A Controlled Experiment to Explore Potentially Undetectable Defects for Testing Techniques

Martin Solari, Santiago Matalonga
Universidad ORT Uruguay
martin.solari@ort.edu.uy, smatalonga@uni.ort.edu.uy

Abstract—Software testing practitioners have an array of testing techniques to choose from to test their software. Nevertheless, there is little empirical evidence about the capability of each technique to detect specific types of defects. As a result, when selecting and combining the testing techniques for a project, practitioners must rely on their own experience. This paper studies the behaviour of two specific techniques, equivalence partitioning and decision coverage, to determine which types of defect are potentially undetectable to either one. This paper presents a differentiated experiment replication based on a previous experimental design, but using different artifacts. The experiment confirms the hypothesis that some defect types are undetectable to each technique. Even with a correct application of each technique, some defects will only be detected by chance. This study adds new empirical evidence for constructing a classification of defects that takes into account technique detection capabilities.

Keywords: Software testing, experiment, defect detection.

I. INTRODUCTION

Software testing is one of the software quality assurance’s key activities. Nowadays, the software testing practitioner has a myriad of tools and techniques at his disposal to carry out his task effectively. And yet, our experimental knowledge on these tools and techniques is limited [1].

An important contribution to the experimental knowledge on defect detection techniques has been achieved by the execution of a family of experiments started by Basili [2]. This family of experiments has been adapted and replicated by other researchers [3]–[6]. The objective of this family of experiments is to study the impact of the human factor on the application of different defect detection techniques. In the different iterations of the experiment, several manual static (inspection) and dynamic (testing) defect detection techniques were compared. The most common result is that these techniques have similar defect detecting capability, and that the efficacy is dependent on the defect and the context where the technique is applied [6].

This paper presents an experimental replication that uses a set of seeded defects to compare the efficacy of functional and structural manual testing. These seeded defects were designed to test the hypothesis that certain defects could be potentially undetectable for the functional or structural testing technique.

The differentiated replication presented in this paper is based on an previous experiment designed by Juristo and Vegas [5]. We call this replication differentiated because the experiment was modified in two key aspects. First, experimental programs were adapted to fit the knowledge and training of the experimental subjects. Secondly, seeded defects were modified to test a new hypothesis that defects detected by each technique are different.

The results presented by this replication confirm previous replications of this family of experiments regarding technique efficacy, but also adds new data about the types of defects detected by each technique. Our results show that some defects types are potentially undetectable to structural or functional techniques. Furthermore, the techniques efficacy to detect a defect is dependent to the program and the specific seeded defect.

This paper uses Carver’s guidelines to report replications of software engineering controlled experiments [7]. These guidelines suggest describing in separate sections the previous experiment and the replication, and also including an interpretation of results in the context of the experimental family. Following this, the rest of the paper is structured as follows. Section II details the history of the family of experiments that this replication is part of. Section III presents the information about the previous experiment. Section IV describes this experimental replication. Section V discusses the potentially undetectable defects for each technique. Section 0 presents a comparative analysis of the replication with the previous results. Section VII presents the interpretation of the results in the context of the experimental family and finally in Section VIII we present our conclusions.

II. HISTORY OF THE FAMILY OF EXPERIMENTS

Basili started the family of experiments to which this replication is part of in the 1980s. Since then, this experiment has been extensively evolved and replicated [3]–[6]. Some of these replications have introduced changes in the techniques under study or have adapted the materials to suit different contexts. Some replications have focused on code reviews, and other in testing techniques. Some have even considered comparing these approaches.

We consider the first stage of the experiment to be the replications carried out at Maryland University and the Software Engineering Laboratory at NASA Goddard Centre [2]. The second stage of this family of experiments is based on the laboratory package that was built by Kamsties y Lott at Kaiserslautern University [3], and replicated by Roper et al [4]. Juristo and Vegas carried out the following stage at Universidad Politécnica de Madrid (UPM) [5]. This stage produced the laboratory package that has driven the replication.
presented in this paper. Arguably, this family of experiments is one of the longest running families of experiments within the experimental software engineering community.

In addition to providing another replication to this family of experiments, the motivation for this work is to identify new empirical evidence about which types of defects could be potentially undetectable for structural or functional testing techniques. It has been hypothesized that functional and structural techniques complement each other, but there is no empirical evidence to confirm this hypothesis. This replication uses defects that have been seeded in order to evaluate if they are undetectable to either technique. We define a defect to be potentially undetectable to a technique, if the technique capability to find it can only be attributed to chance, and not to the technique prescription for test case generation.

III. INFORMATION ABOUT THE PREVIOUS EXPERIMENT

The replication reported in this paper is based on a previous experiment design proposed by Juristo and Vegas [6] performed at UPM. This experiment is in turn a replication based in the laboratory package created by Kamsties and Lott [3], later refined in multiple replications [4], [5]. When referring to the UPM experiment here, we use the term previous experiment in order to avoid confusion with other experiments in the same family.

One of the main evolutionary changes in the Juristo and Vegas replication was the separation of test design and test execution activities. Each subject performed two separate exercises: one for designing the test cases and other to execute the test cases against the executable program. In both activities, the influence of the human factor can be observed. This reductionist approach has enabled a more focused study of the phenomenon of applying a dynamic software evaluation technique.

Along the replication process, with the objective of studying the behaviour of the techniques against different types of defects, several defect taxonomies have been used. One of the replication used the same defect classification introduced in the Basili experiment. Different types of defects were seeded generating multiple versions of the programs. This design allowed the exploration of defect type as a factor in a significant number of cases. However, this modification did not yield to conclusive results and the defects were re-seeded following a different scheme.

The research question of the previous experiment is: Which is the efficacy of different software evaluation techniques (equivalence partitioning, decision coverage and code review by stepwise abstraction) for detecting defects? This question could be expressed in a null hypothesis format: \( H_0: \) There is no difference in defect detection efficacy between equivalence partitioning, decision coverage and code review by stepwise abstraction.

For the dynamic evaluation techniques (equivalence partitioning and decision coverage) the response variable is measured as the percentage of subjects that generate at least one test case that can detect the failure associated with each fault. For the static evaluation technique (code review by stepwise abstraction) efficacy is measured as the percentage of subjects that report each fault. The previous experiment uses a factorial design. The factor of the experiment (independent variable) is the evaluation technique with three alternatives: equivalence partitioning, decision coverage and code review by stepwise abstraction.

Subjects of the previous experiment were fourth-year students of the course Software Evaluation at UPM. The students profile was medium level of experience in programming. The training on the testing techniques was received during the course, so the testing experience level was low. At an operational level, the experiment was performed in Spanish language. In the previous experiment all the artefacts provided to the subject to accomplish the task are paper form. The programs are coded in C language and have between 200 and 300 lines of code. Three different programs are used: `cmdline`, `nametbl` and `ntree`. Each program has 6 seeded faults, named F1 to F6. Faults F1, F2 and F3 are seeded with the intention to be more difficult to find for the structural technique. Faults F4, F5 and F6 are seeded with the intention of being more difficult to find for the functional technique.

The results of the previous experiment confirm (at a statistically significant level) the hypothesis that some defects are much more difficult to detect for structural of functional testing techniques. The code review technique has an overall lower efficacy, but could potentially detect all the seeded defects.

IV. DESCRIPTION OF OUR REPLICATION

Performing the replication had a double motivation: start an experimental research project about software testing and studying the communications mechanisms in experimental replications [9]. Our replication was performed at Universidad ORT Uruguay from 2010 to 2012. The experimental design was derived from the previous experiment, but changing the number of alternatives in the independent variable (the code review technique was not studied). New artifacts were created in Java to match subjects skills. More importantly, new defect types were seeded to test a new research hypothesis.

The experiment subjects were final year undergraduate students taking a Software Testing course. The experiment was run with three different groups, one in each year using the same experimental design and operation. The number of subjects was 8 in 2010, 9 in 2011 and 7 in 2012. All the cases were jointly analyzed with a procedure already used for obtaining bigger statistical samples in software engineering experiments [8].

To the previous experiment hypothesis about technique efficacy, we added a new null hypothesis in our replication: \( H_{eq}: \) All defects are potentially detectable for equivalence partitioning and decision coverage testing techniques.

Although, the main experimental design remains the same, the code review technique was not studied in this replication. This change has no effect in the possibility of aggregation and comparison of results with previous replications, considering only the shorter list of studied alternatives.
The subjects’ background in programming is different from the previous experiment subjects. To fit this constraint, new programs in Java language were used in this replication. The two generated programs Export and MesConv are of similar size, each one having a 2 page specification and around 200 lines of source code.

Another change of the replication was in the data gathering mechanism. In the previous experiment paper forms were used, while in the replication the subjects had to upload a file with the test cases in an authenticated web system. Each subject could use JUnit as a framework to create the test cases (either functional or structural) but this step was not mandatory.

V. DISCUSSION OF POTENTIALLY UNDETECTABLE DEFECTS

In order to present the results of the replication, a discussion regarding the potentially undetectable defects for each technique is necessary. Although there are several defect classification taxonomies, to the best of our knowledge, none of them has the capability to differentiate technique defect detection capability. Because of this, it is necessary to have a preliminary theoretical discussion about defects types potentially undetectable to structural or functional testing techniques.

Structural and functional techniques use different strategies for test case generation. In the case of equivalence partitioning the program specification is used as the main source, analyzing the semantic and dividing the input space in different equivalence classes. This technique uses heuristics to identify the classes and generate associated test cases. In the case of the structural technique the program source code is normally the main source for generating the test cases. The code is analyzed form a control flow or data flow perspective to generate test cases that cover certain degree of these structures. For example, in decision coverage each branch of execution is tested by at least one test case. This strategies use different foundations, so the generated test cases could lead to some potentially undetectable types of defect for each technique.

Functional techniques could have problems detecting:

- Defects related to combination of equivalence classes. The technique does not prescribe in which way the valid classes should be combined. If the defect is only detectable if certain classes are combined in the same test case, this type of defect could be potentially undetectable.
- Defects present within code of unspecified functions. Since the specification is the basis for the equivalence partitioning, if there is a defect in code not associated with the specification, this type of defects are potentially undetectable for the functional technique.
- Faults associated with implementation structures. Is possible that the developer makes decisions in the implementation structures, not necessary mandated in the specification but following implementation constraints or developer preferences.
- Faults detectable only if specific data set are used in the test case. Equivalence partitioning technique mandates to use at least one data input representative of each class in the test cases.

Structural technique could have problems detecting other types of defects:

- Faults of complete omission of functionalities in implementation. Since there are no internal structures to generate the test cases, the omission could only be detected if by chance the generated test cases cover the omitted function.
- Faults associated with specific combination of predicates or test data, not necessarily exercised by executing decision branches. This is an intrinsic limitation of the decision coverage criterion.

In our replication, 4 defects of similar type were seeded in each program. Defects e1 to e4 were seeded in the Export program, while defects m1 to m4 were seeded in the MesConv program. Based on this classification, we hypothesise that the first two faults are potentially undetectable for the structural technique and the last two potentially undetectable for the functional technique. We try to seed faults of the same type and approximately equal in detection difficulty, although this assumption should be confirmed with the collected empirical data.

VI. ANALYSIS OF RESULTS

In a preliminary analysis a set of descriptive statistics were used to confirm the potentially invisible defects for the structural and functional techniques. The response variable is the percentage of subjects that detect each defect. Table II shows the detection rate and the classification of each defect after the preliminary analysis. Values below 50% detection rate are written in bold.

The number of participants was limited to 24 subjects, each performing two exercises. Since the sample size is near the conventional statistical practice, both parametric (ANOVA: Analysis of Variance) and non-parametric (Mann-Whitney) statistical tests were used. Both tests showed significance in the same independent variables (technique and program).

### TABLE I. DETECTION RATE FOR EACH FAULT AND TECHNIQUE

<table>
<thead>
<tr>
<th>Fault</th>
<th>Structural det. rate</th>
<th>Functional det. rate</th>
<th>Preliminary analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>0.33</td>
<td>0.92</td>
<td>Potentially undetectable to structural</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>1</td>
<td>Highly detectable</td>
</tr>
<tr>
<td>e3</td>
<td>0.5</td>
<td>0.92</td>
<td>Potentially undetectable to structural</td>
</tr>
<tr>
<td>e4</td>
<td>0.33</td>
<td>0.83</td>
<td>Potentially undetectable to structural</td>
</tr>
<tr>
<td>m1</td>
<td>0.25</td>
<td>0.42</td>
<td>Potentially undetectable for both</td>
</tr>
<tr>
<td>m2</td>
<td>0.92</td>
<td>0.92</td>
<td>Highly detectable</td>
</tr>
<tr>
<td>m3</td>
<td>0.08</td>
<td>0.67</td>
<td>Potentially undetectable to structural</td>
</tr>
<tr>
<td>m4</td>
<td>0.17</td>
<td>0.17</td>
<td>Potentially undetectable for both</td>
</tr>
</tbody>
</table>

Regarding the general efficacy of structural and functional techniques, efficacy of the functional technique is significantly better than the structural technique in this replication. After performing the preliminary analysis, this result can be attributed to the fact that only potentially undetectable defects for the structural technique were actually seeded. Besides the
technique factor, the statistical analysis also shows that the program is a significant factor that affect efficacy. This is explained because the MesConv program has specific defects that are more difficult to detect than the Export program counterpart. This is a confirmation that the detection capability of each defect is more dependent on the specific seeded defect and program, than the intended preliminary classification.

VII. INTERPRETATION OF RESULTS WITHIN THE FAMILY OF EXPERIMENTS

There are a considerable number of experiments and replications studying software evaluation techniques, but results are not always consistent. Variations among replications on technique and training factors are not conclusive about which technique to use. Because of this, the safest advice for a practitioner is to use a combination of techniques. The replication reported in this paper focused the investigation in which types of defects could be potentially undetectable to each testing technique.

This family of experiments has also contributed with observations about the human factor in the application of software evaluation techniques. The potential theoretical efficacy of certain technique could be considerably different with the result when applied by practitioners. With the same training, different subjects generate different set of test cases, even for well-established testing techniques. Because of this variation is sensible to promote that the evaluation of critical systems is done involving several persons to archive the maximum potential defect detection capability of the technique.

The experiments in the family show that technique efficacy is dependent of the program and the seeded defect. However, we do not have yet a complete and practical defect classification that is sensitive to technique defect detection capability.

VIII. CONCLUSIONS

This paper has presented a replication of an experiment about software testing techniques that belongs to one the longest running families of experiments in the experimental software engineering community. The results from the replication show that the efficacy of the structural technique (decision coverage) and functional techniques (equivalence partitioning) are similar, although the last set of seeded defect yield to greater efficacy of the functional technique. The observed difference can only be statistically attributed to the natural variation of the subjects involved in the application of the techniques. This result confirms previous observations of other replications of this family of experiments. Nonetheless, this replication adds a new element to the existing empirical knowledge. Our result adds empirical evidence that there are certain types of defects that can be potentially undetectable to certain testing techniques. This means that even with the correct application of the technique, the defect can only be detected by chance when a test case is defined that specifically stimulates the conditions for that defect.

According to the current level of empirical knowledge in the field, exploring the hypothesis that different techniques can detect different types of defects is still relevant. These defect types must be mapped to real defects in production software, which will allow for the development of another classification to continue this line of research. Furthermore, the impact of the human factor in the application of each technique still requires more research. For instance: How many testers are necessary for applying a technique to reach its theoretical defect detection limit? How can testing techniques complement each other according to their patterns of potentially undetectable defects? Only by continuous experimentation and replication iterations, we will be able to produce enough variations to provide practitioners with empirical knowledge to answer these questions.

IX. REFERENCES