Comparing Inspection Methods using Controlled Experiments

Andrea De Lucia[1], Fausto Fasano[1], Giuseppe Scanniello[2], and Genoveffa Tortora[1]

[1] Dipartimento di Matematica e Informatica, University of Salerno
Via Ponte Don Melillo, Fisciano (SA), ITALY
{*adelucia,ffasano,tortora*}*@unisa.it*

[2] Dipartimento di Matematica e Informatica, University of Basilicata
Viale Dell'Ateneo, Macchia Romana, Potenza, ITALY
giuseppe.scanniello@unibas.it

Abstract

Objective: In this paper we present an empirical study that was aimed at comparing three software inspection methods, in terms of needed time, precision, and recall values. The main objective of this study is to provide software engineers with some insight into choosing the inspection method to adopt.
Method: We conducted a controlled experiment and a replication. These experiments involved 48 Master students in Computer Science at the University of Salerno. In the experiments, 6 academic researchers were also involved. The students had to discover defects within a software artefact using inspection methods that differ in terms of discipline and flexibility. In particular, we selected a disciplined but not flexible method (the Fagan's process), a disciplined and flexible method (a virtual inspection), and a flexible but not disciplined method (the pair inspection).
Results: We observed a significant difference in favour of the Pair Inspection method for the time spent to perform the tasks. The data analysis also revealed a significant difference in favour of the Fagan's inspection process for precision. Finally, the effect of the inspection method on the recall is not significant.
Conclusions: The empirical investigation showed that the discipline and flexibility of an inspection method affect both the time needed to identify defects and the precision of the inspection results. In particular, more flexible methods require less time to inspect a software artefact, while more disciplined methods enable the identification of a lower number of false defects.

Keywords: Code Inspection, Controlled Experiment, Distributed Inspection, Fagan's Method, Pair Inspection

1. INTRODUCTION

Software inspection is one of the most widely employed review practices to reach consensus on a software artefact and approve it for use in the project. The goal of an inspection is to identify defects in software artefacts ranging from software requirements specifications to source code.

During the last years formal inspection and other types of review techniques have increased in popularity. This has caused the proliferation of several methods [3] [5] [6] [7] [10] [11] [12] [15] [16] [18] to support software inspection. A possible classification of inspection methods has been proposed by Tervoven *et al.* [16], who defined two dimensions: discipline and flexibility. The discipline dimension concerns the formal aspect of an inspection approach. The flexibility dimension is strongly related to the simplicity of organising and conducting a meeting. This dimension is measured with respect to: place and/or time independence (i.e., the difficulty of having reviewers in the same place at the same time), network (i.e., the base solution for place and/or time independence), tailorable (i.e., how the approach may be adapted according to different inspection scenarios), and varying numbers of participants (what does it happen in case a lower number of participant is available). However, this classification does not help software

quality manager to select the most suitable inspection method to be adopted. The large number of available methods can still generate confusion in the management of a software company.

Empirical studies could be conducted to support the selection of a suitable inspection method. In fact, this kind of studies aims at acquiring general knowledge about which process, method, technique, language, or tool is useful for whom to conduct which task in which environment [1] [9] [14].

In this paper we present the results of a controlled experiment and its replication. These experiments aim at comparing different inspection methods that have been selected to investigate whether the choice of very different approaches significantly affects the inspection results. The choice had to cover the main inspection method categories, according to the classification proposed in [16]. As a consequence, we have compared an inspection method with a high level of discipline and a low level of flexibility (i.e., the Fagan's inspection process [5]), an inspection method with a high level of discipline and flexibility (i.e., a virtual inspection [3]), and a method with a low level of discipline and a high level of flexibility (i.e., the pair inspection method [7]).

The reminder of the paper is organised as follows. Section 2 introduces the experimented inspection methods. The design of the controlled experiments is presented in Section 3, while the achieved results are presented in Section 4. Final remarks and future work conclude the paper.

2. BACKGROUND

In the following subsections we describe the inspection methods considered in the controlled experiments.

2.1. Fagan's process

The first conventional inspection process was proposed by Michael Fagan [5], who defines software inspection as a formal, efficient, and economical method of finding errors in design and code. To inspect a software artefact he proposes a formal and structured process, composed of five sequential phases: Overview, Preparation, Inspection, Rework, and Follow Up. In the Overview phase the designer first describes the overall domain area being addressed and then provides details about the specific area he/she has designed. Documentation concerning the software artefact to inspect is distributed to all the participants of the inspection team in the Preparation phase. During the Inspection phase a face-to-face meeting is carried out to identify the defects. The moderator should produce a written report, which is provided to the author of the artefact, to address the identified defects, during the Rework phase. In the Follow Up phase the moderator checks the quality of the rework and determines whether a re-inspection is required.

2.2. Virtual inspection method

In [3] an inspection method with a high level of discipline and flexibility is presented. The inspection process is composed of seven phases, namely Planning, Overview, Discovery, Refinement, Inspection Meeting, Rework, and Follow Up. The process is implemented by a web based tool named WAIT (Web Based Inspection Tool) [3]. In the Planning phase the quality manager specifies which artefact version must undergo a review process. In the Overview phase, the artefact author explains the design and the logic of the software artefact to the inspectors. During the Discovery phase, each inspector analyses the artefact and takes note of the candidate defects. When this phase is completed, the moderator accesses the defect log, containing all the defects identified by the inspectors. The moderator depending on the results of the Discovery phase can decide if the refinement phase must be enacted. In the Refinement phase, the inspector accesses the merged defect list and selects one of the defects that caused the conflict. By accessing the defect details the inspector decides whether it is an actual defect or a false positive. To this end, the inspector may decide to further analyse the considered software artefact. This phase is concluded when all the conflicts are solved or when the inspectors are not able to reach an agreement on the

conflicts. In the Inspection Meeting phase the unsolved conflicts are synchronously discussed. In case defects have been identified, the author has to fix them during the Rework phase. Once the defects have been addressed, the author creates a new version of the artefact that is validated by the moderator during the Follow Up phase. During the Follow Up phase the moderator checks the quality of the rework and decides whether a re-inspection is needed.

2.3. Pair Inspection

Pair inspection is a very flexible and undisciplined way to review software artefacts [7]. It requires only two participants: the author and an inspector who reviews the author's artefact. Pair inspection requires continuous iterations, so any strict rules can be formulated to guide the inspection. Due to the iterative nature of the process, the recording of defects found in informal meetings is not required. A form or a template is adopted to annotate the defects of the software artefact under revision. In some cases the defects are typically marked on paper documents and corrected during the next correction cycle. A checklist can be optionally used to drive the inspectors during the review of a software artefact. In addition the author and inspector together can also write down comments for the acceptance review. As shown in [8] pair inspection is generally useful for small size software artefacts.

3. DESIGN OF THE CONTROLLED EXPERIMENTS

In this section, we present the design of the controlled experiment and its replication. The aim of the replication was to increase confidence in the conclusion validity [2]. Both the controlled experiments follow the template suggested by Wohlin *et al.* [17].

3.1. Experiment definition and context

The context of the first experiment was constituted of master students in Computer Science at the University of Salerno. The students were 24 volunteers with comparable background (they were graduated students with basic software engineering and programming experience) and 6 academic researchers (acting as moderators), who were randomly grouped in 9 teams. Among the teams 6 were composed of 3 inspectors and 1 moderator (i.e., one author). It is worth noting that the moderators were asked to lead the meeting and manage the inspection process, without taking part to the defect identification. The remaining 3 teams were composed of 2 inspectors. The experiment has been performed online within a Software Engineering Laboratory of the University of Salerno.

The following is the selected task to be performed in the experiment:

T_1: inspecting a Java class composed of 166 LOCs implementing a binary tree data structure and the algorithms to traverse and modify it.

Even in the replicated experiment we selected 24 master students in Computer Science with comparable background. The same 6 academic researchers acted as moderators in the replicated experiment. The subject were volunteers and were grouped similarly to the the first experiment. The only difference between the first experiment and its replication consists of the task to perform:

T_2: inspecting a Java class composed of 145 LOCs enabling the construction and the execution of queries on a database.

The tasks were expected to be accomplished within one hour and a half and were selected to analyse the effects of changing the number of true defects within the tasks. In particular, 78 and 56 were the true defects of the classes of the tasks T_1 and T_2, respectively. The following inspection methods were considered: Fagan's method (FAG), virtual inspection (WAIT), and pair inspection method (PI). To avoid biasing the results due to the choice of different approaches (e.g., checklist, scenario, or perspective based) all the investigated methods were based on checklists. Moreover, as the results could be also affected by the checklist, we also provided the same checklist to the subjects involved in both the experiments.

3.2. Hypotheses

The first goal of the experiments was to verify whether an inspection method influences the time required to identify the defects in source code. To this end, the following *null hypothesis* has been formulated:

Hn_1: the use of one of the considered methods (i.e., FAG, WAIT, or PI) does not significantly affect the time to complete the inspection task;

The alternative hypothesis can be easily derived:

Ha_1: the use of one of the experimented methods significantly affects the time to complete the inspection task;

In order to asses the first *null hypothesis*, the time spent to accomplish the task has been considered. The overall quality of the results achieved by applying the selected methods has also been analysed to investigate the effect of using one of the selected inspection methods. To this end, the recall and the precision measures have been considered. In our case, the *recall* is defined as the ratio between the number of actual defects identified by the team over the total number of actual defects, while the *precision* is the number of actual defects identified by the team over the total number of identified defects. Both the precision and the recall range between 0% and 100%. If the recall is 100%, it means that all the true defects have been identified, though there could be identified defects that are false defects. If the precision is 100%, it means that all the identified defects are correct, though there could be correct defects that were not identified. To assess the presence of a significant difference among the considered treatments in terms of recall and precision we have formulated the following *null hypotheses*:

Hn_2: the use of one of the experimented methods does not significantly affect the recall value of the inspections performed by the teams;

Hn_3: the use of one of the experimented methods does not significantly affect the precision value of the inspections performed by the teams;

Even in this case the alternative hypotheses can be easily derived:

Ha_2: the use of one of the experimented methods significantly affects the recall value of the inspections performed by the teams;

Ha_3: the use of one of the experimented methods significantly affects the precision value of the inspections performed by the teams;

3.3. Selected variables and experiment design

In order to properly design the experiment and analyse the results, the following independent variable was considered:

Method : this variable indicates the factor on which the study is focused, i.e. FAG, WAIT, and PI.

To verify the formulated hypotheses, we considered the following dependent variables:

Time : the average time that the inspectors of a given team spent to perform the task;

Recall : the recall value that an inspection team achieved by performing the task;

Precision : the precision value that an inspection team achieved by performing the task.

Table 1 summarises the design of the experiments presented in this paper. This table also shows the id of the inspection teams, the composition of each team, and the treatment and task used

TABLE 1: design of the controlled experiment and its replication

Team ID	Team Composition	Treatment	Task
EXP 1.1	3 inspectors + 1 moderator	FAG	T_1
EXP 1.2	3 inspectors + 1 moderator	FAG	T_1
EXP 1.3	3 inspectors + 1 moderator	FAG	T_1
EXP 1.4	3 inspectors + 1 moderator	WAIT	T_1
EXP 1.5	3 inspectors + 1 moderator	WAIT	T_1
EXP 1.6	3 inspectors + 1 moderator	WAIT	T_1
EXP 1.7	2 inspectors	PI	T_1
EXP 1.8	2 inspectors	PI	T_1
EXP 1.9	2 inspectors	PI	T_1

First controlled experiment

Team ID	Team Composition	Treatment	Task
EXP 2.1	3 inspectors + 1 moderator	FAG	T_2
EXP 2.2	3 inspectors + 1 moderator	FAG	T_2
EXP 2.3	3 inspectors + 1 moderator	FAG	T_2
EXP 2.4	3 inspectors + 1 moderator	WAIT	T_2
EXP 2.5	3 inspectors + 1 moderator	WAIT	T_2
EXP 2.6	3 inspectors + 1 moderator	WAIT	T_2
EXP 2.7	2 inspectors	PI	T_2
EXP 2.8	2 inspectors	PI	T_2
EXP 2.9	2 inspectors	PI	T_2

Replicated experiment

TABLE 2: Survey questionnaire

ID	Question
Q_1	I had enough time to perform the inspection task
Q_2	The task objectives were perfectly clear
Q_3	Performing the task was easy
Q_4	The supporting material to perform the task provided enough information
Q_5	The checklist was clear and well structured
Q_6	The defect identification was easy

by each team. It is worth noting that the subjects were randomly assigned to the treatments and each team performed only one inspection task.

3.4. Preparation

The subjects attended an introductory lesson on the usefulness of the inspection methods to detect and remove defects early in the development process of software systems. Depending on the treatment, they also attended a training session aimed at presenting detailed instructions on the Fagan's process, the virtual inspection, and the pair inspection method. We also proposed some examples that were not related to the tasks to avoid biasing the experiment. Furthermore, the web based tool was fully described to the subjects that experimented the WAIT method. The training sessions aimed at providing all the subjects an equal prior knowledge and to deeply describe the tool. The lessons for each treatment were all concluded presenting detailed instructions on the tasks to be performed in the laboratory session.

At the end of the laboratory sessions of the controlled experiments the subjects were asked to fill in a survey questionnaire. The questionnaires aimed at assessing the overall quality of the material provided to the subjects, the perceived effort to perform the inspection tasks, and the clearness of the inspection tasks and used checklist. The answers of the survey questionnaires are based on a five-point Likert scale [13]: from 1 (strongly agree) to 5 (strongly disagree). Table 2 shows the survey questionnaire used in both the experiments.

3.5. Material, execution, and data analysis

The supervisors collected the log files containing the information traced by the web based tool during the inspection as well as the defect reports. The supervisors also gathered the information (i.e., the times and the defect log reports) of the experiment performed using the treatment FAG and PI. To perform the experiments each inspector was provided with a folder containing a pencil, white sheets, and the following hard copy material:

- the guidelines to perform the assigned tasks according to the inspection method (i.e., FAG, WAIT, or PI);
- the source code of the class to be inspected in the task;

TABLE 3: data of the controlled experiments

Method	Time (min.)	Recall (%)	Precision (%)
FAG	72	41.03	84.21
FAG	78	58.97	100
FAG	43	48.72	97.44
WAIT	78	42.31	78.54
WAIT	79	74.36	93.55
WAIT	49	84.62	84.62
PI	46	75.64	70.24
PI	38	55.13	93.48
PI	36	88.46	85.19

First controlled experiment

Method	Time (min.)	Recall (%)	Precision (%)
FAG	58	51.79	82.86
FAG	42	53.57	96.77
FAG	64	51.79	93.55
WAIT	42	64.29	85.71
WAIT	57	44.64	89.29
WAIT	66	62.50	92.11
PI	39	55.36	70.45
PI	38	51.79	82.86
PI	42	53.57	75.00

Replicated experiment

TABLE 4: mean values for the selected variables of both the controlled experiments

Method	Time (min.)	Recall (%)	Precision (%)
FAG	59.5	50.97	92.47
WAIT	61.8	62.12	87.30
PI	39.8	63.32	79.53

- a checklist and an inspection defect log template to be used to perform the inspection task using the treatments FAG and PI;
- the survey questionnaire to be filled in at the end of the laboratory session.

The inspection teams performed the tasks without time limit.

To verify the *null hypotheses* of the experiments, we planned to adopt the two-way ANOVA test. To apply this test the following four assumptions have to be verified: the observations are independent, the scale of measurement for the observations is interval, the distribution is normal, and the variance of the observations is constant. For the considered dependent variables the first and the second assumptions are easily verified due to the experiment design and the scale of measurement. On the other hand, to verify the normality of the distributions we used the Shapiro-Wilk tests, while to verify the last assumption the Levene statistic test has been adopted. In case the two-way ANOVA test cannot be applied the non-parametric Kruskal-Wallis test has been used [4]. Let us note that the results of the tests used here are intended as statistically significant at $\alpha = 0.05$.

In the following, we perform the analysis on the data from both the experiments simultaneously. This was possible as the second experiment is an exact replication of the first one (they only differ in the task to be performed).

4. RESULTS

The times to accomplish the laboratory sessions as well as the identified defects have been collected after the execution of the experiment. To compute the precision and recall values, two researchers worked together to analyse the defect log reports produced by the teams and to classify the defects as false positives or true defects. After the experiments executions, the survey questionnaires were collected as well.

4.1. Influence of the Method factor

The values of the variables Time (expressed in minutes), Recall, and Precision corresponding to the inspection teams of the first controlled experiment and its replication according to the Method factor are shown in Table 3. The mean values of the considered dependent variables grouped by the Method factor are shown in Table 4.

The Shapiro-Wilk test revealed that the normality of the distribution of the Time variable is constant. On the other hand, the Levene test revealed that the variance of the observations is

TABLE 5: two-way ANOVA results on Precision (R-Squar.=0.451 and Adjusted R-Squar.=0.223)

Source	Type III Sum of Squares	df	Mean Square	F	p-value
Method	508.760	2	254.38	4.122	0.043
Task	19.355	1	19.355	0.314	0.586
Method*Task	81.322	2	40.661	0.659	0.535
Error	740.511	12	61.709		
Total	135834.197	18			

not constants. Hence, the two-way ANOVA test cannot be applied. Consequently, to verify the *null hypothesis* Hn_1 the Kruskal-Wallis test has been adopted. This test revealed that this hypothesis can be rejected (i.e., *p-value* = 0.011), so the effect of the Method factor is statistically significant. In fact, the subjects experimented the Fagan's inspection process spent on average more time than the subjects experimented the other two inspection methods. Let us also note that on average the time spent to perform the inspection tasks was the smaller, when PI was used.

The Shapiro-Wilk test revealed that the normality of the distribution of the Recall variable is not constant, so the two-way ANOVA test cannot be applied. Hence, the *null hypothesis* Hn_2 has been verified using the Kruskal-Wallis test, which revealed that this hypothesis cannot be rejected (i.e., *p-value* = 0.179). This means that the effect of the Recall dependent variable on Method is not statistically significant. Therefore, a further analysis on the recall values revealed that the subjects achieved similar results independently on the used inspection methods.

The Shapiro-Wilk and Levene tests reveled that the distribution and the variance of the Precision variable are constant. Hence, two-way ANOVA can be applied. The results achieved by applying this test are shown in in Table 5. The interaction between Method and Task is not significant, while there was a significant effect of Method overall. In fact, we observed that the subjects benefit more from the use of the Fagan's inspection method. On the other hand, the subjects benefit less in case the Pair Inspection method was used. Furthermore, ANOVA also showed that the influence of the Task factor is not significant.

4.2. Survey questionnaire results

The survey questionnaire of the controlled experiments are considered and analysed together. In particular, the data collected according to the methods FAG, PI, and WAIT are visually summarised in Figure 1. The agreement level considering the answers of the survey questionnaire can be generally considered concordant. For example, the boxplots of the questions Q_1 and Q_2 show that the time to perform the inspection tasks and the objectives of the experiment was considered appropriate. The distributions of the question Q_3 concerning the methods FAG and WAIT are nearly the same. Two outliers were present in the distribution of the WAIT method. Despite the different distributions, the medians of the observations are the same. We can conclude that the subjects found the inspection task simple to perform in both the controlled experiments. A positive judgement on the hard copy material provided as support to accomplish the experiments was also expressed (see boxplots of the question Q_4). However, the distribution of the answers of the subjects that used the WAIT method presents three outliers. A good judgement on the checklist was expressed by the subjects as the boxplots of the question Q_5 show. On average, the subjects found the defect identification simple when the methods FAG and PI were used (see boxplots of the question Q_6).

4.3. Threats to validity

We describe here the threats to validity (i.e., internal, construct, external, and conclusion validity threats) that could affect the presented controlled experiments. The internal validity threats of this experiment are mitigated by the experiment design. In fact, each group performed only one task adopting only one treatment (i.e., FAG, WAIT, and PI). The survey also confirmed that the subjects found clear everything regarding the tasks. Moreover, the subjects did not know exactly the hypotheses of the experiment and were not evaluated on their performance.

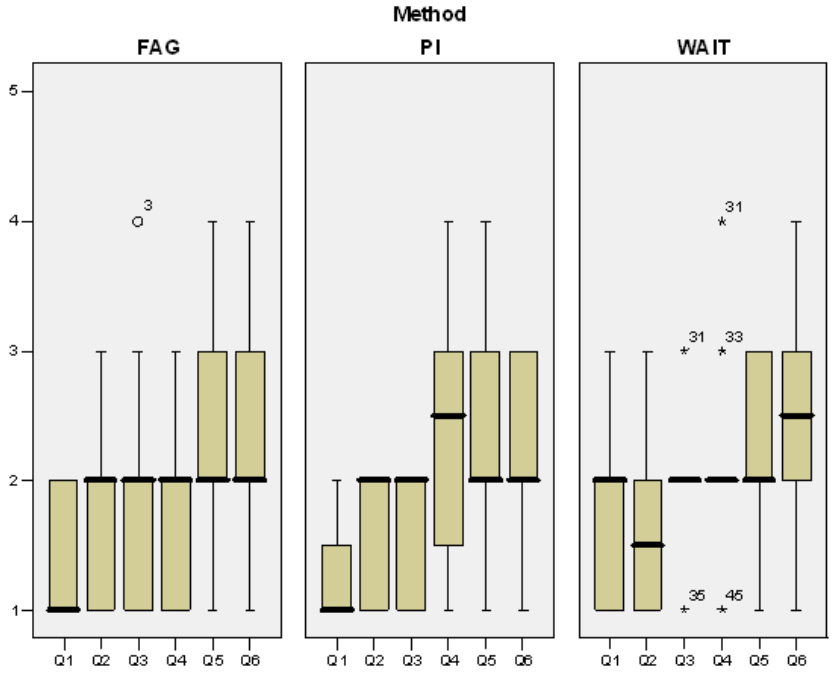

FIGURE 1: boxplots of the survey questionnaire of both experiments

The construct validity threats (i.e. if the relationship between cause and effect is causal) were mitigated by a proper design that allowed separating the analysis of the different factors and of their interactions. Moreover, the measurements of the dependent variables were performed considering the times gathered by the experiment supervisors for the treatments FAG and PI and analysing the log files produced by the tool when the task was performed using the treatment WAIT. Moreover, the precision and recall measures have been widely employed in the past and well reflect the overall quality of the inspections. To compute these measures true and false positive defects that experts manually identified have been used. The experts worked independently and iterated until an agreement was reached on each inspection. However, the method to assess the inspection quality could also threat the construct validity. Finally, the survey questionnaire was designed using standard ways and scales [13].

External validity is always present when experiments are performed with students. However, last-year master students are not far from junior industry programmers and have a good analysis, development and programming experience. The subjects of both the experiments were also familiar with the source code of the tasks. The tasks are also representative and are sufficiently complex to generalise the results achieved on the low level software artefacts (i.e., the source code). Nevertheless, further replications on larger datasets should be performed with different subjects in different contexts to confirm or contradict the achieved results. Indeed, it will be worth replicating the experiment within professional development environments using both high and low level software artefacts. It is worth noting that no subjects abandoned the experiments.

Conclusion validity is concerned with the relationship between the considered treatments and the outcomes. In the controlled experiments presented in this paper conclusion validity threat was mitigated by the experiment design and by the properly selection of the population. Regarding the recruited subjects, we drew a fair sample from that population and conducted our experiments with subjects belonging to this sample. However, to confirm or contradict these results further replications on larger datasets are required. Proper tests were also performed to statistically reject the defined *null hypotheses*. In cases where differences were present but not significant,

this was explicitly mentioned. Furthermore, when there were not the conditions necessary to use parametric tests non-parametric tests were used.

5. DISCUSSION AND FUTURE WORK

This paper has presented an empirical investigation aimed at comparing three different inspection methods that have been selected considering the discipline and flexibility dimensions [16]. In particular, we have considered the Fagan's inspection process, an inspection method with a high level of discipline and a low level of flexibility, the WAIT inspection process, a virtual inspection method with a high level of discipline and flexibility, and the pair inspection, a method with a low level of discipline and a high level of flexibility. The Fagan's inspection process and the pair inspection method require face-to-face meeting, while the WAIT inspection process supports geographical dispersed reviews.

In this paper, we have presented the results of two controlled experiments both performed with Master students in Computer Science. These experiments were conducted within a Software Engineering laboratory at the University of Salerno. The replicated experiment defers from the first experiment on the source code used within the inspection task. Owing to this fact the data analysis has been performed considering the experiments together. This analysis revealed a significant difference in favour of the Pair Inspection method for the time spent to perform the tasks. This could be due to the fact that Pair Inspection is more agile and flexible than the other considered inspection methods. Moreover, because of the lower number if participants, an agreement is more easy to be achieved with respect to the other methods. On the other hand, this also limits the possibility to identify false defects.

A significant difference was observed on the precision values that the subjects achieved by performing the tasks using the Fagan's method. This result could be due to the fact that this inspection method is more rigorous and is based on structured process. Even the virtual inspection method benefited from the use of the structured process. The worse results were achieved when the Pair Inspection method was adopted.

The data analysis also revealed that the inspection methods did not affect the recall values that the subjects achieved by accomplishing the tasks. This means that the number of true defects identified by the subjects is nearly the same whatever inspection method has been used. As results, the recall is not a key issue in the choice of an inspection method.

One of the main contributions of the empirical investigation presented here concerns the identification of some directions to select the more suitable inspection process within a productive environment according to the software project requirements. For example, in case the inspection team is geographically dispersed a distributed inspection process could be adopted without affecting the recall values achieved by the practitioners. An agile approach could be used in case the time required to inspect a software artefact should be minimised. Such an approach reduce on average the time to inspect a software artefact preserving the number of identified defects. The Fagan's inspection method should be adopted in case a low number of false defects is admittable and the cost to detect defects within a software artefact is not an issue.

In the future, we plan to conduct controlled experiment replications and case studies with different subjects and tasks. Furthermore, it will be worth conducting industrial case studies with practitioners on actual software artefacts. We also plan to investigate the effect of using the selected inspection methods on high level software artefacts.

6. ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their insights and feedbacks to several key issues covered in this work. The work described in this paper is supported by the project METAMORPHOS, funded by MiUR (Ministero dell'Universitá e della Ricerca) under grant PRIN-2006-2006098097.

REFERENCES

[1] Victor R. Basili, R. W. Selby, and D H. Hutchens. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, 12(7):733–743, 1986.

[2] Victor R. Basili, Forrest Shull, and Filippo Lanubile. Building knowledge through families of experiments. *IEEE Trans. Softw. Eng.*, 25(4):456–473, 1999.

[3] Andrea De Lucia, Fausto Fasano, Giuseppe Scanniello, and Genoveffa Tortora. Integrating a distributed inspection tool within an artefact management system. In *Proceedings of 2nd International Conference and Data Technologies*, pages 184–189, Barcellona, Spain, 2007.

[4] Jay L. DeVore and Nicholas R. Farnum. *Applied Statistics for Engineers and Scientists*. Duxbury, 1999.

[5] Michael E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[6] Daniel P. Freedman and Gerald M. Weinberg. *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*. Dorset House Publishing Co., Inc., New York, NY, USA, 2000.

[7] Juha Iisakka, Ilkka Tervonen, and L. Harjumaa. Experiences of painless improvements in software inspection. In *Project Control for Software Quality, ESCOM-SCOPE'99*, pages 321–327. Shaker Publishing B.V., 1999.

[8] Juha Iisakka, Ilkka Tervonen, and Paula Hiitola. How to inspect minor software projects. In *Project Control for Software Quality, ESCOM-SCOPE'00*, pages 359–366. Shaker Publishing B.V., 2000.

[9] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.*, 28(8):721–734, 2002.

[10] John C. Knight and E. Ann Myers. An improved inspection technique. *Communications of the ACM*, 36(11):51–61, 1993.

[11] Filippo Lanubile, Teresa Mallardo, and Fabio Calefato. Tool support for geographically dispersed inspection teams. *Software Process: Improvement and Practice*, 8(4):217–231, 2004.

[12] Paul Murphy and James Miller. A process for asynchronous software inspection. In *STEP '97: Proceedings of the 8th International Workshop on Software Technology and Engineering Practice (STEP '97) (including CASE '97)*, page 96, Washington, DC, USA, 1997. IEEE Computer Society.

[13] A.N. Oppenheim. *Questionnaire Design, Interviewing and Attitude Measurement*. Pinter Publishers, 1992.

[14] Shari Lawrence Pfleeger and Winifred Menezes. Marketing technology to software practitioners. *IEEE Software*, 17(1):27–33, 2000.

[15] Chris Sauer, D. Ross Jeffery, Lesley Land, and Philip Yetton. The effectiveness of software development technical reviews: A behaviorally motivated program of research. *IEEE Transactions on Software Engineering*, 26(1):1–14, 2000.

[16] Ilkka Tervonen, Juha Iisakka, and Lasse Harjumaa. Software inspection – a blend of discipline and flexibility. In *Proceedings of ENCRESS-98*, pages 157–166. Shaker Publishing B.V., 1998.

[17] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[18] Takuya Yamashita. Evaluation of jupiter: A lightweight code review framework. Master's thesis, University of Hawaii, 2006.