Applying Testing to Enhance Software Product Quality

Etienne Lamas, Luiz Alberto Vieira Dias, Adilson Marques da Cunha

Computer Science Division
Aeronautics Institute of Technology, ITA
Sao Jose dos Campos, Brazil

Abstract

Software free from defects is a goal to be ideally achieved. Defects in software do arise from the development process. A possible solution for quality improvement is to apply a reference framework for software testing during all development phases. What is needed is a software testing process performed along with the development, in an orderly and simple way, even if the software development have been conducted in an organization without maturity certification. This investigation proposes to validate the hypothesis: software product quality is dependent on the software testing. The Operational Reference Architecture for a framework under development, named Organizational Testing Management Maturity Model (OTM3), as in [1, 2, 3, 4], focuses on software testing process improvement, based upon Software Process Engineering Meta-model (SPEM), as in [5]. It is structured by direct experiences on issues related to the measurement of software testing results, with continuous improvements on its process, and specifications and design techniques using the method of Goal-Question-(Indicator)-Measure (GQ(IM)), as in [6]. Using OTM3 simplifies the software testing, because it becomes part of the development process in an orderly and simple way, instead of an ad hoc.

This article is organized as follows. Section 2 describes the Software Testing process. Section 3 deals with software testing for a software product line. Section 4 and 5 presents some experimental results and describes basic statistical concepts. Section 6 specifies the statistical technique for Pearson’s Chi-Square applied to variable independence. Next, in Section 7, it is discussed the statistical technique for Pearson’s Chi-Square from software testing results. Section 8 presents some general considerations. Finally, Section 9 highlights conclusions.

1. Introduction

Software free from defects is a goal to be ideally achieved. Defects in software do arise from the development process.

A possible solution for quality improvement is to apply a reference framework for software testing during all development phases.

What is needed is a software testing process performed along with its development, in an orderly and simple way, even if the software development is been conducted in an organization without maturity certification. This investigation proposes to validate the hypothesis: software product quality is dependent on the software testing.

The Operational Reference Architecture for a framework under development, named Organizational Testing Management Maturity Model (OTM3), as in [1, 2, 3, 4], focuses on software testing process improvement, based upon Software Process Engineering Meta-model (SPEM), as in [5]. It is structured by direct experiences on issues related to the measurement of software testing results, with continuous improvements on its process, and specifications and design techniques using the method of Goal-Question-(Indicator)-Measure (GQ(IM)), as in [6]. Using OTM3 simplifies the software testing, because it becomes part of the development process in an orderly and simple way, instead of an ad hoc.

This article is organized as follows. Section 2 describes the Software Testing process. Section 3 deals with software testing for a software product line. Section 4 and 5 presents some experimental results and describes basic statistical concepts. Section 6 specifies the statistical technique for Pearson’s Chi-Square applied to variable independence. Next, in Section 7, it is discussed the statistical technique for Pearson’s Chi-Square from software testing results. Section 8 presents some general considerations. Finally, Section 9 highlights conclusions.

2. The Software Testing Process

If the software testing is conducted in an ad hoc way, the expended effort is unnecessary and still worse, faults do occur undiscovered, as in [7]. The literature presents some testing maturity models such as: (i) Testing Maturity Model (TMM), as in [9]; (ii) Test Process Improvement (TPI), as in [10]; (iii) Testing Improvement Model (TIM), as in [11]; and (iv) The Test Maturity Model integration (TMMi), as in [12]. Models (i), (ii), and (iii) have a complementary relationship with the Capability Maturity Model [13].
The model (iv) is based on the Capability Maturity Model Integration (CMMI) [14]. These four models were based on a maturity level, with no impact caused by already implemented previously processes.

The OTM3, as in [1, 2, 3, 4], under development, has an advantage over the other maturity models for software testing. It is independent from the maturity model used on its software development.

The OTM3 is comprised of three different architectures: (i) Operational Reference Architecture, to be detailed in this paper; (ii) Collaborative Reference Architecture, to deal with Cloud Testing; and (iii) Decision Support Reference Architecture. The last two Reference Architectures are under development.

The Operational Reference Architecture, as in [1, 2, 3, 4], is also composed of a set of process maps, based upon the SPEM and the Plan-Do-Check-Act cycle, as in [15].

The Operational Reference Architecture overall view comprises the following main six phases of software testing: Ph1 - Planning; Ph2 - Construction; Ph3 - Test Execution; Ph4 - Test Finalization; Ph5 - Approval; and Ph6 - Measurement and Analysis.

Test Case Scenarios are based on requirements used for the Use Case creation, as in [16]. It is very important to emphasize that the activities in the Test Execution and in the Test Finalization Phases in general apply the Regression Test technique. Regression testing means rerunning test cases from existing test suites to build confidence that software changes have no unintended side effects.

The Planning Phase (Ph1) consists of developing a test proposal, which will be used to minimize the key business risk and to provide paths for the next phases. In this Planning Phase, as seen in [2], a Test Plan was developed.

The Construction Phase (Ph2) consists of testing design and implementation, in order to prepare the environment for the next Test Execution Phase (Ph3).

In general, software testing management activities use specific tools. For example, the Mantis tool, as in [17] can be used to provide each defect control within a software life-cycle, by using customizable workflows to report defects until an activity closure.

It is also important to use a version control system like the Subversion tool, as in [18]. It provides an effective screening for tracking where and who made changes. Thus avoiding possible conflicts caused by unwanted changes.

The Mantis tool, used in this research, also provided integration with the TestLink tool, as in [19]. The TestLink is a management system for test execution and tracking Test Cases, organizing them into a Test Plan. A Test Case is a set of inputs, execution preconditions, and expected results (outcomes) developed for a particular objective to verify compliance with a specific requirement.

The existence of these three integrated tools adds reporting on project-levels, analyses, and management capabilities. This integration occurs when used in conjunction with information for requirements traceability. It also allows that the defects control could be made by the management system. Other tools with the same functionalities could be also used instead.

The main objective of software testing is the measurement of Test Case Scenarios to evaluate its capability in terms of efficiency to detect defects.

At the end of the Test Execution Phase (Ph3) variables for primary metrics may be collected from Mantis and TestLink tools. These metrics are related to the measurement of software testing results. By using the GQ(1)M method, as in [6], secondary metrics are calculated from primary metrics. Metrics should provide product quality attributes, such as process control.

At this point, the statistical technique for Pearson’s Chi-Square ($\chi^2$) is used, as discussed in Section 7. This has to be carried out, in order to determine whether there is a significant relationship between two categorical variables.

This investigation proposes to validate the following hypothesis: software product quality is dependent on the software testing. A case study based on an actual project at the Brazilian Aeronautics Institute of Technology within its Software Engineering Research Group (Grupo de Pesquisa em Engenharia de Software), as in [20], was performed, using the Operational Reference Architecture proposed based on the OTM3.

3. Testing For a Software Product Line

The adopted strategy in the R & D Project at ITA was to develop software according to a Model Driven Architecture (MDA) approach. It was integrated within a Unified Process based on software product line components for each subsystem, as in [21].

The approach for software product line, as in [22], is a concept used in the computing environment for projects software development. This encourages the creation of groups with similar software systems from a common set of specifications, by using common means of production. The software product line represents a software development technique that has been transposed from other production areas, used in engineering.

In all phases of software testing, a Test Plan was developed and integrated for each Software Product.
Line (SPL), before the occurrence of SPLn phase changes, where “n” refers to each product line.

For the R & D Project at ITA, eight software product lines were planned, but only two of them had been available at the time of this research. Thus, only the Software Product Line 2 (SPL2) and Software Product Line 3 (SPL3) were utilized. At this point, data were collected from two different experiments and placed within the following software testing artifacts: Test Plan, Test Procedures, Test Cases, and Test Scenarios (also commonly called scripts or suites), as in [23]. This was done by using Functional and Structural Techniques, as in [24], and Defect Prevention Analysis, as in [3].

4. Experimental Software Testing Results

To obtain the software testing results, Test Cases, as in [24], based on Use Case requirements, as in [16], were manually performed for SPL2 and SPL3. In this case, TestLink and Mantis were used to collect the results for each SPLn.

Within this project methodology, not only the testers (software testing team), but also all developers had the responsibility to detect defects.

During this research, while the software testing team has detected failures from Test Cases and registered them by using the TestLink tool shown in Figure 1, the project team (different from software testing team) has detected failures and also registered them by using the Mantis tool.

The TestLink tool shown in Figure 1 enables monitoring the percentage amount of test cases for each outcome status. Those, which passed are in green, failed in red, blocked in blue, and non-executed in black.

Through the failures registered in the Mantis tool by the project team, as shown in Figure 2, it becomes possible to calculate the metric of Effectiveness in Fault Detection (EFD) and the other two metrics: (i) Failures Found by Software Testing (FFST); and (ii) Failures Found by Other (FFO), formulated by Correa, as in [25].

4.1. Effectiveness in Fault Detection for the SPL2

From primary metrics collected at SPL2, it was possible to calculate the metric of Effectiveness in Fault Detection (EFD), from (1):

\[
EFD = \left( \frac{80}{80 + 53} \right) \times 100 = 60\%
\]

4.2. Effectiveness in Fault Detection for the SPL3

From primary metrics collected at SPL3, it was possible to calculate the metric of Effectiveness in Fault Detection (EFD), from (1):

\[
EFD = \left( \frac{114}{114 + 27} \right) \times 100 = 81\%
\]

References [2, 4] show that the software testing team, using the Operational Reference Architecture, has obtained a value of 60% for Effectiveness in Fault Detection within SPL2 and 81% for SPL3, in relation to the project team.

From experimental data taken from Mantis and TestLink tools, as shown in Figures 1 and 2, it was enabled the development of a product with more quality.

It is important to mention that most of the Test Cases for the SPL3 were developed before the system
code had been generated. This way, certain defects were detected very early in the process.

4.3. Software Testing Results from the 1st Integration Test for SPL3

The 1st Test Integration for SPL3 is presented first, because of its relevance mainly considering: (i) It had 7 available Use Cases (UC1 to UC7); (ii) It had 361 performed Test Cases; and (iii) The large number of failures indicated in the TestLink charts, as shown in Figure 3.

![Figure 3. The Testlink Chart - 1st Test Integration for SPL3.](image)

Table I, at the end of this article, presents all results obtained from the 1st Integration Test, where primary metrics were collected from the TestLink tool for SPL3 as in [2].

4.4. Software Testing Results from Integration Test and Regression Test for SPL2 and SPL3

The main result from the 1st Integration Test, and from the 1st and the 2nd Regression Test, was the primary metric that depicted the number of failures detected in the software and acquired by Mantis for SPL2 and SPL3.

This Software Testing Results were grouped in failures classified by characteristics: Group 1 - without unit test performed; Group 2 - failures from data base or server activities; Group 3 - failures detected by project team (excluding testing team personnel); and Group 4 - failures in documentation, as in [2].

Failures found by software testing team were registered on Groups 1, 2, and 4, as in [2].

4.5. Software Testing Results from the 1st Integration and the 2nd Regression Test for SPL2 and SPL3

In this section are the main results from the 1st Integration Test and also from the 2nd Regression Test. The primary metric collected from the TestLink tool was the number of software failures detected for SPL2 and SPL3, as in [2].

5. Basic Statistical Concepts

In order to analyze the main results of this research, it is necessary to understand some basic statistical concepts.

An experimental study aims to collect data in a controlled environment to confirm or deny a hypothesis, as in [26, 27, 28]. Usually, two hypotheses are set:

5.1. The null hypothesis (H0)

The H0 indicates that observed differences are coincident (i.e., it assumes that the analyst wants to reject with the greatest possible significance); and

5.2. The alternative hypothesis (H1)

The H1 means the opposite of the null hypothesis, which will be accepted if the null hypothesis is rejected. Statistical tests allow the determination if it is possible to reject the null hypothesis, according to a set of observed data and their statistical properties.

Hypotheses have dealt with the risk of a parsing error occurrence. The “Type I error” (a Type I error is like a false positive) occurs when the statistical test indicates a relationship between two categorical variables.

The Significance Level (\( \alpha \)) of a test is such that the probability of rejecting by mistake the null hypothesis is no more than the stated probability. The most commonly used significance levels are 10%, 5%, 1%, and 0.1%.

The lowest significance level with which one can reject H0 is the p-value, as in [27, 28]. There is a statistical significance, when the p-value is less than the adopted level.

In order to Test a Hypothesis, it is necessary to follow six steps: (i) Determine the significance level of the test; (ii) Choose statistics with a known distribution under H0; (iii) Build the critical region under which the test passes, from the statistical test, and significance level; (iv) The sample is used in order to obtain the statistical value; (v) Reject the null hypothesis, if the
statistical value belongs to the critical region, accepting the alternative hypothesis; and (vi) Otherwise, the null hypothesis is not rejected. At this point, nothing can be said about the alternative hypothesis.

6. Using the Chi-Square Technique for Independence

In order to validate this research, the statistical technique for Pearson’s Chi-Square ($\chi^2$) was selected, as in [2, 27, 28]. An independence test assesses whether paired observations on two categorical variables, expressed in a standard contingency table, are independent of each other.

In order to analyze the quantitative results of this research, two samples of software testing variables were considered.

In this research, the statistical technique for Pearson’s Chi-Square, as in [2, 27, 28], was considered by comparing the Observed Frequency (OF) from the software testing for categorical variables with Expected Frequency (EF) obtained from primary metrics.

These samples were comprised of non-paired variables (those with no relationship among them) and nominal variables. The variables obeyed a non-normal distribution. An expected frequency is a theoretic frequency obtained from an experiment. It is presumed to be true until statistical evidence in the form of a hypothesis test indicates otherwise. An observed frequency, on the other hand, is the actual frequency that is observed.

At this point, it is important to mention that, in software testing, the expected frequency values are unknown. The standard contingency tables were used (with “r” lines and “s” columns) to calculate EF, when not known. For the calculation, it was multiplied totals from marginal column by the marginal line, and divided the product by the overall total of the table.

After that, the adopted procedures were the hypotheses testing regarding the statistical independence of two variables. The statistical technique for Pearson’s Chi-Square, as in [2, 27, 28], to determine this independence, was based on the magnitude of the difference between OF and EF.

It’s also important to mention that this difference should be assigned to the variability of software testing at the level of significance.

Initially, the statistical technique for Pearson’s Chi-Square was calculated by finding the difference between OF and EF for each possible outcome, as in [2, 27, 28].

A second important part of determining the statistical technique for Pearson’s Chi-Square was defining the Degrees of Freedom (DF) of the experiments, as in [2, 27, 28]. This has represented essentially the adjustment of the observed frequency number for using some of those observations to define the expected frequency.

The number of Degrees of Freedom, from the data in a standard contingency table, was calculated from (2):

$$DF = (r - 1) \cdot (s - 1).$$

At this point, two statistics were obtained: a calculated $\chi^2$ value; and another tabulated $\chi^2$ value. To calculate $\chi^2$ statistics, the formulae shown in (3), was used

$$\chi^2_{calc} = \sum_{i=1}^{r} \sum_{j=1}^{s} \frac{(OF_{ij} - EF_{ij})^2}{EF_{ij}}$$

Notice that the OF was obtained directly from data samples, while the EF was also calculated from the OF.

It is important to mention that the deviation was the mismatch between OF and EF in a class. When OF was very close to EF value, $\chi^2$ was considered small. But when differences between OF and EF were large, the deviation became too large and therefore $\chi^2$ was assumed as a high value.

For the tabulated $\chi^2$ value, it was queried the Pearson’s Chi-Square table, as in [2, 27, 28], where the number of Degrees of Freedom corresponded to the line value and Type I error ($\alpha$) related to the column. At the intersection between the line and the column, the tabulated $\chi^2$ was determined.

Then, a decision was taken, by comparing the two values: (i) If the calculated $\chi^2 \geq$ the tabulated $\chi^2$, H0 is rejected; or (ii) If the calculated $\chi^2 <$ the tabulated $\chi^2 \rightarrow$ Accepts H0.

The $\chi^2$ test of this experiment was a non-parametric one (i.e., it did not depend on population parameters such as mean and variance). The basic principle of this method was to compare proportions (i.e., possible differences between the OF and EF for each experiment).

In this case, it can be said that two samples behave similarly, if and only if the differences between OF and EF within each category are very small, or close to zero.
7. Chi-Square Statistics from Software Testing Results

In general, failures detected from software testing processes would allow the removal of defects, which would produce software with more quality. Software product quality requires software with minimum failures, consequently, small OF. Software testing reduces OF.

To evaluate the quality of software, the statistical technique of Pearson’s Chi-Square was used, as in [2, 27, 28], in order to determine whether there was a significant relationship between OF and EF after testing. The OF was obtained directly from the software testing results, while the EF was calculated from the OF. Thus, the calculated $\chi^2$ was derived from experimental data, taking into account the OF and EF, to evaluate their independence from each other.

7.1. Chi-Square Statistics from the 1st Integration Test for SPL3

Through the software testing results, the measurements obtained from the primary metrics were collected at SPL3, as shown in Table I, at the end of this article.

It was possible to calculate $\chi^2$, from formulae (3). Considering DF = 6, from formulae (2), and the Type I error = 0.05, hypotheses were: H0 = the software product quality was independent from software testing; and H1 = the software product quality was dependent from software testing.

Calculated $\chi^2 = 28.63 >$ tabulated $\chi^2 = 12.59$

As the achieved value was greater than the critical value, deviations were not as a result of chance. Therefore, H0 was rejected, H1 accepted, and the software product quality was dependent on the software testing.

Notice that in this experiment, the calculated p-value was 7.15E-15. In this case, there was also a statistical significance, because the p-value was less than the adopted level.

7.2. Chi-Square Statistics from Integration Test and Regression Test for SPL2 and SPL3

Through the software testing results, the measurements obtained from the primary metrics were collected at SPL2 and SPL3.

It was possible to calculate $\chi^2$, from formulae (3). Considering DF = 6, from formulae (2), and the Type I error = 0.05, hypotheses were: H0 = the software product quality was independent from software testing; and H1 = the software product quality was dependent from software testing.

Calculated $\chi^2 = 155.01 >$ tabulated $\chi^2 = 12.59$

In this case, as the calculated value was greater than the tabulated, H0 was rejected and H1 accepted. Then, the numbers for OF and EF were statistically different. The differences were statistically significant, enabling the assertion that the software product quality was also dependent on the software testing.

Notice that in this experiment, the calculated p-value was 6.7E-31. In this case, there was also a statistical significance, because the p-value was less than the adopted level.

7.3. Chi-Square Statistics from the 1st Integration and the 2nd Regression Test for SPL2 and SPL3

Through the software testing results, the measurements obtained from the primary metrics were collected at SPL2 and SPL3.

In this case, it was possible to calculate $\chi^2$, from formulae (3). Considering DF = 1, from formulae (2), and the Type I error = 0.05, hypotheses were: H0 = the software product quality was independent from software testing; and H1 = the software product quality was dependent from software testing.

Calculated $\chi^2 = 6.35 >$ tabulated $\chi^2 = 3.84$

In this case, deviations were also significant, because the calculated value was greater than the tabulated one. Variables were not independent. The association between variables was not as a result of chance. Following, the probability of such features was dependent and, statistically, it was stated that the software product quality was dependent on the software testing.

Notice that in this experiment, the calculated p-value was 0.011. In this case, there was a statistical significance, because the p-value was less than the adopted level.

8. General Considerations

As a natural continuation of this work, some ongoing research has been done for applying testing to enhance software product quality, supported by graduate courses on Software Quality and Software Testing at the Brazilian Aeronautics Institute of Technology [29, 30, 31].
On the last years, this research work has also motivated the development of some Master [2] and PhD Theses [32] involving the Cloud Computing environment and related investigations directly applied to the Embraer and other Brazilian Aerospace Industries.

9. Conclusion

Experimentally, the software testing team, using the Operational Reference Architecture, has obtained the value of 60% for Effectiveness in Fault Detection within Software Product Line 2 (SPL2) and 81% for Software Product Line 3 (SPL3), in relation to the project team.

In order to determine whether there was a significant relationship between two categorical variables (Observed Frequency - OF and Expected Frequency - EF) from software testing results, by using the Operational Reference Architecture, the statistical technique for Pearson’s Chi-Square was successfully used in this article.

Results from these analyses revealed that there was statistical significance because, in each one of the two experiments (SPL2 and SPL3), with different tools (Mantis and TestLink) and at different test phases (Integration and Regression), the calculated p-value was less than the adopted level.

The Operational Reference Architecture application has permitted an appropriate and simple management for software testing through process maps, allowing the improvement of product quality through prevention defects and failures. It has addressed both the management of defects and measurements of the software testing results, through a method of using effective metrics.

So, for these experiments, in order to reach a product with software quality, it was necessary to implement a quality process evolution.

By using this Operational Reference Architecture, it was possible to statistically state that: (i) The product with software quality depends on the appropriate application of a software testing process; and (ii) The statistical technique for Pearson’s Chi-Square can be used to ensure the categorical variables (OF and EF) independence from software testing results.

Several iterative and incremental investigations to support the application of testing to enhance software product quality on different projects, including Cloud Testing, have been developed at the Brazilian Aeronautics Institute of Technology, for the benefit of the Brazilian Aerospace Industries.

10. Acknowledgment

Authors would like to thank: the Brazilian Aeronautics Institute of Technology – ITA, for its scientific development incentives by using Research & Development Projects at ITA; the Brazilian National Water Agency – ANA, for the opportunity of participating in its Project; the Finance Agency for Studies and Projects (FINEP); and also the Casimiro Montenegro Filho Foundation (FCMF), for its available infrastructure.

11. References


Table 1. TestLink Tool Results from 1st Integration Test for SPL3, as in [2]

<table>
<thead>
<tr>
<th>Use Case 1</th>
<th>Use Case 2</th>
<th>Use Case 3</th>
<th>Use Case 4</th>
<th>Use Case 5</th>
<th>Use Case 6</th>
<th>Use Case 7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software failure</td>
<td>7</td>
<td>29</td>
<td>42</td>
<td>14</td>
<td>4</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>Software success</td>
<td>21</td>
<td>57</td>
<td>37</td>
<td>37</td>
<td>21</td>
<td>36</td>
<td>33</td>
</tr>
<tr>
<td>Total Test Cases</td>
<td>28</td>
<td>86</td>
<td>79</td>
<td>51</td>
<td>25</td>
<td>40</td>
<td>52</td>
</tr>
</tbody>
</table>