CLEI ELECTRONIC JOURNAL, VOLUME 16, NUMBER 1, PAPER 9, APRIL 2013

Conducting Empirical Studies to Evaluate a Technique to Inspect Software Testing Artifacts

Jardelane Brito, Arilo Claudio Dias-Neto

Federal University of Amazonas (UFAM), Institute of Computing (IComp)
Manaus, Amazonas, Brazil, CEP 69077-000

{jardelane.brito, arilo}@icomp.ufam.edu.br

Abstract

Experimentation is becoming increasingly used in the Software Engineering field. Several methodologies adopt empirical studies as an instrument to support the software technologies' conception and maturation. This paper presents the application of a methodology based on the conduction of primary studies to develop a new checklist-based technique, named *TestCheck*, for inspection of software testing artifacts (test plan, cases, and procedures). Thus far, three empirical studies have been conducted (two feasibility studies and one observational study). The results of the first feasibility study indicate that *TestCheck* is more efficient and effective when detecting defects in software testing artifacts than an Ad-hoc approach. The results of the second feasibility study suggest an improvement in *TestCheck* regarding the defects detection and the reduction of the number of false positives per inspector. Finally, the results of the third study (observational study) evaluated positively the application of *TestCheck* by software engineers inspecting testing artifacts developed in an industrial environment. Thus, it would be qualified for the execution of the methodology's next steps (case studies in the software industry).

Keywords: Software Inspection, Software Testing, Checklist, Software Quality.

1 Introduction

The application of experimentation concepts in the area of software engineering (known as Empirical Software Engineering – ESE) goes back to the 70s and 80s [1] and has been growing significantly every year [2]. As a consequence of this development, the technical literature has provided research methodologies [3][4], ontologies and knowledge repositories [5][6], and environments to support experimental studies [7][8], among others. Thus, more and more experimentation has been used for the conception and evaluation of new software technologies.

In this context, this paper describes the application of concepts published in the ESE area for the experimental evaluation of a new checklist-based inspection technique for reviewing artifacts produced during the software testing process, called *TestCheck*. Despite the advances in Software Engineering regarding the utilization of methods and techniques to obtain quality systems, a piece of software still needs to be tested before it can be delivered to a client. Thus, software testing presents itself as one of the most effective means to ensure the quality of software, and it is also one of the most complex areas in software engineering, as well as one of the most studied [9][10].

In order to achieve the goals outlined by the software testing activities, it is necessary to follow a process that manages the tasks from their planning through to the analysis of their results. The testing activities, if carried out without an objective, planning or adequate techniques, can present unpleasant consequences to a software project, since the discovery of defects at an advanced stage, among other things, increases correction costs, leads to delays in the schedule and provokes customer dissatisfaction. However, merely following a process does not guarantee its quality. The quality of the tests is directly related to the possibility of revealing failures in the system. If the artifacts produced during the tests (e.g.

test plan, cases, and procedures) present defects, it is unlikely that the objective of providing quality tests will be achieved and the final result of the tests will not be satisfactory [11].

Thus, applying quality assurance techniques to the artifacts that are produced in the testing process may be a viable solution to guarantee the quality of tests in software projects. In this context, software inspection [12] is a quality assurance technique that has been established to detect defects at an early stage in the system's development, and it is applied to any software artifact. In [13], the authors present an approach based on a checklist for the inspection of test artifacts, called *TestCheck*. Its objective is to minimize and prevent defects occurring in the artifacts that are produced in the testing process. This paper presents the procedures followed for the experimental evaluation of *TestCheck* and their results. The studies we carried out were based on the study packages for the evaluation of checklist-based inspection techniques published in [14] and [15].

The application of ESE concepts generally occurs in different Software Engineering subareas, but it has a special scenario in the Software Inspection subarea, where we can see a large volume of evidence coming from experimental studies, as can be seen in a systematic literature review published in 2005 [16]. This scenario results in study packages that can be replicated by other researchers in order to provide an experimental evaluation of inspection techniques. For the evaluation of *TestCheck* we will apply a scientific methodology based on different experimental studies [3]. According to the methodology, the first study to be carried out for the evaluation of a new technology is a feasibility study, which should be repeated until the feasibility of the new proposed technique in relation to its goal is known. Then, an observation study should be carried out in order to evaluate the behavior of software engineers when they use the new technique. Continuing the methodology, case studies in a real lifecycle in the industry need to be carried out to evaluate the proposed technology from its definition right through to its transfer to the industrial environment. Thus far, in our research, the first two stages of this methodology have been completed in three experimental studies (two feasibility studies and one observation study). This paper describes the evolution process of the *TestCheck* technique guided by the results obtained in these studies.

This paper is an extended version of the paper published in [17], where new details, including a new version, regarding the proposed approach to inspect testing artifacts and results from the first two evaluation (feasibility) studies are described. Moreover, a third (observational) study performed with practitioners is described. The results indicate efficiency and effectiveness of the proposed techniques to detect defects in testing artifacts in controlled and industrial environments.

Besides this Introduction, this comprises more 5 sections. In Section 2 we present the checklist-based technique to support the inspection of software testing artifacts (*TestCheck*), which is the main instrument of evaluation in the studies carried out and described in this paper. In Section 3 we present the planning, design, and results of the first experimental study carried out, with the objective of evaluating the feasibility of *TestCheck* when compared to an Ad-hoc technique. In Section 4 we present the second feasibility study, with the objective of evaluating the evolution of *TestCheck* in continuation from the first study, analyzing whether it satisfies the necessary criteria for the next step in the methodology (evaluation in industry). In Section 5 we present the third (observational) study, which evaluated *TestCheck*'s behavior in an industrial environment of software development. Finally, in Section 6, we present the conclusions, limitations and next steps to be taken to follow through this research.

2 *TestCheck* – Checklist-Based Technique to Support the Inspection of Software Testing Artifacts

The *TestCheck* approach has been developed with the aim of complying with the technical features required in an inspection approach, as described in [18]. Acting on these features, the following tasks were carried out, as detailed in the subsections that follow:

- Definition of the technique used to detect the defects in the inspected artifacts;
- Determination of the software artifact that needs to be evaluated by the approach;
- Specification of the evaluation items that need to comprise the proposed approach.

2.1 Defect Detection Technique to be adopted

Within the 3 categories of defect detection techniques that can be applied in an inspection approach (Ad-hoc, checklist-based, or reading technique) it is not possible to instantiate the use of an Ad-hoc technique, as such a technique does not use any support tools and depends exclusively on the experience and previous knowledge of the inspector, without providing any additional support. When analyzing the feasibility of creating a reading technique, we perceived that this requires the document to be inspected to follow a standard format to guide the reading process, and this scenario cannot as yet be attained in the Software Testing area, as there is no standardization for representing test artifacts. Despite the existence of the IEEE-Std 829 standard [19] as a guideline for the documentation of test artifacts, in practice it is little used, as can be seen in [20]. Furthermore, this standard only describes templates that can (must) still be customized in the process of applying them to software projects.

Finally, analyzing the technique based on checklist, we can see that it is an evolution from the Ad-hoc technique and, as opposed to the reading techniques, it does not require a standardization of the artifacts' representation. When reviewing software documents using a checklist, the inspector is supplied with a set of questions that help him identify defects by indicating in which part of the artifact to be evaluated he needs to look for them. Therefore, due to the features of the Software Testing area, the defects detection technique based on checklists has features that make it feasible to be applied in an inspection of test artifacts. It was this technique, therefore, that we selected to detect defects in this study.

2.2 Software Testing Artifacts to be inspected

To develop *TestCheck*, we selected the artifacts that would be evaluated by the proposed technique. We looked for testing artifact templates that would be candidates to be reviewed in different sources, such as software development methodologies/frameworks, including predictive (process-based, like RUP – Rational Unified Process) and adaptive (agile) methods. In this analysis, we observed that the templates suggested by these methods are originated from the same source: IEEE-Std 829 [19]. In general, these methods suggest test documentation templates that consist of a sub-set of templates proposed by IEEE-Std 829. This standard proposes a set of eight templates that can be applied to different development methodologies, such as predictive and adaptive methods. It was published initially in 1983, and reviewed in 1998 and 2008. The current version (published in 2008 [19]) describes templates for eight different testing artifacts: Master Test Plan (MTP), Level Test Plan (LTP), Level Test Design (LTD), Level Test Case (LTC), Level Test Procedure (LTPr), Level Test Log (LTL), Anomaly Report (AR), Level Interim Test Status Report (LITSR), Level Test Report (LTR), and Master Test Report (MTR).

The selection of which testing artifacts would be reviewed by the proposed technique was based, initially, on the **test planning stage**. We decided to evaluate, in the first moment, only artifacts produced in this stage as an attempt to minimize and prevent the incidence of defects during the testing execution stage, which could make the tests' execution in a software project unfeasible. Thus, this decision limited the choice to the artifacts MPT, LTP, LTD, LTC, and LTPr.

In a second decision, we decided to select testing artifacts included in the versions of IEEE-Std 829 published in 1998 and 2008, because the old version is still applied intensely in software projects [20]. Therefore, templates included in only one of these versions were not selected. This decision limited the choice to the artifacts LTP, LTD, LTC, and LTPr.

Finally, in the third decision, we decided to review the testing artifacts most used in the software industry. In [20] the authors presented the results of a survey conducted with practitioners regarding the preferable IEEE-Std 829 test documentation in an international perspective. The authors reported that the following three testing artifacts are the most applied in real software projects in the testing planning phase: test plan (LTP), test cases (LTC), and test procedures (LTPr). Therefore, they were selected to be reviewed by the proposed technique. The remaining testing artifacts can be considered in the evolution of this research. A very brief description of the selected artifacts is as follows:

- **Level Test Plan** presents the planning to execute tests, including its scope, goals, tasks allocation, resources distribution, schedule, and risks associated to testing [19].
- **Level Test Case** defines inputs (test data), expected results/behaviors, actions and conditions to execute the test case [19].
- **Level Test Procedure** specifies the steps to execute a set of test cases, their pre-conditions and special requirements to be executed [19].

The selected artifacts can be applied to tests at different levels (e.g. unit, integration, system and acceptance). Thus, the checklists that compose the *TestCheck* approach were elaborated in such a way as to apply them to these different levels, consisting of items that inquire into the artifacts' structure and content without being directed toward a specific level.

Figure 1 graphically represents an evolution of the sub-process of test planning described in [21]. It indicates the artifacts to be inspected and at what moment (represented by a diamond) of the testing process this occurs.

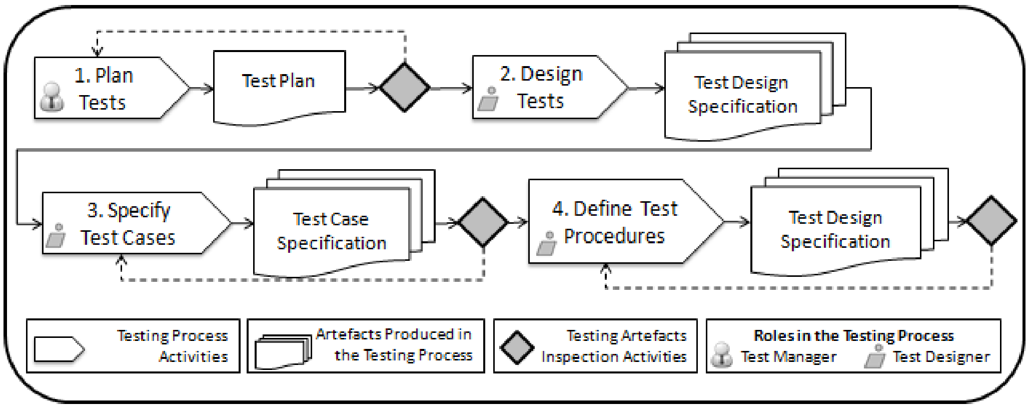

1. Plan Tests
Test Plan
2. Design Tests
Test Design Specification
3. Specify Test Cases
Test Case Specification
4. Define Test Procedures
Test Design Specification
Testing Process Activities
Artefacts Produced in the Testing Process
Testing Artefacts Inspection Activities
Roles in the Testing Process
Test Manager
Test Designer

Figure 1: Evolution of the Test Planning Sub-process including Inspection Activities.

2.3 Checklists for the Inspection of Software Testing Artifacts

Three checklists were developed, one for each selected testing artifact: Test Plan, Test Case, and Test Procedure.

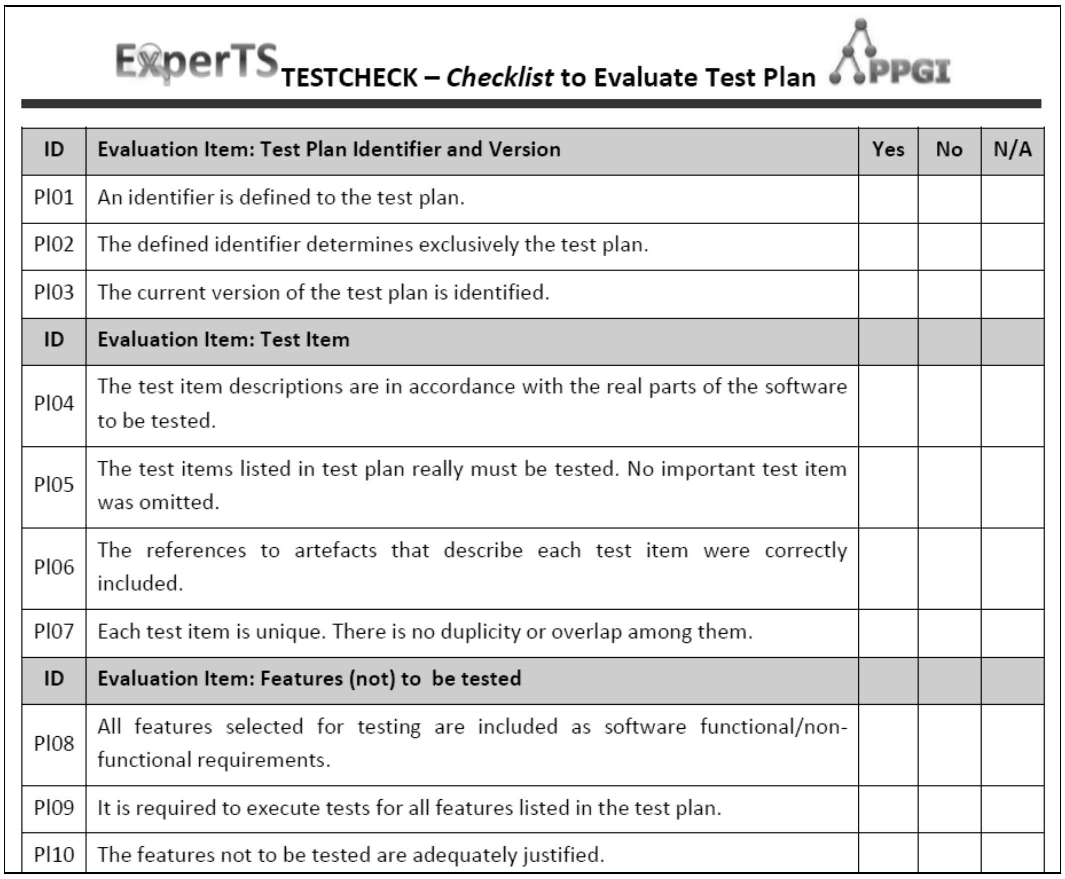
- **Checklist to Inspect Testing Plans**

Starting with the testing plan, this artifact needs to contain all the information that is relevant to the planning of testing activities, as any defects existing in this document can undermine the final result of the tests. The *checklist* developed to check this artifact has 13 items (listed below) to be evaluated by 38 questions (see Table 1).

Table 1: Evaluation Items Comprising the Checklist for Inspection of Test Plans

Evaluation Items	Description	# of Questions
1. Test Plan Identifier and Version	Duplications/Omissions of test plan identifier	3
2. Test Item	Completeness of the description of the software items to be tested	4
3. Features (not) to be tested	Description of the quality features to be tested (or not)	3
4. Type of Testing	Adequacy of the specified test types (levels)	2
5. Testing Approach	Description of the testing approaches and support tools	2
6. Item Pass/Fail Criteria	Objectivity and clarity of the pass/fail criteria defined for the test items	3
7. Suspension Criteria / Resumption Requirements	Objectivity and clarity of the criteria defined for the suspension and resumption of tests	1
8. Testing Tasks and Deliverables	Completeness and correctness of the tasks listed for the test	5
9. Responsibilities / Team Skills	Description and profile of the test team	3
10. Environmental Needs	Description/Adequacy of the physical features and the software/hardware needed to carry out the tests	2
11. Staffing and Training Needs	Description of the necessary training and the compatibility with the tests to be carried out	3
12. Testing Schedule	Correctness of the testing schedules defined for each test activity	2
13. Risks and Contingencies	Completeness of assessing the possible risks associated with the tests in a project, as well as their mitigation and contingency plans	5

Due to the limited space, we cannot present the full checklist proposed to inspect test plans. An extract of this checklist is shown in Figure 2.



ExperTS TESTCHECK – *Checklist* to Evaluate Test Plan

ID	Evaluation Item: Test Plan Identifier and Version	Yes	No	N/A
PI01	An identifier is defined to the test plan.			
PI02	The defined identifier determines exclusively the test plan.			
PI03	The current version of the test plan is identified.			
ID	**Evaluation Item: Test Item**			
PI04	The test item descriptions are in accordance with the real parts of the software to be tested.			
PI05	The test items listed in test plan really must be tested. No important test item was omitted.			
PI06	The references to artefacts that describe each test item were correctly included.			
PI07	Each test item is unique. There is no duplicity or overlap among them.			
ID	**Evaluation Item: Features (not) to be tested**			
PI08	All features selected for testing are included as software functional/non-functional requirements.			
PI09	It is required to execute tests for all features listed in the test plan.			
PI10	The features not to be tested are adequately justified.			

Figure 2: Partial Extract of the Checklist to Evaluate the Test Plan [13].

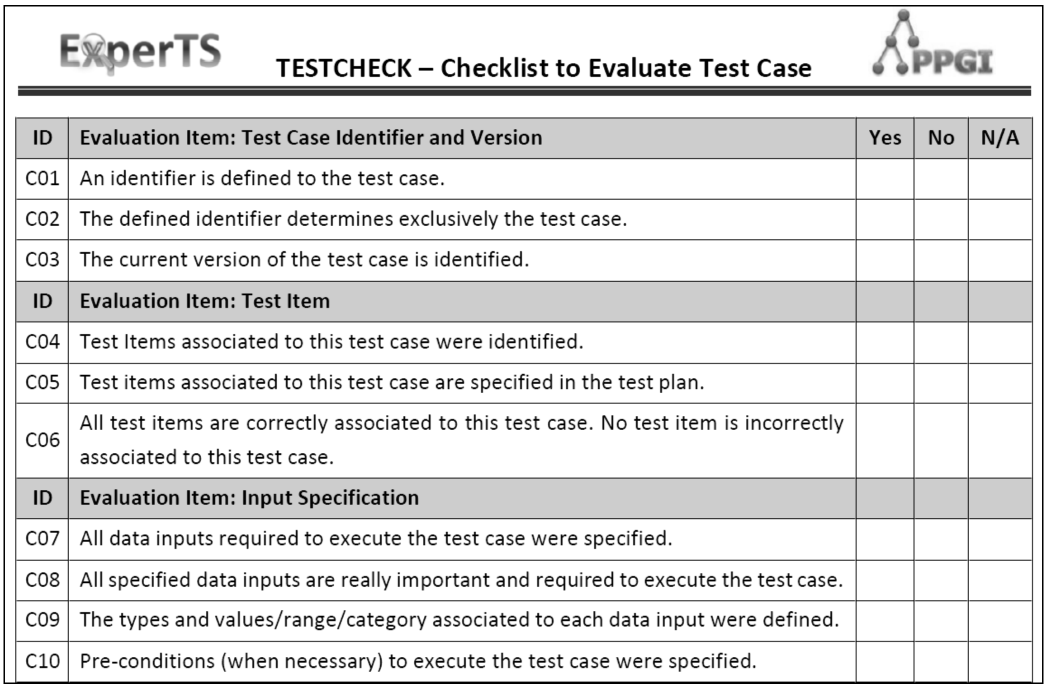
- **Checklist to Inspect Test Cases**

The checklist to inspect specification of test cases evaluates 6 items (listed below) that comprise this artifact, detailed in 19 questions, as shown in Table 2.

Table 2: Evaluation Items Comprising the Checklist for Inspection of Test Cases

Evaluation Items	Description	# of Questions
1. Test Case Identifier and Version	Duplications/Omissions of test case identifier	3
2. Test Items	Completeness of the description of software items to be tested	3
3. Input Specifications	Correctness and completeness of the described entries for the test case, their values, types and pre-conditions	4
4. Output Specifications	Correctness and completeness of the results or expected behaviors after the tests	4
5. Environmental Needs	Description of the physical features and the software/hardware necessary to carry out a test case	2
6. Inter-case Dependency	Correctness of the dependency relationship between the project's test cases	3

Again, due to the limited space, we will not also present the full checklist proposed to inspect test cases. An extract of this checklist is shown in Figure 3.



TESTCHECK – Checklist to Evaluate Test Case

ID	Evaluation Item: Test Case Identifier and Version	Yes	No	N/A
C01	An identifier is defined to the test case.			
C02	The defined identifier determines exclusively the test case.			
C03	The current version of the test case is identified.			
ID	Evaluation Item: Test Item			
C04	Test Items associated to this test case were identified.			
C05	Test items associated to this test case are specified in the test plan.			
C06	All test items are correctly associated to this test case. No test item is incorrectly associated to this test case.			
ID	Evaluation Item: Input Specification			
C07	All data inputs required to execute the test case were specified.			
C08	All specified data inputs are really important and required to execute the test case.			
C09	The types and values/range/category associated to each data input were defined.			
C10	Pre-conditions (when necessary) to execute the test case were specified.			

Figure 3: Extract of checklist to support inspection of test cases [13].

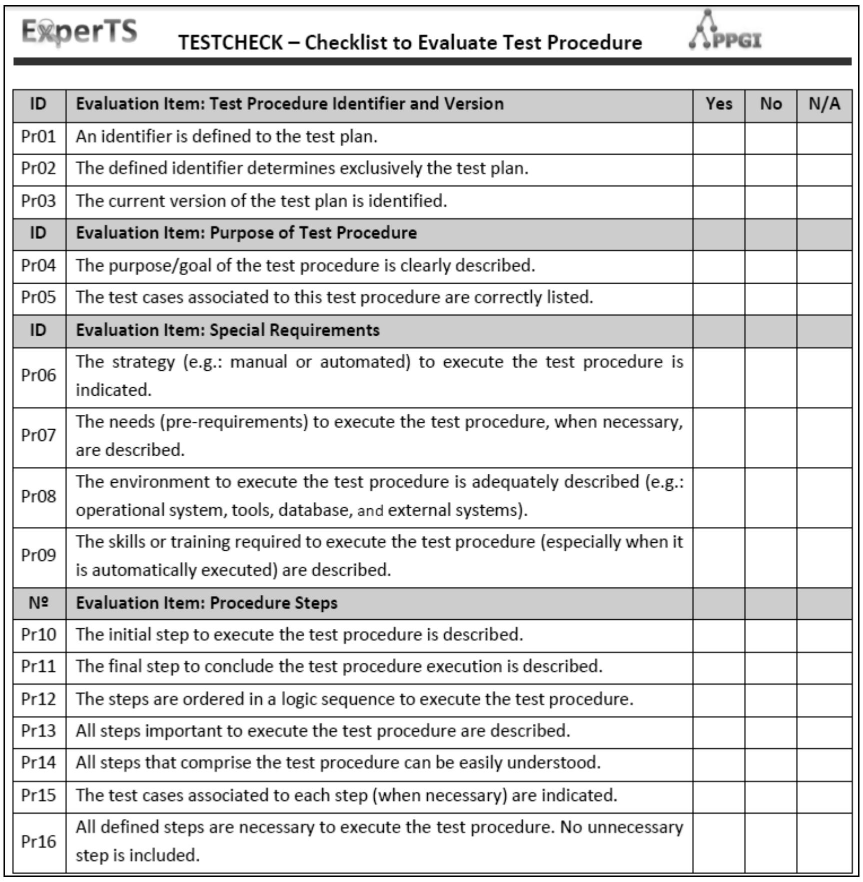
- **Checklist to Inspect Test Procedures**

The checklist for inspecting the specification of test procedures evaluates 4 items (listed below), detailed in 16 questions, as shown in Table 3.

Table 3: Evaluation Items Comprising the Checklist for Inspection of Test Procedures

Evaluation Items	Description	# of Questions
1. Test Procedure Identifier and Version	Duplicates/Omissions of test procedure identifier	3
2. Purpose of Test Procedure	Description of the test procedure's purpose/objectives in relation to its scope	2
3. Special Requirements	Description of the necessary requirements for the correct execution of the test procedure	4
4. Procedure Steps	Completeness and correctness of the steps described for a test procedure and their sequencing	7

As it is smaller than the other two, in Figure 4 we present the full checklist proposed to inspect specifications of test procedures, including all evaluation items and questions.



ExperTS TESTCHECK – Checklist to Evaluate Test Procedure PPGI

ID	Evaluation Item: Test Procedure Identifier and Version	Yes	No	N/A
Pr01	An identifier is defined to the test plan.			
Pr02	The defined identifier determines exclusively the test plan.			
Pr03	The current version of the test plan is identified.			
ID	Evaluation Item: Purpose of Test Procedure			
Pr04	The purpose/goal of the test procedure is clearly described.			
Pr05	The test cases associated to this test procedure are correctly listed.			
ID	Evaluation Item: Special Requirements			
Pr06	The strategy (e.g.: manual or automated) to execute the test procedure is indicated.			
Pr07	The needs (pre-requirements) to execute the test procedure, when necessary, are described.			
Pr08	The environment to execute the test procedure is adequately described (e.g.: operational system, tools, database, and external systems).			
Pr09	The skills or training required to execute the test procedure (especially when it is automatically executed) are described.			
Nº	Evaluation Item: Procedure Steps			
Pr10	The initial step to execute the test procedure is described.			
Pr11	The final step to conclude the test procedure execution is described.			
Pr12	The steps are ordered in a logic sequence to execute the test procedure.			
Pr13	All steps important to execute the test procedure are described.			
Pr14	All steps that comprise the test procedure can be easily understood.			
Pr15	The test cases associated to each step (when necessary) are indicated.			
Pr16	All defined steps are necessary to execute the test procedure. No unnecessary step is included.			

Figure 4: Extract of checklist to support inspection of test procedures [13].

Currently, *TestCheck* is in its third version. The complete version of the checklists can be found at http://www.icomp.ufam.edu.br/experts/testcheck-en.zip. More details about *TestCheck*'s conception process, the content of its checklist and previous versions can be obtained in [13].

Having concluded the technique's conception phase, in the next section we will present the first experimental study carried out to evaluate the proposed technique.

3 Feasibility Study 1

According to the scientific method for evaluation of software technologies based on experimental studies defined in [3], the first experimental study that has to be carried out to evaluate a new technology is a feasibility study, which aims to assess whether the new technology is in fact feasible and whether the time spent using it is well employed.

The feasibility of the technique was assessed in relation to its effectiveness (number of defects in relation to the total number of discrepancies encountered) and efficiency (number of defects identified per unit of time) in identifying defects.

3.1 Planning and Execution of the Feasibility Study

The premise for an inspection approach is that it should guide inspectors in the detection of defects in artifacts in a reduced space of time and generate a low number of false positives (discrepancies reported as defect by the inspector, but that are not a real defect). To evaluate this premise, we carried out a comparison between the defects detected by the proposed approach (*TestCheck*) and the results obtained by an *Ad-hoc* approach, as we did not find any other approach in the technical literature based on a checklist or reading technique with a similar objective to *TestCheck*. In addition to evaluating this premise, this first study also aims to provide a baseline for the *TestCheck* approach, allowing for future comparisons in the course of its evolution. Comparing it initially with the *Ad-hoc* approach contributes to this objective. Thus, we defined the Null Hypothesis (and the alternatives ones) for this study, as described below:

- **Null Hypothesis (H0):** There is no difference between the results obtained by the *TestCheck* and the *Ad-hoc* approaches as regards the detection of defects (effectiveness and efficiency) in test artifacts.
- **Alternative 1 Hypothesis (H1):** *TestCheck* obtains a better result in regard to the detection of defects (effectiveness and efficiency) than the *Ad-hoc* approach.
- **Alternative 2 Hypothesis (H2):** The *Ad-hoc* approach obtains a better result in regard to the detection of defects (effectiveness and efficiency) than *TestCheck*.

This first study was carried out in the Amazonas state in Brazil and 12 undergraduate and graduate students of a class about software quality at the Federal University of Amazonas (UFAM) took part in it. In order to assess each student's knowledge and experience, they were asked to answer a Profile Questionnaire, which evidenced the participants' low level of technical knowledge regarding the application of inspection techniques in the industry, as most of their previous experience had been only in an academic environment. Each participant carried out the inspection off-line and did not know who else participated in the study in order to avoid the exchange of information between participants.

In the study design, the participants were divided into 2 groups (A and B) based on the information in their profile questionnaires so as to mix participants with different levels of experience. Test artifacts of two distinct projects (called *Industrial* and *Academic*) were used during the experiment. The test artifacts of the *Industrial* project originated from a real project being developed in industry for managing users, their profiles and functionalities in a web system for the management of research projects. The test artifacts of the *Academic* project originated from an assignment given in a subject about software testing taught in 2010, in which students (not participants of the study) designed tests for a library's control system. At no time did the researchers insert defects into these artifacts. It was the versions created by their authors that were evaluated. The participants carried out the study in 3 different phases, as described in Table 4.

Table 4: Procedure followed in the study

Group	PHASE 1 *Ad-hoc* Inspection	PHASE 2 Training	PHASE 3 Inspection with *TestCheck*
Group A	Industrial Project	Training about the *TestCheck* technique	Academic Project
Group B	Academic Project		Industrial Project

In phase 1 everybody applied an *Ad-hoc* technique. At this point each student received documents pertaining to one of the projects, according to which group they belonged to, containing 1 test plan, 5 test cases, 5 test procedures, 1 list of discrepancies to be filled out by the participants during the inspection, and 1 requirements specification document for any clarification about the test artifacts. The requirements specification document was not inspected.

After phase 1, the students received training on how to employ the second technique, *TestCheck*. In this phase, they were given an explanation of the items that comprise the checklists and of the new template of the list of discrepancies to be filled out. After the training, phase 3 was started where artifacts of a new software project were distributed to the students, inverting the artifacts distributed between the groups in phase 1.

After the conclusion of phase 3, the data obtained were organized. The individual lists of discrepancies were integrated into one single list. Then, each discrepancy was evaluated and classified either as a defect or a false positive. This analysis was performed by the researcher involved and a specialist in software testing. In the case of doubt, a third researcher was consulted.

3.2 Results of the Feasibility Study 1

In order to assess the feasibility of the proposed inspection technique, we analyzed the variables: defects detected, occurrence of false positives, and the time each inspector needed for the inspection, as shown in Table 5. Two other variables were also analyzed, derived from the data obtained from the simple variables cited. They are:

- **Effectiveness:** the percentage of defects detected in relation to the total of reported discrepancies.
- **Efficiency:** number of defects detected per minute.

Table 5: Summary of the data obtained in the feasibility study 1.

Participants	Inspection Technique	Project	Defects	False Positives	Time (minutes)	Effectiveness (defects/discr.)	Efficiency (defects / min)
P01	*Ad-hoc*	Academic	0	15	210	0.00%	0.00
P02		Academic	9	6	285	60.00%	0.03
P03		Academic	2	15	76	11.76%	0.03
P04		Academic	0	4	42	0.00%	0.00
P05		Academic	21	3	120	87.50%	0.18
P06		Academic	6	22	180	21.43%	0.03
P07		Academic	8	11	100	42.11%	0.08
P08		Industrial	28	0	75	100.00%	0.37
P09		Industrial	4	6	120	40.00%	0.03
P10		Industrial	28	12	121	70.00%	0.23
P11		Industrial	24	4	100	85.71%	0.24
P12		Industrial	6	16	143	27.27%	0.04
P01	*TestCheck*	Industrial	40	7	190	85.11%	0.21
P02		Industrial	32	10	220	76.19%	0.15
P03		Industrial	30	10	34	75.00%	0.88
P04		Industrial	43	9	135	82.69%	0.32
P05		Industrial	28	2	127	93.33%	0.22
P06		Industrial	7	6	240	53.85%	0.03
P07		Industrial	36	17	180	67.92%	0.20
P08		Academic	19	8	155	70.37%	0.12
P09		Academic	36	12	85	75.00%	0.42
P10		Academic	57	27	122	67.86%	0.47
P11		Academic	84	20	180	80.77%	0.47
P12		Academic	58	22	275	72.50%	0.21

After that, the averages for the collected variables were analyzed, as shown in Table 6.

Table 6: Average of the defects, time, effectiveness and efficiency.

Factors		Defects	False Positives	Time (minutes)	Effectiveness	Efficiency
Techniques	*Ad-hoc*	11.33	9.5	131.0	45.48%	0.11
	TestCheck	39.17	12.5	161.9	75.05%	0.31
Projects	Industrial	25.50	8.0	131.0	71.42%	0.24
	Academic	25.00	13.5	138.5	49.11%	0.17

Starting with the analysis of the total number of defects detected, we see that the *TestCheck* technique found, on average per participant, more defects than the *Ad-hoc* technique (39.17 x 11.33 defects), while the analysis of the total number of defects per software project showed, on average, a similar results (25.0 x 25.5 defects). However, we notice that *TestCheck* takes, on average, more time to carry out the inspection than the *Ad-hoc* technique (161.9 x 131.0 minutes). This result was expected, as *TestCheck* provides additional instruments for the detection of defects during an inspection (checklists). Analyzing the time spent per software project, we see that, on average, the results of the two are similar (131.0 x 138.5 minutes).

Another variable considered during the analysis was the number of false positives each technique generated, which in this first study did not show such a positive result for the *TestCheck* technique, having generated on average 12.5 false positives per inspector, whereas the *Ad-hoc* technique generated 9.5. This result suggested a possible point for the technique's evolution in a new version. Analyzing the results per software project, one also notices a discrepancy between the two, with one of the projects (Academic) obtaining an average of 8.0 false positives, against 13.5 of the other project (Industrial), suggesting that the software project used in the study could be a factor that influences the amount of false positives generated during an inspection.

In regard to the variables' effectiveness and efficiency, the *TestCheck* technique obtained positive results for both when compared to the *Ad-hoc* technique. The average in relation to effectiveness was 75% for *TestCheck*, against 45% for the *Ad-hoc* technique. The average regarding efficiency was 0.31 defects/minute for *TestCheck*, against 0.11 defects/minute for the *Ad-hoc* technique. Analyzing the results of the two variables per software project, we can see that the average of one project is disproportional to the other in regard to the variable effectiveness (71% x 49%), but fairly similar for the variable efficiency

(0.24 x 0.17 defects/minute).

However, despite this preliminary result, it was necessary to apply some statistical tests to the obtained data in order to evaluate the existence of any statistically significant differences between them. For this, we first carried out an outlier analysis to check if there are any points substantially diverging from the values obtained. The results show that there were no outliers between participants. After that an Analysis of Variance (ANOVA) was applied with a level of accuracy of 95% (error rate α = 0.05). All the variables listed in Table 6 were analyzed considering the two factors evaluated in this study: the inspection technique applied (*Ad-hoc* or *TestCheck*) and the software project inspected (*Industrial* and *Academic*). The results obtained are described in Table 7.

Table 7. Summary of the Analysis of Variance (ANOVA) applied to Feasibility Study 1.

Variable	**Analysis per Technique**		**Analysis per Project**	
	p-value	**Statistical Difference?**	***p-value***	**Statistical Difference?**
Defects	0,0003	**YES**	0,9555	NO
False Positives	0,3078	NO	0,0539	NO
Time	0,2697	NO	0,6699	NO
Effectiveness	0,0101	**YES**	0,0616	NO
Efficiency	0,0120	**YES**	0,3951	NO

Analyzing the technique that was applied, we can observe that there is a statistical difference (since the *p-value* obtained is inferior to the error rate adopted in the study, α = 0.05) between the *TestCheck* and *Ad-hoc* techniques for the variables *Number of defects*, *Effectiveness* and *Efficiency*, and the *TestCheck* technique always would be the factor influencing positively on the result. This indicates that *TestCheck* detects a higher number of defects in the course of test artifacts inspections in absolute value, as well as a higher number of defects in relation to the total number of discrepancies reported by the inspectors (effectiveness), and also in relation to the number of defects detected per unit of time spent on the inspection (efficiency).

Analyzing the variables *False positives* and *Time*, the data indicates that there would be no statistical difference between the techniques. This result was considered to be positive in regard to evaluating the feasibility of the *TestCheck* technique, as this technique introduces new elements (checklists) which could turn into factors that make the inspection of the software testing artifacts more difficult. However, this did not happen with these variables.

Evaluating the results obtained for each software project, one notices that the projects used in the study did not influence the results. All the *p-values* for all the variables (Table 7) were above the error rate (α = 0.05) defined in the study, thus indicating that there would be no statistical difference between the projects employed.

Therefore, the null hypothesis was rejected, as the results indicated that *TestCheck* makes a positive contribution to the detection of defects when compared to an *Ad-hoc* approach.

3.3 Evolution of the Checklists

Despite the good performance of *TestCheck* in this first evaluation study, we know that this technique is still in a phase of development, and so it is important to pay attention to points that can be improved upon. One of the points to be improved was the number of **false positives** generated when applying *TestCheck*, which amounted to 12.5 per participant and were thought to be high by the researchers, considering that the rate of defects detected by the technique was 39.16 per participant. Thus, for every 3 defects, approximately, 1 false positive would be generated by *TestCheck*. The objective for evolution, therefore, is to reduce the rate of false-positives encountered in the first study and maintain the high rate of defects encountered when using *TestCheck* as compared to the *Ad-hoc* technique.

For this reason, as a more detailed analysis of *TestCheck*'s performance, an individual analysis was carried out for each question that comprises the checklists that constitute the technique. The objective was to identify questions that can be developed and improved on for having had a low performance during the study. For this analysis we investigated the relation between the number of false positives that the question generated and the number of defects it detected. All the questions of the three checklists that constitute *TestCheck* that had a **defect rate per false positive below or equal to 3.0** (this being a value obtained in the general scenario, as cited previously) would be a potential candidate to be evolved for the next version of the technique. This analysis is described in Table 8, where the questions that did not attain this standard are highlighted in gray. The items that did not generate a single false positive were not included in the analysis and are not shown in this table so as to simplify the description.

Table 8. Defects and false positives per *TestCheck* question (version 1.0),

Test Plan			***Test Case***			***Test Procedure***		
Question	Defects	False Positives	Question	Defects	False Positives	Question	Defects	False Positives
Pl02	**0**	**3**	C02	1	1	Pr01	3	1
Pl04	1	2	C06	44	6	Pr02	2	1
Pl05	1	3	C07	11	1	Pr04	4	3
Pl07	2	2	C08	1	2	**Pr07**	**0**	**40**
Pl08	**0**	**3**	**C10**	**42**	**18**	Pr08	3	1
Pl09	**0**	**3**	C11	1	1	Pr12	0	1
Pl11	1	2	C12	5	1	Pr13	2	2
Pl13	1	2	C13	44	1	Pr14	3	4
Pl14	0	1	C15	2	5	Pr15	3	2
Pl19	4	1	C19	10	2	Pr16	0	2
Pl20	0	1						
Pl21	1	1						
Pl22	1	1						
Pl26	1	2						
Pl30	1	2						
Pl31	0	1						
Pl32	0	2						
Pl33	6	1						

A total of 31 questions (out of the 75 questions that constituted the 3 checklists in version 1.0) were selected as candidates for evolution. They were revised and rewritten (not necessarily all of them) to create a new version of *TestCheck*. Some questions required more attention, as is the case with questions Pl02, Pl08 and Pl09 of the test plan, C10 in the test case, and Pr07 in the test procedure (all boldfaced in Table 8), which had a high rate of false positives. Several factors may have had an influence on this result, one of them being the question's wording, which could have led inspectors to different interpretations.

3.4 Threats to the Validity

The threats to the viability of this study will be described here according to their type:

- **Conclusion Validity:** classification of each discrepancy as a defect by the experimentalists and heterogeneity of the participants, who were undergraduate and graduate students of Software Engineering.
- **Internal Validity:** the sample of participants was not random, as it depended on the availability of students attending a class conducted by the researchers, and there may have been plagiarism, which would be the potential exchange of information between the participants about the study's tasks.
- **Construction Validity:** mono-operation bias, which would be the uncertainty of whether the study's results were arrived at due to the application of the *TestCheck* or rather because of the "ease" of detecting defects in the document used in the selected projects, as well as the probability of participants "touching up" their data due to personal expectations about the evaluation of their results.
- **External Validity:** interaction between selection and treatment, as academic students may not be a substitute for inspectors coming from the industry; interaction between environment and treatment, since the academic environment may not be able to entirely simulate the conditions of an industrial environment.

This process of evolution generated a new *TestCheck* version. Having concluded this phase of the technique's evolution, we will now present the second study we carried out and its results.

4 Feasibility Study 2

In order to evaluate the evolution of *TestCheck* it is necessary to assess the results obtained in a second feasibility study to analyze if the implemented changes lead to improvements in the proposed technique. The second study followed the same framework as the first one, in which the objective was to evaluate the application of *TestCheck* compared to an *Ad-hoc* technique, and it was made up of two inspection rounds using the *Ad-hoc* and *TestCheck* techniques and it was carried out in December 2011 in an academic environment.

4.1 Planning and Execution of Feasibility Study 2

Initially, 19 undergraduate students of a software engineering class at the Federal University of Amazonas (UFAM) participated in the study; however, only 10 students concluded it. As in the first study, each student's knowledge and experience was evaluated by means of a profile questionnaire, and the inspection was again carried out off-line to avoid the exchange of information between participants.

In the study design, the participants were once more divided into two groups (A and B) based on their questionnaires, thus mixing participants with different levels of experience. The same software projects (*Industrial* and *Academic*) and their test artifacts were used during the experiment (1 test plan, 5 test cases, 5 test procedures, 1 requirements specification document for any clarification about the test artifacts). The same procedure was followed for carrying out the study and consolidating the final list of discrepancies, as shown in Table 4. After carrying out the study, the analysis of the data was initiated.

4.2 Result of Feasibility Study 2

To evaluate the *TestCheck* technique, we collected the data for the same variables as in study 1, which we described in Section 3. Table 9 shows a summary of the data obtained for each participant.

Working with this data, we analyzed the averages obtained for each variable, analyzing the defect detection techniques evaluated and the software projects used. After that, the statistical method ANOVA was applied to the data. As the main focus of this second study was to evaluate the evolution of *TestCheck* in relation to the number of false positives encountered by each inspector and ascertain the maintenance of the high defect detection rate, and also due to the limited space, the results obtained in the statistical tests of the variables will not be shown in this article. They can, however, be calculated from the data presented in Table 9.

Table 9. Summary of the data obtained in feasibility study 2.

Participants	*Inspection Technique*	*Software Project*	*Defect*	*False Positive*	*Time*	*Effectiveness (defects / disc)*	*Efficiency (defects / tempo)*
P01	*Ad-hoc*	Industrial	38	4	75	90.48%	0.51
P02		Industrial	4	2	40	66.67%	0.10
P03		Industrial	6	1	95	85.71%	0.06
P04		Industrial	0	5	80	0.00%	0.51
P05		Academic	9	1	130	90.00%	0.10
P06		Academic	5	1	58	83.33%	0.06
P07		Academic	12	1	60	92.31%	0.51
P08		Academic	8	2	60	80.00%	0.10
P09		Academic	17	2	176	89.47%	0.06
P10		Academic	20	10	458	66.67%	0.51
P01	*TestCheck*	Academic	56	8	330	87.50%	0.10
P02		Academic	22	2	180	91.67%	0.06
P03		Academic	40	8	230	83.33%	0.51
P04		Academic	22	1	240	95.65%	0.10
P05		Industrial	14	5	60	73.68%	0.06
P06		Industrial	8	2	134	80.00%	0.51
P07		Industrial	29	6	342	82.86%	0.10
P08		Industrial	20	4	200	83.33%	0.06
P09		Industrial	76	11	370	87.36%	0.51
P10		Industrial	63	2	360	96.92%	0.10

Analyzing the evolution of *TestCheck,* Table 10 presents a comparison between the inspection techniques focusing on the number of defects detected and the false positives generated per participant. The rate of false positives generated was 4.9 per inspector, a better result as the one obtained in the previous study, in which the rate was 12.5. This means that, when looking only at the average obtained, the objective for *TestCheck*'s evolution was achieved without undermining the second objective of the study, which was to maintain *TestCheck*'s high rate of defect detection when compared to the *Ad-hoc* technique. On average, for every 7 defects, 1 false-positive is generated by the *TestCheck* technique, whereas in the previous study the ratio was 3:1.

Table 10. Comparative Results of the Techniques regarding false positives.

Measures	***Ad-hoc***		***TestCheck***	
	False Positives	**Defects**	**False Positives**	**Defects**
Total	29	119	49	350
Average	2.9	11.9	4.9	35.0
Standard Deviation	2.846	10.969	3.314	22.852

Analyzing the data (number of defects and false positives) obtained for each question that comprise the 3 checklists that make up *TestCheck*, we notice a better distribution of the false positives, no longer concentrated in a reduced number of questions, as observed in the first study (Table 11). However, nine questions, highlighted in Table 11, that generated 3 false positives or more will be investigated and possibly evolved in a new *TestCheck* version.

It is not possible to eliminate the presence of false positives, as this is an element which is part of the inspection process and depends on various factors that cannot necessarily be controlled. However, they can be optimized by reducing some of the factors that lead to the occurrence of false positives, mainly in relation to the wording of the questions that comprise the checklists, thus eliminating ambiguities that can lead to multiple interpretations of the questions presented.

Table 11. Defects and false positives per *TestCheck* question (version 2.0).

Test Plan			***Test Case***			***Test Procedures***		
Question	Defects	False Positives	Question	Defects	False Positives	Question	Defects	False Positives
Pl03	7	1	C03	26	4	Pr03	30	2
Pl04	0	2	C04	0	1	Pr05	14	2
Pl05	0	2	C05	2	3	Pr06	25	1
Pl07	0	1	C06	30	5	Pr07	2	3
Pl08	4	2	C07	3	1	Pr09	20	3
Pl10	1	3	C09	4	1	Pr10	20	1
Pl11	0	1	C10	30	3	Pr12	1	1
Pl14	3	1	C11	2	1	Pr13	4	4
Pl15	7	1	C12	2	2	Pr15	55	12
Pl25	5	1	C13	22	2			
Pl26	2	1	C14	0	1			
Pl27	3	1	C15	6	1			
Pl28	3	1	C17	25	3			
Pl34	2	1	C18	1	3			
Pl38	1	1						

The threats to the validity of this study are the same as those of the previous one, and for this reason they are not listed again in this section.

After analyzing the averages, we again carried out statistical tests on the data obtained to check for the existence of statistically significant differences between results. An Analysis of Variance (ANOVA) was applied, with a level of accuracy of 95% (error rate $\alpha = 0.05$). The same variables analyzed in study 1 were again analyzed in this study, considering the two factors evaluated in the study: applied inspection technique applied (*Ad-hoc* or *TestCheck*) and inspected software project (*Industrial* and *Academic*). The results obtained are described in Table 12.

Table 12. Summary of the Analysis of Variance (ANOVA) applied to feasibility study 2.

Variable	***Analysis per Technique***		***Analysis per Project***	
	p-value	**Statistical Difference?**	*p-value*	**Statistical Difference?**
Defects	0.0099	**YES**	0.6313	NO
False-Positives	0.1649	NO	0.6845	NO
Time	0.0306	**YES**	0.8122	NO
Effectiveness	0.0472	**YES**	0.6419	NO
Efficiency	0.0857	NO	0.9869	NO

Analyzing the techniques, we observed a statistical difference (p-value < 0.05 [α]) between *TestCheck* and *Ad-hoc* for the variables *Defects*, *Time*, and *Effectiveness*. For the variables *Defects* and *Effectiveness* the result was the same as observed in the first study, suggesting *TestCheck* detects more defects than *Ad-Hoc*. However, *Time* was a variable where *TestCheck* obtained a negative result. One possible reason was lower expertise level of the participants on software engineering when compared to the first study's participants. We need to conduct new experiments to obtain more evidence regarding the impact of *TestCheck* with beginners on software engineering. The result of *Time* also impacted in the variable *Efficiency*, since *TestCheck* detected more defects, but required more time to be invested. For the last variable, False-positive, no statistical difference was found between the techniques.

Analyzing the software project, no statistical difference was found for any of the variables, suggesting the technique would be the factor that influences the results.

5 Observation Study in the Industry with Practitioners

5.1 Planning and Execution of the Observation Study

Proceeding within the methodology adopted in this study [3], the new technique is now to be applied in a second phase in order to analyze its evolution. Since the results obtained in the first two studies presented indications of *TestCheck*'s feasibility, this phase was conducted in a Brazilian research institute and aimed to assess whether the approach is adequate for use in a real-life environment. The items analyzed were its capacity to detect defects (quantitative evaluation) and the level of satisfaction of the inspectors from industry that applied the approach (qualitative evaluation). Table 13 presents information about the participants. In this study, one of the participants is the author of the documents inspected and he participated in the inspection section as a facilitator, evaluating the discrepancies and assessing their impact (severity) on the software project.

Table 13: Profile of the participants of the observation study.

Role	Position	Experience in the Industry	Experience with Inspections	Experience with Inspection Techniques	Experience with Software Testing Inspection
Inspector 1	Test Analyst	2 years	Average	Low	Average
Inspector 2	Test Analyst	More than 4 years	Average	Average	High
Inspector 3	Test Intern	6 months	Average	Low	Low
Author	Test Manager	More than 4 years	Average	Low	Average

This observation study proposes to respond to one question: Do the checklists proposed by *TestCheck* detect defects that would affect the quality of the execution of tests in a software project? To answer this question we analyzed the opinions of the participants of the inspection, evaluating whether the *TestCheck* approach allows the inspectors to identify defects in test artifacts (test plans, cases and procedures) within a reasonable amount of time, thus showing the effectiveness/efficiency of the checklists that comprise the approach.

As we were dealing with professionals from industry, with limited time on their hands, the inspection process was modified. The observation study in industry was conducted in only one session and all the participants carried out the inspection at the same time and in the same environment. In this session the candidates completed the Profile Questionnaire so as to identify their competency and experience with regard to the study's objective, as well as to identify the participant's knowledge of the problem domain to which the inspected artifact belongs. After the profiling, the participants were oriented to complete the Consent Form.

The test artifacts inspected (test plan, cases and procedures) consist of documents of a real project of that organization and they had already been evaluated by means of a self-devised (*Ad-hoc*) approach, however without keeping record of its performance (e.g. time and number of defects). After the first stage, the inspectors received training about the *TestCheck* approach and the procedures they should follow during this activity. They started the inspection by completing the list of discrepancies.

Having concluded the inspection, each participant was asked to complete the Post-Experiment Evaluation Questionnaire so as to obtain their qualitative evaluation of working with the *TestCheck* approach.

In possession of the Discrepancies Reports produced by each participant, a single meeting was held to classify the discrepancies either as defects or false positives. To qualify a discrepancy as a defect or a false positive we relied on the participants' own judgment as well as that of specialists in software testing (as happens in any traditional inspection process).

5.2 Quantitative Analysis

Considering that the evaluated testing artifacts had been previously inspected and approved by other professionals for tests execution, the number of defects detected using *TestCheck* was considered satisfactory by the software engineers that participated in the study (Table 14). All reviewers detected a high number of new defects.

Table 14: Result of the observation study carried out in the industry.

Inspector	Discrepancies (DI)	False Positives (FP)	Defects (D)	Time (T) [minutes]	Effectiveness [D / DI]	Efficiency [D / T]
01	8	1	7	85	87%	0.08
02	8	0	8	90	100%	0.09
03	12	0	12	80	100%	0.15
Total	**28**	**1**	**27**	**265**	**96%**	**0.10**

Another aspect observed in Table 14 was the low number of false positives generated using *TestCheck* (only 1 false positive was generated). This result can be related to several factors, such as the evolution of the checklists after the two feasibility studies or the previous experience of the reviewers. Thus, new studies would be necessary to get more evidence regarding this observation.

TestCheck obtained, on average, 96% of effectiveness (defects per discrepancies) and efficiency of 0.10 defects per minute. Therefore, we assessed as positive the use of *TestCheck* in this study.

5.3 Qualitative Analysis

Analyzing just the total number of defects detected by *TestCheck* in these artifacts cannot represent how the technique contributed to the quality of testing in the software project. Thus, we investigated which defects were found and their estimated impact (severity) to the software project. The goal was to observe if they were related to important system features that would affect the testing quality, if not fixed.

The 27 reported defects were analyzed by the researchers and the artifacts' author, who would be the professional responsible for the tests' execution in this project. Removing duplicated defects 20 new defects were identified (Test Plan: 11 defects; Test Cases: 6 defects; Test Procedures: 3 defects). The author classified them according to their impact to the quality of testing activity in three possible levels (Figure 5):

- High: it could make the tests' execution unfeasible;
- Medium: it would affect the testing, but the tester could observe and fix it before the execution.
- Low: it would affect the testing management, but have minimum impact on test execution.

Describing the defects classified as "high impact", they were related to:

- Test Plan: absence of quality attribute to be tested and incorrect addition of features not developed in the current iteration (sprint) as test item.
- Test Case: one incorrect expected result was described and the dependency among test cases was not defined.
- Test Procedures: the execution order of test cases was not optimized, causing additional effort for the tests' execution.

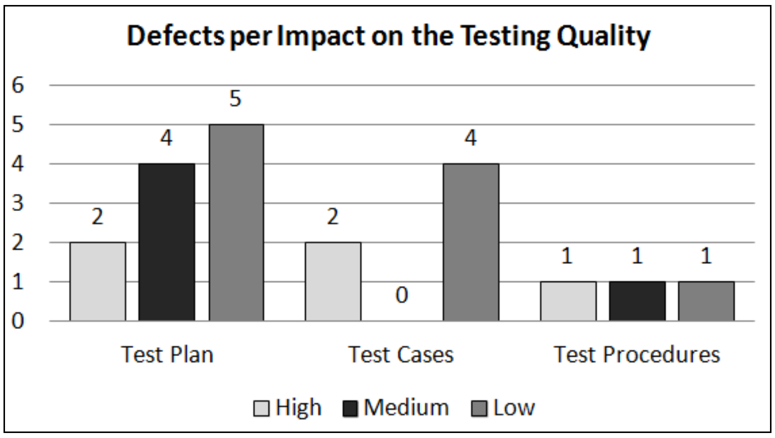


Figure 5: Classification of Defects per Impact on the Testing Quality.

5.4 Analysis of the Participants' Satisfaction

Analyzing the results obtained and each inspector's opinion (Table 15), we can identify the potential of this approach being applied in an industrial context. Some modifications would be necessary, and based on the post-experiment evaluation questionnaire it is possible to improve the approach for future experiments.

Table 15: Evaluation reported by each inspector.

Questions	Inspectors		
	01	**02**	**03**
Level of difficulty to apply the approach.	Very easy	Very easy	Easy
How did the technique help to identify defects in the artifacts?	3	3	3
Can the approach point to where errors can be found?	Yes	Yes	Yes
Can the approach evaluate the quality of a test artifact?	Yes	Yes	Yes
Is there any evaluation item that has not been understood?	No	No	Yes
Would you employ it in future inspections?	Yes	Yes	Yes

Keys:

1. Negatively. The approach acted as an obstacle. My performance would have been better if I hadn't used it.
2. Neutral. I think I would have found the same defects if I hadn't used the approach.
3. Positively. The approach helped to detect defects. Some defects I may not have detected if I hadn't used it.

Analyzing the answers to the questions we come to the following conclusion: With regard to detecting defects, all inspectors believe that the approach contributed positively, as it was possible to detect defects even after the document had already been reviewed. Another positive point highlighted by the inspectors is the fact that the approach allows to point defects in the artifacts, in addition to the evaluation of the quality of the test artifact. On account of these benefits, all inspectors believe that it is possible to use the approach in an industrial context, giving due consideration to some points of improvement described in Table 16.

Table 16: Inspectors' suggestions for improvement.

Inspector	Suggestions for improvement
01	There are some points where documents used in the Scrum framework cannot be applied. I believe that developing a checklist only for agile methodologies would be rather interesting.
02	A glossary with explanations of some of the items could be interesting for professionals initiating in this area.
02	Customize, taking into account the company's methodology.
03	Direct the checklist toward companies that use agile methodologies.

The improvement suggested by inspectors 01 and 03 refers to the creation of checklists directed toward companies that use agile methodologies. However, in this study we aimed to develop an approach that could be applied in different contexts, independently of the development methodology employed by an organization. That way, the *TestCheck* approach would be adaptable, bearing in mind that there is a wide range of items on the checklists, which allows firms to choose the items that fit the context of their company.

All suggestions were taken into account and analyzed, as the study focuses on the suitability of this approach for an industrial context. The suggestion of inspector 02 was considered to be very pertinent. Thus a glossary of terms will be included in a new version of *TestCheck*. Regarding customization, as suggested by Inspector 02, the checklists of the approach can be customized. However, this would have to be done manually, which would be costly in a project. One possible improvement would be to implement the *TestCheck* approach in a support tool in such a way as to allow for its customization, thus keeping to a minimum the work required for this task.

5.5 Threats to the Validity

Despite the identification of possible threats to the validity of the results, no conclusions were drawn beyond the limits imposed by these threats, and therefore they do not invalidate the study's results.

- **Conclusion Validity:**
  - **Reliability of the Measures:** The aim of this study was to identify the number of real defects and the type of defect the *TestCheck* approach identified with greater intensity. This being so, a potential threat to the validity of the study's conclusion refers to the classification of each discrepancy as a defect by the experimentalists. However, this process is subjective, and it therefore represents a threat.
- **Internal Validity:**
  - **Participants:** a potential problem is related to the sample of participants. However, as they are professionals in this field and work in industry, we can assume that they constitute a representative sample of inspectors, with participants having different levels of knowledge.
  - **Competency:** A possible threat to the validity is the necessary competency for carrying out the inspection. However, as they are professionals in this area with their skills and competency described in the profile questionnaire, this threat will be minimal.
- **Design Validity:**
  - **Mono-operation bias:** the observation study in the industry will use real projects containing different artifacts: one single Test Plan, five Test Cases and five Test Procedures as object of inspection. Therefore, as they are artifacts developed in industry, it can be assumed that they are representative of the theory on which they are based, and therefore the result obtained will be due to the application of *TestCheck*, and not to the "ease" of detecting defects in the document used, remembering that those artifacts had already been previously inspected using an *Ad-hoc* technique adopted by the organization.

6 Conclusions

This article presented a sequence of studies and their results, the objective of which is to provide a sequential evaluation of the *TestCheck* technique (a technique to support inspection of software testing artifacts), envisaging its evolution and, consequently, transfer to the software industry.

The first experimental study carried out evaluated *TestCheck*'s efficiency and effectiveness by comparing it to an *Ad-hoc* technique. The results of this study were rather positive and indicate that *TestCheck* would be more efficient and effective in detecting defects than an *Ad-hoc* technique. However, an opportunity for improvement was identified for its further development: the rate of false positives encountered during inspections. Thus a new version of the technique was produced aiming to reduce the rate of false positives being generated.

Then a second experimental study was conducted with the objective of evaluating the modifications made in *TestCheck* and their impact on the rate of false positives encountered. The aim of the technique continued to be providing a higher rate of defect detection when compared to an *Ad-hoc* technique. The data obtained points to a positive result, with the rate of false positives per inspector having gone down from 12.5 to 4.9 when compared to the results obtained in the first study.

Finally, a third experimental study was conducted, with the objective of observing *TestCheck*'s application in the hands of practitioners (software engineers in the industry). The results indicate that *TestCheck* contributed positively to the detection of defects in test artifacts that had already been inspected previously with an *Ad-hoc* approach. The qualitative analysis carried out with the participants indicated their satisfaction in regard to working with the proposed approach.

The data obtained in both studies are not conclusive; they do however indicate that the approach has undergone an evolution and matured in the course of the studies. A further development will be the construction of a support tool for the application of *TestCheck*, allowing for the customization of the checklists that comprise it. The scientific methodology (proposed in [3]) for evaluating the approach will continue to be followed and *TestCheck* will proceed to the third phase, which is applying the method to a case study in an industrial environment. Thus new points for evolution will be identified and applied to the technique with the aim of eventually establishing it in an industrial environment.

Acknowledgments. The authors thank FAPEAM, FAPESP, CNPq (process 575696/2008-7) and INCT-SEC (processes 573963/2008-8 and 08/57870-9) for their kind support and assisting us in this research.

References

[1] Basili, V.R., Selby, R.W., Hutchens, D.H.; “Experimentation in Software Engineering”, IEEE Transactions on Software Engineering, 12 (7), 1986, pp. 733-743.

[2] Hannay, J. E., Hansen, O., By Kampenes, V., Karahasanovic, A., Liborg, N., and C. Rekdal, A; A Survey of Controlled Experiments in Software Engineering. IEEE Transaction on Software Engineering, 31, 9, 2005, pp. 733-753.

[3] Shull, F., Carver, J., Travassos, G. (2001) “An Empirical Methodology for Introducing Software Processes”, In: Joint 8th ESEC and 9th ACM SIGSOFT FSE-9, pp. 288-296.

[4] Dias Neto, A. C.; Spinola, R. ; Travassos, G. H. . Developing Software Technologies through Experimentation: Experiences from the Battlefield. In: XIII Congresso Iberoamericano en "Software Engineering" (CIBSE). Cuenca : Universidad del Azuay, 2010. v. I. p. 107-121.

[5] Biolchini, J. ; Mian, P. ; Conte, T. U. ; Natali, A. C. C. ; Travassos, G. H.; Scientific research ontology to support systematic review in Software Engineering. Advanced Engineering Informatics, v. 21, p. 133-151, 2007.

[6] Siy, H., & Wu, Y. (2009). An Ontology to Support Empirical Studies in Software Engineering. In: International Conference on Computing, Engineering and Information (ICC '09), April, 2009, pp. 12-15. doi:10.1109/ICCEIS.2009.72.

[7] Lopes, V. P. ; Travassos, G. H. . Knowledge Repository Structure of an Experimental Software Engineering Environment. In: XXIII Brazilian Symposium on Software Engineering, Fortaleza, 2009. v. 1. p. 32-42.

[8] Travassos, G.H.; Santos, P.S. M.; Mian, P.; Dias-Neto, A.C.; Biolchini, J.; An Environment to Support Large Scale Experimentation in Software Engineering. In: IEEE International Conference on Engineering of Complex Computer Systems, 2008, Belfast, p. 193-202.

[9] Luo, L. (2010), Software Testing Techniques - Technology Maturation and Research Strategies; In: Institute for Software Research International - Carnegie Mellon University, Pittsburgh, Technical Report 17-939A.

[10] Poon, P.L.; Tse, T. H.; Tang, S.F.; Kuo, F.C. (2011), “Contributions of tester experience and a checklist guideline to the identification of categories and choices for software testing”, Software Quality Control 19, 1 (March 2011), pp. 141-163. DOI=10.1007/s11219-010-9109-4 http://dx.doi.org/10.1007/s11219-010-9109-4.

[11] Itoken, J. V.; Mantyla, M,V.; Lassenius, C. (2007), “Defect Detection Efficiency: Test Case Based vs. Exploratory Testing”, In: First International Symposium on Empirical Software Engineering and Measurement, pp. 61-70, DOI 10.1109/ESEM.2007.56.

[12] Fagan. M. (1976); “Design and Code Inspections to Reduce Errors in Program Development”. IBM Systems Journal. Riverton. NJ. V.15. n.3.p.182-211.

[13] Brito, J.; Dias-Neto, A.C., *TestCheck* – A Checklist-Based Approach to Inspect Software Testing Artifacts (In Portuguese), In: Brazilian Symposium on Software Quality, Fortaleza, Junho, 2012.

[14] Barcelos, R. F.; Travassos, G. H., “An Approach to Inspect Architectural Artifacts Based on Checklist”. In: Braziliam Symposium on Software Quality, 2006, Vila Velha.

[15] Conte, T. U. ; Vaz, V.T. ; Massolar, J. ; Mendes, E. ; Travassos, G. H. . Improving a Web Usability Inspection Technique using Qualitative and Quantitative Data from an Observational Study. In: Simpósio Brasileiro de Engenharia de Software, 2009. v. 1., Fortaleza, p. 227-235.

[16] Mafra, S. N.; Travassos, G. H.; A Skeptical Discussion Regarding Experimentation in Software Engineering Based on Reading Techniques Studies. In: 2nd Experimental Software Engineering Latin American Workshop, 2005, Uberlandia, v.1.

[17] Brito, J.; Dias-Neto, A. C.; (2012), "Conduzindo Estudos Experimentais para Avaliação de uma Técnica de Inspeção de Artefatos de Teste de Software" (*in Portuguese*), In: Experimental Software Engineering Latin American Workshop (ESELAW 2012), pp. 41-50, Buenos Aires.

[18] Laitenberger, O., DeBaud, J.M., (1997). Perspective-based Reading of Code documents at Robert Bosch GmbH. Information and Software Technology, 39:781–791.

[19] IEEE Standard 829-2008: Standard for Software Test Documentation, IEEE Press, 2008.

[20] Sidek, R.M.; Noraziah, A.; Wahab, M.H.A.; "The Preferable Test Documentation Using IEEE 829", In: International Conference on Software Engineering and Computer Systems (ICSECS), Vol. 181, 2011, pp.109-118.

[21] Dias-Neto, A. C.; Travassos, G. H. . Evolving a Computerized Infrastructure to support the Selection of Model-Based Testing Techniques. In: IFIP International Conference on Testing Software and Systems (ICTSS'10), PP. 85-90, Natal, 2010.