the cutting, it will be more convincing; and if you use the knife to cut your own stones you will have verified that indeed it works in a wider setting than originally demonstrated. You will probably be more convinced, but you still don't really know what is going on. But if I explain how the knife works, and you manufacture your own knife and cut your own stones, this provides a much more convincing form of verification. This is the ideal of reproducible research.

Replicating or reproducing scientific results is also important because of a possible file-drawer effect [51]. This is the notion that published papers are largely a biased sample of type-I errors (false positives) while many negative results languish in laboratory filing cabinets around the world (because it is uncommon to publish negative results). This notion is based on the common definition of statistical significance that is often used, where a study's result is considered significant if it could occur by chance with a probability of no more than 5%. So the published results may be hoped to contain all those that uphold hypotheses that are indeed true, but also 5% of what should have been negative results, but masquerade as positive results due to a statistical fluke. If studies of what are actually false hypotheses are few, this is only a small bias. But if negative results are actually the norm (just like 9 out of 10 startups fail), those 5% false positives can lead to a big shift in our perception of reality.

Evidence that this analysis may be correct comes from focused efforts to replicate previous findings, which generally lead to dismal results [31]. For example, one industry effort to reproduce 53 basic results in cancer successfully did so for only 6 of them [8]. But such efforts are rare, and in general experiments are rarely replicated. One reason is that the notion that in natural sciences experiments are easily replicable is actually wrong. There are often very many (technical) difficulties in setting up an experimental platform that was developed elsewhere, and it often does not succeed. Another reason is the misconception that repeating someone else's results does not count as a real contribution, and therefore such work is harder to publish and may negatively affect employment decisions. An exception to this occurs only when the implications are very significant. For example this was the case with the possibility of cold fusion due to its potentially enormous impact on energy [2]. And indeed the original results could not be replicated, and the emerging consensus was that they were wrong. What *is* often done is internal repetition: scientists repeat their own experiments at least twice before publication, and perhaps many times as they adjust and optimize the experimental procedure.

All the above is also valid for computer science. But in addition, in computers the assumption of stability is sometimes (or perhaps often) shaky, because the "world" is our infrastructure. Regrettably, computational infrastructure can change in strange and non-monotonic ways [29]. Moreover, it may be hard to actually measure the value of interest (e.g. the question of using microbenchmarks versus complete applications [45, 9, 38]). For example, one anecdote recounts measurements of Java virtual machines, that turned out to measure the quality of floating-point emulation on a certain platform rather than the desired factors [58]. There is also a problem of noise and variability, where it is not always clear that results really reflect the distribution of possible outcomes [27], or else they may depend on supposedly uninteresting hidden factors [48, 18]. Recommendations for improvement often emphasize the need to replicate research findings, and the use of rigorous statistical techniques [65, 58, 10, 16]. Facilities for automated replication and sensitivity checking have also been proposed [18]. In the following we consider what is meant by replication, and suggest different terms for different approaches.

An interesting case study is provided by the question of how to test software. Two main approaches are to use static analysis, where the code is inspected by automatic tools or code reviews, and dynamic testing where the code is executed and the outcome is compared to the specification. But which of these is better? in 1978 Myers conducted a controlled experiment at IBM, comparing the achievements of testers using black-box, white-box, and inspection in finding problems in a short PL/I text-formatting program [47]. This was followed in 1987 by a study by Basili and Selby, using essentially the same methodologies but on different programs and with different subjects [4]. Among other things, this exposed an interaction between method and subjects: code reading turned out to work better only if professionals were involved. This experiment was replicated by Kamsties and Lott [39] and by Wood et al. [64] some years later, again with different programs, and focusing exclusively on students as subjects. These exposed an interaction between the approach used and the types of faults that are discovered. These results prompted yet another version of the experiments by Juristo and Vegas, which included a deeper investigation of this interaction [36]. Later, Juristo et al. replicated these experiments yet again to reveal more interactions and contextual effects [37]. Together, this sequence of studies led to an elaboration of experimental techniques and artifacts, and to the formation of a more complete picture of the circumstances which favor the use of each method (rather than identifying a "winning" method which is always better than he others). Similar sequences may be expected to improve our understanding of computer systems too.

3. MOTIVATION

At the risk of some repetition (no pun intended), let's elaborate on the motivation for redoing experiments. We emphasize here those that are not related to verification per se, namely compensating for human errors and enhancing our understanding. This discussion is taken from [23].

In some fields the propensity for mistakes is well-documented, and accepted as part of life. A prime example is software engineering.

Practically all software life-cycle models are based on the notion of iteration, where successive iterations of the development correct the shortcomings of previous iterations [52]. As mistakes are typically found by testing the software, testing has become a major part of development. In the Unified Process, testing is one of four main workflows that span the duration of a software development project [33].

But mistakes happen in all domains of human endeavor, and finding them is a social activity that requires a time investment by multiple participants. De Millo et al. list several illuminating examples from mathematics, where proofs of theorems were later found to be flawed [15]. The history of science has witnessed several great controversies among eminent scholars, who can't all be right [30].

A recent example closer to computer science is provided by the SIAM 100-digit challenge [11]. The challenge was to compute 10 digits of the answer to each of 10 difficult computational problems. 94 groups entered the challenge, and no fewer than 20 won, by correctly computing all 100 digits; 5 additional teams got only one digit wrong. But still, three out of four groups made mistakes, including groups with well-known and experienced computational scientists. Another interesting observation was that it was not necessary to known all the answers in advance: when multiple groups from different places using different methods got the same numbers, they were most probably right, whereas unique results were probably wrong.

The lesson from these examples is that we cannot really be sure that published research results are correct, even if they were derived by the best scientists and were subjected to the most rigorous peer review. But we can gain confidence if others repeat the work and obtain similar results. Such repetitions are part of the scientific process, and do not reflect specific mistrust of the authors of the original results. Rather, they are part of a system to support and gain confidence in the original results, and at the same time to delimit the range of their applicability.

While the basic reason for attempting to reproduce previous results is to verify them, this is not the only reason. Verification takes time, and by the time we are sure of the validity of results "beyond a reasonable doubt" they may be no longer relevant. However, a more important reason may be to improve our understanding of the measured system. This is especially true in an academic setting, where basic understanding is arguably more valuable then putting results to actual use.

One of the arguments against requiring results to be verified is that it is too hard to do to be practical. Michael Foster [46] writes

> The difficulty with validating results is the myriad of details in a simulation or experiment that may affect the measurement. Reproducing a result means determining which details are important and which are inessential...

This claim is perfectly true. But a central point in studying the performance of a system is just this: finding out what are the important parameters that affect performance, the mechanisms by which they affect performance, and the degree to which they affect performance. If we manage to do this, we have learned something from the study. And if verification is the means to achieve such a level of understanding, this is a good reason to perform verification.

A rare example of actually trying to repeat measurements done by others is presented by Clark et al. [14]. Despite being essentially successful, this example underscores the difficulties of reproducibility, as the reproducing authors seem to have needed significant help from the original authors in order to achieve similar results. One of their findings was that disabling SMP support in the operating system turned out to be crucial for the reported performance. This interesting observation would not have been made if they were not attempting to repeat previous measurements.

4. REPEATABILITY

Repeatability concerns the exact repetition of an experiment, using the same experimental apparatus, and under the same conditions.

In fields ranging from psychology to software engineering repetitions are typically not really 100% exact, because experiments are done with human subjects. The expertise and behavior of subjects at different locations may vary, the recruitment policies may vary, and even with the same subjects their behavior may exhibit statistical fluctuations or change with time and experience [13, 43]. In natural sciences and computer performance evaluations repetitions may also not be 100% exact, due to statistical or environmental variations and measurement error [42, 1]. Therefore repetitions are often used as an important element of experimental design, allowing for confidence intervals to be calculated [34]. But under certain conditions in computer systems exact repetitions are in principle possible.

It is important to note that exact repetition does not provide increased confidence in the experimental result [20]. At best it shows that the setting is indeed identical for all intents and purposes. Consider this analogy: if you receive a program from a colleague, and you manage to compile and execute it and get the same result, this does not mean that the program is correct. So if the program is a simulator or even a complete system, and you have all the needed inputs and get the same result, this does not verify the result — it just verifies that you managed to repeat the original execution [53]. The only exception is when non-trivial claims were made about the execution itself, e.g. about its power consumption being unusually low, and therefore repeating the execution itself is of value. Putting it more bluntly, exact repetition can expose fraudulent representation of results.

But what are the other reasons to perform repetitions?

The first reason is that while the actual repetition has no value, preparing for it does. Packaging all the artifacts used in the simulation and listing all the configuration options are necessary steps in order to prepare the material for archiving. And given this material, others will be able to study the details of what you have done if needed. In effect, the experimental artifacts serve as the ultimate operational documentation of the experiment.

Conversely, failure to achieve exact repeatability can expose hidden assumptions about the experiment or the environment [49]. For example, in web-based systems one may perform A/A experiments (comparing version A to itself, instead of the usual A/B experiments comparing to alternative designs) to verify that the achieved results are consistent [41]. When using a workload trace as input to a simulation, one may divide the trace in two and compare the

results obtained from each half. This helps to increase our understanding of the scope and limitations of the experimental results.

Another reason is that repeatability provides a baseline for future studies [58]. When a new system design is proposed, it needs to be compared against previous designs that represent the state of the art. It is unwise and inefficient to expect the developers of the new design to do so from scratch, because of the effort needed to replicate the earlier designs, and the danger that their replication will not do justice to them [60]. A much better alternative is to provide the definitive implementation of the earlier designs, which can then serve for comparisons with newer competing designs. In a related vein, a collection of repeatable measurements can serve as a foundation for learning about how tunable parameters of the system affect performance, thereby facilitating modeling and machine-learning approaches to performance prediction and auto-tunning [26].

Finally, it should be noted that some experiments are inherently not exactly replicable. In such cases repetition is used to assess the inherent variability and compute confidence intervals [34]. Also, it is usually not really important to reproduce exactly the same data — it is more important to reproduce the patterns that lead to conclusions, e.g. about relative performance [17].

As noted above, in the context of computer systems exact repetitions are in principle possible. Conditions for such exact repeatability are

- Using the same platform.

  This can be achieved if the experimental platform is completely standard, such as an Intel-based computer of a known model from a reputable brand. Alternatively, non-standard infrastructure can be reused. An example may be a certain cluster or supercomputer, which is one-of-a-kind, but open to access by others.

  Note that the same needs to apply also to the software infrastructure, e.g. using a Linux kernel of known version with all default configuration settings. Furthermore, all this needs to be specified unambiguously.
- Using common experimental artifacts.

  This refers to community-developed or shared facilities that are widely used and serve as a de-facto standard. This includes both software and data.

  In terms of software, one class of artifacts is the software used to run experiments. Examples include simulators such as the ns2 network simulator [12] or the SimpleScalar architecture simulator [3]. Importantly, by working within a common framework individual additions (e.g. the implementation of a new feature) also become available to all [49]. Another class is benchmarks used to assess systems, such as SPEC and TPC.

  In terms of data, an important artifact is the description of the workload that is used to evaluate a system. For example, the Parallel Workloads Archive contains workload traces for evaluating job scheduling on parallel systems [25].
- Sharing all unique experimental artifacts.

  Each individual experiment adds its own unique elements to the common ones identified above. In order to enable exact repeatability these should be shared.

  The unique artifacts can be quite minimal, such as a configuration file for some common experimental infrastructure. On the other hand they may include large-scale software systems that were developed specifically for this experiment.

  In particular, when a new system design is the focus of the experiment, the full implementation of the system needs to be shared. This is often done by making it open-source, using a site such as GitHub.
- The experiment is completely self-contained.

  This means that the experiment is isolated from the world, and does not depend on any random inputs (e.g. external network activity which may interfere with the system being studied). In particular, if random numbers are used, the random number generator can be seeded to reproduce exactly the same sequence.

Conversely, impediments for exact repeatability include

- The experimental platform is unique and access to it cannot be obtained.
- The precise configuration of the experimental system is hard to ascertain and the details induce an effect. This includes elements like the exact version of all software libraries, the underlying BIOS and chipset, etc.
- Unique experimental artifacts cannot be shared due to intellectual property rights or other considerations [44].
- Shared artifacts suffer from portability problems. If the infrastructure is indeed the same this should not happen. But seemingly innocuous variations, such as using a different compiler that supports the same language standard, may still lead to unintended consequences.
- The experiments are complex in the sense that thousands of simulations or measurements were performed with different parameters. This problem can be alleviated by automatic tools that perform all the individual steps and record their details, but still it may happen that some details are lost.
- The experimental platform cannot be isolated from the world. For example, this is the case in platforms like PlanetLab, whose performance characteristics are affected by cross-traffic on the Internet [7].

As a result of such impediments, repeatability is not at all as straight forward and trivial as may be assumed [44].

5. REPLICATION

Replicability is the recreation of the same experimental apparatus, and using it to perform exactly the same experiment.

Replicability, or at least partial replicability, is widely used in the natural sciences. The basis is that the natural world is the same everywhere, so experimental procedures used to study bacteria in one place will work in the same way on other bacteria in other places. This leads to the establishment of experimental protocols which are described in laboratory manuals and replicated by everyone (e.g. [50]). It also leads to the common structure of research papers, where experimental techniques and methods (which are common) are described separately from the experimental results (which are unique to the application of the said methods to a particular situation).

Replicability is more meaningful in terms of experimental validation in fields that involve humans, such as psychology or software engineering. Here variability always exists between the subjects, so we cannot expect to get precisely the same results [43]. Thus replications can provide important data about the robustness of the original experiments in the face of such variability [40]. In particular, replications that fail to produce the same result are actually a success because they teach us something new [55].

However, replicability does not add much to our confidence in reported results. The essence is to repeat or replicate as exactly as possible what others have done. Regrettably this includes replicating their flaws as well. But on the other hand it can serve to elucidate the precise conditions under which certain results hold. This is a consequence of imperfect replication, where the unintended differences actually turn out to make a difference.

In more detail, replication hinges on having a full and detailed description of the original experiments [10]. But descriptions are never really full. A recurring problem is tacit knowledge: important things that are not verbalized in the recipe of what to do because they are taken to be self-evident, but may not be so for whoever is doing the replication [54]. Another problem is parameterization and configuration settings that are hardcoded rather than being accessible [44]. As a consequence of such problems, replications may fail to achieve the expected results. And such failures, once their cause is found, can provide potentially important information about the scope and applicability of these results. Alternatively, they can provide interesting information about the volatility of the infrastructure being used and its possible effects [29].

6. VARIATIONS

Variations on an experiment are replications with some intended modification.

In both repetition and replication the goal is to rerun exactly the same experiment. We may fail to create an exact replication, and we may learn something from that failure, but the goal was to be exact. The first step away from exact replication is to induce a controlled modification. For example, if the original experiment measured the bandwidth of a network using message sizes that are powers of two, the variation may use message sizes in jumps of 10KB. This may expose fragmentation effects that were not seen before. In the context of software engineering, using multiple teams leads to replication — essentially the same experiment, but with statistical variations due to subjects [5]. Using multiple projects as test cases, in contrast, is an explicit variation. And using both allows for better analysis, including different assignments of projects to teams, different randomized orders of performing the tasks, and so on.

The motivation for running variations is to extend our understanding of the experiment and the system being studied. In particular, the original experiment made some claim about system behavior under certain circumstances. In many cases there is an implicit notion that this behavior is not unique to these specific circumstances, but that it generalizes to other circumstances as well. Variations on the experiment are meant to check this notion and make it explicit. Thus variations help to establish the scope of the result.

A special case of establishing scope is establishing persistence. This is achieved when the modifications are time-based, as when new versions of environmental artifacts are used. In the physical world, once a fact is established it is expected to remain valid. But in computer systems, updating to a new version of an operating system kernel or software library may cause unanticipated consequences [29]. In this sense the digital world is more brittle, and persistence cannot be taken for granted.

A subtle point is the difference between generalization or persistence and verification. Variations essentially use the original experimental artifacts, or an exact replica. Thus, if there was some flaw in the experiment, this flaw will propagate to the variations. So establishing the scope of the result does not necessarily increase the confidence in the result — it just provides a better characterization of exactly what the result is.

That being said, variations can at least help ensure us that the original result is not a complete fluke. A single experiment is a sample of size 1. We tend to think that this is in some vague way representative in general, but of course this is not necessarily the case. Performing several variations can provide a distribution of results, and establish whether the original result is "in the middle" of this distribution or in its tail. For example, inducing random variations in the workload is a good way to turn a single point estimate into a point estimate with a confidence interval, where the new point estimate is not the original one but rather the average of all the measurements in the different variations [57, 66]. Importantly, such variations should not change the main independent variables, so as not to lead to a systematic (and expected) change in the results. Thus inducing variations by changing job runtimes is a bad idea if the performance metric is response time.

An important class of variations is variations in the experimental procedure, not in parameters of the experiment. Thus a family of replicated experiments with variations can facilitate a better characterization of the phenomenon being studied than any single experiment could [6]. This is discussed next.

7. REPRODUCIBILITY

Reproducibility is the reproduction of the gist of an experiment: implementing the same general idea, in a similar setting, with newly created appropriate experimental apparatus.

The essence of reproducibilty is to reproduce the result that others have obtained, where the term "result" is restricted to mean the outcome of a certain procedure, not a scientific fact. Thus reproducibility implies using essentially the same procedure. This shows that this procedure indeed leads to this result, which is what we typically want in order to confirm or refute previously obtained results. An alternative is obtaining evidence supporting the same hypothesized scientific fact by other means. This is obviously useful for increasing confidence in the result, by providing corroborating evidence. But it only provides indirect support for the procedure, so we don't call it reproduction.

The focus in reproduction is on concepts rather than artifacts — we try to recreate the essence of the experiment rather than the experiment itself. This is the difference between reproduction and replication. As a consequence reproduction will typically include some change of details or circumstances. But these changes are not pre-conceived modifications intended to map out the scope of the result. Rather, they are a result of how different people interpret and implement the experimental idea. Nevertheless, this enables us to learn something about the limitations and scope of the experimental approach. However, a problem may occur if the reproduction

fails to produce the expected result. In that case we do not know exactly what changed, so it is hard to understand the source of the difference [55]. To identify the source, controlled variations of the original experiment (or the new one) are needed.

Reproducibility also allows for meta studies, where the results of multiple independent studies are combined and patterns are sought. This is the way to find whether different reincarnations of the experiment support each other, and to weed out specific instances that are outliers that may be suspected to be flawed.

While reproducibility fundamentally implies the recreation of experimental artifacts, the original artifacts of previous work may still be extremely important. Specifically, such artifacts enable a detailed comparison with that work, and a precise examination of the differences. And these small differences may turn out to be crucial. This is again a difference between the brittle digital world and the more continuous physical one.

Finally, a special case of reproducibility is competitions. Competitions are used to accelerate progress in a certain field by creating a common evaluation framework, and inviting researchers to submit systems to be evaluated (e.g. [32, 59]). Strictly speaking this is not really reproducibility, because the element being changed is the system itself. However, it does emphasize the use of the same experimental procedures.

8. CORROBORATION

Corroboration is providing evidence in support of a hypothesis or result, but using a different approach from the one used originally.

Importantly, using a different approach has the best potential not only to corroborate a given result, but also to refute it. Such negative results are extremely important for identifying the limits to the scope of both results and experimental techniques.

In terms of increasing our confidence in a scientific result, corroboration is the best approach [20]. The reason is that it focuses on the result, and intrinsically steers clear of all potential flaws and shortcomings of the original experimental procedure. But in doing so it dissociates from the concept of reproducibility.

Another advantage of corroboration studies is that they skirt the problem of being "just a replication" [35], and therefore may have a better chance of getting published. This depends on the degree to which the new experimental technique is innovative.

9. AN EXAMPLE

To illustrate all the above definitions, consider the question of how performance depends on the relationship between a program's working set size and the available cache. The underlying hypothesis is that caching works best when the working set of the program fits in the cache.

The basic experiment demonstrating this uses direct measurements of the execution of a test program that can be configured to operate on different amounts of data (hence changing the working set size), and produces the expected result.

A repetition just does this again. Due to measurement noise we may expect the results to be slightly different, even if qualitatively the conclusions are the same. The repetitions nevertheless contribute to a characterization of the variability in the measurements, and enable an assessment of statistical significance.

A replication can be achieved by providing a description of the system and the test program to allow others to run the same simulation. This can demonstrate that the provided artifacts include all the details that may be needed about the experiment. It may also provide some additional support for the results if a slightly different system is used for the measurements.

Variations on the original experiment can use different data sizes than those used in the original study. This may improve the precision of the result, and better identify the threshold beyond which performance changes.

A reproduction of the result would use different test programs or perform the measurement on a different architecture altogether. Such extensions stay within the original framework, but remove any suspicion that the results depend in some unknown way on the original experimental artifacts.

Finally, a corroboration can be achieved by using alternative methodologies. For example one can use hardware counters of cache activity when running the test program. We can then see the effect of the working set size on the caching activity and not only on the overall performance, thereby supporting the notion that the performance changes are due to caching activity. Alternatively, we can use a simulator such as SimpleScalar to simulate the execution of the program with different cache sizes, thereby corroborating the results obtained from the measurements.

10. CONCLUSIONS

Good experimentation is the best and fastest way to make progress in both science and technological development [23]. In particular, it avoids the hobbling effect of baseless assumptions. The elements of experimentation are observation and measurement, which ground us in reality, modeling and hypothesis formulation, which encapsulate the obtained knowledge, and reproducibility, which allows for verification and refutation, and helps steer us in the right direction.

But using "reproducibility" as a catch-all term loses fidelity. There are actually several levels of redoing previous experimental work, with differences in generalizability and scope (see Table 2). Thus a mere repetition of previous work has no claim for generalizability. If replication or reproduction are employed then there is some modest or perhaps even significant generalizability in terms of method. And if a result is corroborated by other means then we have evidence for wider scope of the theory. Repeatability is an important step, but generally not ambitious enough. We need reproducibility and corroboration to really make progress. And we also need to recognize these activities as being significant research contributions, worthy of publication in top scientific venues.

In parallel to reproducibility work, there is a need for complementary work to publish, archive, and curate artifacts, mainly software and data. For example, a study of recent papers published in leading psychology journals found that for 3/4 of them the necessary data cited in the paper were not obtainable [61]. This is in general also the problem with independent repositories, which seldom survive beyond the interest or funding of their originators. A better approach is to establish repositories that are curated by professional organization, and to link them directly to publications. For

Approach	*Requirements*	*Use*
Repetition	Access to equivalent (or possibly similar) infrastructure and original artifacts	Share artifacts to enable others to ascertain details of your work, to compare directly with it, and possibly to assess its sensitivity to infrastructure and environment variations
Replication	Access to equivalent or similar infrastructure and a full detailed description of artifacts	Verify that your description is detailed enough to allow others to replicate your setup, and possibly increase confidence that result is robust against minute variations
Variation	Access to equivalent or similar infrastructure and artifacts or their description	Map out the effect of measured variations to assess the scope and generality of the result
Reproduction	Access to similar infrastructure and conceptual description of artifacts	Increase confidence in both procedure and result by replicating it in a similar (but not identical) setup, identify scope for generalization, and provide inputs for meta studies
Corroboration	Conceptual understanding of the original hypothesis and result	Increase confidence in a result by obtaining it with different means, and increase scope of result

Table 2: *Uses of different approaches.*

example, biologists have GenBank, an annotated collection of published DNA sequences maintained by the NIH, and submitting data to GenBank is typically a pre-requisite for publishing a paper in a leading journal. Ideally such a repository will also include raw results for alternative analysis and possible reinterpretation, rather than just the artifacts needed for reprodicibility.

That being said, it is important to realize that reproducibility in and by itself is not enough, and perhaps also not the most important activity. For example, it may be more important to perform a deep analysis drilling down to the root causes of observed results (e.g. [22, 29]). This cannot be done by mere reproduction, but failure to reproduce a result can help to flag a problem if one exists.

One also needs to consider other dimensions of experimental methodology, which may be even more problematic than reproducibility. A recurring problem with computer systems is that performance may be brittle, inconsistent, or even chaotic [57, 48, 41, 18]. A related problem is the effect of human factors, which we typically like to ignore [19, 28]. Another is lack of using sufficient statistical and experimental procedures such as randomization to eliminate measurement bias [65, 48, 58]. In particular, computer systems workloads and performance are often characterized by modal, skewed, or heavy-tailed distributions, and perhaps also long-range correlations [24]. As such they tend to violate common assumptions which underlie classical parametric statistical methods, so modern non-parametric methods should be learned and used [21, 62, 63, 16].

Experimental procedures are described, among others, in the books by Jain [34] and Lilja [42]. But having books is not enough if the research culture does not promote the use of experimental approaches. In psychology, for example, study programs typically include courses in experimental methodology, and students are required to participate in experiments themselves. In computer science, in contradistinction, there are typically no such courses, and the study programs emphasize the mathematical and theoretical approach, without any explicit grounding in reality. However, it seems that awareness of experimental approaches is increasing, and it may be hoped that this will herald the needed cultural change [23].

11. REFERENCES

[1] A. Abedi, A. Heard, and T. Brecht, "*Conducting repeatable experiments and fair comparisons using 802.11n MIMO networks*". *Operating Syst. Rev.* **49(1)**, Jan 2015.

[2] I. Amato, "*Pons and Fleischmann redux?*" *Science* **260(5110)**, p. 895, 14 May 1993, DOI:10.1126/science.260.5110.895.

[3] T. Austin, E. Larson, and D. Ernst, "*SimpleScalar: An infrastructure for computer system modeling*". *Computer* **35(2)**, pp. 59–67, Feb 2002, DOI:10.1109/2.982917.

[4] V. R. Basili and R. W. Selby, "*Comparing the effectiveness of software testing strategies*". *IEEE Trans. Softw. Eng.* **SE-13(12)**, pp. 1278–1296, Dec 1987, DOI:10.1109/TSE.1987.232881.

[5] V. R. Basili, R. W. Selby, and D. H. Hutchens, "*Experimentation in software engineering*". *IEEE Trans. Softw. Eng.* **SE-12(7)**, pp. 733–743, Jul 1986, DOI:10.1109/TSE.1986.6312975.

[6] V. R. Basili, F. Shull, and F. Lanubile, "*Building knowledge through families of experiments*". *IEEE Trans. Softw. Eng.* **25(4)**, pp. 456–473, Jul/Aug 1999, DOI:10.1109/32.799939.

[7] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "*In VINI veritas: Realistic and controlled network experimentation*". In *ACM SIGCOMM Conf.*, pp. 3–14, Sep 2006, DOI:10.1145/1151659.1159916.

[8] C. G. Begley and L. M. Ellis, "*Raise standards for preclinical cancer research*". *Nature* **483(7391)**, pp. 531–533, 29 Mar 2012, DOI:10.1038/483531a.

[9] S. M. Blackburn et al., "*The DaCapo benchmarks: Java benchmarking development and analysis*". In 21st *Object-Oriented Prog. Syst., Lang., & Appl. Conf. Proc.*, pp. 169–190, Oct 2006, DOI:10.1145/1167473.1167488.

[10] S. M. Blackburn et al., *Can You Trust Your Experimental Results?* Tech. Rep. #1, Evaluate Collaboratory, Feb 2012. URL http://evaluate.inf.usi.ch/sites/default/files /EvaluateCollaboratoryTR1.pdf.

[11] F. Bornemann, D. Laurie, S. Wagon, and J. Waldvogel, *The SIAM 100-Digit Challenge: A Study in High-Accuracy Numerical Computing*. SIAM, 2004.

[12] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "*Advances in network simulation*". *Computer* **33(5)**, pp. 59–67, May 2000, DOI:10.1109/2.841785.

[13] A. Brooks, J. Daly, J. Miller, M. Roper, and M. Wood, *Replication's Role in Experimental Computer Science*. Tech. Rep. EFoCS-5-94 [RR/94/172], University of Strathclyde, 1994.

[14] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. N. Matthews, "*Xen and the art of repeated research*". In *USENIX Tech. Conf.*, Jun 2004.

[15] R. A. De Millo, R. J. Lipton, and A. J. Perlis, "*Social processes and proofs of theorems and programs*". *Comm. ACM* **22(5)**, pp. 271–280, May 1979, DOI:10.1145/359104.359106.

[16] A. B. de Oliveira, S. Fischmeister, A. Diwan, M. Hauswirth, and P. F. Sweeney, "*Why you should care about quantile regression*". In 18th *Intl. Conf. Architect. Support for Prog. Lang. & Operating Syst.*, pp. 207–218, Mar 2013, DOI:10.1145/2451116.2451140.

[17] A. B. de Oliveira, J.-C. Petkovich, and S. Fischmeister, "*How much does memory layout impact performance? a wide study*". In *Intl. Workshop Reproducible Research Methodologies*, pp. 23–28, Feb 2014.

[18] A. B. de Oliveira, J.-C. Petkovich, T. Reidemeister, and S. Fischmeister, "*DataMill: Rigorous performance evaluation made easy*". In 4th *Intl. Conf. Performance Engineering*, pp. 137–148, Apr 2013, DOI:10.1145/2479871.2479892.

[19] P. A. Dinda, G. Memik, R. P. Dick, B. Lin, A. Mallik, A. Gupta, and S. Rossoff, "*The user in experimental computer systems research*". In *Workshop Experimental Comput. Sci.*, art. no. 10, Jun 2007, DOI:10.1145/1281700.1281710.

[20] C. Drummond, "*Replicability is not reproducibility: Nor is it good science*". In 4th *Workshop Evaluation Methods for Machine Learning*, Jun 2009.

[21] D. M. Erceg-Hurn and V. M. Mirosevich, "*Modern robust statistical methods: An easy way to maximize the accuracy and power of your research*". *Am. Psych.* **63(7)**, pp. 591–601, Oct 2008, DOI:10.1037/0003-066X.63.7.591.

[22] D. G. Feitelson, "*Experimental analysis of the root causes of performance evaluation results: A backfilling case study*". *IEEE Trans. Parallel & Distributed Syst.* **16(2)**, pp. 175–182, Feb 2005, DOI:10.1109/TPDS.2005.18.

[23] D. G. Feitelson, "*Experimental computer science: The need for a cultural change*". URL http://www.cs.huji.ac.il/~feit/papers/exp05.pdf, 2005.

[24] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, 2015.

[25] D. G. Feitelson, D. Tsafrir, and D. Krakov, "*Experience with using the Parallel Workloads Archive*". *J. Parallel & Distributed Comput.* **74(10)**, pp. 2967–2982, Oct 2014, DOI:10.1016/j.jpdc.2014.06.013.

[26] G. Fursin, R. Miceli, A. Lokhmotov, M. Gerndt, M. Baboulin, A. D. Malony, Z. Chamski, D. Novillo, and D. Del Vento, "*Collective Mind: Towards practical and collaborative auto-tuning*". *Scientific Prog.* **22(4)**, pp. 309–329, 2014, DOI:10.3233/SPR-140396.

[27] J. Y. Gil, K. Lenz, and Y. Shimron, "*A microbenchmark case study and lessons learned*". In *SPLASH'11 Workshops*, pp. 297–308, Oct 2011, DOI:10.1145/2095050.2095100.

[28] S. Hanenberg, "*Faith, hope, and love: An essay on software science's neglect of human factors*". In *Object-Oriented Prog. Syst., Lang., & Appl. Conf. Proc.*, pp. 933–946, Oct 2010, DOI:10.1145/1932682.1869536. (Onward track).

[29] A. S. Harji, P. A. Buhr, and T. Brecht, "*Our troubles with Linux kernel upgrades and why you should care*". *Operating Syst. Rev.* **47(2)**, pp. 66–72, Jul 2013, DOI:10.1145/2506164.2506175.

[30] H. Hellman, *Great Feuds in Science: Ten of the Liveliest Disputes Ever*. John Wiley & Sons, 1998.

[31] J. P. A. Ioannidis, "*Why most published research findings are false*". *PLOS Medicine* **2(8)**, pp. 0696–0701, Aug 2005, DOI:10.1371/journal.pmed.0020124.

[32] L. D. Jackel, D. Hackett, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "*How DARPA structures its robotics programs to improve locomotion and navigation*". *Comm. ACM* **50(11)**, pp. 55–59, Nov 2007, DOI:10.1145/1297797.1297823.

[33] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison Wesley, 1999.

[34] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[35] B. E. John, "*Avoiding "it's JUST a replication"*". In *CHI2013 Workshop on Replication of HCI Research*, pp. 3–7, Apr 2013.

[36] N. Juristo and S. Vegas, "*Functional testing, structural testing and code reading: What fault type do they each detect?*" In *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, R. Conradi and A. I. Wang (eds.), pp. 208–232, Springer-Verlag, 2003, DOI:10.1007/978-3-540-45143-3_12. Lect. Notes Comput. Sci. vol. 2765.

[37] N. Juristo, S. Vegas, M. Solari, S. Abrahao, and I. Ramos, "*Comparing the effectiveness of equivalence partitioning, branch testing and code reading by stepwise abstraction applied by subjects*". In 5th *Intl. Conf. Software Testing, Verification, & Validation*, pp. 330–339, Apr 2012, DOI:10.1109/ICST.2012.113.

[38] T. Kalibera, J. Hagelberg, P. Maj, F. Pizlo, B. Titzer, and J. Vitek, "*A family of real-time Java benchmarks*". *Concurrency & Computation — Pract. & Exp.* **23(14)**, pp. 1679–1700, Sep 2011, DOI:10.1002/cpe.1677.

[39] E. Kamsties and C. M. Lott, "*An empirical evaluation of three defect-detection techniques*". In 5th *European Softw. Eng. Conf.*, pp. 362–383, Springer-Verlag, Sep 1995, DOI:10.1007/3-540-60406-5_25. Lect. Notes Comput. Sci. vol. 989.

[40] R. A. Klein et al., "*Investigating variation in replicability: A "many labs" replication project*". *Social Psychology* **45(3)**, pp. 142–152, 2014, DOI:10.1027/1864-9335/a000178.

[41] R. Kohavi and R. Longbotham, "*Unexpected results in online controlled experiments*". *SIGKDD Explorations* **12(2)**, pp. 31–35, Dec 2010, DOI:10.1145/1964897.1964905.

[42] D. J. Lilja, *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2000.

[43] J. Lung, J. Aranda, S. Easterbrook, and G. Wilson, "*On the difficulty of replicating human subjects studies in software engineering*". In 30th *Intl. Conf. Softw. Eng.*, pp. 191–200, May 2008, DOI:10.1145/1368088.1368115.

[44] I. Manolescu et al., "*The repeatability experiment of SIGMOD 2008*". *ACM SIGMOD Record* **37(1)**, pp. 39–45, Mar 2008, DOI:10.1145/1374780.1374791.

[45] L. McVoy and C. Staelin, "*lmbench: Portable tools for performance analysis*". In *USENIX Ann. Technical Conf.*, pp. 279–294, Jan 1996.

[46] T. Mudge, "*Report on the panel: How can computer architecture researchers avoid becoming the society for irreproducible results?*" *Comput. Arch. News* **24(1)**, pp. 1–5,

Mar 1996.

[47] G. J. Myers, "*A controlled experiment in program testing and code walkthroughs/inspections*". *Comm. ACM* **21(9)**, pp. 760–768, Sep 1978, DOI:10.1145/359588.359602.

[48] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "*Producing wrong data without doing anything obviously wrong!*" In 14th *Intl. Conf. Architect. Support for Prog. Lang. & Operating Syst.*, pp. 265–276, Mar 2009, DOI:10.1145/2528521.1508275.

[49] L. Peterson and V. S. Pai, "*Experience-driven experimental systems research*". *Comm. ACM* **50(11)**, pp. 38–44, Nov 2007, DOI:10.1145/1297797.1297820.

[50] J. Sambrook and D. W. Russell, *Molecular Cloning: A Laboratory Manual*. Cold Spring Harbor Laboratory Press, 3rd ed., 2001.

[51] J. D. Scargle, "*Publication bias: The "file-drawer" problem in scientific inference*". *J. Sci. Explor.* **14(1)**, pp. 91–106, 2000.

[52] S. R. Schach, *Object-Oriented and Classical Software Engineering*. McGraw-Hill, 6th ed., 2005.

[53] S. Schmidt, "*Shall we really do it again? the powerful concept of replication is neglected in the social sciences*". *Rev. General Psychology* **13(2)**, pp. 90–100, Jun 2009, DOI:10.1037/a0015108.

[54] F. Shull, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri, "*Replicating software engineering experiments: Addressing the tacit knowledge problem*". In *Intl. Symp. Empirical Softw. Eng.*, pp. 7–16, Oct 2002, DOI:10.1109/ISESE.2002.1166920.

[55] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "*The role of replications in empirical software engineering*". *Empirical Softw. Eng.* **13(2)**, pp. 211–218, Apr 2008, DOI:10.1007/s10664-008-9060-1.

[56] D. J. Simons, "*The value of direct replication*". *Perspective on Psychological Sci.* **9(1)**, pp. 76–80, Jan 2014, DOI:10.1177/1745691613514755.

[57] D. Tsafrir, K. Ouaknine, and D. G. Feitelson, "*Reducing performance evaluation sensitivity and variability by input shaking*". In 15th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 231–237, Oct 2007, DOI:10.1109/MASCOTS.2007.58.

[58] J. Vitek and T. Kalibera, "*Repeatability, reproducibility and rigor in systems research*". In 9th *Intl. Conf. Embedded Software*, pp. 33–38, Oct 2011, DOI:10.1145/2038642.2038650.

[59] E. M. Voorhees, "*TREC: Continuing information retrieval's tradition of experimentation*". *Comm. ACM* **50(11)**, pp. 51–54, Nov 2007, DOI:10.1145/1297797.1297822.

[60] S. Wartik, "*Are comparative analyses worthwhile?*" *Computer* **29(7)**, p. 120, Jul 1996.

[61] J. M. Wicherts, D. Borsboom, J. Kats, and D. Molenaar, "*The poor availability of psychological research data for reanalysis*". *Am. Psych.* **61(7)**, pp. 726–728, Oct 2006, DOI:10.1037/0003-066X.61.7.726.

[62] R. Wilcox, *Introduction to Robust Estimation & Hypothesis Testing*. Academic Press, 3rd ed., 2012.

[63] R. R. Wilcox, *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy*. Springer, 2nd ed., 2010.

[64] M. Wood, M. Roper, A. Brooks, and J. Miller, "*Comparing and combining software defect detection techniques: A replicated empirical study*". In *European Softw. Eng. Conf. & Intl. Symp. Foundations of Softw. Eng.*, pp. 262–277, Springer-Verlag, Sep 1997, DOI:10.1007/3-540-63531-9_19. Lect. Notes Comput. Sci. vol. 1301.

[65] J. J. Yi, D. J. Lilja, and D. M. Hawkins, "*Improving computer architecture simulation methodology by adding statistical rigor*". *IEEE Trans. Comput.* **54(11)**, pp. 1360–1373, Nov 2005, DOI:10.1109/TC.2005.184.

[66] N. Zakay and D. G. Feitelson, "*Workload resampling for performance evaluation of parallel job schedulers*". *Concurrency & Computation — Pract. & Exp.* **26(12)**, pp. 2079–2105, Aug 2014, DOI:10.1002/cpe.3240.