XVII International Scientific Conference on Industrial Systems (IS'17)
Novi Sad, Serbia, October 4. – 6. 2017.
University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Available online at http://www.iim.ftn.uns.ac.rs/is17

Lessons learned from a partial replication of an experiment in the context of a software engineering course

Robert Ramač
(University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Serbia, ramac.robert@uns.ac.rs)

Itir Karac
(M3S Research Unit, University of Oulu, Finland, Itir.karac@oulu.fi)

Burak Turhan
(Department of Computer Science, Brunel University London, United Kingdom, burak.turhan@brunel.ac.uk)

Natalia Juristo
(Escuela Tecnica Superior de Ingenieros Informaticos, Universidad Politecnica de Madrid, Spain, natalia@fi.upm.es)

Vladimir Mandić
(University of Novi Sad, Faculty of Technical Sciencies, Trg Dositeja Obradovića 6, Serbia, vladman@uns.ac.rs)

Abstract

Replications are an integral component of experimentation through which the validity and reliability of the observed outcome in a previous experiment can be probed. In a strict replication, the experiment is executed in the same conditions as the original by following the same protocol and thus the evidence is strengthened statistically by means of increased sample size. Another objective for running replications is generalizing the experimental results beyond the limitations of one study and its context. For this purpose, certain elements of the original experiment, such as experimenters, experimental objects, and construct operationalization are altered and their impact is investigated. This paper presents lessons learned from a replication that was conducted as a part of an undergraduate university course in Serbia. The focus of the experiment was investigating the effectiveness of writing tests during the development process. The original experiment investigated the effectiveness of test-first programming and was conducted in Italy (Politecnico di Torino) with third-year computer science students during an intensive Java course. Lessons learned from this partial replication are that the given task descriptions and structure has an impact on the experiment outcome and that variations in metrics collection can occur when multiple researchers analyse the data, which requires metrics consolidation.

Key words: *software testing process, empirical software engineering, controlled experiments, partial replication, software development process*

1. INTRODUCTION

Today research in software engineering (SE) is considered to be of great importance. Every good research in the field of SE must be based on some evidence, and one of the ways to collect evidence is through experimentation. In SE experimentation can be quite difficult, and one reason for that is that there is a large number of context variables and creating a cohesive understanding of experimental results requires a community of researchers that can replicate studies, vary context variables and build abstract models that represent the common observation about the discipline [3]. Empirical methods in SE are gaining popularity in the last few years and experimentation is being moved to the centre of the research process [1,2,3]. This is because there is a need to somehow validate assumptions or claims and the need to verify the research results. Some authors claim that experimentalism in SE is necessary and that common wisdom, intuition and speculation are not reliable sources of credible knowledge, thus experimentation can help build a reliable knowledge base by collecting various evidences about the phenomenon under observation [2,3].

Empirical work is complex and time consuming, especially in SE. As Basili et al. say “We can not a priori assume that the results of any study apply outside the specific environment in which it was run." [3]. In other words, the uniqueness of the SE research is intricately tied with the context. Software engineering is specific because every new software product is different from the last, so these products do not provide a large set of data points that would permit sufficient statistical power for confirming or rejecting hypothesis [3]. Therefore, the focus of the SE research is always on a process and often the human factor has the significant effect on the findings. Another characteristic of SE is that empirical investigators are presented with a challenge to design the best study that the given context allows and are expected to generalize the research results, with a certain level of validity [3].

In today's scientific community experiments are considered to be an indispensable part of the experimentation process and scientific process as they provide a way to test what effect some variables have on the variables in observation and as such confirm or refute some hypothesis that the researchers previously set. However, in SE research it is not just about identifying the causal relationships but gaining insight into the context, variables, various effects and so on [1,2].

In order to generate significant and valid results researchers have to use various empirical methods that should have a strict design and a precisely defined procedure to follow. Therefore, it is considered a good practice to plan the experimentation process in detail to avoid certain bumps in the road. During the planning of an experiment it is always helpful to have some references in the sense of best practices and problems that other researchers faced [1,3,10,14].

When it comes to the practitioners there is an ongoing debate whether if using students as research subjects is acceptable or not. One of the most common scenarios in which students are used as research subjects is within the context of a university course. There are various viewpoints on this subject and some researchers are in favour of using students as subjects in experiments while some are against it [10,11,12,13]. Some benefits are: training junior researchers, gaining data to prove or refute hypothesis, education, industrial relevance, hands on practical experience and etc. [11,12,13]. On the other hand, the drawbacks are usually tied to validity issues in the context of experience, skills and so on [10].

Experiments with students as subjects have shown to be particularly useful for pilot experiments before they are carried out in the industrial environment [11]. Carver et al. conducted an overview of various benefits and costs of using students in experiments [11]. According to Carver et al., using students is mainly beneficial to researchers as it helps with obtaining preliminary results, is vital to showing the industry the importance of research, fine tunes the research before it is conducted in some company, helps with training junior researchers etc. [11]. Other studies can neither reject nor accept the hypothesis on the difference between using students and industry people in experiments [12]. Höst et al. argue that the only minor differences between students and professionals can be shown in their ability to perform small tasks of judgement [13]. With everything said using students as research subjects is something that should be taken into account when conducting research in SE, even if there are certain limitations to this practice.

This paper presents lessons learned from a partial replication of an experiment in the context of a software development course. The replication investigated the effectiveness of a specific software development and testing technique. The replication was designed as practical tasks that the students did within the course and on which they were graded in order to pass the course, i.e. the experiment was embedded within the course itself. Various lessons are drawn out of the replication process itself and from the experience of working with students.

The rest of the paper is organized as follows. Background about experiments and related terms are given in Section 2. In Section 3 a description of the general setup of the replication is provided, while Section 4 contains the lessons learned from the replication. Finally, Section 5 is used to give some conclusions about the material presented in this paper.

2. BACKGROUND AND RELATED WORK

Although experiments are considered to be a vital part of SE research [1,2,3], and because of the uniqueness of SE one of the ways to increase the validity of research results is through the process of repeating experiments. This process represents a core component of the experimentation process [1]. The importance and value of experiment repetitions has been widely recognized in various scientific disciplines, and from a scientific viewpoint not conducting a sufficient amount of experiment repetitions can lead to the acceptance of not robust enough results [2]. In SE experiment repetitions can have many purposes like: verifying that researchers do not influence the results, that the results are independent of the experiment site, verification that the original experiment results are not a product of chance and more [2]. In what follows, basic experimental terminology and concepts, along with some related work on using students in SE experiments are introduced.

2.1 **Terminology**

Experiments can be considered as *controlled experiments* when every variable and condition is held in control by the researchers. In other words, it represents a closely monitored and controlled study in which an intervention is deliberately introduced in order to view its effects. The effect of independent variables on the dependant variable is measured during the application of treatments on the independent variables [14].

Experiment design represents the way an experiment is structured, and describes how the experiment is supposed to run. The most important parts of the experiment design are the definition of variables (dependent and independent), and treatment that will be applied. There are various experiment designs, and the one used in this paper is the *crossover design* [15].

The crossover experiment design represents an experiment design in which values for independent variables are switched. In this way the risk of subjects being biased to variable values is eliminated [15].

A *quasi experiment* is considered to be an experiment in which the researcher does not have full control of every aspect of the experiment. Often it leads to the inability to obtain a satisfactory sample [14]. Generally, quasi experiments are typical in SE because of the researcher's inability to control every factor.

Experiment replications are repeated executions of the original experiment. Experiment replications serve to consolidate knowledge that is built upon some experimentation results [5]. By running replications of some experiment and confirming the results of the original experiment researches are one step closer to inferring that such results are regularities existing in the phenomenon that is under study by the experiment [1]. Running more and more replications of one experiment further increase the credibility of the results [4]. There are many classifications of replications [1,2,3,6,7]. For example, there are *strict replications* that are used in order to replicate a base experiment as precisely as possible, *differentiated replications* that alter the aspects of the original experiment in order to test the limits of that studies conclusions, *partial replications* that have the same goal in focus as the original experiment but in some way alter the design or procedure of the experiment etc. [2, 16]. Also, some researchers strive to conduct as many replications as possible in one study in order to widen the sample as much as possible and by confirming the results try to generalize them to the whole population that lies beneath the study [9]. It is common thought in literature that more replications whose results are in compliance with the base experiment results equal more reliable results about the phenomenon under study.

2.2 **Experiments with students**

Software engineering experiments require subjects to apply treatment, e.g. to apply a software development technique. In SE research these subjects come in the form of professionals who work for some company or students who attend a certain course. This papers main focus is on running experiments with students as subjects [10,11,12,13].

In literature there are various mentions of this matter and Table 1 shows some of the main characteristics (benefits) with which this paper can relate to.

Table 1. Benefits of using students as subjects

Ref	Characteristic	Description
[11]	Obtaining evidence needed to confirm or refute hypotheses	New hypothesis need to undergo empirical validations before their use in the industry
[11]	Train junior researchers	The academic environment tends to be more "soft" for junior researchers to generate some experience
[11]	Education on modern topics	Using the research to train students in some popular technologies and techniques
[11]	Industrial relevance	Students gain some insight into various industrial problems
[11]	Hands-on practice and empirical methods usefulness	Students gain first hand examples of some real world problems instead of just theoretical classes. Also students are demonstrated the usefulness of using quantitative methods
[13]	Mimic professionals using students in experiments	Minor differences between students and professional lead to good test runs of the experiment

Besides the characteristics shown in the table, in literature, there is mention of some drawbacks to using students as subjects. One of the main drawbacks is the generation of validity problems for formal experiments. Some researchers and practitioners claim that the use of students as subjects reduces the practical value of an experiment because of validity issues such as the lack of experience and skills. In other words, authors argue that professionals are a more credible source of data because of their knowledge base, and that results gathered from students might not be suitable for generalization. Also there are those that neither side with, nor discourage the use of students as research subjects, and believe that the students are the next generation of professionals and that they are really close to the population of interest [10, 11, 12, 13].

The replication that is the topic of this paper used students as research subjects, as did the original experiment that was conducted in Torino, Italy. The focus of the original experiment was on the effectiveness of the test-first approach to programming as opposed to the test-last approach [8]. The researchers evaluated the external quality and programmer productivity in the context of incremental development and an undergraduate object-oriented course. Researchers used a standard design with one factor and two treatments, where the treatments corresponded to the two development techniques (test-last and test-first approach). The study consisted of 35 students who were at their third year of studies who were split into two groups. During the experiment one group was compared with the other group. The main result of the experiment was that the test-first approach produces more tests then more conventional test-last approaches, and thus a higher level of productivity is reached. Also the results showed better decomposition levels as well as improved understandability of underlying requirements. The researchers pointed out the need for a larger sample then 35 subjects as well as new supporting tools for process conformance [8]. Afterwards, this experiment was replicated on several occasion: University of Oulu, Finland [19, 20] and University of Basilicata, Italy [9].

3. EXPERIMENTAL SETUP

As mentioned earlier the focus of this paper is to present and reflect upon lessons learned from a partial replication conducted in the context of a software engineering course. The replication was carried out at the University of Novi Sad, Serbia and was conducted with students on their second academic year.

The replication was designed as a crossover experiment with two sessions of repeated measurements in which students were tasked with conforming to test-last and test-first software development process respectively. There were two tasks in total where each task represented a small API that needed to be developed by the students in the *Java* programming language. Each task had detailed explanations and some initial code for the students to start with. Task 1 was about developing a score keeper for a bowling game taken from [8], while Task 2 represented an API for driving a small vehicle in two dimensional space. The students were split into four groups where each group first used one development process and during the second session the next development process in order to solve the given task.

Before the first session the students were trained in the software development and testing technique (test-last) required for the first treatment. Also, the experiment environment was set up. On every lab computer the *Eclipse* IDE was installed along with the *Besouro*[1] plugin used for monitoring compliance with a specific technique. The treatment for groups was randomly selected by flipping a coin.

During the *first session* two groups were given task 1, while the other two groups were given task 2. In this session both groups implemented the test-last approach. Because of the course schedule the groups did the tasks sequentially over the course of two days.

After the first session the metrics for productivity, test code coverage, code quality and number of written tests were extracted from student code. The extractions were done in part automatically using the plugin and *Python* scripts and in part manually by two researchers. The metrics that were extracted manually needed to undergo a consolidation process because both researchers extracted the metrics separately from the whole sample and on site. After the metrics were extracted the students were given feedback of how they did in the first task in the form of a grade. The grade was calculated based on the metrics extracted and basically depended on two components: quality of written code and conformance to the applied software development technique. Afterwards they were trained in the test-last software development technique and given the main pointers of how to adhere to the process.

During the *second session* the tasks were switched between groups and the students implemented the test-first approach in order to solve the tasks.

[1] Besouro plugin (accessed: 08.02.2017.) https://github.com/brunopedroso/besouro

After the second session the students were again given feedback in the form of a grade and the data analysis process for the whole experiment was ready to begin. This time the metrics were extracted on site by one researcher, while the second researcher extracted the metrics off site. This time the consolidation of extracted metrics was done over *Skype*.

Table 2 illustrates this crossover design and task switching of the experiment.

Table 2. *Crossover design task distribution*

	Session 1	Session 2
Group 1	Task 1	Task 2
Group 2	Task 1	Task 2
Group 3	Task 2	Task 1
Group 4	Task 2	Task 1

What differs in this replication from the original experiment, and the reason it is partial is that the experiment had to undergo some minor changes in order to accommodate the course under which it was conducted as a replication. The main difference between the replication and the original experiment is that the replication is designed using a cross experiment design technique, as well as repeated measurements for students where each student was compared to him/herself. Also one important modification that makes this replication partial was the fact that the sessions were treated as tests for students and the results of the first session were announced to the students in order to grade them. All of the modifications were carried out in such a way so they do not change the base of the original experiment. The main question that arose from these modifications was will the feedback to students in some way impact the second session.

4. LESSONS LEARNED

During the experiment researchers took note of some observations that could be useful for others and should be taken into account for future replications of this experiment.

Time constraint – One of the modifications in the experiment was to the time that was given to subjects to complete the task. In the replication the subjects had one and a half hours to complete the given task. This was proven to be insufficient and the students complained that they had little time to complete the task relative to their coding skills. This was also visible in the data analysis process because in some cases the subjects were on the right track and then just stopped because of insufficient time. Based on this observation the recommendation for future replications of this experiment would be to extend the time.

Task understandability – One of the tasks that was given to subjects was to create and test a bowling score keeper program. Because bowling is little to not at all popular in the region where the replication was conducted the subjects had trouble with understanding the rules that were described in the program specification. The recommendation for future replications in this case is to keep the scope of the task but replace its theme based on the region it is conducted in so that this drawback could be eliminated.

Programming skills – It is well known that skill increases with practice. As mentioned the students used in this replication were at their second year of studies and as far as researchers know they are the youngest. This has to be factored in when analysing their code and also if future comparisons with other replications are to be conducted.

4.1 **The measurement process**

Measurement and data analysis was conducted by two researchers. Each of the researchers did the whole analysis process, meaning that each researcher went through all the data in order to collect the required metrics.

Because the results were analysed by two researchers it was expected for some variations to occur between the generated metrics. This was proven to be so in the context of this replication and was considered as a normal side effect of the two researcher data analysis process.

In order to analyse the mentioned variation and agreement between researchers the inter-rater reliability was measured using the *Intraclass Correlation Coefficient* (ICC) [17]. The ICC is computed using the *irr* R package [18]. The *Computed ICC* in Table 3 represents the computed value. The interpretation of the results is taken from the work of Koo et al. and states that if the ICC values are close to 0 the reliability is low as opposed to values being close to 1 resulting in higher reliability. In other words, the ICC will be high if there is little variation and small if there is some significant variation. Table 3 shows various scenarios for which the ICC metric was computed.

Table 3. Intra-rater reliability

ICC measurement scenario	Computed ICC
Overall ICC: Computed over all measurements (including both sessions and tasks)	0.765
ICC on session 1: Computed over the measurements for the first session	0.599
ICC on session 2: Computed over the measurements for the second session.	0.862
ICC on Task 1: Computed over the measurements for task 1 in both sessions.	0.895
ICC on Task 2: Computed over the measurements for task 2 in both sessions.	0.367

When the results were analysed the largest variations seemed to be on generating metrics for the second task during both sessions. It was concluded that all of the variations were due to the need of implementing some small corrections to subject code in order to be suitable for metric generation and different corrections being applied by the two researchers. Generally, the code changes were kept to a minimum in order to avoid significant effect on the student work. Most of the changes were introduced because the majority of students did not comply with the API specification, so the changes needed to wrap the students code in order to pass the acceptance tests. Because these changes led to variations the need for consolidation of the results arose. On the count of this need the two researchers consolidated their generated metrics and used the newly generated metrics for further analysis.

The situations where more than one researcher analyses the generated data brings some advantages and disadvantages to the scene. For one the main benefit of having multiple researchers analysing the data is that the risk of missing some steps or some data is reduced. Also the researchers can coordinate between each other and provide help for some situations in the data analysis process. The main drawback of multiple researchers analysing the data is that there is a need for result consolidation but this is acceptable in such a scenario.

4.2 **The impact of intersession feedback**

One main concern that the researchers had was will there be impact of the researcher's feedback to students in the first session on the second session. Because the replication was organized in the context of a software testing course the treatments were disguised as tests for students, who had no knowledge of the experiment that was running in the background. On the count of the treatments being treated as tests the students needed to have some feedback in the form of a grade.

Hypothetically, we assumed the following scenario. Students who did the first test poorly will be more motivated for the second test, and that the students that showed good results on the first test will be more careless on the second test; i.e.:

H_0 - *The feedback from the first session will have some impact on the second session*.

For the purpose of this analysis every student was compared with him/herself and a statistical paired t test was used in order to generate the needed statistics.

Table 4. Paired t test

t = 0.74549	df = 50	p-value = 0.4595
alternate hypothesis: true difference in means is not equal to 0		
95% confidence interval: [-1.162738, 2.535287]		
sample estimates: mean of differences 0.6862745		

Based on the generated metrics and analysis it was possible to conclude that the feedback from the first test did not impact the outcome of the second test, and that it neither motivated or demotivated the subjects. After these results were analysed a question if the one task was more difficult for students to implement then the other arose.

Further data analysis indicated that the tasks indeed had some impact on student performance in the context of their grades. The t tests were used in order to analyse the effect of task switching on student grades. It is hypothesised that there will be no difference between sessions and that student grades will not change. Two tests were conducted, one for the groups that did task 1 in the first session and then task 2 in the second, and another for the groups that did task 2 in the first session and task 1 in the second. The resulting statistics and generated p-values (0.0008755 and 0.001313 respectively) indicated that the null hypothesis can be rejected in favour of the alternate and the fact that tasks indeed had impact on student performances. It was concluded that task 1 was more difficult for students to implement then task 2. Task analysis reviled that students had trouble with task 1 because they did not fully understand the specification and also because in this task they had to conform to a certain project template, which was not the case with task 2.

5. CONCLUSIONS

The main focus of this paper was on presenting the lessons learned from running a partial replication in the context of a software testing course. The researchers conducted a replicated study of the effects of implementing test-last and test-first software development techniques. The study was conducted using students on their second year of studies. The research itself was designed using a crossover experiment design with repeated measurements, and which consisted of two experimental sessions. The replication was designed in such a way that each subject was compared to him/herself. After the last session and the data analysis the researchers were able to reflect upon the whole research process and draw out some lessons. The first lesson showed the benefits and drawbacks of multi-researcher involvement in the whole data analysis process, while the second lesson showed the that feedback between sessions might not have any impact on the following sessions. This was concluded for this replication, while it might not hold for other cases and researchers in future replications should have this in mind.

This replication showed that this form of experimentation can be imbedded into a university course and as a part of future work we plan to continue this practice and to improve the replication design based on the lessons learned.

6. REFERENCES

[1] Tichy, W. F. (1998). Should computer scientists experiment more?. Computer, 31(5), 32-40.

[2] Juristo, N., & Gómez, O. S. (2012). Replication of software engineering experiments. In *Empirical software engineering and verification* (pp. 60-88). Springer Berlin Heidelberg.

[3] Basili, V. R., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, *25*(4), 456-473.

[4] Juristo, N., & Vegas, S. (2009, October). Using differences among replications of software engineering experiments to gain knowledge. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement* (pp. 356-366). IEEE Computer Society.

[5] Shull, F. J., Carver, J. C., Vegas, S., & Juristo, N. (2008). The role of replications in empirical software engineering. *Empirical software engineering*, *13*(2), 211-218.

[6] Krein, J. L., & Knutson, C. D. (2010, May). A case for replication: Synthesizing research methodologies in software engineering. In *RESER2010: proceedings of the 1st international workshop on replication in empirical software engineering research*.

[7] Van IJzendoorn, M. H. (1994). A process model of replication studies: On the relation between different types of replication.

[8] Erdogmus, H., Morisio, M., & Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *IEEE Transactions on software Engineering*, *31*(3), 226-237.

[9] Fucci, D., Scanniello, G., Romano, S., Shepperd, M., Sigweni, B., Uyaguari, F., ... & Oivo, M. (2016, September). An external replication on the effects of test-driven development using a multi-site blind analysis approach. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 3). ACM.

[10] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering*, *28*(8), 721-734.

[11] Carver, J., Jaccheri, L., Morasca, S., & Shull, F. (2003, September). Issues in using students in empirical studies in software engineering education. In *Software Metrics Symposium, 2003. Proceedings. Ninth International* (pp. 239-249). IEEE.

[12] Runeson, P. (2003, April). Using students as experiment subjects–an analysis on graduate and freshmen student data. In *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering* (pp. 95-102).

[13] Höst, M., Regnell, B., & Wohlin, C. (2000). Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, *5*(3), 201-214.

[14] Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N. K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, *31*(9), 733-753.

[15] Vegas, S., Apa, C., & Juristo, N. (2016). Crossover designs in software engineering experiments: Benefits and perils. *IEEE Transactions on Software Engineering*, *42*(2), 120-135.

[16] Mandić, V., Markkula, J., & Oivo, M. (2009, June). Towards multi-method research approach in empirical software engineering. In *International Conference on Product-Focused Software Process Improvement* (pp. 96-110). Springer Berlin Heidelberg.

[17] Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, *15*(2), 155-163.

[18] Gamer, M., Lemon, J., Gamer, M. M., Robinson, A., & Kendall's, W. (2012). Package 'irr'. *Various coefficients of interrater reliability and agreement*.

[19] Fucci, D., & Turhan, B. (2013, October). A replicated experiment on the effectiveness of test-first development. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on* (pp. 103-112). IEEE.

[20] Fucci, D., Turhan, B., & Oivo, M. (2014, September). Impact of process conformance on the effects of test-driven development. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 10). ACM.