Towards a Framework to Compare Formal Experiments that Evaluate Testing Techniques

Diego Vallespir[1], Silvana Moreno[1],
Carmen Bogado[1], and Juliana Herbert[2]

[1] Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay.
[2] Herbert Consulting
Porto Alegre, RS, Brazil.
dvallesp@fing.edu.uy , {silvanamoren, cmbogado}@gmail.com,
juliana@herbertconsulting.com

Abstract. There are many formal experiments to evaluate the performance of different software testing techniques. The first we know about is from 1978 [1].The most recent one is currently under execution and some initial results have already been obtained [2]. Having a comparison framework of experiments is necessary in order to be able to formally compare them and make progress on the construction of new experiments based on previous ones. This paper presents a comparison framework and four known formal experiments are compared.

1 Introduction

It is normal to use a hammer to hammer a nail into a wall. There are many types of hammers but it is easy to choose one and even more, a lot of hammers do the same job. It is normal to use a software testing technique to verify a software unit. Unfortunately, it is not known which one to choose nor if different techniques perform the same way for the same task.

The performance of each technique (cost, effectiveness and efficiency) has to be known at the time of choosing a testing technique. But to obtain this kind of knowledge is not easy given the variability of their performance, which depends on the subject that applies it, the programming language and the application type that is being tested (information system, robotics, etc). Some advances have been made but there is still a long way to go.

There are many formal experiments to study the performance of different software testing techniques. The first we know about is from 1978 [1]. The most recent one is currently under execution and some initial results have already been obtained [2].

Many years of empiric investigation in the subject have gone by though there are not definite results yet. In *A look at 25 years of data*, the authors have reached the same conclusion after studying various experiments on software testing [3].

Also, they found that it is really difficult to compare different experiments, however, they do not present any solution to it.

Having a comparison framework of experiments is necessary in order to be able to formally compare them and make progress on the construction of new experiments based on previous ones. This paper presents a comparison framework and takes four known formal experiments as example: [4], [5], [6], [7].

2 Comparison Framework

In this section we present what we consider the most relevant characteristics to make a comparison of formal experiments to evaluate testing techniques. First of all, it is interesting to compare the goals to know in which aspects and up to which level the experiments can be compared. In second place, it is possible to identify the chosen factors for the experiment and the alternatives for each one. These are also compared together with the identification of the set of parameters in each experiment.

As subjects are a basic component in Software Engineering experiments, it is interesting to compare their main characteristics such us experience, abilities and motivation.

Given the experiments to be compared refer to the application of testing techniques, it is relevant to compare particular aspects of this kind of experiments, for example: the defect classification used, the size, and language of each program together with their number of defects. Making a comparison focused on the chosen design for each experiment is of main importance. This includes the time the experiment takes, the distribution of the subjects, the guidelines followed for the assignment of the set of techniques and programs, the division of the experiments into sessions, the number of unitary experiments in which each subject participates, etc. A comparison of the way in which each design is applied, thus, the process followed in each experiment, is also made. The response variables chosen by the authors of the different experiments are identified. Finally, the similarities and differences between the conclusions are studied.

3 Articles Comparison.

In this section we make a comparative analysis using the framework presented in the previous section along with some known experiments conducted by, Basili and Selby (B-S) [4], Kamsties and Lott (K-L) [5], Macdonald and Miller (M-M) [6], and Juristo and Vega (J-V) [7].

Every experiment have a goal in common, which is to evaluate both the efficiency and effectiveness of verification techniques meant for defect detection. In order to conduct that evaluation the subjects must execute a series of testing techniques on programs or fractions of code.

B-S carried out an experiment in 1987, later, in 1995 K-L carried out another basing on the preceding from B-S. However, it is far from being a replication as they differ on the language used as well as on the verification process as they incorporated defect detection once the failures were found. In 1997 M-M carried out a new experiment, though it varies from B-S's and K-L's ones. Finally, in 2001, J-V conducted a first experiment basing on B-S's and K-L's ones, and a second experiment basing on the same experiments as before and the experienced gained due to their first experiment.

The factors considered in each experiment are listed below in Table 1.

Table 1. Factors

Factor	B-S	K-L	M-M	J-V(1)	J-V(2)
Technique	$\surd$	$\surd$		$\surd$	$\surd$
Program	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Subjects experience	$\surd$				
Application order		$\surd$			
Defect type				$\surd$	$\surd$
Inspection Method			$\surd$		
Program version					$\surd$

In the studies B-S, K-L and J-V conducted, the same testing techniques are used as alternatives to the technique factor, these are: code reading, functional testing and structural testing. The authors M-M apply the code reading technique with two different approaches: inspections based on paper and inspections based on software tools. Not only they have in common the technique, but also they have in common the program as a factor. Each experiment has also factors that differ from the ones in the other experiments: B-S consider the experience of the subjects, K-L the order in which the techniques are applied, M-M the inspection method, J-V consider the defect type in both experiments, and in the last one add the program version.

The parameters considered in each experiment are shown in Table 2.

Table 2. Parameters

Parameter	B-S	K-L	M-M	J-V(1)	J-V(2)
Language	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Program size	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Defects	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Subjects		$\surd$	$\surd$	$\surd$	$\surd$

Every experiment sets its parameters, such us the program size, the language in which these are programmed and their defects. Subjects are considered as parameter by K-L, J-V y M-M, while for B-S they are a factor, given the different levels of experience in their design.

The characteristics of the programs used in the experiments are presented in Table 3.

Table 3. Characteristics of the programs

Characteristics	B-S	K-L	M-M	J-V(1)	J-V(2)
Quantity of programs	4	3	2	4	3
Program size(LOCS)	169, 145, 147 y 365	set of functions from 10 to 30	147 and 143	200	200
Quantity of defects	34 in whole	not specified	12 for each program	9 for each program	7 for each program
Language	Fortran and Simpl-T	C	C++	C	C

All the programs to be tested are considered small in every experiment (contain less than 500 LOCS); the development language used by K-L and J-V is C, by M-M is C++ and the one used by B-S is Fortran and Simpl-T. The number of defects in J-V's first experiment is 9, and in the second one is 7, while in M-M's is 12. In those experiments the number of defects is the same for every program. The four programs used by B-S differ on their quantity of defects, a total of 34. In the case of K-L, the number of defects is not specified.

The main characteristics of the subjects are presented in Table 4.

Table 4. Characteristics of the subjects

Characteristics	B-S	K-L	M-M	J-V(1)	J-V(2)
Quantity	74	50	43	196	46
Experience	8 advanced, 24 intermediate, 42 junior	only one level considered	only one level considered	only one level considered	only one level considered
Experience level	Students from the University of Maryland, Programming professionals from NASA and Sciences Corporation	3rd and 4th grade students	3rd grade students	5th grade students	5th grade students

The studies level as well as the experience of the subjects vary greatly from experiment to experiment. In B-S's experiment, subjects with different levels of experience (advanced, intermediate and junior) are chosen in order to be representative of different levels of knowledge of reality. While in K-L's the subjects are students in third and fourth year from the University of Kaiserslautern. They have some experience in C, but even so, they have a previous training in the usage of techniques and language. This previous training is also put into practice in both J-V's experiments, who choose inexperienced students in the fifth year from the Computer Sciences School, Polytechnic University of Madrid. The subjects involved in M-M's experiment are in the third year of the career, have a solid basis on Scheme, C++ and Eiffel programming, are highly motivated by the experience given it is associated with a major class project and therefore they are graded on their work.

Some design decisions related to each experiment are shown in Table 5.

Both B-S and K-L used the same classification scheme for defects, J-V a subclassification from that scheme while M-M did not classified any defect found. The scheme used is that proposed in Basili's article, in which defects are classified as omission and commission at a first level. The commission defects are those which appear as a result of an incorrect segment of existing code. Omission defects are those which result when the programmer omits including an entity. Besides, the classification system is divided to distinguish the defect types: initialization, calculation, control, interface, data and cosmetics. According to the designs, all the authors decide to make a training prior to the experiment in order to present the subjects with the techniques to be used. The B-S's experiment is divided in 5 sessions. The first one is training, the following three consist of the experiment itself and a follow-up session. K-L's and M-M's experiments consist of two experiment sessions after a training phase. J-V organize the experiment in five sessions: the training session and four execution ones. All through B-S's experiment four programs are used, but in each session three out of the four are tested, thus there is a combination of programs never tested. In the case of K-L's, M-M's and J-V's experiments all the programs are tested in every session. In K-L's are three programs, in M-M's two, and four and three in J-V's experiments, respectively.

Table 6 presents the characteristics of the process followed in each experiment.

The total number of subjects in B-S's experiment is 72, divided in: 8 advanced, 24 intermediate and 42 junior. At the same time they are organized as follows: 29 in the first session, 13 in the second and 32 in the third one. In the first two sessions only subjects with intermediate and junior levels participate, while in the third session advanced subjects are also involved. In K-L's fewer subjects are involved: 27 in the first session and 23 different subjects in the second one. The number of subjects in M-M's experiment is close to K-L's: 43. In this case they are organized in two sessions: 22 and 21 subjects respectively. At the same time they are divided in working groups: six groups of three subjects and one of four in the first session, and seven groups of three subjects in the second one. The total number of subjects in J-V's first experiment rises to 196, organized in 8 groups of 12 subjects (four groups apply structural techniques and the other four apply functional techniques), and four groups of 25 subjects

Table 5. Design decisions

Exp	Techniques	Defect classification	Response variables
B-S	Functional testing, Structural testing and Code reading	Defined by Basili	Number and percentage of defects detected, total time of detection and defect detection rate. For functional testing and Structural testing: number of executions, CPU time consumed, maximum coverage of sentences obtained, time of connection used, number and percentage of defects observable from the data, and percentage of defects observable from the data that are actually found by the subjects.
K-L	Functional testing, Structural testing and Code reading	Defined by Basili.	Each time motivation. Language abilities. Working time. Abilities applying defect detection techniques. Time spent on each step. Number of failures revealed. Number of failures observed. Total number of defects detected. Number of defects detected by chance. Number of defects detected by applying the techniques.
M-M	Code Reading	Does not classify	For each subject and group the number of defects correctly detected and the number of false positives are registered. The gains and losses on inspections made in groups. Frequency of detection for each defect, both on the inspections based on paper and on the tool based ones.
J-V(1)	Functional testing, Structural testing and Code reading	Subclassified by Basili (types considered: initialization, control and cosmetics)	For each defect number of subjects that generate a test case able to detect the defect.
J-V(2)	Functional testing, Structural testing and Code reading	Idem experiment 1	Number of subjects that generate a test case able to detect the defect.

(apply code reading). In the second experiment the subjects are 46, divided in six groups from 7 to 8 subjects.

In the process followed by Basili, three out of the four programs are used in each phase, and every subject uses the three techniques and tests the three programs. In each phase every subject tests the same program on the same day. The K-L's experiment consists of two internal replications, for which three days of a week are settled. Each day a different program is tested by applying the three techniques. The first day participate 23 students, the second 19 and the third day 15. The inspection process of M-M's experiment takes two sessions. The first session is of individual detection and the second is when the consolidation is achieved after working in groups. The experiment is conducted for a period of 10 weeks. During the first six weeks the students are trained. During the remaining four the experiment is performed. The inspection consists of inspecting the source code using a check list, considering the specification of the program.

In M-M's every subject applies both inspections and works on the two programs. In J-V's first experiment a session is carried out each day using one program, and each of the three groups execute a different technique from the three possible ones. The intention of J-V's design is to eliminate the validity threat of the learning, while for K-L the intention is to minimize this threat by assigning the subjects each technique and program only once. In the second experiment of J-V the design is adapted so all the subjects apply every technique. It is organized in three days, and each day only one program and all the techniques are executed. Each of these is executed by two different groups.

The general conclusions reached in each experiment are presented in Table 7.

The complementary conclusions reached in each experiment are presented in Table 8.

Both B-S and K-L reached the conclusion that code reading technique is as effective as structural and functional testings when it comes to the number of defects detected. They also concluded that subjects were more efficient when applying functional testing. M-M concluded that there is no big difference between inspections based on paper and those based on the tool, either it is individual or working in group. With regard to J-V's first experiment, they concluded that the number of subjects that detect a defect depends not only on the program used but also on the technique applied and the defect itself. In addition to this, code reading technique is less sensitive to defect hiding than the other techniques. Functional testing behaves better than the structural testing technique in most cases, but the cases in which the two techniques behave identically, or one better than the other, occur indistinctly for each defect type. According to their second experiment, they concluded that regardless of the defect type, code reading technique is not as effective as functional and structural techniques. Another conclusion is that functional and structural techniques behave identically. The addition of the program version in this last experiment has an impact on the number of subjects that finds a defect. More subjects develop test cases that detect the defects in a version while fewer do it in the other, regardless of the program, technique and defect.

4 Comments on the Articles.

In most of the cases, the defects present on the programs are injected in the code. In comparison with the real practice of the industry, we consider this practice tends to make the conditions in which the experiment is executed less credible.

The differences between the abilities and experience of the subjects are considered only by B-S, as the level of experience is considered a factor. In the rest of the experiments the level of experience is homogenized by similar amounts of training, and students with similar characteristics are chosen (for example: in the same year of the career, or course). We think B-S's choice allows us to gather information more specific on the impact that the differences between subjects and the number of defects that these detect has. This type of information cannot be obtained from the other experiments. It is important to consider that classifying the subjects according to their level of abilities is not always possible, in order to do so the subjects have to be adequately classified.

One aspect in common to all the experiments is the intention of minimizing the risk of information to be shared among the subjects. Several strategies are applied such us to organize the subjects so they work on the same program on the same day.

5 Conclusions

This paper presents a comparison framework for formal experiments intended to study the effectiveness, cost and efficiency of different testing techniques. This framework is developed in order to provide formality when comparing different experiments. These comparisons are not trivial due to the great variability of the characteristics of the experiments.

In addition to this, the application of the framework in four formal experiments is presented. The application of the framework is the comparison of each of these experiments itself. As a result, a better understanding of the more relevant aspects of every experiment is achieved, and can be easily compared considering various relevant aspects. However, the framework needs refining and other characteristics of the experiments should be added. For this kind of experiments, some aspects should also be detailed, such us the techniques applied.

A refining of the framework is currently underway. The intention is not only to improve the comparison framework, but also to be able to count with a group of basic characteristics that any researcher should determine when conducting a formal experiment. If all these characteristics were clearly described by each researcher in every report after performing an experiment, more general conclusions could be achieved and it would be easier to replicate experiments.

Determining the best technique according to the case is far from being done, however, many formal experiments that have already been conducted in addition to others under execution contain lots of important information. The framework presented shares light on how to analyze the information gathered in a precise way.

Table 6. Characteristics of the process

Exp	Organization of the subjects	Training	Process
B-S	8 advanced, 24 intermediate and 42 junior. First session 29, Second 13 and third 32.	The first session consists of the initial training. Subjects are presented with similar types of trainings.	In each phase, three out of the four programs are used, and every subject applies the three techniques and tests the three programs. Each phase consists of 5 sessions: an initial training, three testing sessions and a follow-up session. In each phase every subject tests the same program on the same day.
K-L	First replication: 27 subjects. Second replication: 23 students the first day, 19 the second and 15 the third day	The subjects are presented with the testing techniques and trained prior to the execution	The experiment consists of two internal replications, which are conducted in three settled days out of a week. Each day a different program is tested with the three techniques. The same guidelines as in the training were used.
M-M	Session 1 consists of six groups of three subjects and one of four, while session 2 consists of seven groups of three subjects.	During the first six weeks of the experiment the subjects are trained	The process of inspection takes two sessions. The first one is of individual detection and the second consists of working in groups to reach consolidation. The experiment is conducted for 10 weeks, during the last four the inspections are executed. These consist of inspecting the source code using a check list, considering the specification of the program.
J-V(1)	Consists of 8 groups of 12 subjects (4 test with structural techniques and the other four with functional techniques), and 4 groups of 25 subjects (testing with code reading)	During the initial session the training is conducted	The experiment is organized in 5 sessions, during the last four the testing of the programs is performed. Each day a session is executed, each of the three groups execute a program with a different technique from the three available ones.
J-V(2)	Consists of six groups from seven to eight subjects		The design is changed so every group execute all the techniques. It is organized in three days, and each day only one program and all the techniques are executed. Each of these is executed by two different groups.

Table 7. General Conclusions of the authors

Exp	General conclusion
B-S	According to the number of defects detected and the associated cost, code reading technique is as effective as functional and structural testings. The efficiency, effectiveness and cost depend on the type of software under test.
K-L	When detecting defects, any technique can be as effective as the others, if time is not considered an important aspect and every subject lacks of experience in the language as well as in the three techniques under study.
M-M	There is no big difference between inspections based on paper and those based on the tool, either it is individual or working in group.
J-V(1)	The number of subjects that detect a defect depends not only on the program used but also on the technique applied and the defect itself. Some defects behave better when certain programs are used as well as defects that do so when certain techniques are applied.
J-V(2)	Code reading always behaves worse than the functional and structural techniques, indistinctly for the defect type. With regard to functional and structural techniques, they both behave identically. The program version influences on the number of subjects that detect a defect.

Table 8. Complementary Conclusions of the authors

Exp	Complementary conclusions
B-S	The advanced subjects detected more defects and were more efficient when applying code reading than functional and structural testing. Besides, the number of defects found with functional testing was larger than with structural testing. Intermediate and junior subjects were almost as efficient and effective when applying the three techniques. Code reading technique detected more interface defects than did either of the other techniques, while functional testing did so with control defects. When applying code reading, the subjects gave the most accurate estimates, while functional testers gave the least accurate estimates.
K-L	Subjects were more efficient when applying functional testing.
M-M	Significant differences are not found between both methods as to the number of false positives found, nor in the gained or lost cost due to the meetings.
J-V(1)	Code reading technique is less sensitive to defect hiding than the other techniques. Functional testing behaves better than the structural testing technique in most cases, but the cases in which the two techniques behave identically, or one better than the other, occur indistinctly for each defect type.
J-V(2)	The number of subjects that detect a defect by applying the reading technique does not depend on the observability of the defect. More subjects develop test cases which detect more defects with one version than with the other, regardless of the program, the technique and the defect.

References

1. Myers, G.J.: A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM **21**(9) (September 1978) 760–768
2. Vallespir, D., Herbert, J.: Effectiveness and cost of verification techniques: Preliminary conclusions on five techniques. In: Proceedings of the Mexican International Conference in Computer Science. (2009)
3. Moreno, A., Shull, F., Juristo, N., Vegas, S.: A look at 25 years of data. IEEE Software **26**(1) (Jan.–Feb. 2009) 15–17
4. Basili, V.R., Selby, R.W.: Comparing the effectiveness of software testing strategies. IEEE Transactions on software engineering **13**(12) (1987) 1278–1296
5. Kamsties, E., Lott, C.M.: An empirical evaluation of three defect-detection techniques. In: Proceedings of the Fifth European Software Engineering Conference. (1995) 362–383
6. Macdonald, F., Miller, J.: A comparison of tool-based and paper-based software inspection. Empirical Software Engineering **3**(3) (1998) 233–253
7. Juristo, N., Vegas, S.: Functional testing, structural testing, and code reading: What fault type do they each detect? Empirical Methods and Studies in Software Engineering **2765/2003** (2003) 208–232