

A Controlled Experiment to Assess the Effectiveness of a Distributed Method for Code Inspection

Andrea De Lucia, Fausto Fasano, and Genoveffa

Tortora

Dipartimento di Matematica e Informatica,

University of Salerno, Via Ponte Don Melillo,

84084, Fisciano (SA), ITALY

{adelucia, ffasano, tortora }@unisa.it

Giuseppe Scanniello

Dipartimento di Matematica e Informatica,

University of Basilicata, Viale Dell'Ateneo,

Macchia Romana, 85100, Potenza, ITALY

giuseppe.scanniello@unibas.it

1. Introduction

In this report we propose a distributed inspection method, which tries to minimize the synchronous collaboration among team members to identify defects in software artifacts. The main idea of our approach consists of identifying conflicts on the potential defects and then resolving them using an asynchronous inspection meeting before performing a traditional synchronous meeting. A synchronous inspection meeting is often not performed since it is generally used to remove conflicts on the defects that the inspection members have not solved. This is because at the end of the asynchronous meeting unsolved conflicts are rare or absent at all. This approach has been implemented in a web based tool and assessed using a controlled experiment, which context was constituted of master students in Computer Science at the University of Salerno.

The remainder of the report is organized as follows: Section 2 summarizes the Fagan's method. The migration process and the tool are described in Section 3, while the controlled experiment and the achieved results are presented in Section 4. Final remarks, lesson learned, and future work conclude the document.

2. Fagan's Method

Generally, the inspections activity finds defects in software artifacts in a formal meeting. The first structured inspection process was Michael Fagan's inspection method [1]. This method consists of six phases: *Planning*,

Overview, Preparation, Inspection Meeting, Rework, and Follow Up. During the Planning, the moderator selects the reviews, organizes them in a team, and schedules the next phases of the inspection. In the Overview phase the author of the software artifact briefly presents the purpose and the scope of the artifact as well as the inspection aims. The team members analyze the software artifact and individual detect its defects in the Preparation phase. Defects are identified exploiting check items, which are grouped in a checklist. Successively, a face-to face meeting is performed to discuss the defects identified in the previous phase. During the Rework phase the author revises the inspected software artifact. Finally, the moderator checks the quality of the rework and determines whether a re-inspection of the artifact is required.

3. The inspection process and tool

The inspection process we propose is shown in Figure 1. Ellipses represent mandatory phases of the process, while optional phases are shown as outlined ellipses. This process replaces Preparation phase of the Fagan's inspection process with two new sequential phases: *Discovery* and *Refinement*. Essentially, the goal of the Preparation and Discovery phases consists of identifying defects in the inspected software artefact. Indeed, for the Preparation phase the goal changes from pure understanding to defect identification, so the team members individually take notes of the candidate defects, which should be considered in the Refinement phase. In the Refinement phase first the defects are putted together to identify potentially conflicts. The potential conflicts are then asynchronously solved by the team members.

Conflicts can be produced at two different granularity levels: at check item and within a check item. In the first case at least a review declares that a check item is not verified or not applicable and the remaining reviewers state that the check item is verified. Differently, we are in the latter case when a concern on a check item (i.e., unique defect) of the checklist is raised by a single review. For example, a single review detects a variable declaration that is not compliant with the naming convention. The main goal of the Refinement phase is to remove false defects and to build the consensus on the true defects. As suggested by the empirical study proposed in [4] we consider a defect as a true defect when at least two reviews recognize it. Indeed, the minimum number of reviews required to get an agreement can be differently chosen according to the inspection process constraints. Whether an agreement has not been obtained the reviewers have to discuss on the unsolved conflicts using either face-to-face meeting or web based messaging tool. Let us note that the phases Planning, Overview, Rework and Follow-Up are the same as the Fagan's method.

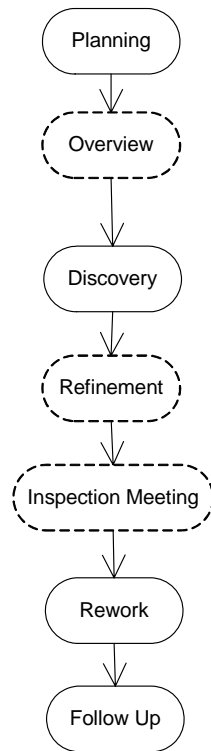


Figure 1. Reengineered Inspection Process

This process has been implemented within a web based tool, which in turns has been integrated within ADAMS [2]. In particular, this tool during the Planning enables the moderator to selects the team members, and schedules the inspection phases of the software artifacts. To this end the human resource management functionality of ADAMS is used. In the Overview phase the author of the software artifact eventually produces a document that briefly describes the purpose and the scope of the artifact to be inspected and then deploys it in ADAMS. Successively, each member of the inspection team downloads the document and becomes familiar with it. Once the software artifact has been analyzed, each review fills the checklist in and specifies for each check item where the defect has been detected. Indeed, in case of textual defect the review has to specify its location within the software artifact in terms of page and line number (see Figure 2). Differently, in case a defect has been identified on a picture of the software artifact its number as well as its page number has to be indicated.

Figure 2. Inspection tool

An event is notified to the inspection manager, when all the reviewers have accomplished the Discovery phase. Successively, the manager starts the Refinement phase and the inspection tool sends an email containing the conflict list to the inspection team. This mail also aims at notifying the team members that the conflicts can be analyzed in order to get an agreement. Depending on the inspection process constraints, a conflict is removed when at least two members reach an agreement. Indeed, a defect is highlighted in green when conflicts at both check item level and within a check item have not been raised, while is yellow when a unique defect has been identified. A defect is red when a conflict has been raised at check item level. Figure 3 shows a snapshot of the tool during the conflict resolution. Once the refinement phase is concluded the reviews can use the inspection synchronous chat of the tool to discuss on the unsolved defects.

Question	Complies	Text
I nomi delle variabili e delle classi sono significativi	Yes	COMPLIES
Il codice è opportunamente indentato	No	Yes=1 No=1 Not Applicable=1
Gli identificatori di costante, di variabile, i nomi delle classi e i nomi dei p...	No	Defect in Text pp.1..1 line 2..2 identified by 1 inspector(s) on 3 Defect in Text pp.1..1 line 33..33 identified by 1 inspector(s) on 3

■ No Conflict
 ■ Conflicts on defects only
 ■ Conflict on pass/fail answer

Refinement Complete

Figure 3. The defect log report

4. The controlled experiment

The usefulness of the approach in the inspection of source code was evaluated through a controlled experiment. In this section we describe this experiment following the template suggested by Wohlin *et al.* [6].

4.1 Experiment definition and context

The context of the experiment was constituted of master students in Computer Science at the University of Salerno. The students were 24 volunteers with comparable background, who were randomly grouped in inspection teams each composed of 3 students and 1 moderator. The experiment has been performed online at the University of Salerno. The PCs involved in the experiment had a comparable hardware and software configuration, and were connected each other by a LAN.

The following are the selected migration tasks that the subjected had to perform:

- **T₁**: inspecting a java class composed of 166 LOCs implementing a binary tree data structure and the algorithms to traverse and modify it;
- **T₂**: inspecting a java class composed of 145 LOCs enabling the construction and the performing of queries on a database.

The tasks were expected to be accomplished within one hour and a half and were selected to analyze the effects of changing the complexity of the tasks. Moreover, these tasks have also been selected to present the majority of the problems that teams could encounter in the inspection of source code.

Two different inspection methods, namely *FA* and *NOFA*, are used to perform the inspection tasks. Both methods aimed at inspecting the component of the two tasks. The difference is that the *FA* method is performed using the traditional Fagan's method [1], while *NOFA* is performed using our inspection process as well as the web based tool implementing it.

4.2 Hypotheses

Since one of the aims of this experiment is to verify whether the use of the proposed tool influences the productivity in the identification of defects in source code with respect to the use of the Fagan's inspection method [1], we have formulated the following null hypothesis:

- **H_{n1}**: the use of the proposed approach and the tool does not significantly affect the effort to inspect the source code;

The alternative hypothesis is

- **H_{a1}**: the use of the proposed approach and the tool significantly affects the effort to inspect the source code;

The productivity is not straightforward to compare the proposed approach and the Fagan's method [1] approaches. Thus, we also considered in this experiment whether the tool supports the defect identification by applying our inspection process. To this end the following null hypotheses have been formulated:

- **H_{n2}:**

teams in each laboratory session. We randomly assigned two inspection teams to the group A and B, and one inspection team to the group C and D.

Table 1. Experiment design

	Groups			
	A	B	C	D
Lab1	T1_FA	T1_NOFA	T2_FA	T2_NOFA
Lab2	T2_NOFA	T2_FA	T1_NOFA	T1_FA

4.4 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. In this lesson the Fagan's inspection method was deeply presented and discussed. Examples of java and C++ source code (not related to the tasks to avoid biasing the experiment) were proposed too.

Successively, the subjects also attended a training session on the approach and the tool. The training sessions aimed at providing all the subjects an equal prior knowledge and to deeply describe the software application and inspection process. To give subjects more confidence with the tool some examples (not related to the tasks to avoid biasing the experiment) were also presented. The training sessions were concluded presenting detailed instructions on the tasks to be performed.

Depending on the Method variable, at the end of each laboratory session a survey questionnaire has been presented to the subjects. In particular, Table 2 and Table 3 present the questionnaires for the methods FA and NOFA, respectively. The questionnaires aimed at assessing the overall quality of the provided material, the effort required to perform the inspection tasks, and the clearness of the laboratory session goals and of the inspection tasks. Regarding the method NOFA the questionnaire also aimed at assessing the usefulness of the tool and its ease of use. It is worth noting that the questions ranging from Q1 to Q5 of both the questionnaires are the same as well as the question Q8 of the Table 2 and question Q9 of the Table 3.

The questions of the two survey questionnaires expect answers according to the five-point Likert scale [5]: from 1 (strongly agree) to 5 (strongly disagree).

4.5 Material and execution

The inspection teams accomplished each laboratory session of the experiment without time limit. The inspection team members were asked to fill in the survey questionnaire corresponding to the inspection method assessed in the accomplished laboratory session. The supervisors also 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 gather the information (i.e., the times and the defect log reports) of the experiment performed using the FA method.

Table 2. FA method questionnaire

Id	Question
Q1	I had enough time to perform the inspection task
Q2	The task objectives were perfectly clear
Q3	The task was perfectly clear
Q4	The supporting material to perform the task provided enough information
Q5	The checklist was clear and well structured
Q6	The defect identification was simple
Q7	Performing the face-to-face meeting was simple
Q8	How much time (in terms of percentage) did you spend to identify the defect according to the provided checklist? (A < 25% - B >= 25% and < 50% - C >= 50% and < 75% - D >= 75%)?
Q9	How much time (in terms of percentage) did you spend to carry out the synchronous meeting? (A < 25% - B >= 25% and < 50% - C >= 50% and < 75% - D >= 75%)?

Table 3. NOFA method questionnaire

Id	Question
Q1	I had enough time to perform the inspection task 52. well structured

1. The introductory presentation of the training session;
2. We provided the guidelines to perform the assigned tasks according to our inspection method and the Fagan's method;
3. The source code of the component to be inspected in the two tasks;
4. We provided a checklist and an inspection defect log template to be used to perform the inspection task using the FA Method;
5. The survey questionnaires to be filled in at the end of the two laboratory sessions.

4.6 Experiment Results

After the execution of the experiment, we collected the times to accomplish the laboratory sessions as well as the identified defects². To classify the defects as false positive and true defects, two researchers working together identified the actual defects of the inspection tasks and then analyzed the report defect logs produced by the teams.

The times to perform the inspection tasks according to the considered treatments are summarized by the boxplots of Figure 4, while the descriptive statistics are reported in Table 4. The members of the inspection teams spent on the average 43 and 36 minutes to perform the tasks using the method FA and NOFA, respectively. Moreover, we can observe that mean times to accomplish the task T_1 using the methods FA and NOFA are comparable (i.e., 43 and 44 respectively). On the other hand, to perform the task T_2 the inspection teams spent on the average 49 and 36 using the methods FA and NOFA, respectively.

Table 4. Descriptive statistics

Task	Method	Min.	Max.	Mean	Std. Dev.
ALL	FA	43	90	67.3889	15.57577
	NOFA	36	92	62.4444	18.04968
T_1	FA	43	90	72.2222	18.06085
	NOFA	44	92	68.7778	16.51346
T_2	FA	49	80	62.5556	11.69520
	NOFA	36	82	55.4444	16.89757

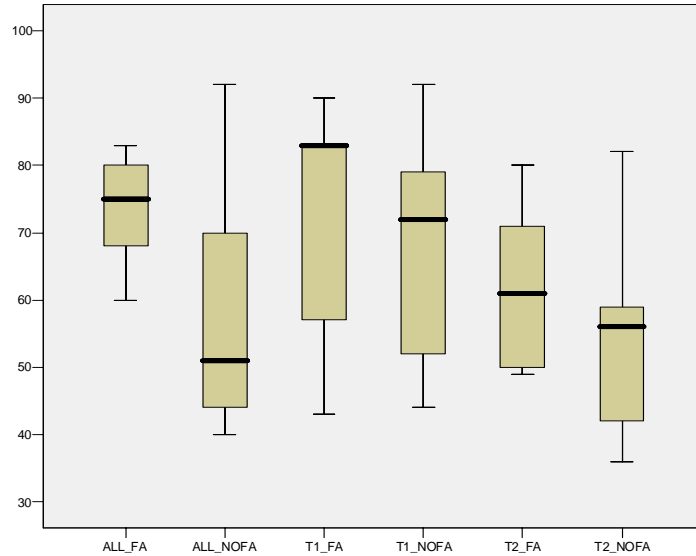


Figure 4. Boxplots of the times

To verify the null hypotheses of the experiment we decided to use 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. In case the two-way ANOVA test can be applied, the basic outputs are analyzed, and then whether a statistical significance difference between the observations and dependent variables was revealed the causes have been further investigated [3].

The first and the second assumptions were easily verified due to the experiment design and the scale of measurement for the dependent variables, respectively. To verify the normality of the distributions we used both the Kolmogorov-Smirnov and Shapiro-Wilk tests. The results of these tests revealed that the normality is verified for the observations of the dependent variables IdD, TD, and FP, while is not verified for the observations of the Time variable (p -value = 0.031).

To verify the last assumption on the variables IdD, TD, and FP, we adopted the Levene statistic test. This test revealed that the null hypothesis cannot be rejected on the variables IdD (p -value = 0.134) and FP (p -value = 0.248), thus the variances of the observations are considered constant. Differently, the variance of the observations of the variable TD is not constant (p -value = 0.037).

Concluding, for the variables IdD and FP the assumptions to apply the two-way ANOVA test were verified, while were not satisfied for the variables Time and TD. Thus, we decided to use a k non parametric test (i.e., Friedman test) to verify the null hypotheses H_{n1} and H_{n3} .

The Friedman test applied on the Time dependent variable and the independent variables Method and Task revealed that the null hypothesis H_{n1} can be rejected (p -value = 0.000). The achieved results were further confirmed

by applying the Kendall's W Test. Therefore, the alternative hypothesis H_{a1} can be accepted. Let us note that the teams using our approach spent on the average 90% of the time to inspect the same program using Fagan's method.

To verify the null hypothesis H_{n2} we adopted the two-way ANOVA (see Table 5). We first observe that the interaction between *Method* and *Task* is not significant. There is very highly significant (p -value = 0.016) effect of the Task overall, while the Method factor is not significant (p -value = 0.097). We also applied the Friedman non-parametric test to confirm the results of the two-way ANOVA test. Let us note that the teams using the Fagan's method identified on the average 77% of the total number of the defects identified by applying our method.

Table 5. ANOVA table of similarity on the IdD dependent variable $R^2= 65.3$ (Adjusted $R^2 = 52.3$)

Source	Type III Sum of Squares	Df	Mean Square	F	p-value
Task	918.750	1	918.750	9.188	0.016
Method	352.083	1	352.083	3.521	0.097
Interact.	234.083	1	234.083	2.341	0.165
Error	800.000	8	100.000		
Total	23557.000	12			

The Friedman test applied on the TD variable and the selected independent variables revealed that the null hypothesis H_{n3} can be rejected (p -value = 0.000). The same result was obtained applying Kendall's W Test. Therefore, the alternative hypothesis H_{a1} can be accepted. Let us note that the teams using our method identify on the average more true defects (i.e., 39) than using the Fagan's method (i.e., 34).

The results achieved by applying the two-way ANOVA to verify the null hypothesis H_{n4} are summarized in Table 6. The interaction between *Method* and *Task* is significant. On the other hand, there is a significant (p -value = 0.024) effect of the Method overall, while the Task factor is not significant (p -value = 0.088). On the other hand, the Friedman non-parametric test revealed that the null hypothesis H_{n4} can be rejected (p -value = 0.018). The inspection teams that used the Fagan's method on the average identified less false positive defects.

Table 6. ANOVA table of the FP dependent variable $R^2= 67.7$ (Adjusted $R^2 = 55.6$)

Source	Type III Sum of Squares	Df	Mean Square	F	p-value
Task	44.083	1	44.083	3.779	0.088
Method	90.750	1	90.750	7.779	0.024
Interact.	60.750	1	60.750	5.207	0.052
Error	93.333	8	11.667		
Total	641.000	12			

Figure 5 visually summarizes the data collected from the survey questionnaire. In particular, the answers of the survey questionnaire corresponding to the method FA and NOFA are summarized by the boxplots on the left hand side and on the right hand side, respectively. The analyses of the collected data showed that the time to perform the inspection tasks was considered appropriate as well as their objectives (see boxplots of the questions Q1 and Q2). The distributions of Q3 and Q4 concerning the methods FA and NOFA present some differences. In particular, two and three outliers in the distributions of Q3 and Q4 of the NOFA methods are shown, respectively. On the other hand, the Q3 distribution of the FA method shows a single outlier. Despite these differences the medians of the observations are the same. Therefore, the subjects expressed on the average a positive judgment on the inspection task clarity and on the hard copy material provided as support to accomplish the experiment. Also, the checklist was considered clear and well structured (see boxplots of the questions Q5). Regarding the question Q6 the medians of the distributions of the methods FA and NOFA are different and the corresponding boxplots are skewed. Hence, the subjects that performed the task using the FA method on the average found the defect identification simpler than the subjects that used the tool. We further investigated the answers provided to the questions ranging Q1 to Q6 using the non parametric Mann Whitney U-test. This test revealed that the subjects did not perceive a significant difference when used the methods FA and NOFA were used. We also adopt the Mann Whitney U-test to analyze the correlation between the answers of the questions ranging Q1 to Q6 and inspection tasks. The test revealed that the subjects did not perceive a significant difference when performed the tasks T_1 and T_2 . For space reason details on the results achieved employing the Mann Whitney U-test are not reported. For scant relevance we do not also report details on the data analysis on the questions from Q7 to Q9 and from Q7 to Q10 of the survey questionnaire presented to the subjects that performed the inspection task using the method FA and NOFA, respectively.

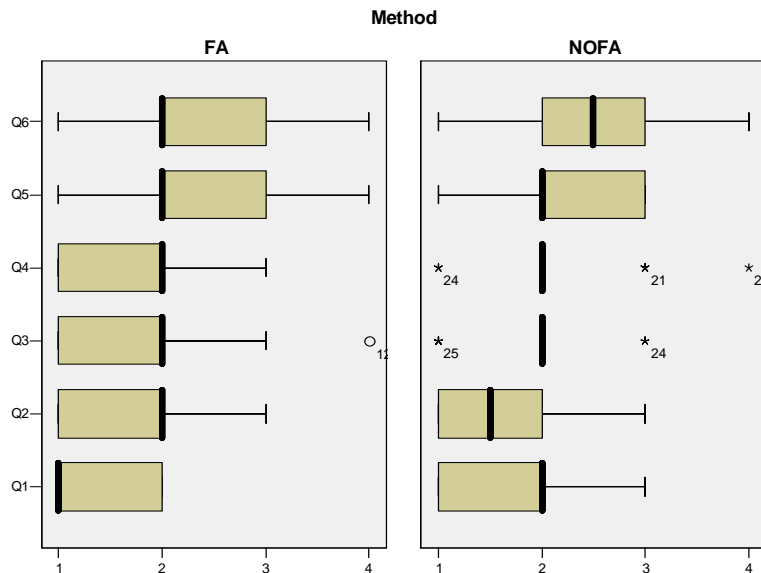


Figure 5. Boxplots of the survey questionnaire

4.7 Threats to validity

In this section we describe the threats to validity (i.e. *internal*, *construct*, *external*, and *conclusion* validity threats) that could affect the experiment.

The internal validity threat is relevant in studies that try to establish a causal relationship. In this case we aim at concluding that our method made a difference in the traditional Fagan's method to inspect source code. Generally, the key question is whether the observed differences can be attributed to the learning effect and not to other possible causes. The internal validity threats of this experiment are mitigated by the experiment design, since each group worked, over the two Labs, on different Tasks and with two different Methods (FA and NOFA). The survey also confirmed that the subjects found clear everything regarding the tasks of the experiment. Moreover, the subjects did not know exactly the hypothesis of the experiment although they applied the inspection methods on real inspection tasks. Finally, the subjects were not evaluated on their performance.

The construct validity threats (i.e. the interactions between different treatments) were mitigated by a proper design that allowed separating the analysis of the different factors and of their interactions. Moreover, depending on the treatment, the measurements of the dependent variables were performed either analyzing the log files produced by the tool or considering the times gathered by the experiment supervisors (one for each team). Moreover, true and false positive defects were manually identified by experts. Finally, the survey questionnaire was designed using standard ways and scales [5].

External validity refers to the approximate truth of conclusions involving generalizations within different contexts. External validity threats are always present when experiments are performed with students. Generally, last-year master students have a very good analysis, development and programming experience, and they are not far from junior industry programmers. Moreover, the selected inspection tasks are representative and are sufficiently complex to generalize the presented results. Nevertheless, replications in different contexts should be performed with different subjects in different contexts to confirm or contradict the results.

Conclusion validity concerns the issues that affect the ability of drawing a correct conclusion. The conclusion validity threats were 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 experiment with subjects belonging to this sample. Moreover, proper tests were performed to statistically reject null hypotheses. In cases where differences were present but not significant, this was explicitly mentioned. Non-parametric tests were used in place of parametric tests where there were not the conditions necessary to use parametric tests. Also, ANOVA results were confirmed by non-parametric test.

5. Conclusion and Lesson Learned

In this report we have described an inspection method to identify defects in software artifact using global distributed inspection. This process aims at minimizing synchronous collaboration among geographical dispersed

reviews. To this end conflicts on potential defects are resolved using an asynchronous inspection meeting before performing an optional traditional synchronous meeting. Indeed, synchronous inspection meetings are often not performed since it is generally used to remove conflicts on the defects that the inspection members have not solved.

The proposed method is based on an inspection process that has been implemented in a web based tool and assessed using a controlled experiment, which context was constituted of master students in Computer Science at the University of Salerno. The data analysis revealed that reviews adopting our process spent less time to inspect software artifact with respect to the traditional Fagan's method. Moreover, the inspection teams are able to identify more defects and true defects when our inspection process is adopted. Conversely, the number of false defect decrease when the Fagan's method is employed to perform the inspection tasks. Due to this concern we further investigated the defect logs of the inspection teams in order to obtain precision and recall values. 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. The further analysis of the inspection defect logs revealed that a better values of recall and worse values of precision are achieved when our method is adopted. It is worth noting that the recall value is always not known a-priori, while the precision can be improved manually analyzing the defects identified by the team members. Thus, in case our method is used, the inspection manager could filter out false defects and get more quickly inspection report log containing a larger number of true defects with the respect to the Fagan's method. Moreover, the experiment revealed that synchronous meetings are only required to discuss on unsolved conflicts, which are generally rare or absent at all.

The data analysis also provided a number of directions to improve the usefulness of the inspection tool. A first direction would be to add some features to further simplify the defect localization within the software artifact. A second direction should aim at adding new features to better support optional synchronous meetings. Finally, some usability problems that the subjects detected will be also addressed.

References

- [1] M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development", *IBM Systems Journal*, vol. 15, no. 3, 1976, pp. 182–211.
- [2] A. De Lucia, F. Fasano, R. Francese, and G. Tortora, "ADAMS: an Artefact-based Process Support System", *Proceedings of 16th International Conference on Software Engineering and Knowledge Engineering*, Banff, Alberta, Canada, 2004, pp. 31-36.
- [3] J.L. Devore, N. Farnum, *Applied Statistics for Engineers and Scientists*, Duxbury, 1999.
- [4] F Lanubile and T. Mallardo, "An Empirical Study of Web-Based Inspection Meeting", *Proc. of International Symposium on Empirical Software Engineering*, Rome, Italy, IEEE CS Press, 2003, pp. 244- 251
- [5] N. Oppenheim, "Questionnaire Design, Interviewing and Attitude Measurement", Pinter Publishers, 1992.
- [6] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering – An Introduction*, Kluwer, 2000.