Empir Software Eng (2014) 19:378–417
DOI 10.1007/s10664-013-9267-7

Effectiveness for detecting faults within and outside the scope of testing techniques: an independent replication

Cecilia Apa · Oscar Dieste · Edison G. Espinosa G. · Efraín R. Fonseca C.

Published online: 8 August 2013

Abstract The verification and validation activity plays a fundamental role in improving software quality. Determining which the most effective techniques for carrying out this activity are has been an aspiration of experimental software engineering researchers for years. This paper reports a controlled experiment evaluating the effectiveness of two unit testing techniques (the functional testing technique known as *equivalence partitioning* (EP) and the control-flow structural testing technique known as *branch testing* (BT)). This experiment is a literal replication of Juristo et al. (2013). Both experiments serve the purpose of determining whether the effectiveness of BT and EP varies depending on whether or not the faults are visible for the technique (InScope or OutScope, respectively). We have used the materials, design and procedures of the original experiment, but in order to adapt the experiment to the context we have: (1) reduced the number of studied techniques from 3 to 2; (2) assigned subjects to experimental groups by means of stratified randomization to balance the influence of programming experience; (3) localized the experimental

Communicated by: Jeffrey C. Carver, Natalia Juristo, Teresa Baldassarre and Sira Vegas.

C. Apa
Universidad de la República, Julio Herrera y Reissig 565, Montevideo, Uruguay
e-mail: ceapa@fing.edu.uy

O. Dieste
Universidad Politécnica de Madrid, Boadilla del Monte 28660, Madrid, Spain
e-mail: odieste@fi.upm.es

E. G. Espinosa G.
Escuela Politécnica del Ejército Sede Latacunga, Latacunga, Ecuador
e-mail: egespinosa1@espe.edu.ec

E. R. Fonseca C. (✉)
Escuela Politécnica del Ejército, Sangolquí, Ecuador
e-mail: erfonseca@espe.edu.ec

materials and (4) adapted the training duration. We ran the replication at the Escuela Politécnica del Ejército Sede Latacunga (ESPEL) as part of a software verification & validation course. The experimental subjects were 23 master's degree students. EP is more effective than BT at detecting InScope faults. The session/program and group variables are found to have significant effects. BT is more effective than EP at detecting OutScope faults. The session/program and group variables have no effect in this case. The results of the replication and the original experiment are similar with respect to testing techniques. There are some inconsistencies with respect to the group factor. They can be explained by small sample effects. The results for the session/program factor are inconsistent for InScope faults. We believe that these differences are due to a combination of the fatigue effect and a technique x program interaction. Although we were able to reproduce the main effects, the changes to the design of the original experiment make it impossible to identify the causes of the discrepancies for sure. We believe that further replications closely resembling the original experiment should be conducted to improve our understanding of the phenomena under study.

Keywords Replication · Experiment · Unit testing · Reporting guidelines

1 Introduction

Verification and validation (V&V) activities play a fundamental role in improving software quality. There are many approaches for carrying out V&V, but we do not know for certain which technique or combination of techniques is more effective for each type of software validation: unit, integration or system testing.

Determining the effectiveness of unit testing techniques soon attracted the attention of experimental software engineering (SE) researchers. Way back in 1978, Myers compared the effectiveness of functional and structural testing techniques (Myers 1978). Soon after, Basili and Selby studied the effectiveness of functional and structural techniques and code reading (Basili and Selby 1985), and started up a much replicated family of experiments.

In this paper, we report the replication of a controlled experiment belonging to Basili's family. The original experiment was conducted at the Universidad Politécnica de Madrid (UPM) in December 2005. According to Gómez et al. (2010) and Gómez (2012), this replication can be classified as a literal (that is, the replication resembles the original experiment as closely as possible), joint (some of the original experimenters participated in the replication) and external (the replication was conducted at a different site) replication of the original experiment. The replication was conducted at the Escuela Politécnica del Ejército Sede Latacunga (ESPEL) in Ecuador in December 2011.

The purpose of conducting the replication was to verify the results of the original experiment, and, in the event of inconsistencies, identify which factors or parameters could explain the differences between the experiments. The alternative, if this were not possible, would be to conduct further replications. Another goal of this paper was to use Carver's guidelines (Carver 2010) and evaluate their strengths for reporting replications.

Following Carver's guidelines, the paper describes information about the original study in Section 2, details information about the replication in Section 3, compares

the replication results to the original results in Section 4 and, finally, reports the conclusions across studies in Section 5.

2 Information About the Original Study

The original experiment was carried out by Juristo et al. (2013). Its aim was to study how effective structural and functional testing techniques are at detecting faults. Juristo et al.'s (2013) experiment is, in turn, a differentiated replication of the family of experiments started by Basili and Selby (1985, 1987) in 1982, and later replicated by several researchers like Kamsties and Lott (1995) and Wood et al. (1997). The main difference between the original experiment and Basili's family of experiments lies in the way in which the faults were seeded in the programs.

Basili, Selby and others referred to the types of faults used in their experiments as fairly representative of the defects that tend to occur in software development (Basili and Selby 1987). Therefore, those experiments evaluated how effective the testing process is expected to be in practice. However, the fault types used in Juristo et al.'s (2013) experiment were simpler. They can be divided into two major categories: faults that can and faults that, at least in theory, cannot be detected by the structural or functional test case generation strategy (InScope and OutScope faults, respectively).

The differentiation of the two fault types is useful for studying the effectiveness of the testing techniques more precisely than before, as it helps to distinguish the effect of the actual technique (that is, the detection of InScope faults) from the positive effects that the technique has on the tester but that cannot be attributed to the technique per se (that is, OutScope faults). Both effects were confounded in previous experiments.

2.1 Research Questions

The original experiment report does not include an explicit research question. However, this question can be easily inferred from the experimental design. It can be stated in GQM (Basili 1992) as follows:

Analyse the application of software testing techniques for the purpose of finding out how effective they are at unit testing level with respect to different fault types (InScope, OutScope) from the point of view of testers in the context of a controlled experiment in academia.

This research question is useful for identifying the key elements of the original experiment.

(A) **Main Factor:** The original experiment studied two testing techniques: the functional testing technique known as *equivalence partitioning* (EP) and the control-flow structural testing technique known as *branch testing* (BT).
The original experiment also tests another technique: code reading by stepwise abstraction technique (CR). As CR is capable of detecting all fault types, it is used in the original experiment for the purpose of control not as a main factor. According to the original experimenters (Juristo et al. 2013), *"As a control group, we have used a technique* (the CR technique) *with a strategy capable, at least in theory, of detecting all program faults"*.

(B) **Response Variables:** Technique effectiveness was measured as the percentage of faults located by the techniques over the total seeded faults. As the research question is concerned with **InScope** and **OutScope** faults, the effectiveness of the techniques was calculated separately for each fault type.

(C) **Hypotheses:**

H_{10}: There is no difference in the effectiveness of EP, BT and CR with respect to the detection of faults within their scope.

H_{11}: The effectiveness of EP, BT and CR differs with respect to faults within their scope.

H_{20}: There is no difference in the effectiveness of EP, BT and CR with respect to the detection of faults outside their scope.

H_{21}: The effectiveness of EP, BT and CR differs with respect to faults outside their scope.

2.2 Participants

The subjects participating in the experiment were 46 Universidad Politécnica de Madrid undergraduate computing engineering students. The students were taking the 4th-year software verification and validation course as part of their five-year degree programme. Students had little or no professional experience of software development.

2.3 Design

As a between-subjects design with a total of 46 experimental subjects would result in very few subjects per group ($46/3 \simeq 15$) and low statistical power, the authors of the original study used a within-subjects design, where each subject applied each of the tested techniques at three different times (sessions). Sessions had no set time limit, and each session lasted on average four hours. Table 1 summarizes the experimental design.

Although the within-subjects design increases statistical power, it also poses several validity threats. For example, carryover, learning effects (maturation) and fatigue are a possibility:

(a) **Carryover:** The residual effect that administering one treatment to a subject has on another treatment administered later to the same subject, where the residual

Table 1 Original experiment design

Program	Cmdline			Ntree			Nametbl		
Session	Session 1			Session 2			Session 3		
Techniq.	CR	BT	EP	CR	BT	EP	CR	BT	EP
Group 1	X	–	–	–	X	–	–	–	X
Group 2	X	–	–	–	–	X	–	X	–
Group 3	–	X	–	–	–	X	X	–	–
Group 4	–	X	–	X	–	–	–	–	X
Group 5	–	–	X	X	–	–	–	X	–
Group 6	–	–	X	–	X	–	X	–	–

effect increases or decreases the effectiveness of the later treatment, is known as carryover (Brown 1980). Carryover is an important risk in medical experiments, as drug residues can remain in the body for quite some time and interact with later treatments (Senn 2002). It is harder to imagine what underlying cause could produce a carryover effect in SE. However, carryover has been explicitly cited in the SE literature (Kitchenham et al. 2003) and hence should be taken into account.

(b) **Learning:** Subject performance may increase irrespective of the applied treatments as a result of practice acquired in successive experimental sessions.

(c) **Fatigue:** Subject performance may drop as a result of fatigue caused by applying treatments at short intervals in successive experimental sessions.

To minimize all these threats, the sessions were held one week apart. Even so, as shown in Table 1, the original experimenters added the session and group variables to the design in order to determine whether such threats materialize. Specifically, according to the original experimenters, the session variable can identify the presence of learning or fatigue effects, and the group variable can detect carryover effects.

2.4 Artefacts

Several artefacts were used to operationalize the original experimental design:

- Training materials to improve or consolidate subject knowledge of the techniques under study
- Experimental objects on which to apply techniques
- Experimental materials to support experimental task performance.

The above-mentioned artefacts are described in the following. They are all available at Juristo et al. (2013), with the exception of the programs and fault descriptions. This is meant to assure that students participating in the experiments are not acquainted with them beforehand. However, programs and fault descriptions are available from the original experimenters via email.

2.4.1 Training Materials

The material used to train subjects on the application of the software testing techniques under study includes:

- **Reference guide:** Document containing the theoretical foundations and practical exercises for training experimental subjects in the application of software testing techniques.
- **Slides:** Support material for the trainer containing a summary of the reference guide.
- **Training programs:** Material used to supplement the theoretical groundwork, focusing on acquainting experimental subjects with the experiment execution environment.

2.4.2 Experimental Objects

For testing technique application, there should be at least as many programs as sessions, seeded with the right faults, because subjects cannot test the same

program with the same faults twice. The programs and faults used in the original experiment were:

(A) **Programs:** Three programs written in C were used: **cmdline**, **nametbl** and **ntree**. These are the same programs that were used in experiments run by Kamsties and Lott (1995) or Roper et al. (1997). As these programs are likely to affect (increase or decrease) the effectiveness of techniques, the programs were considered as blocking variables for analysis purposes. These programs perform the following functions:

  (a) **Cmdline** parses a command line to determine whether it is valid. If it is, it displays a description of the input command line; if it is not, it specifies why it is incorrect. Note that the program does not execute any of the commands, but merely validates the input.
  (b) **Nametbl** reads and processes file commands to test a series of functions used to manage an abstract data type, specifically a particular programming language symbol table.
  (c) **Ntree** reads and processes file commands to test a series of functions used to manage an abstract data type, specifically an n-ary tree.

(B) **Faults:** As mentioned earlier, different fault types were seeded in the original experiment than were used in earlier experiments (e.g.: Kamsties and Lott 1995; Roper et al. 1997).
The foremost difference between the original experiment and earlier experiments is the InScope and OutScope fault categorization for EP or BT. An InScope fault is a fault that can be detected by a correctly applied technique; faults are classed as OutScope otherwise.
Thus, for example, *unimplemented parts of the specification,* is a possible example of a fault that EP can detect (InScope). EP is capable of revealing this fault type because it generates test cases for all program specifications, including non-implemented specifications. Alternatively, *code for functionalities that are not in the specification* (e.g. when a programmer has mistakenly written or copied code that implements functions not accounted for in the specification) is an example of a fault that BT can detect. Branch testing's strategy prescribes that test cases should be generated to cover all the alternatives of 100 % of the code decisions, in which case it should detect superfluous code.
In order to systematically seed programs with faults, faults would have to have been classified by technique sensitivity. However, the original experimenters were unable to find any such classification, and they therefore generated the necessary fault types, shown in Table 2.

Table 2 Fault types (FT)

FT	Description
1	Unimplemented specification
2	Specific test data for achieving coverage
3	Combination of invalid equivalence classes
4	Chosen combination of valid equivalence classes
5	Test data for combining classes
6	Implementation detail
7	Implementation of unspecified functionality

Each of the fault types proposed by Juristo et al. (2013) are further detailed below.

(a) **Unimplemented specification:** The program does not implement the code for a particular specification. The BT technique is unable to identify this fault type, whereas EP will generate test cases capable of revealing this type of faults.
(b) **Specific test data for achieving coverage:** The program does not cover all the values specified in the requirements. The BT technique protocol does not indicate which data to select to assure that the test cases cover code decisions, whereas EP will generate test cases capable of detecting this fault type.
(c) **Combination of invalid equivalence classes:** Faults entered in programs by adding code that does not comply with the program specification. BT is capable of detecting this fault type, but EP is not.
(d) **Chosen combination of valid equivalence classes:** Unnecessary functionalities added to the code, which are already covered by another program specification. BT is capable of detecting this fault type, whereas EP is not.
(e) **Test data for combining classes:** Coding faults caused by the inclusion of functionalities that are not stated in the specifications. EP is not always able to detect this fault type, because, although the EP strategy prescribes that test cases should cover all the identified equivalence classes, it does not state exactly which class data should be selected for the test case. On the other hand, BT will generate test cases to detect this fault type.
(f) **Implementation detail:** Faults entered by programmers as a result of the choice of programming strategy to comply with the program specification. EP is unable to find this fault type, whereas BT can generate test cases capable of discovering such faults.
(g) **Implementation of unspecified functionality:** The program implements a functionality that is not in the specification. The EP technique is unable to identify this fault type. On the other hand, if applied correctly, BT will reveal such faults.

The programs were each seeded with six faults of the seven fault types. Three of these faults (called F1-F3) were InScope faults for EP, and the other three (F4-F6) were InScope faults for BT. Remember that the faults that are InScope for one technique are OutScope for the other, and vice versa. Table 3 details the faults seeded in each program.

2.4.3 Experimental Materials

The material used to execute the experiment includes the program specifications, experimental data collection forms, source code listing with seeded faults, executable code with seeded faults and guidelines for executing the experiment.

2.5 Context Variables

The original experiment explicitly considered the following contextual variables:

- **Environment:** Academia
- **Subject type:** Undergraduate students
- **Experience:** Students with little or no professional software development experience

Table 3 Faults seeded in programs

Technique	FT	F1	F2	F3	F4	F5	F6
Cmdline							
EP	1	X		X			
	2		X				
BT	3				X		
	4						X
	5					X	
Nametbl							
EP	1	X					
	2		X	X			
BT	3					X	
	5						X
	7				X		
Ntree							
EP	2	X	X	X			
BT	3					X	
	5				X		
	6						X

- **Program type:** Small-sized programs (150–220 LOC), with cyclomatic complexities ranging from 21 to 61
- **Program language:** C.

2.6 Execution Procedure

The original experiment was executed in three clearly defined stages, which the original experimenters called pre-session, during-session and post-session.

2.6.1 Pre-session

The pre-session is the stage of the experiment during which experimental subjects receive training on how to apply the testing techniques. The materials to be used to execute the experiment are also prepared at this stage. These materials, available in Juristo et al. (2013), are as follows:

- Experimental objects
- Forms
- Guides

2.6.2 During-Session

This is the stage during which the experiment proper is executed. The subjects take their places, far enough apart so that they cannot copy, in a room equipped for the purpose. Subjects will have been randomly assigned to groups before the first session. This assures that experimenters deliver the right materials to subjects during the session.

The first phase of the experimental session is test case generation. To do this, the subjects applying EP are supplied with:

- Experimental object specification (specification of the cmdline, nametbl or ntree program)

– Data collection forms for recording the test case and the expected output of each test case.

For the BT technique, subjects are supplied with:

– Source code listing with seeded faults in order to generate the test cases
– Data collection forms for recording the test cases and the expected output of each test case.

Note that none of the experimental subjects are given executable code during the test case generation phase, as the subjects might be tempted to read the programs instead of generating test cases to detect faults. This would frustrate the purpose of the experiment, which is to test how effective testing techniques, not software testers, are at detecting faults.

After they have generated the test cases, they are given:

– Executable code with seeded faults
– Data collection forms for recording the observed output of each test case execution
– The program specifications in order to test the generated cases
– Data collection forms for recording the detected faults.

The experimental subjects are not allowed to add any other test cases once they have been given the executable code at the end of the test case generation stage. This is controlled by the person responsible for monitoring the experiment execution.

Finally, the subjects fill the respective form with the outputs observed during the execution of each test case. The program faults are inferred from the comparison of the observed outputs with the expected outputs.

2.6.3 Post-Session

During this stage, all the material used by the experimental subjects is collected, and the data collection forms containing the test cases designed by the experimental subjects are detached for analysis. To do this, the test cases generated by the subjects are compared with previously designed test case templates. This analysis reveals the faults discovered by each subject applying each technique. This is not a strict comparison protocol, and test cases containing some sort of formal error that, with a minor correction, would generally identify faults are rated positively.

2.7 Summary of Results

Repeated measures ANOVA (rANOVA) was used to analyse the results, where the technique (BT, EP, CR), the program (ntree, cmdline, nametbl) and the group (six groups, termed G1-G6) were considered as factors. The sessions were not explicitly considered as factors, as they were confounded with the programs. The session/program and technique factors were considered as within-subjects factors, whereas the group factor was treated as a between-subjects factor. The fitted model was strictly additive and took the form:

$$Y = TECHNIQUE + PROGRAM + GROUP + e$$

The results obtained by the original experimenters can be summarized as follows.

2.7.1 InScope Response Variable

The rANOVA provided significant results for all factors: technique, program/session and group:

(A) **Technique:** The significant rANOVA result leads to the null hypothesis being rejected. Pairwise comparison showed that the code reading technique was less effective than the *equivalence partitioning* and *branch testing* techniques. In actual fact, code reading detected approximately 54 % of faults, whereas branch testing and equivalence partitioning detected 67.67 and 78.70 % of the seeded faults, respectively.

(B) **Program/session:** The significant rANOVA result suggests that either the program or the session influences effectiveness, but, as they are confounded, there is no way of reliably telling which one really has a bearing. Although the program/session interaction could theoretically have an effect, it is unlikely to do so because there is no reason why a program should increase the effectiveness of a session or vice versa.
The pairwise comparison reveals that cmdline<nametbl<ntree (we use the "<"symbol to indicate that subjects are less effective for cmdline than for nametbl and so on for all other variables). Alternatively, S1<S3<S2. The differences are only significant for cmdline/S1 and ntree/S2. As the design type used confounds the session and program variables, there are three possible grounds for the observed results: learning, fatigue or the possibility of one program being easier to test than another. If there were a learning effect, we should observe S1<S2<S3. If there were a fatigue effect (very unlikely, however, as the sessions were held one week apart), we should find that S3<S2<S1. Consequently, the program rather than the session is more likely to have a bearing on effectiveness.

(C) **Group:** As already mentioned, the group factor was introduced to detect the presence of carryover, as the original experiment had a within-subjects design. Pairwise comparison reveals significant differences of effectiveness among the sequences EP-CR-BT and EP-BT-CR and BT-CR-EP with respect to the order in which the techniques were applied. With a fault detection rate of 82.41 %, the EP-CR-BT sequence was more effective than the EP-BT-CR and BT-CR-EP sequences with rates of 55.55 and 57.41 %, respectively. As there are some significant differences between some levels of this factor but not between others, the original experimenters claimed that there does not appear to be any carryover effect from one technique to another.

2.7.2 OutScope Response Variable

The rANOVA provided significant results for all the factors: technique, program/session and group.

(A) **Technique:** The significant rANOVA result leads to the null hypothesis being rejected. Pairwise comparison shows that the *equivalence partitioning* technique was less effective (14.12 %) than the *branch testing* technique (29.09 %).

(B) **Program/session:** In this case again, the rANOVA result is significant and suggests that either the program or the session influences effectiveness. Because the program and session are confounded, there is no way of reliably telling which has a bearing on effectiveness.

The pairwise comparison shows that cmdline<nametbl (or, in session terms, S1<S3). Ntree (or S2) does not have significant differences with respect to the other two programs (sessions). This difference of effectiveness between S1 and S3 could be attributed to the session and not to the program. However, the results on technique effectiveness for Inscope faults, plus the fact that there are no statistical differences between S3 and S2, led the original experimenters to conclude that it is the program and not the session that makes the difference. As for the InScope faults, there would again be no learning effect.

(C) **Group:** Pairwise comparison reveals significant differences of effectiveness among sequences EP-CR-BT (8.33 %), and CR-BT-EP and EP-BT-CR (35.42 and 36.11 %, respectively) only with respect to the order in which the techniques were applied. As with the InScope variable, the original experimenters claimed that there does not appear to be any carryover effect improving technique effectiveness for faults outside their scope due to the order of technique application.

3 Information About the Replication

The replication was conducted at the Escuela Politécnica del Ejército sede Latacunga (ESPEL), Ecuador, with Master in Software Engineering students taking a software verification and validation course. The duration of this course is 80 h divided into seven, 10-h face-to-face sessions and a 10-h off-campus period set aside for homework and academic administrative matters. The face-to-face sessions were divided into three stages. In the first stage, the first part of the theoretical groundwork of the training was taught across three consecutive sessions. This was followed by a three-day break. Then the second stage (two consecutive sessions) covered the second theoretical part and training exercises. The third stage was the replication, which was run across two consecutive sessions as of the following day. The replication was one of the course assessment tests that carried most weight. This was done purposely to assure that students were motivated.

3.1 Motivation for Conducting the Replication

The main reason for conducting a replication was to confirm the results of the original experiment or, in the event of inconsistencies, identify the factors and parameters that might have caused such inconsistencies. This could, if necessary, trigger another replication cycle. The independent experiment replication should, secondarily, help to improve our competence at applying empirical methods in SE research.

3.2 Level of Interaction with the Original Experimenters

There was a lot of interaction with the original experimenters in the phases prior to the execution of the experimental replication. The replication was executed without the involvement of the original experimenters. Finally, the original experimenters played a very minor role in data collection and analysis.

(A) At the start of the replication we had to gain a profound understanding of the original experiment. For this purpose we held several meetings with

the original experimenters. Once we had a thorough understanding of the experiment, the next step was to adapt the design of the original experiment to the context where the replication was to be carried out. As a result of this adaptation, we generated a design document, which was validated by the original experimenters. Finally, we were given the artefacts of the original experiment.

(B) We did not have any interaction with the original experimenters during the preparation of the training materials, training proper and replication execution.

(C) After we had executed the replication, the original experimenters explained the procedure for identifying whether or not a test case reveals a fault from the questionnaires completed by each experimental subject.

3.3 Changes to the Original Experiment

Generally speaking, the replication is quite true to the original experiment in all respects. The hypotheses, factors, faults, response variable, materials and experimental procedure are all unchanged, save the following exceptions:

- We have eliminated one of the main factor levels. Specifically, we did not test the code reading technique, primarily because it was impossible to run three experimental sessions in the time available for running the replication.
- As we have omitted one of the techniques, one of the three sessions and one of the three programs are unnecessary. In fact, we have omitted the cmdline program and reduced the number of sessions from three to two.
- We have altered the order in which the programs were used in order to study whether it is the program or the session that influences technique effectiveness.
- We have run the replication with a group of experimental subjects that have different characteristics than the subjects of the original experiment.
- We have applied stratified randomization (Kernan et al. 1999) to assign subjects to experimental groups in order to assure balanced groups.
- We have adapted the training to the time constraints of the course on which the replication was run.
- We have localized the experimental materials to the dialectal differences of Ecuadorian Spanish.

Table 4 summarizes the changes. The changes are described in more detail in the following.

3.3.1 Changes to the Main Factor Levels

According to the software verification and validation course schedule, only two days were available for running the experiment, whereas three days (one per session) had been spent on the original experiment. We had two options in this respect:

1. Squeeze two experimental sessions into one day.
2. Eliminate one of the technique factor levels.

A third option, which meant splitting a session across two days, was rejected outright as there was a risk of the intermission affecting student performance or simply of students swapping notes.

Table 4 Differences between UPM and ESPEL

Activity	Characteristic	UPM	ESPEL
Design	Randomization	Normal	Stratified
	Sessions	3	2
	Main factor	CR, BT, EP	BT, EP
	Programs	cmdline, ntree, nametbl	nametbl, ntree
	Groups	6	2
	Dialectal differences in materials	Castilian Spanish	Ecuadorian Spanish
Recruitment	Number	46	23
	Type	Undergraduate students	Master's students
	Professional experience	Generally inexperienced	Yes
Training	Training type	Face-to-face	Semi-distance
	Duration	12 h	50 h
Execution	Duration	3 sessions, unlimited time	2 sessions, unlimited time

Finally, we went for the second of the two options. The fact that the sessions were scheduled for two consecutive days (Saturday and Sunday) went against the first option: it would have been very demanding on students to take part in three sessions without a break, and the resulting fatigue could have had a negative influence on the results. Also, the fact that the original experiment really tested the functional and structural techniques, whereas code reading was primarily a control technique, and therefore optional, favoured the second option.

Consequently, the factor levels were the functional testing technique known as *equivalence partitioning* (EP) and the control-flow structural testing technique known as *branch testing* (BT), each applied in one session by two groups of experimental subjects (Group 1 and Group 2). Table 5 shows the resulting experimental design.

3.3.2 Changes to the Secondary Factor Levels

The omission of one the testing techniques (code reading) makes one of the three sessions unnecessary. Consequently, it was also necessary to eliminate one of the programs used in the original experiment. We left out the cmdline program for two key reasons:

1. Experimental subjects stated, during the original experiment and other experiments of the same family (Juristo and Vegas 2003), that the *cmdline* program was a harder to understand and test than *ntree* and *nametbl*. Testing techniques are generally less effective on cmdline, suggesting that this program is more complex, as discussed in Section 2.7.
2. The ntree and nametbl programs are similar to each other (146-172 LOC and a cyclomatic complexity of 21-29, respectively), and both are different to

Table 5 Replication design

Session	Session 1		Session 2	
Program	Nametbl		Ntree	
Technique	BT	EP	BT	EP
Group 1	X	–	–	X
Group 2	–	X	X	–

cmdline (209 LOC and a cyclomatic complexity of 61). cmdline's high cyclomatic complexity is a possible explanation for it being harder to test.

In the original experiment (see Table 1), the cmdline program was used in Session 1, whereas ntree and nametbl were used in Sessions 2 and 3, respectively. This would apparently signify that the sessions in the original experiment and the replication are not comparable, as they are associated with different programs. However, this should not be a problem as there is unlikely to be any relationship between session and program, as already stated.

3.3.3 Change to the Order of Program Use

As discussed in Section 2.7, the analysis is unable to distinguish whether the effects are due to either factor because program/session are confounded. This is a peculiarity of the cross-over design used in the original experiment. In this replication, they are again confounded because we adhere to the original design. On this ground, we have decided to change the order in which the programs are used from ntree then nametbl in the original experiment to nametbl then ntree in the replication. By comparing the original and replicated experiment, we expect to be able to identify whether the possible effects are really due to the program or the session.

3.3.4 Change to Population Type

The subjects of the original experiment at the UPM were undergraduate students, most of whom had little or no professional experience. On the other hand, the subjects of the replication at ESPEL were master's students, many of whom did have professional programming, architectural design or other software development experience.

Because of their higher academic and professional level, the marginal means of ESPEL subjects should be greater than for UPM subjects. However, this should not affect the comparisons between factors and treatments, as the implemented stratified randomization (see Section 3.3.5) assures that the experimental groups are homogeneous.

3.3.5 Balancing Experimental Groups

The design of the replication called for the formation of two groups of experimental subjects called Group 1 and Group 2. These two groups were formed from 23 experimental subjects. Because the subjects of the ESPEL replication have a different level of professional experience than the subjects of the original experiment, we decided to stratify the groups depending on this characteristic to assure a more reliable balancing of groups. To do this, we conducted a survey of 21 subjects (two did not attend on the day that the survey was administered) to get a better picture of their professional experience, assuming that professional programmers or testers would apply the techniques more effectively than subjects with less experience.

In the survey, the experimental subjects were asked about their general programming experience, C programming experience and software testing technique experience. The results are illustrated in Figs. 1, 2 and 3, and tabulated in Table 21 of Appendix B.

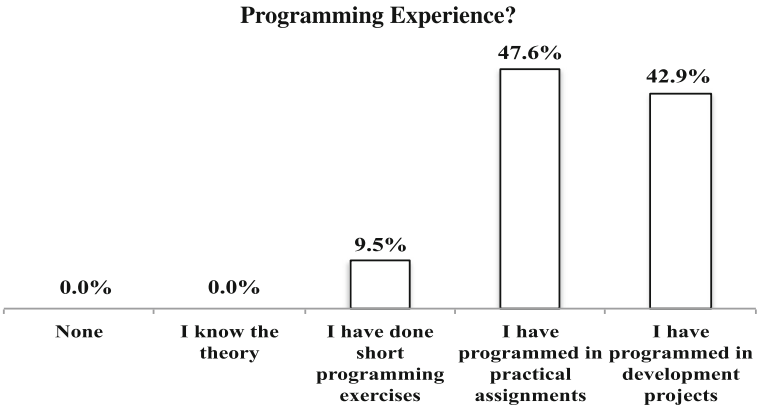

Fig. 1 Survey results—programming

We found that almost half of the experimental subjects are professional programmers, whereas the other half have only programmed as part of practical assignments. C is not the most popular programming language used by subjects for formal development, but they are all acquainted with the language, at least in theory. A considerable percentage (20 %) of experimental subjects are not familiar with software testing techniques. None are professional testers, and generally they have only used testing techniques as part of practical assignments.

Experience in both C and the testing will definitely have an effect. However, 81 % of subjects have no professional experience in C, whereas none of the subjects have professional testing experience. The difference in the effectiveness between inexperienced and experienced subjects will be very small for both variables, as the value range is from *No experience* to *I have done practical assignments*. Therefore, we can assume that experience in C and testing will have a rather small effect or, at least, less than professional experience in programming is likely have.

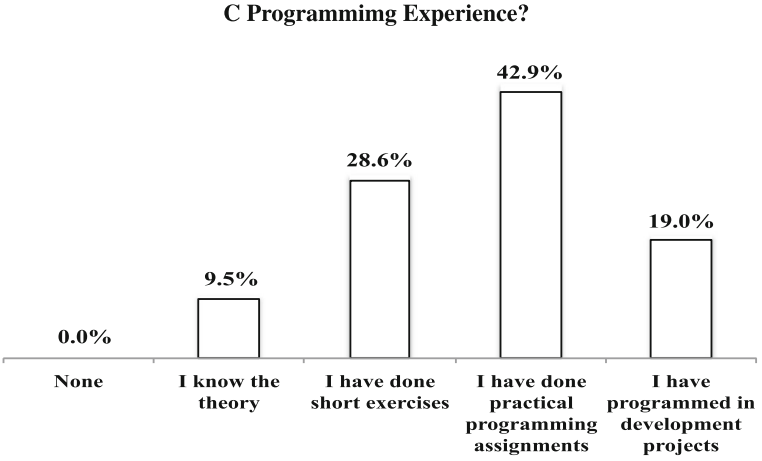

Fig. 2 Survey results—C programming

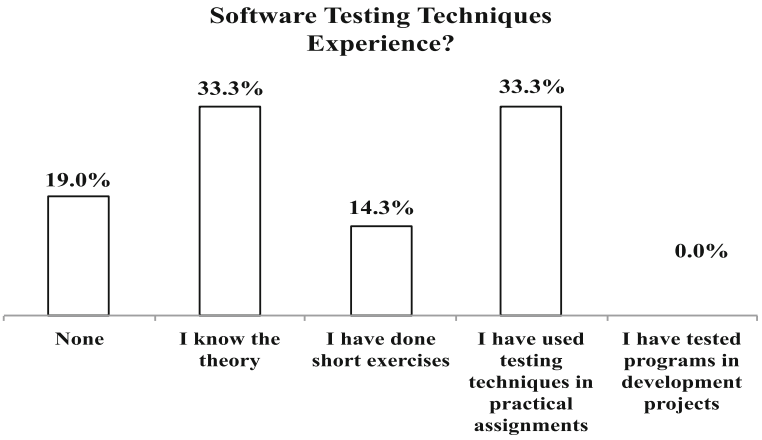

Fig. 3 Survey results—software testing

On the other hand, programming experience did vary considerably (roughly 40 % of subjects have professional programming experience, compared with 60 % who do not), and it is reasonable to assume that this experience may well have a bearing. On this ground, we conducted a stratified randomization (Kernan et al. 1999) based on programming experience. The two subjects that were not surveyed were each allocated to one group (G1 (EP-BT) or G2 (BT-EP)) at random.

In this way, we produced two groups of experimental subjects that were balanced with respect to programming knowledge. With respect to C programming experience, both groups were balanced regarding the number of subjects that used C in industry or academia (around 20 and 80 %, respectively), as shown in Fig. 4. G1 subjects had a slight advantage in terms of the type of experience that they had acquired in academia. Of G1 group subjects, 54 % have used C in practical assignments, whereas the remaining 27.3 % know the theory or have completed short exercises. The respective percentages for the G2 group are 30 and 50 %.

The G1 group also appears to have slightly more software testing technique experience, as shown in Fig. 5. Of these subjects, 45.5 % have experience in practical assignments, whereas the remaining 54.6 % know the theory or have completed short exercises. The respective percentages in the G2 group are 20 % and 80 %.

This imbalance between groups could result in subjects from the EP-BT group being generally more effective than subjects from the BT-EP group. Although we think this is a remote possibility, it should be taken into account during the discussion of the results of the replication. The between-group differences are confined to some subjects in the G1 group having completed more practical assignments than G2 group members. We have the feeling that any difference there is will be small. Additionally, all subjects have received special-purpose training in software testing before the experiment, which includes practical exercises. This training should have further reduced the differences between the groups.

3.3.6 Training Adaptation

The teaching method applied in the original experiment was divided into three four-hour sessions plus homework, each of which was held one week apart. The

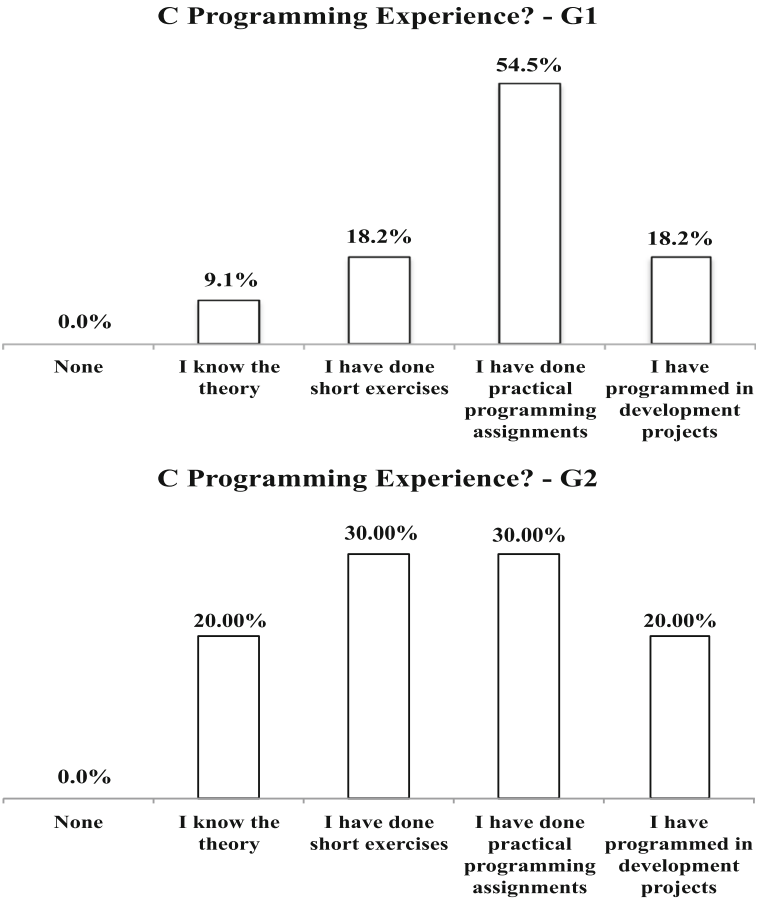

Fig. 4 Results of the stratified randomization for C programming

replication training, on the other hand, had to be adapted to the semi-distance teaching method at ESPEL, with five consecutive 10-h sessions, during which students completed all the practical exercises. In view of this training method, the fatigue factor is a validity threat, which may have influenced subject performance.

3.3.7 Localization

Although similar syntactically, the Spanish spoken in Spain (Castilian) and the Spanish spoken in Latin America, particularly Ecuador, are slightly different with respect to the words and idioms used locally. On this ground, we modified details, such as terms and phrases, that are not common in the dialect spoken in the area where the replication was conducted to ease understanding. We also corrected some minor ambiguities found in the original material. The material used for replication purposes is available at: http://www.grise.upm.es/sites/extras/12/.

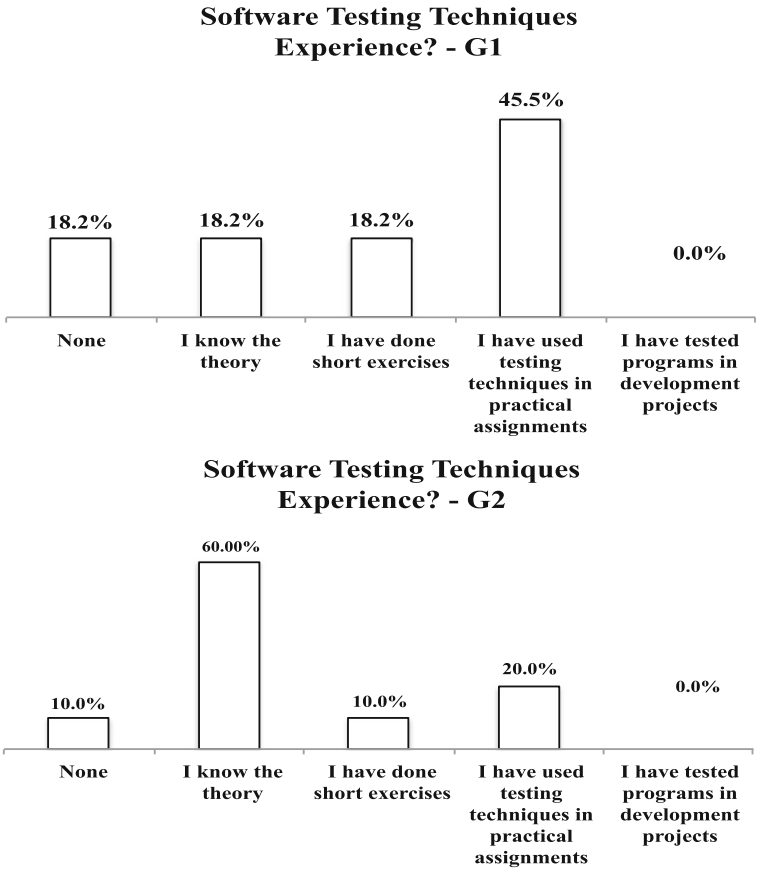

Fig. 5 Results stratified randomization for testing experience

3.4 Replication Execution

The replication was executed according to a similar procedure to the original experiment. The subjects were given the same experimental materials (experimental objects, program specifications, source code listing with seeded faults, forms, guides and executable code with seeded faults), and generated and proceeded to execute test cases in order to check that the identified faults caused failures. There were no major events (e.g. drop-outs, errors in materials delivery, etc.) during the replication execution. The data were analyzed according to the same procedure as in the original experiment.

Table 6 Test of within-subjects effects

Source	Type III sum of squares	df	Mean square	F	Sig.
Technique	1999.054	1	1999.054	4.517	0.046
Session/program	4655.948	1	4655.948	10.520	0.004
Error (technique)	9294.241	21	442.583		

Table 7 Test of between-subjects effects

Source	Type III sum of squares	df	Mean square	F	Sig.
Intercept	68201.322	1	68201.322	100.846	0.000
Group	6028.757	1	6028.757	8.914	0.007
Error	14202.195	21	676.295		

3.5 Replication Results

This section describes the results of the replication. Specifically, we report the hypothesis tests and multiple comparisons for each response variable (InScope or OutScope). The raw data and descriptive statistics are reported in Appendix C.

Like the original experiment, the experimental design used in the replication lends itself to a repeated-measures analysis of variance (rANOVA) and the same additive model. SPSS V.20 was used for all calculations.

3.5.1 Response Variable: Effectiveness for Faults Within Technique Scope

There are two requirements for applying a rANOVA: homogeneity of the covariance matrices and sphericity.

Box's M test is used to check the condition of homogeneity of covariance matrices. Box's M tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups (Meyers et al. 2006). For our sample, M = 3.755, F = 1.122, df1 = 3, df2 = 111064.484, sig. = 0.338, that is, the results verify the null hypothesis and the data are, therefore, homogeneous.

Mauchly's test is used to check the sphericity condition. In our case, however, there are only two levels of repeated measures (for both technique and session/program), which precludes a sphericity violation (Meyers et al. 2006), and, therefore, the test is unnecessary.

As the analysis contains within- and between- subjects factors, we obtain two different tables, one for each factor type, instead of the standard ANOVA table (Tables 6 and 7). The results suggest that the technique, session/program and group factors all influence the effectiveness with respect to the detection of faults within technique scope. Therefore, the null hypothesis (there is no difference in the effectiveness of *equivalence partitioning* and *branch testing* with respect to the detection of faults within their scope) for this response variable is rejected.

The results of the pairwise comparison for the **Technique** factor, which are shown in Table 8, suggest that *branch testing* is less effective than *equivalence partitioning* (with an effectiveness of 31.943 and 45.140 % respectively).

The pairwise comparisons for the **session/program** factor suggest that **S1/nametbl** is more effective than **S2/ntree** (with an effectiveness of 48.612 and 28.471 %, respectively), as shown in Table 9. As only one program is used in a session, it is not possible at this point to distinguish whether the effect is produced by the

Table 8 Pairwise comparisons test for technique

Tech1	Tech2	Mean dif.	Std. dev.	Sig.
BT	EP	−13.197	6.210	0.046

Table 9 Pairwise comparisons test for program

Prog1	Prog2	Mean dif.	Std. dev.	Sig.
S1/nametbl	S2/ntree	20.140	6.210	0.004

Table 10 Pairwise comparisons test for group

Group1	Group2	Mean dif.	Std. dev.	Sig.
BT-EP	EP-BT	−22.918	7.676	0.007

Table 11 Test of within-subjects effects

Source	Type III sum of squares	df	Mean square	F	Sig.
Technique	2905.36	1	2905.36	5.567	0.028
Session/program	6.577	1	6.577	0.013	0.912
Error (technique)	10959.95	21	521.902		

Table 12 Test of between-subjects effects

Source	Type III sum of squares	df	Mean square	F	Sig.
Intercept	17357.588	1	17357.588	20.086	0.000
Group	354.129	1	354.129	0.41	0.529
Error	18147.296	21	864.157		

Table 13 Pairwise comparisons test for technique

Tech1	Tech2	Mean dif.	Std. dev.	Sig.
BT	EP	15.910	6.743	0.028

Table 14 Pairwise comparisons test for program

Prog1	Prog2	Mean dif.	Std. dev.	Sig.
S1/nametbl	S2/ntree	0.757	6.743	0.912

Table 15 Pairwise comparisons test for group

Group1	Group2	Mean dif.	Std. dev.	Sig.
BT-EP	EP-BT	5.554	8.677	0.529

session, by the program or by both. We will try to clarify whether the session or the program caused the observed effect when we examine the OutScope faults and, especially, when we compare the replication results with the original experiment.

The results of the pairwise comparison for the **Group** factor suggest that the BT-EP group (which applies *branch testing* in session 1 followed by *equivalence partitioning* in session 2) is less effective than the EP-BT group (with an effectiveness of 27.082 and 50.00 %, respectively), as shown in Table 10.

There are two possible explanations for this result:

- A carryover effect could be influencing the effectiveness of techniques depending on whether they are applied in first or second place. Carryover signifies an increase (or decrease) in the effectiveness of the technique that a subject applies in second place.
- As mentioned earlier in Section 3.3.5, the EP-BT group is slightly more experienced than the BT-EP group in C and software testing. This superior experience could explain why the EP-BT group is more effective.

We will try to determine whether the observed effect is to due to between-group differences or carryover when we study the OutScope faults and, later, when we compare the results of the replication with the original experiment.

3.5.2 Response Variable: Effectiveness for Faults Outside Technique Scope

We check whether the sample has the sphericity and homoscedasticity properties before conducting the statistical analysis. Mauchly's W and Box's M statistics again confirmed those properties ($W = 1.000$, Approx. Chi-Square $= 0.000$, $df = 0$, Sig. $= 0.000$; $M = 11.947$, $F = 0.429$, $df1 = 3$, $df2 = 111064.484$, Sig. $= 0.732$).

The analysis results, which are shown in Tables 11 and 12, suggest that there are significant differences for the technique factor, but not so for the session/program and group factors. Therefore, as in the case of the InScope response variable, the null hypothesis is rejected.

Regarding the **Technique** factor, Table 13 shows that *branch testing* is more effective than *equivalence partitioning* for faults outside technique scope (with an effectiveness of 31.943 and 11.489 %, respectively). These results reveal the opposite pattern to the analysis of the InScope faults reported in Section 3.5.1 (that is, *branch testing* is less effective than *equivalence partitioning* for faults within technique scope). This suggests that *equivalence partitioning* is more sensitive to faults within its scope, but *branch testing* is better at detecting faults outside its scope.

The multiple comparisons for the **session/program** factor, shown in Table 14, suggest that there are no significant differences between the levels of this factor. However, the results of the InScope response variable, which were reported in Section 3.5.1, did suggest that there were significant differences. There is no apparent reason why the session/program factor should behave differently depending on fault types. Therefore, we are unable to venture any hypothesis to explain this discrepancy considering just the replication results.

The multiple comparisons for the **Group** factor, shown in Table 15, suggest that, unlike our findings for the InScope response variable, there are no significant differences between BT-EP and EP-BT.

In Section 3.5.1, we ventured two hypotheses to explain the results for InScope faults: the existence of a carryover effect or, alternatively, the EP-BT group's supe-

rior C and testing technique experience. In either case, we would expect this effect to be the same irrespective of the fault type. Again considering only the replication results, we are unable to venture any reason why effects should be significant for the InScope response variable.

4 Comparison of Replication Results to Original Results

Because our replication uses a subset of factor levels of the original experiment, we will not be able to contrast all the results of the original experiment with the replication, and some will be only partially comparable. The *branch testing* and *equivalence partitioning* levels of the technique factor are comparable in both experiments, but we left out the *stepwise abstraction* technique, which is used only in the original experiment.

The session/program factor is partially comparable, as the programs used in the two experiments (nametbl and ntree) correspond to different sessions in each experiment (sessions 1 and 2 in ESPEL and sessions 3 and 2 in UPM, respectively). We clearly define whether we are referring to the program or session in each case.

In the case of the group factor, it is the technique factor levels and experiment sessions that define the levels of each group. Six different groups were formed in the original experiment and only two in the replication. The two groups formed in the replication are subsets of two of the six groups of the original experiment. Therefore, they are only comparable at a very high level.

Sections 4.1 and 4.2 contain the similarities and differences, respectively, between the original experiment and the replication.

4.1 Consistent Results

The consistent results between the original experiment and the replication refer to the technique factor for both the InScope and OutScope response variables, as shown in Table 16. This table summarizes the results obtained in the multiple comparisons for the original experiment (UPM) and the replication (ESPEL), specifying whether the null hypothesis (there is no difference in the effectiveness of *equivalence partitioning* and *branch testing* with respect to the detection of faults) is rejected or accepted and showing the observed pattern (order relationship) between the factor levels.

4.1.1 InScope Response Variable

Branch testing is less effective than *equivalence partitioning* in both experiments, albeit with some fine distinctions. Firstly, as shown in Table 16, at ESPEL we have obtained significant differences between the technique levels, whereas the difference was not so grand at UPM. However, the patterns are the same (BT < EP) in both cases.

Table 16 UPM-ESPEL comparison—technique factor

Response	H0		Tendency	
Variable	Upm	Espel	Upm	Espel
InScope	Accept	Reject	$BT < EP$	$BT < EP$
OutScope	Reject	Reject	$BT > EP$	$BT > EP$

Secondly, the marginal means for the technique factor at ESPEL are lower than at UPM (*branch testing* with 31.943 vs. 67.670 % and *equivalence partitioning* with 45.140 and 78.704 %, respectively). Finally, *branch testing* technique dispersion is lower at ESPEL than at UPM, as shown in Fig. 6.

Branch testing's lower dispersion at ESPEL may explain why the difference between *branch testing* and *equivalence partitioning* is significant at ESPEL. BT's wider dispersion at UPM may be due to the fact that UPM subjects are undergraduate students, in general without professional experience, and therefore their testing abilities may vary considerably. This would generate wide interquartile ranges. As a consequence, the null hypothesis could not be rejected at UPM, causing the

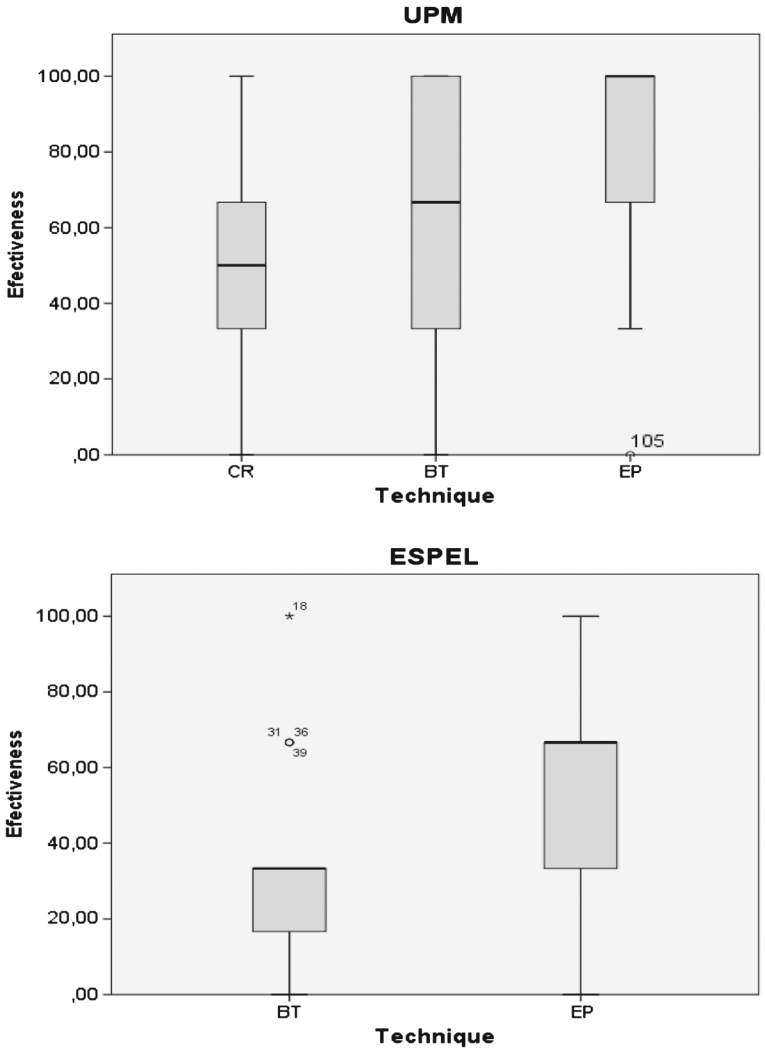

Fig. 6 Boxplot for technique at UPM and ESPEL—InScope

impression that the results at UPM and ESPEL are slightly different, when they really are consistent.

Why the subjects are less effective at ESPEL than at UPM is another question. The most likely reason is that the course on which the experiment was run is intensive (long lecture hours concentrated over just a few days). This may have influenced technique effectiveness, as subjects may not have had enough time to practice and consolidate the usage of the techniques. *Branch testing* would be more adversely affected, since subjects told us at post-experimental meetings that BT is harder to understand and apply than EP. Another potential factor, besides the teaching method, is trainer inexperience in teaching the software verification and validation course, especially under such circumstances.

4.1.2 OutScope Response Variable

The results obtained at UPM for this response variable were confirmed at ESPEL, as the null hypothesis is rejected in both cases, and, besides, the trend is the same as shown in Table 16, where *branch testing* is more effective than *equivalence partitioning* for faults that are outside their scope. The values of the marginal means are slightly lower at ESPEL (27.398 % for *branch testing* and 11.489 % for *equivalence partitioning*) than at UPM (29.089 % for *branch testing* and 14.119 % for *equivalence partitioning*). The dispersions are generally quite similar, albeit, predictably for undergraduate students, slightly wider at UPM, as shown in Fig. 7. It is interesting to note that the low technique effectiveness at ESPEL is much more marked for the InScope than for the OutScope faults, considering that the differences in the training (both course intensiveness and possibly trainer inexperience) should (in principle) affect both response variables more or less equally.

A possible explanation for better *branch testing* performance with OutScope faults is that the test case generation strategy requires an analysis of source code, at which point subjects could informally apply code inspection and thus round out the technique. This is a convincing explanation for two reasons:

- Subjects applying the *equivalence partitioning* technique do not have the source code of the program that they are testing (only the executable), as mentioned in Section 2.6.2. Consequently, their fault detection proficiency should be very low. This is precisely what we found, as the mean effectiveness of experimental subjects is 11.5 %, that is, each subject identifies on average 0.3 faults.
- Without the help of testing techniques, subjects with comparable characteristics (programming experience, years of experience industry, etc.) should locate more or less the same faults. We expect to observe that ESPEL subjects are more effective than UPM subjects because they have some professional experience. We found that the effectiveness of both subject groups (ESPEL and UPM) is more or less equal. Now, this is precisely what we would expect to find if the ESPEL subjects were to be suffering from fatigue as a result of experiment planning, as discussed in Section 4.1.1. Another reasonable hypothesis in the light of the results is that subjects apply informal code reading during the application of the *branch testing* technique.

Even though this is a reasonable hypothesis, it needs to be corroborated in future replications.

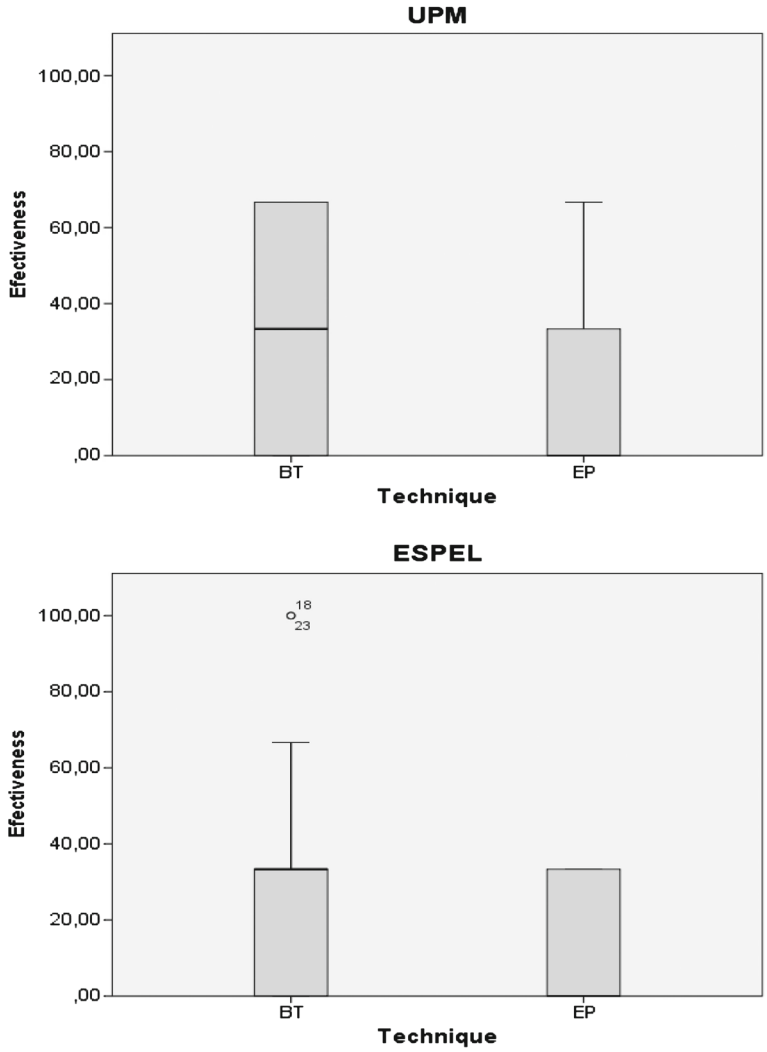

Fig. 7 Boxplot for technique at UPM and ESPEL—OutScope

4.2 Differences in Results

The differences between the results of the original experiment and the replication refer to the session/program and group factors. Both are dealt with separately in the following.

4.2.1 Differences with Respect to the Session/Program Factor

Table 17 summarizes the comparison for the session/program factor.

(A) **InScope Response Variable** With respect to **session/program** factor, testing was more effective at UPM when the ntree program was applied, whereas just

Table 17 UPM-ESPEL comparison—session/program factor

Response	H0		Tendency	
Variable	Upm	Espel	Upm	Espel
InScope	Accept	Reject	$S3 < S2$	$S1 > S2$
			$ntbl < ntree$	$ntbl > ntree$
OutScope	Accept	Accept	$S3 > S2$	$S1 < S2$
			$ntbl > ntree$	$ntbl < ntree$

the opposite was the case at ESPEL, where nametbl was more effective as shown in Fig. 8. Additionally, the differences are significant at ESPEL, whereas at UPM they are not.

We changed the order in which the ntree and nametbl programs were applied in the replication. This change is designed to clarify whether it is the session or program (as the original experimenters claim) that is causing the observed effect.

ESPEL results contradict UPM results, suggesting that causal factor is probably the session and not the program.

Fatigue could be the mechanism through which the session influences effectiveness. Note that the experiment was run on an intensive academic programme and experimental sessions were held only one day apart. Therefore, subjects might well have been fatigued when they arrived at the experimental sessions (notice, in this respect, that the marginal means of effectiveness are always lower at ESPEL than at UPM and that, in particular, S2 was less effective than S1 (as Fig. 8 clearly shows).

The possibility of a fatigue effect is a convincing hypothesis on two grounds. Firstly, the sharp drop in effectiveness from S1 to S2 would explain why the S1/nametbl and S2/ntree differences turned out to be significant. If fatigue had had no effect, the differences would have been smaller and possibly not significant, which is what was found at UPM (where the sessions were held one week apart). Secondly, the effect size for the session/program factor (shown in Table 9) is abnormally high compared with the technique factor (shown in Table 8). The fatigue effect is also compatible with this finding.

In any case, the data gathered in the two experiments are still not enough to determine which factor (session or program) is really having a bearing, so yet more replications will be necessary to clarify this point.

(B) **OutScope Response Variable** Neither experiment observed significant differences with respect to the OutScope variable. The above-mentioned possibility of a fatigue effect is entirely consistent with the fact that S2/ntree is less effective than S1/nametbl, as shown in Fig. 9.

The ESPEL and UPM results have a completely different pattern, giving the visual impression that the two experiments are inconsistent. Note, however, that the S1–S2 and ntree-nametbl differences are not significant in either experiment. The fact that the results are not significant shows that neither the program nor the session (with the possible exception of a fatigue effect) has any effect on OutScope fault detection effectiveness, which makes sense. This strengthens the plausibility of OutScope fault detection depending exclusively on the expertise of the experimental subjects, as specified in Section 4.1.2.

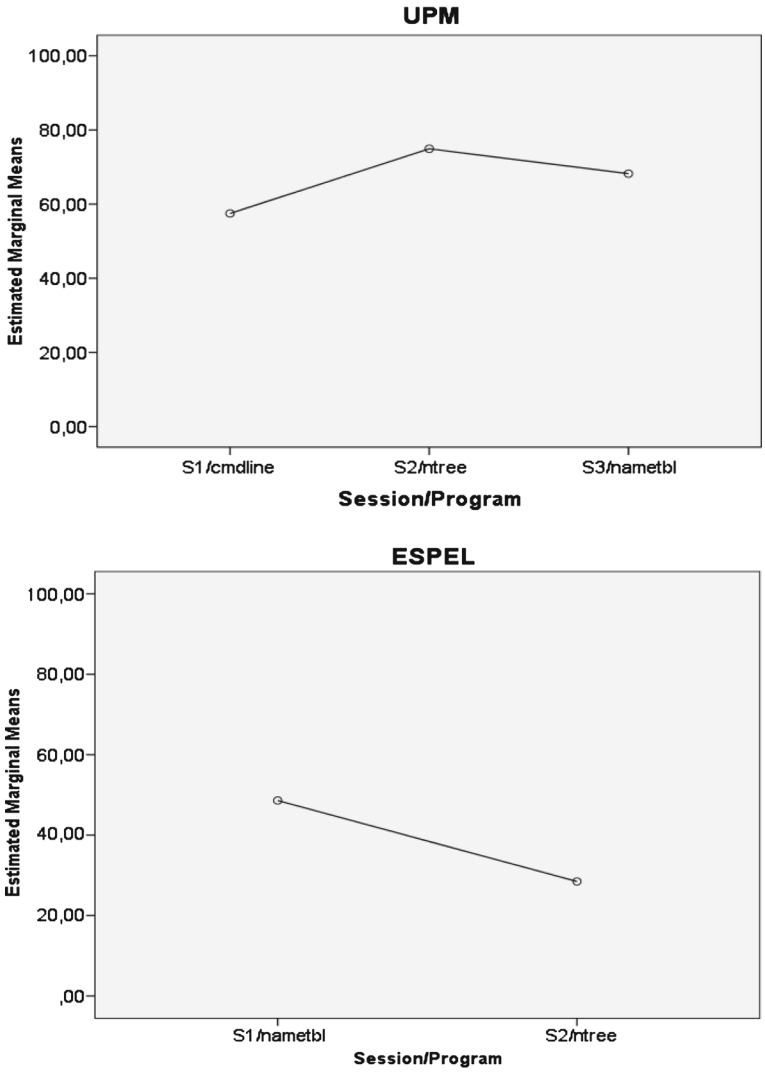

Fig. 8 Estimated marginals means for Session/Program at UPM and ESPEL—InScope

4.2.2 Differences Regarding the Group Factor

The between-group differences at ESPEL and UPM cannot be tabulated and plotted as above, because they are too profound. Taking into account the statistical significance of the results, however, we can compare the two experiments as shown in Table 18.

(A) **InScope Response Variable** The results of the experiments at both UPM and ESPEL are statistically significant for the group factor. The analysis suggests that ESPEL results may be due to either a carryover effect or an imbalance in experience in C programming and testing technique use across experimental

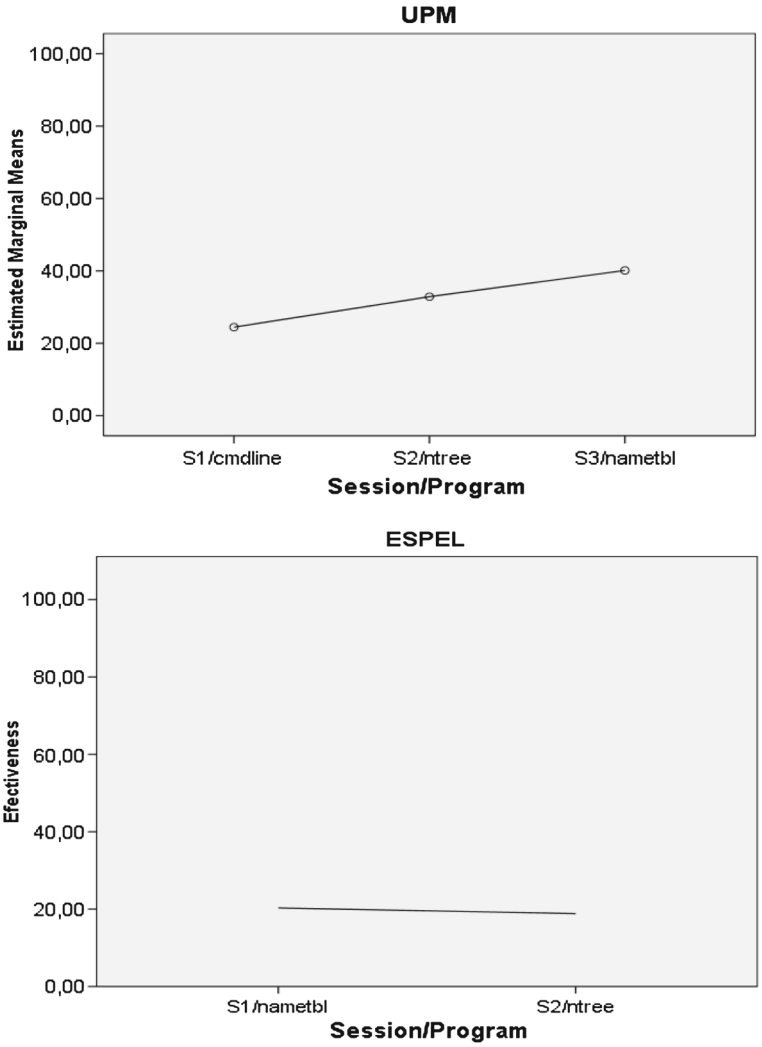

Fig. 9 Estimated marginals means for session/program at UPM and ESPEL—OutScope

groups. At UPM, we do not know whether or not the experimental groups are balanced (the original experimenters do not provide this information). Additionally, the original report (Juristo et al. 2013) indicates that there was no carryover.

Table 18 UPM-ESPEL comparison—group factor

Response	H0	
Variable	Upm	Espel
InScope	Reject	Reject
OutScope	Reject	Accept

We take the view that the between-group imbalance has not had a decisive impact. The imbalance between experimental groups at ESPEL should show up consistently across the InScope and OutScope faults. However, this is not the result that we observed, as the between-group difference for OutScope faults is not significant (and, besides, is different to the pattern for InScope faults).

In our view, the carryover effect cannot be ruled out. This opinion is based on two observations:

- The analysis of the technique and session/program factors at both ESPEL and UPM appears to suggest that OutScope fault detection depends exclusively on the experimental subjects, that is, on their knowledge, experience, etc. If this were the case, we should not observe any carryover effect for OutScope faults in the replication, as *equivalence partitioning* and *branch testing* have no influence on OutScope fault detection. This would not apply to InScope faults, and it would make sense if we were to observe a carryover effect.
- Nonstatistically, using EP first appears to improve the effectiveness of the techniques applied subsequently in both experiments (groups EP-CR-BT and EP-BT-CR in the original experiment and group EP-BT in the replication appear to be more effective).

Small sample effects offer a possible explanation, which does not support the carryover effect, for the significant differences that show up with respect to the group factor. The original experiment had 46 experimental subjects, so the number of subjects per group is $46/6 \simeq 8$. At ESPEL, the number of subjects per group is $23/2 \simeq 12$. With so few subjects per group, the significant effects may be a product of chance.

We take the view that the small sample effects can add noise to the data analysis (that is, cause some groups to exhibit significant differences from others purely by chance), but this does not fully explain the results. Note that the small sample effect should act consistently across InScope and OutScope faults, which is not the case. At UPM, three groups exhibit significant differences with respect to InScope faults, whereas only one of the groups has significant differences with respect to OutScope faults (which could quite possibly be a small sample effect). At ESPEL, the differences are significant for InScope but not for OutScope faults. Consequently, there may well be a carryover effect with respect to InScope faults.

Unfortunately, the carryover hypothesis is rather speculative. The groups at ESPEL and UPM are not directly compatible; hence all inferences are based on indirect evidence. Consequently, more replications need to be conducted to confirm or reject the existence of a carryover effect.

(B) **OutScope Response Variable** We have been obliged to explain all our findings with respect to the OutScope variable in the analysis of the InScope response variable. On this ground, we will merely state our conclusions at this point:

- We ascribe the existence of significant differences with respect to the group factor for OutScope faults to a small sample effect.
- It appears from the analysis of the technique and session/program factors that subjects draw on their own expertise to detect OutScope faults. On

this ground, there should be no carryover effect (which, basically, is the end result of a relationship between testing techniques and cannot, therefore, exist unless they have a bearing on the OutScope response variable).

5 Conclusions and Lessons Learned Across Studies

5.1 Conclusions

Comparing non-identical replications is a complex issue. The changes caused by eliminating one of the technique factor levels on logistic grounds (insufficient time to run all the experimental sessions) have had a waterfall effect on the program and session factors. Consequently, neither the sessions nor the groups are comparable in every respect. Even so, the replication has helped to get a better understanding of the influence of the factors under study:

- Firstly, we have confirmed that the *equivalence partitioning* technique is more effective at detecting faults that are within its scope and *branch testing* is more effective for faults outside its scope. A possible reason for this difference of effectiveness in the case of the InScope variable is that subjects find the *branch testing* technique harder to use or, at least, are better at applying the *equivalence partitioning* technique. Regarding the difference in effectiveness for the OutScope response variable, we hypothesize that the structural technique is more effective because students use code review to round out the technique. As they have access to the source code (not so for the functional technique), they can inspect the code to gain a better understanding of the program. The statistical results show that the measures of effectiveness were generally lower at ESPEL than at UPM. This could be attributed to the influence of the setting. To be precise, we believe that the extremely intensive teaching method applied in training could have had an influence. Apart from the teaching method, another consideration is trainer inexperience in teaching the software verification and validation course, especially under the circumstances.
- Secondly, the results for the session/program and group factors with respect to the OutScope response variable support the hypothesis that neither the EP nor the BT technique really influence OutScope fault detection. In both cases (session/program and group), the results were not significant in either the original experiment or the replication.
- Thirdly, the existence of some sort of carryover effect for the InScope variable appears to be confirmed. In this case, the problem is that the groups are formed differently in the two experiments and are hence not comparable. Consequently, further replications need to be conducted before we can state that this effect really does exist.

The hardest thing to figure out was the influence of the session/program factor with respect to the InScope variable. The original experimenters had concluded that the program was responsible for the session/program effect on effectiveness at UPM (remember that the two factors are confounded). The original experimenters arrived at this conclusion after comparing the InScope and OutScope fault detection effectiveness for session/program. This comparison suggested that the differences

between cmdline, nametbl and ntree together offered a better explanation than the differences across sessions S1, S2 and S3 for the resulting data.

However, the results at ESPEL for InScope have a completely opposite pattern to UPM findings. Not only do the ESPEL and UPM patterns differ with respect to the programs (ntree>nametbl at UPM, whereas nametbl>ntree at ESPEL), but the differences at ESPEL are also statistically significant.

One of the changes made to the replication with respect to the original experiment was to reduce the number of sessions from three to two. This led to one of the programs used in the original experiment (cmdline) being omitted. Under these circumstances, it is hard to reach any sort of reliable conclusion. Our analysis tends to ascribe the observed effects to the session rather than to the program. However, there are other alternative explanations. The existence of some sort technique/program interaction is a particularly convincing cause.

Figure 10 shows boxplots for the technique factor. In contrast to Fig. 6, they have been further decomposed by session/program. Figure 10 is not easy to interpret because there are several outliers, but the median for EP x nametbl is clearly much greater than for the other combinations (BT x nametbl, etc.). The profile diagram shown in Fig. 11 illustrates these values more clearly (Note that this diagram plots means not medians). Neither the original experiment nor the replication (which reuses the original analytical model for the sake of comparability) account for this interaction, as a repeated-measures ANOVA cannot calculate this effect.

Irrespective of the analytical model used, the existence of such an interaction would be compatible with the results for technique and program at ESPEL and would explain why a carryover effect was observed. (Note that the technique/program interaction, that is, where a technique may be more effective when applied to certain programs, is confounded with the group factor, and thus the statistical analysis is unable to distinguish the two effect types.) A possible technique/program effect does not explain all the findings, however. In particular, there is the question of

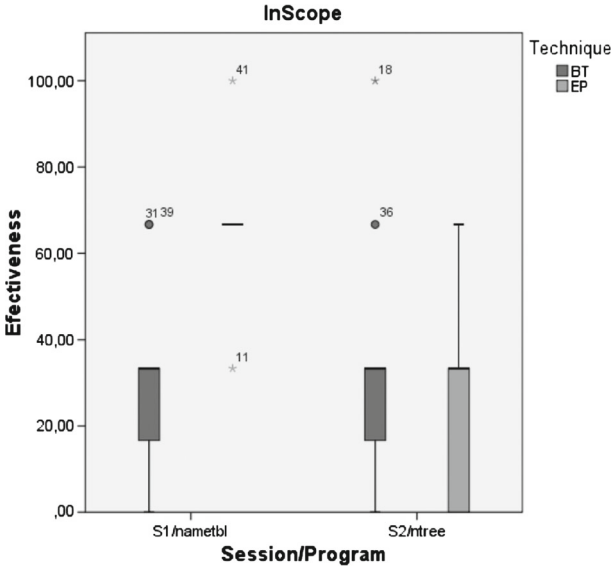

Fig. 10 Boxplot for the interaction technique × session/program at ESPEL—InScope

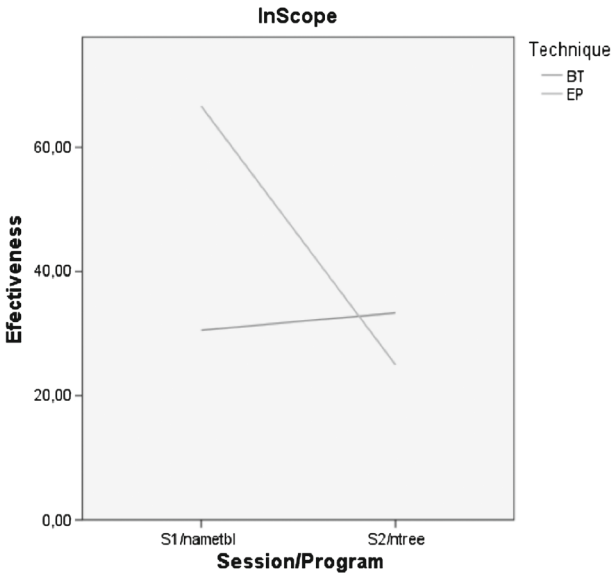

Fig. 11 Profile diagram for the technique × session/program interaction at ESPEL—InScope

why the EP-BT-CR and EP-CR-BT groups are so effective in the original experiment when the program tested in the first session was cmdline and not nametbl, and all the subjects participating in the experiments regarded cmdline as a program that is hard to understand and test. In actual fact, all the above explanations are tentative. As already mentioned, we will not be able to clearly understand the effects of the technique and session/program factors unless further replications are conducted.

5.2 Lessons Learned

The replication that we have conducted is one of a long line of experiments. This means that information and the experience gathered from multiple replications conducted as part of this family is reasonably thorough, and, as we also had access to the original experiment report, the replication could have been carried out without any additional information.

With hindsight, however, we believe that we would have had very little prospect of success if we had proceeded in this manner (as already mentioned, we had intensive communication with the original experimenters). The likenesses between the two experiments are noteworthy, but the differences are no less marked. We have been able to trace these differences back to characteristics of the experimental setting (e.g., software verification and validation course intensiveness), but this was possible only because relatively few changes were made to the replication. For example, we might have ascribed the low effectiveness of the ESPEL subjects using the *branch testing* technique to the fact that they were inexperienced at programming in C. But, UPM subjects are generally not very experienced C programmers either and they are more effective. Therefore, the low effectiveness is more likely to be due to the training received, which does vary, and very much so, from ESPEL to UPM.

With no more than the original experiment report and the experimental materials, such a close coincidence appears to be very hard to achieve. The experimental material did not describe training issues, such as mentioned above; nor did it detail experiment execution or results measurement, for example. Had we not cooperated with the original experimenters, especially during the early stages, the differences between the original experiment and the replication would probably have been much larger. These differences (such as, for example, the above changes regarding training) are also likely to have caused discrepancies in the results. A post-experimental discussion with the original experimenters could show up design differences, but would not be able to prevent any discrepancies caused by such changes. However, these points were discussed at the pre-experimental meetings that we had with the original experimenters.

As a corollary to the above, the experience of having replicated an experiment previously conducted by other experimenters and, especially, the attempt at comparing the results of the two experiments, has shown that unless replications closely resemble the original experiment it is impossible to ascribe (at least hypothetically) consistent and inconsistent results to particular factors and parameters. For the reasons discussed above, if the experimental settings are not alike, the differences between the results can be attributed to virtually any aspect, making the replication much less enlightening than it would otherwise be.

Obviously, merely replicating an experiment with a similar design does not guarantee that the results can be definitely ascribed to a factor. On the one hand, any experiment is subject to some error probability (α and β). On the other hand, we do not know which aspects of a setting (that is, uncontrolled variables) are likely to alter the effects of the factors. Consequently, even very similar replications can return contradictory results.

We can reduce α and β error fairly simply by increasing the number of experimental subjects. As the number of subjects increases, experimenters can gradually lower the α and β values. In practice, however, the availability of experimental subject is limited (for example, Sjøberg et al. 2005, report that the mean number of experimental subjects per SE experiment is 48.6). In other words, we can at best reckon with enough subjects to assure normality and elude small sample effects (from 30 to 50 subjects) (Richy et al. 2004; Graham and Schafer 1999).

The second problem cannot be solved by running a single replication. The unknown variables are, as their name indicates, inscrutable, and therefore their effect cannot be estimated. Nevertheless, randomization is usually considered effective (that is, cancels out the effects of uncontrolled variables) as of 30 subjects, which is when a sample of a standard population is assumed to meet the normality assumption.

The replication that we have conducted has both of the above characteristics. It has 23 experimental subjects, which is lower than the average 48.6 subjects per SE experiment reported by Sjøberg et al. (2005). However, as a result of the cross-over design, those 23 subjects are equivalent to $23 \times 2 = 46$ experimental units. In the case of repeated-measures designs, it is the number of experimental units, not subjects, that influences the α and β values. Sjøberg et al. (2005) do not report the number of experimental units per SE experiment. However, the number is unlikely to be much greater than 48.6, because within-subjects is not the most

common experimental design in SE. Therefore, this replication can be considered an "average" SE experiment.

Even so, we have observed inconsistencies with the original experiment despite having kept all parameters and factors reasonably well under control. We have hypothesized that such inconsistencies can be put down to certain causes (as in the case of the differences in the results with respect to session/program), but this has only been possible because the experiments are quite alike. If there were more differences between the experiments, any such, even hypothetical, attribution would be out of the question.

These may not, of course, be the real causes of the inconsistencies. Literal replications (that is, replications that closely resemble the original experiment) are just a starting point. We will in any case have to run differentiated replications in order to more generally explore all the factors that potentially have an effect.

Finally, we have found that the preparation of the replication accounts for a much larger workload than the experimental sessions. Gaining a detailed understanding of the original experiment, plus the initial training and processing of the forms submitted by subjects, proved to be much more time-consuming than the experimental sessions. In fact, the session workload was comparatively insignificant.

Briefly the lessons learned were:

- Support from the original experimenters is important, during the early replication preparation phases at least, in order to supplement the information available in reports and materials. It is vital to have as much information as possible to ensure that the original experiment and replication are comparable.
- The replication must be as like the original experiment as possible in order to be able to ascribe the detected differences (or similarities) to specific variables.
- The highest experiment workload is spent not on the experimental sessions per se but on the preparation of the experiment and analysis of the information gathered from subjects.

5.3 Experiences with Reporting Guidelines

Apart from reporting the replication, we have, as part of this research, tested the guidelines for reporting replications proposed by Carver (2010). Generally, the guidelines have proved to be really useful, especially as regards the description of the replication in terms of its differences to the original experiment. However, there are some points that we found not to be fully satisfactory and think should be improved:

- The granularity with which the original experiment should be described is unclear. On the one hand, the original experiment can be assumed to have been published, meaning that a brief description would do the job (interested readers could always refer to the original publication). But, on the other hand, the description should be detailed enough for readers to be able to understand the impact of the changes made to the replication. The guidelines should be clearer in this respect.
- Again regarding the reporting of the original experiment, our impression is that the section contents are very unbalanced. The research questions section has very little content, whereas the experimental design section is packed out. Additionally, it is unclear where the hypotheses should be defined. We have moved some

elements (hypotheses, factors and response variables) to the research questions section, but we consider this procedure to be unsatisfactory.
- In the case of literal replications, it is unclear which parts of the experiment or replication to report. In the replication that we have run, for example, we use only two of the three main factor levels of the original experiment. Should we report the results concerning the third level of the original experiment? On the one hand, it appears that we should, otherwise the report would be incomplete. But, on the other hand, the third level is of no use for comparing the original experiment and the replication. Also, readers could always refer to the original experiment report to look up any information about this third factor.
- There is no separate section for discussing the results of the replication. The replication results should be discussed separately before identifying the similarities and differences between experiments. We have added a separate section, but we think that the guidelines should take this point into account.

5.4 Future Work

The replication that we have run has essentially confirmed the effects observed in the original experiment. However, there are some effects, such as differences of effectiveness associated with sessions/programs or carryover effects, which we have still not been able to positively ascribe to specific variables. Our short-term goal is to continue replicating the UPM experiment, altering the setting as little as possible in order to determine beyond all doubt which variables produce which effects. Once we have a good understanding of how the (*equivalence partitioning* and *branch testing*) testing techniques behave, we will be able to run differentiated replications that explore different settings or populations (e.g., experienced professionals or industrial settings).

Acknowledgements This research has been funded by a grant from the Armed Forces Technical School (ESPE), Republic of Ecuador National Higher Education, Science, Technology and Innovation Secretary's Office (SENESCYT) and partially funded by the Spanish Ministry of Economics and Competitiveness project TIN2011-23216.

We also thank the reviewers for their thoughtful review, which greatly improved the quality of the manuscript.

Appendix

Appendix A: Descriptive Statistics

Table 19 Descriptive statistics InScope variable

Technique	N	Mean	Std. dev.
Branch testing	23	31.883	25.581
Equivalence partitioning	23	44.204	29.988

Table 20 Descriptive statistics OutScope variable

Technique	N	Mean	Std. dev.
Branch testing	23	27.536	32.803
Equivalence partitioning	23	11.593	16.231

Appendix B: Survey's Data

Table 21 Survey's data

Subject	Group	Q1	Q2	Q3
S1	G2	3	3	2
S2	G1	NA	NA	NA
S3	G1	5	4	4
S4	G2	5	3	4
S5	G2	4	2	2
S6	G2	5	2	2
S7	G1	4	2	2
S8	G1	5	4	3
S9	G1	5	5	4
S10	G2	5	4	2
S11	G2	4	4	1
S12	G1	4	4	4
S13	G2	NA	NA	NA
S14	G1	4	5	4
S15	G1	3	4	1
S16	G1	4	3	3
S17	G1	4	4	1
S18	G2	4	3	3
S19	G2	5	5	4
S20	G2	4	5	2
S21	G1	5	3	2
S22	G1	5	4	4
S23	G2	4	4	2

Q1: Question 1
Q2: Question 2
Q3: Question 3
Value 1: None
Value 2: I know the theory
Value 3: I have done short exercises
Value 4: Practical assignments
Value 5: Development projects

Appendix C: Replication's Raw Data

Table 22 Replication's raw data

S	G	T	P	Se	Vis. for EP			Vis. for BT			In	Out
					F1	F2	F3	F4	F5	F6	(%)	(%)
1	2	EP	Na	1	1		1				66.7	0.0
1	2	BT	Nt	2					1		33.3	0.0
2	1	BT	Na	1		1					0.0	33.3
2	1	EP	Nt	2	1						33.3	0.0
3	1	BT	Na	1							0.0	0.0
3	1	EP	Nt	2	1						33.3	0.0
4	2	EP	Na	1			1				33.3	0.0
4	2	BT	Nt	2							0.0	0.0
5	2	EP	Na	1	1		1				66.7	0.0
5	2	BT	Nt	2							0.0	0.0
6	2	EP	Na	1			1		1		33.3	33.3
6	2	BT	Nt	2							0.0	0.0
7	1	BT	Na	1							0.0	0.0
7	1	EP	Nt	2	1	1			1		66.7	33.3
8	1	BT	Na	1							0.0	0.0
8	1	EP	Nt	2							0.0	0.0
9	2	EP	Na	1		1	1				66.7	0.0

Table 22 (continued)

S	G	T	P	Se	Vis. for EP			Vis. for BT			In	Out
					F1	F2	F3	F4	F5	F6	(%)	(%)
9	2	BT	Nt	2					1	1	66.7	0.0
10	1	BT	Na	1							0.0	0.0
10	1	EP	Nt	2							0.0	0.0
11	2	EP	Na	1		1	1				66.7	0.0
11	2	BT	Nt	2						1	33.3	0.0
12	1	BT	Na	1	1	1	1				0.0	100.0
12	1	EP	Nt	2	1					1	33.3	33.3
13	2	EP	Na	1							0.0	0.0
13	2	BT	Nt	2		1	1		1		33.3	66.7
14	1	BT	Na	1							0.0	0.0
14	1	EP	Nt	2							0.0	0.0
15	1	BT	Na	1		1					0.0	33.3
15	1	EP	Nt	2						1	0.0	33.3
16	1	BT	Na	1		1		1			33.3	33.3
16	1	EP	Nt	2			1	1			33.3	33.3
17	1	BT	Na	1		1					0.0	33.3
17	1	EP	Nt	2							0.0	0.0
18	2	EP	Na	1							0.0	0.0
18	2	BT	Nt	2				1		1	66.7	0.0
19	2	EP	Na	1	1	1					66.7	0.0
19	2	BT	Nt	2							0.0	0.0
20	1	BT	Na	1							0.0	0.0
20	1	EP	Nt	2					1		0.0	33.3
21	2	EP	Na	1	1	1	1		1		100.0	33.3
21	2	BT	Nt	2	1	1			1		33.3	66.7
22	1	BT	Na	1							0.0	0.0
22	1	EP	Nt	2							0.0	0.0
23	2	EP	Na	1	1	1					66.7	0.0
23	2	BT	Nt	2					1		33.3	0.0

S: Subject, *G*: Group
T: Technique
(*EP*: Equivalence Partitioning, *BT*: Branch Testing)
P: Program (*Na*: Nametbl; *Nt*: Ntree)
Se: Session, *F1–F6*: Faults, *Vis.*: Visible
In: InScope, *Out*: OutScope (Response Variables)

References

Basili VR (1992) Software modeling and measurement: the goal/question/metric paradigm. Tech. Rep. UMIACS TR-92-96, Departament of Computer Science, University of Maryland, College Park

Basili VR, Selby RW (1985) Comparing the effectiveness of software testing strategies. Tech. Rep. TR-1501, Departament of Computer Science, University of Maryland, College Park

Basili VR, Selby RW (1987) Comparing the effectiveness of software testing strategies. IEEE Trans Softw Eng SE-13:78–96

Brown BW (1980) The crossover experiment for clinical trials. Biometrics 36:69–70

Carver JC (2010) Towards reporting guidelines for experimental replications: a proposal. In: Proceedings of the 1st international workshop on Replication in Empirical Software Engineering Research (RESER). Cape Town, South Africa, 4 May 2010

Gómez O (2012) Tipología de Replicaciones para la Síntesis de Experimentos en Ingeniería del Software. PhD thesis, Universidad Politécnica de Madrid

Gómez O, Juristo N, Vegas S (2010) Replications types in experimental disciplines. In: Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement, no. 3. Bolzano-Bozen, Italy, pp 1–10
Graham JW, Schafer JL (1999) On the performance of multiple imputation for multivariate data with small sample size. In: Hoyle RH (ed) Statistical strategies for small sample research. Sage Publications, pp 1–29
Juristo N, Vegas S (2003) Functional testing, structural testing and code reading: what fault do they each detect? In: Empirical Methods and Studies in Software Engineering Experiences from ESERNET, vol 2785(12), pp 208–232
Juristo N, Vegas S, Apa C (2013) Effectiveness for detecting faults within and outside the scope of testing techniques: a controlled experiment. Available at http://www.grise.upm.es/reports.php. Accessed 15 May 2013
Kamsties E, Lott C (1995) An empirical evaluation of three defect-detection techniques. In: Fifth European Software Engineering Conference (ESEC '95). Lecture Notes in Computer Science, vol 989, pp 362–383
Kernan WN, Viscoli CM, Makuch RW, Brass LM, Horwitz RI (1999) Stratified randomization for clinical trials. J Clin Epidemiol 52(1):19–26
Kitchenham B, Fry J, Linkman S (2003) The case against cross-over designs in software engineering. In: Eleventh annual international workshop on software technology and engineering practice, pp 65–67
Meyers LS, Gamst G, Guarino AJ (2006) Applied multivariate reseach: design and interpretation. Sage Publication
Myers GJ (1978) A controlled experiment in program testing and code walkthroughs/inspections. In: Communications of the ACM, vol 21, pp 760–768
Richy F, Ethgen O, Bruyère O, Deceulaer F, Reginster J (2004) From sample size to effect-size: Small study effect investigation (ssei). In: The Internet Journal of Epidemiology, vol 1
Roper M, Wood M, Miller J (1997) An empirical evaluation of defect detection techniques. Inform Softw Technol 39(11):763–775
Senn S (2002) Cross-over trials in clinical research, 2nd edn. Wiley
Sjøberg DI, Han Hannay JE, Hansen O, Kampenes VB, Karahasanovic A, Liborg N-K, Rekdal AC (2005) A survey of controlled experiments in software engineering. IEEE Trans Softw Eng 31:733–753
Wood M, Roper M, Brooks A, Miller J (1997) Comparing and combining software defect detection techniques: a replicated empirical study. In: Proceedings of the 6th European software engineering conference held jointly with the 5th ACM SIGSOFT international symposium on foundations of software engineering. Zurich, Switzerland, pp 262–277

Cecilia Apa is an Assistant Professor at the Engineering School at the Universidad de la República (UdelaR), coordinator of the Informatics Professional Postgraduate Center (UdelaR) and member of the Organization Committee of the Software and Systems Process Improvement Network in Uruguay (SPIN Uruguay). She received his B.Eng. in Computer Science from the UdelaR. She has several articles published in regional and international conferences. Her main research topics are empirical software engineering and software testing.

Oscar Dieste is research scientist with the Universidad Politécnica de Madrid. Previously, he has been Fulbright Scholar with the University of Colorado at Colorado Springs and assistant professor with the universities Complutense de Madrid and Alfonso X el Sabio. His research interests include empirical software engineering, requirements engineering and their intersections. He received his B.S. in Computing from the University of La Coruña and his Ph.D. from the University of Castilla-La Mancha.

Edison G. Espinosa G. PhD student and Master of software engineering at Universidad Politécnica de Madrid (UPM), Spain. He is full professor of software engineering in Escuela Politécnica del Ejército Sede Latacunga, Ecuador.

Efraín R. Fonseca C. received the MSc degree in 2010. He has ten years of IT industry experience as consultant. He is full professor at Universidad Politécnica del Ejército of Ecuador and now is PhD student at Universidad Politécnica de Madrid. Among his research interests are research process in empirical software engineering, research methods in empirical software engineering, object-oriented analysis and design and ontological representations in software engineering.