Experimental Evaluation of Automatic Tests Cases in Data Analytics Applications Loading Procedures

Igor Peterson Oliveira Santos[1], Juli Kelle Góis Costa[1], Methanias Colaço Júnior [1,2] and André Vinícius R. P. Nascimento[2]

[1]*Postgraduate Program in Computer Science – PROCC, Federal University of Sergipe- UFS, São Cristóvão-SE, Brazil*

[2]*Competitive Intelligence Research and Practice Group – NUPIC, Information Systems Departament – DSI, Federal University of Sergipe - UFS, Itabaiana-SE, Brazil*

Keywords: Business Intelligence, Data Warehouse, Software Testing, Data Quality, Experimental Software Engineering, DbUnit.

Abstract: Business Intelligence (BI) relies on Data Warehouse (DW), a historical data repository designed to support the decision making process. Despite the potential benefits of a DW, data quality issues prevent users from realizing the benefits of a BI environment and Data Analytics. Problems related to data quality can arise in any stage of the ETL (Extract, Transform and Load) process, especially in the loading phase. This Paper presents an approach to automate the selection and execution of previously identified test cases for loading procedures in BI environments based on DW. To verify and validate the approach, a unit tests framework was developed. The overall goal is achieve efficiency improvement. The specific aim is reduce test effort and, consequently, promote test activities in data warehousing process. A controlled experiment evaluation in the industry carried out to investigate the adequacy of the proposed method against a generic framework for DW procedures development. Constructed specifically for database application tests, DbUnit was the generic framework chosen for the experiment by convenience of the programmers. The experiment's results show that our approach clearly reduces test effort when compared with execution of test cases using a generic framework.

1 INTRODUCTION

Information represents a crucial factor for companies in improving processes and decision-making. To assist the strategic areas of the organizations business intelligence (BI) environments presented as sets of technologies that support the analysis of data and key performance indicators (Colaço, 2004).

A central component of BI systems is a Data Warehouse (DW), a central repository of historical data. The idea behind this approach is to select, integrate and organize data from the operational systems and external sources, so they can be accessed more efficiently and represent a single view of enterprise data (Colaço, 2004; Kimball, 2008; Inmon, 2005).

Despite the potential benefits of a DW, data quality issues prevent users from realizing the benefits of a business intelligence environment. Problems related to data quality can arise in any stage of the ETL (Extract, Transform and Load) process, especially in the loading phase. The main causes that contribute to poor data quality in data warehousing identified in (Ranjit and Kawaljeet, 2010).

The lack of availability of automated unit testing facility in ETL tools is also appointed as cause for the poor data quality (Ranjit and Kawaljeet, 2010).The low adoption of testing activities in DW environment is credited to the differences between the architecture of this environment and architectures of the generic software systems. These differences mean that the testing techniques used by the latter need to be adjusted for a DW environment (Deshpande, 2013; Elgamal *et al.*, 2013).

Tests of ETL procedures considered the most critical and complex test phase in DW environment because it directly affects data quality (Golfarelli and Rizzi, 2009). ETL procedures, more precisely the loading routines, exhibit the same behavior as database applications. They operate on initial database state and generate a final consistent

Santos, I., Costa, J., Júnior, M. and Nascimento, A.
Experimental Evaluation of Automatic Tests Cases in Data Analytics Applications Loading Procedures.
DOI: 10.5220/0006337503040311
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 304-311
ISBN: 978-989-758-247-9

database state. So, a black-box approach, which combines the unit and application behavior of loading procedures, is proposed. In this approach, the concern is with the application interface, and not with the internal behavior and program structure (Myers, 2012; Pressman, 2011; Sommerville, 2011). This approach to ETL routines, in some environments, may be the only option, since the use of ETL tools in DW environment produces codes or packages whose internal structure is not known.

Previous studies was presented an experimentation with the proposal of using a unit testing framework (FTUnit) for loading routines in a BI environment based on Data Warehouse (Santos et al, 2016). The motivation for adopting this approach meets the problems pointed out in (Singh and Singh, 2010), as the main causes for the poor quality of data in a DW environment. Through a set of metadata that defines the characteristics of the routines, the framework selects test cases to applied, generates the initial states of the database, executes the routines, performs test cases, analyzes the final state of the database and generates a report with the errors encountered during the execution of each test case. With good results, the use of the framework presents important contributions to increasing the productivity and quality in software engineering for loading routines of DWs.

This paper aims to show the results of an experiment to verify the best performance using the test framework (FTUnit) against a generic database application test framework (DbUnit, 2016). The performance results of FTUnit already presented in (Santos et al, 2016) shows the framework can accelerate and improve the quality of ETL process tests based on SQL. Now to compare with this study, the same use case was used for run the test cases at DbUnit in BI environment, once this framework had been constructed specifically for database application tests.

Thus, considering an industrial environment, this work aims address the following research question: *"in a load context for DW, test cases perform better using the test framework compared with a generic framework?"* To answer, our experimental evaluation analyzed a real context. The results showed numbers that indicate effort reduction using the Testing framework.

The remainder of this paper is structured as follows. Section 2 presents the test framework and DbUnit. Section 3 describes the experiment definition and planning. Section 4 presents the operation of the experiment. Section 5 reports the results of the experiment. In Section 6, related works are presented. Finally, section 7 contains the conclusion and future works.

2 TESTING FRAMEWORK AND DBUNIT

2.1 Testing Framework

Unit testing framework (FTUnit) is used to perform unit tests in loading procedures of a DW environment. It has been developed in C#. It is available for download at <http://ftunit.wordpress.com/>. This tool resembles a framework for performing procedures on test cases in T-SQL (Transact-SQL) language. Therefore, it can be expanded to other SQL languages.

This framework will be used to perform tests under the black-box approach. The code and the internal structure of the routines will not be examined. The test cases previously implemented, will be selected according to the characteristics of the routine being tested. Each routine, in order to be covered by the framework, must have a set of metadata registered. This set of metadata was defined from the schema presented in (Costa et al, 2015).

This paper will not detail the process of implementing the framework for reasons of lack of space.

2.2 DbUnit

DbUnit is a JUnit extension targeted at database-driven projects that puts the database into a known state between test runs. This can avoid the myriad of problems that occurs when one test case corrupts the database and causes subsequent tests to fail or exacerbate the damage (DbUnit, 2016).

Created to implement database operations tests in Java, DbUnit can work with large datasets when used in streaming mode and verify data matches an expected set of values. Using a XML based mechanism for loading test data, DbUnit can export existing test data into the XML format for subsequent use in automated tests. This method compares data, between database tables, flat files and queries (DbUnit, 2016).

This paper will not detail the process of implementing the test cases at DbUnit for reasons of lack of space.

3 EXPERIMENT DEFINITION AND PLANNING

Our work is presented here as an experimental process. It follows the guidelines by Wohlin et al. (2000). In this section, we start introducing the experiment definition and planning. The following sections, will direct to the experiment execution and data analysis.

3.1 Goal definition

Our main goal is to evaluate the best performance using the test framework against a generic database application test framework in a Data Warehouse environment.

The experiment will target developers of ETL processes for BI environments with at least 2 years of experience in the market and one year of experience in ETL programming. The goal was formalized using the GQM model proposed by Basili and Weiss (Basili, 1984):

- **Analyze** the use of a DW unit testing framework
- **With the purpose of** evaluate (against a generic database application test framework)
- **With respect to** the efficiency of the process of executing test cases
- **From the point of** view of developers and decision support managers
- **In the context of** programmers in a BI company.

3.2 Planning

3.2.1 Hypothesis Formulation

The research question for the experiment that needs to be answered is this: *"in a load context for DW, test cases perform better using the test framework compared with a generic framework?"*

To evaluate this question, it will be used a measure: Average time for Testing Framework and Generic Framework. Having the purpose and measures defined, it will be considered the hypothesis:

H_{0time}: the execution of test cases for the testing framework and the generic framework has the same efficiency. ($\mu_{GenericFrameworkTime} = \mu_{TstingFrameworkTime}$).

H_{1time}: the execution of test cases for the testing framework is more efficient than running on generic framework. ($\mu_{GenericFrameworkTime} > \mu_{TstingFrameworkTime}$).

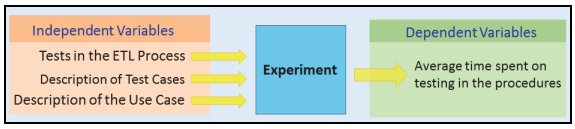

Figure 1: Dependent and Independent variables of the experiment.

Formally, the hypothesis we are trying to reject is H_{0time}. To ascertain which of the hypotheses is rejected, will be considered the dependent and independent variables that are in Figure 1.

3.2.2 Independent Variables

Next, the independent variables of the experiment are described.

Description of Test Cases Used in the Experiment: The loading routines for the DW environment are quite discussed in (Colaço, 2004; Kimball, 2002; Kimball, 2004; Kimball, 2008). Alternative approaches to the loading of dimensions can be found in (Santos, 2011). Algorithms for loading routines for the various types of dimensions can be found in (Santos et al, 2012). Test Cases categories for ETL routines are pointed in (Elgamal *et al.*, 2013; Cooper and Arbuckle, 2002). This material, together with the extensive experience of the authors in DW projects in the public and private sectors, provided the basis for the elaboration of categories and test cases to be considered by the framework. The following categories are contemplated by the framework: a) Unit tests and relationship; b) Number of records between source and destination; c) Transformations between source and destination; d) Processing of incorrect or rejected data; e) Null values processing; f) Behavior Type 1 and Type 2 for dimensions attributes; g) Hybrid approaches for the treatment of historical dimensions.

Description of the Use Case Used in the Experiment: the characteristics of the use case chosen for the validation study were based on practical situations reported by the selected programmers.

For the use case of the experiment, the goal was to generate procedure to perform loads of Staging Area, from employee table to the employee dimension. At this time, the dimension has an historical storage, Type 2 for some attributes. The other attributes are Type 1.

Table 1 shows the characteristics of the employee dimension in DW environment. For the Type 2 treatment in dimensions, new attributes are used for the historical storage. They are the start date, which represents the date on which the record was recorded; the end date which is the date when

the record is no longer current; and finally, the attribute that identifies whether it is or not a current record. These are respectively shown in Table 1, with the names of: dt_initial; dt_end and fl_current.

Tests in the ETL Process: The ETL tests used in this work, have two types of treatments for performing the experiment: 1) Generic Framework: test cases execution using DbUnit for loading data based on use case already presented earlier in a DW environment; 2) Testing Framework: execution of tests based on test case, in the SQL code for the same use case, using the proposed tool of this work, FTUnit.

Table 1: Features of the dimension dbo.dim_employee and its attributes regarding the historical storage.

Dimension Name:	**dbo.dim_employee**	
Treatment historical Type:	**Type 1 and 2**	
Attribute	**Historical Type**	**Attribute Paper**
id_employee		Surrogate Key
cod_registration		Primary Key
name	1	
cpf	1	
title	2	
job	2	
salary	2	
sector	2	
department	2	
dt_initial		Initial Date
dt_end		End Date
fl_current		Current Flag

DbUnit was select to this experiment first to have a framework proposal that is closest to the Testing Framework. Second, for convenience the experiment's programmers are already familiar with JUnit, so would be easier and efficient its use.

3.2.3 Dependent Variables

It were used a measure as a dependent variable: Average time, for testing framework and generic framework, measured using a stopwatch, considering the average time spent on testing in the procedures.

3.2.4 Participants Selection

The selection process of the participants will be done for convenience, making the type of sampling per share in which will be preserved the same characteristics of interest present in the population as a whole. The contributor to be chosen will be Qualycred (www.qualycred.com), a company that provides consulting in BI solutions for industry. This company will provide for the execution of the experiment, eight programmers with four years of experience in other areas and one year of experience working specifically with ETL for DW, in SGBD SQL SERVER.

3.2.5 Experiment Project

The experiment was projected in a paired context, in which a group will evaluate both approaches: Testing Framework and Generic Framework execution. For understanding the execution of the test to be done, ten test cases and one use case (seen in Independent Variables section) were elaborated, which will be presented in a well detailed way to the programmers.

The experiment will be separated into two groups of participants. Will be drawn 5 programmers to start the tests to the rules presented in the Employee Use Case, with the execution of Testing Framework and, shortly after, the execution of Generic Framework. The other participants in parallel, will make the tests made to rules presented in same use case, with the implementation of the Generic Framework and, shortly after, with the execution of the Testing Framework. Thus, the randomness will be enhanced, not prioritizing the manual or automated learning.

3.2.6 Instrumentation

The instrumentation process initially proceeded with the environment setup for the experiment and planning the data collect. It was conducted in a computer lab at Federal University of Sergipe - UFS. The Participants of the experiment had the same working conditions. The computers were adjusted to possess same settings. Listed below are the technology, the installed tools and artifacts used.

SQL Server 2008. It served as a basis for the storage of the identified metadata, and consequently, has been used to store the metadata of FTUnit, described in section 2.

Unit Testing Framework in Sql Code (FTUnit). The FTUnit was described section 2 of this paper. The version to be used in the experiment, due to the participants, runs Unit Testing Cases for charging procedures in SQL code, T-SQL language (Transact-SQL), involving behaviors of Types 1 and 2 for the treatment of historical dimensions.

DbUnit. Using XML files to run test cases in DbUnit, this framework has been prepared to run tests case in ETL procedures to load data to dimensions involving behaviors of Type 1 and 2. At solution the programmers developed methods that could perform test cases for the Employee Use Case,

by means of an XML file with characteristics of employee auxiliary table. To compare the results after loading the data, it was compared with other XML file containing the expected characteristics of the employee dimension, thus confirm whether or not if the data were properly loaded for the dimension.

Environment Created and Produced Artifacts. Some of the tables that make up the DW environment used in the experiments were the following: a) auxiliary employee table; b) dimension of employee with attributes Types 1 and 2. These are described below.

Figure 2 contains the representation of a load data from TB_Aux_Employee (auxiliary table of employees) to the DIM_Employee (dimension of employees) that represents a dimension of Types 1 and 2. To this dimension, the attributes that match the Type 1 are name and CPF. The attributes of Type 2 are title, job, salary, sector and department.

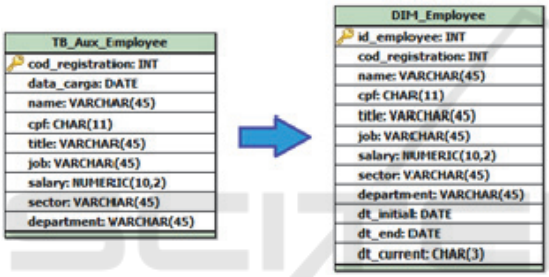

Figure 2: Charge for a dimension of employees with behaviors of Types 1 and 2.

4 EXPERIMENT OPERATION

4.1 The preparation

The following are listed the preparation steps for the execution of the experiment.

1) *DW environment Creation:* in this phase was defined and created the DW environment with the dimensional schemes and staging area. These artifacts served as the basis for the entire experiment.

2) *Definition of Test Cases for loading routines*: were defined test cases to be followed by the developers of the experiment. These test cases were applied to a loading routine previously created.

3) *Review of basic concepts of loading routines for the programmers*: a review of the loading routines, for DW environments with the selected developers, was performed.

4) *Training in Testing Framework (FTUnit):* a training with the programmers was realized to become familiar with the tool.

5) *Training in Generic Framework (DbUnit):* a training with developers was conducted for tool learning in the context of charges for DW environments.

In short, all computers were prepared with the same settings, so programmers were on the same working conditions. Moreover, it was presented to each programmer, a printed document containing a detailed description of Use Case and test cases that would be used by them, in case of any doubts.

4.2 Execution

At the end of the previous steps, the experiment was initiated, it occurred according to the plan described in section 3.

The evaluation of Testing Framework at the end of the experiment, made by the professionals, was positive, since they have commented that the use of this tool have contributed to the reduction of time in the test procedures.

1) Data Collection

Average time for Testing Framework and Generic *Framework.*

It was calculated the time spent by each developer for Testing Framework and Generic Framework tests, of all test cases for the Employee Use Case, taking into account the time for testing and all necessary settings in FTUnit. Under supervision, each programmer reported the completion and was recorded the time on a timer, used for this purpose. The result of these collected data will be presented in section 5 of this paper.

4.3 Data Validation

In order to perform the experiment, one factor was considered, Test of the ETL Process, and two treatments, execution of tests case using the FTUnit tool and DbUnit tool. Facing this context, the average of testing time was computed.

As an aid to analysis, interpretation and validation, we used two types of statistical tests, Shapiro-Wilk Test and the T Test. Shapiro-Wilk test was used to verify normality of the samples. The T test was used to compare the average of the two paired samples (Wohlin *et al*., 2000). All statistical tests were performed using the SPSS tool (SPSS, 1968).

5 RESULTS

5.1 Analysis and Data Interpretation

To answer the question of research, the following dependent variable was analyzed: average time spent on testing in the procedures.

5.1.1 Time Spent in Testing Process

Table 2 displays the results related to the testing time by each participant for the Employee Use Case. The results show that the average time of the developers for Testing Framework was 23.13 minutes, and 166.38 minutes for the Generic one.

These results suggest that the Testing Framework execution have, on average, shorter testing time, as compared with the same test procedure performed in Generic Framework by programmers with experience in the area. Thus, from this preliminary analysis of the data, it is assumed that the answer to the Research Question would be "yes". The execution of test cases using the FTunit can increase the productivity of developers during the testing process in a DW compared with DbUnit, since this tool obtained a difference of approximately 143.25 minutes. However, is not possible to make such a claim without sufficiently conclusive statistical evidence.

Thus, first, we established an apriority significance level of 0.05. The Shapiro-Wilk test ensured that the sample was normally distributed. As seen in Table 3, we found p-values of 0.672 and 0.523 for the use of testing framework and generic framework, respectively. As the p-value is the lowest possible significance with which it is possible to reject the null hypothesis, and they are larger than 0.05, we cannot reject the hypothesis that the data is normally distributed.

Finally, as the samples are dependent, the hypothesis test applied in this context was the T-Test, characterized as parametric for paired samples, which only requires normality of the samples. In Table 4, we obtained the p-value of 0.000. This means the p-value found is less than 0.0001, so we have more than 99% certainty for the valued context. Thus, it was confirmed the evidence of a difference between the averages of 143.25. As the significance test is lower than 0.05, it is possible to reject the null hypothesis. Consequently, we cannot reject the alternative hypothesis that the execution of test cases for the testing framework is more efficient than running on generic *framework*.

Table 2: Average Time (in minutes) to Execute The Tests of the use cases.

	Employee Use Case	
	Testing Framework (minutes)	**Generic Framework (minutes)**
Programmer 1	23	141
Programmer 2	27	154
Programmer 3	19	82
Programmer 4	20	161
Programmer 5	28	209
Programmer 6	21	182
Programmer 7	22	186
Programmer 8	25	216

Table 3: Shapiro-Wilk Test. (SOURCE: SPSS TOOL – IBM (SPSS, 1968)).

	Statistic	**df**	**Sig.**
FTUnit	,946	8	,672
DbUnit	,931	8	,523

Table 4: T-Test Versus Time of the Tests. (Source: SPSS Tool – IBM (SPSS, 1968)).

		Paired Differences					**t**	**df**	**Sig(2-tailed)**
		Mean	**Std. Deviation**	**Std. Error Mean**	**95% Confidence Interval of the Difference**				
					Lower	**Upper**			
Par 1	**DbUnit-FTUnit**	143,25	41,053	14,514	108,929	177,571	9,869	7	,00

5.2 Threats to Validity

In spite of having achieved statistical significance in the study, the following threats to the validity must be considered.

Threats to internal validity: the limited availability of programmers to make new use cases can be considered a threat to validity for having implemented a simple use case. In addition, although participants have been trained to use both framework*s*, they do not use it daily. This lack of constant contact with it may have affected the results, which could be even better, pro-tool. Framework*s* training were conducted at the beginning of the experiment, considering a phenomenon studied by psychology called *Demand Characterization* - which considers that an

experimental artifact may have an interpretation of the purpose of the experiment by the participants. This can lead to change of unconscious behavior, to adapt to this interpretation (Orne, 1962). According to this concept, this training could be harmed the progress of the experiment, but to mitigate this factor, can be said that had been used at least two different approaches: *The More the Merrier* and *Unobtrusive Manipulations and Measures* (Orne, 1962). Respectively, the first, to avoid bias with a single experimenter, the experiment had another researcher to conduct the experiment and an instructor for the tool, not involved with the research. The second guided us not to say which factors and metrics would be assessed, so that the participants had no clues about the research hypothesis.

Threats to external validity: The low number of participants can be a threat, since it can negatively influence the results of the experiment.

Threats to the construction validity: The Specifications for the use case and test cases may not have been very clear to the understanding of some programmers. This threat was mitigated with the prior reading and analysis of the understanding, made by four ETL developers.

6 RELATED WORK

Through literature reviews, with systematic approaches, were not found strongly related work for automated unit tests in ETL tools. Consequently, the absence of ETL tools with these characteristics may contribute to a lower integrity and a lower quality of data, essential in large banks of decision support data.

Some moderately related works also seek solutions for the automatic execution of Test Cases in DW environments. In (Elgamal; Elbastawissy; Galol-edeen, 2013) it is presented a directed models approach for automatic generation and execution of test cases based in formal models of systems. The formal model adopted is based on the UML language. The approach also depends on creating an extension of UML language that can capture the transformations used in a *Data Warehousing* process.

Once the Testing Framework generates test cases based on characteristics of the loads procedures being implemented, it can be extended and used to test load routines created for any ETL tools. So far was not found in literature any similar approach, so we could make a comparison.

It is possible find in (Krawatzeck *et al.*, 2015) an evaluation of unit testing tools suitable for data warehouse testing. The following open source tools were select based on tools that could perform test cases in a BI environment: AnyDbTest, BI.Quality, DbFit, DbUnit, NDbUnit, SQLUnit, TSQLUnit, and utPLSQL. The more similar tool to the proposed work is the DBUnit framework (DbUnit, 2016). However, this one represents a generic framework for database applications and has no particularity regarding to loading routines for a Data Warehouse environment.

7 CONCLUSIONS

Identifying and attempting to solve data quality problems in a Data Warehouse environment is one of the major obstacles faced by large enterprises in the use of Decision Support Systems. Among the many factors that contribute to poor data quality are manual data-loading routines. After delimiting some research questions, the hypothesis was raised that the tests made with Testing Framework support can contribute to quality improvement through the impact on variables such as productivity and coding errors.

To accept or reject the hypotheses presented, we presented the proposal of using a Unit Testing Framework and a Generic Framework for loading routines in a BI environment based on Data Warehouse. The experimentation's results show the use of FTUnit was more efficient than the use of DbUnit. The motivation for adopting this approach meets the problems pointed in (Elgamal *et al.*, 2013; Golfarelli and Rizzi, 2009; Myers *et al.*, 2012) is the need to adopt different strategies, considering the differences between traditional environments and DW environments, which can contribute to the adoption of testing processes.

In this context, this work presents important contributions to increasing the productivity and quality in software engineering for loading routines of DWs, and encourages experimentation in an industrial environment. The Testing Framework encapsulates a method to accelerate and improve the quality of ETL process tests based on SQL. It is noteworthy that the safe and efficient execution of procedures in SQL directly in the database is an option considered by much of the industry, requiring tools to support tests in this type of approach in software engineering.

Although this study did not show satisfactory results in the experiment for the use of DbUnit, a new approach has been set to perform test cases in DW environment. Therefore, it becomes something new in the area, since were not found any work containing the applicable implementing testing

procedures in a BI environment for a generic framework as DbUnit.

The proposed framework (FTUnit) presents test cases previously defined which cover the main categories of tests applied to loading routines. Through a set of metadata that defines the characteristics of the routines, the framework selects test cases to be applied, generates the initial states of the database, executes the routines, performs test cases, analyzes the final state of the database and generates a report with the errors encountered during the execution of each test case.

By virtue of what we have seen above and the framework innovation, the presentation of this experiment will support the adoption of the same or the creation of a similar approach for companies that use this type of strategy. Other contributions obtained were: a) Approach to implementing software testing in BI environments based on DWs; b) Test cases defined for data loading routines in BI environments; c) Testing Framework to meet the execution of unit tests in a BI environment; d) Use of DBUnit for running unit tests in BI environments based on DW; e) Experiments show the benefits of automated testing in BI environments.

As future work, it aims to extend the approach to various SQL languages, as the experiments carried out so far have been only for the T-SQL.

REFERENCES

Basili, V. and Weiss, D. (1984), *A Methodology for Collecting Valid Software Engineering Data*, In: IEEE Transactions On Software Engineering, v.10 (3): 728-738, November.

Colaço Jr. (2004), *Projetando sistemas de apoio à decisão baseados em Data Warehouse*, 1st ed., Rio de Janeiro: Axcel Books.

Cooper, R. and Arbuckle, S. (2002), *How to thoroughly test a Data Warehouse*, Proceedings of STAREAST, Orlando.

Costa, J. K. G., Santos, I. P. O., Nascimento, A. V. R. P., Colaço Jr, M (2015), *Experimentação na Indústria para Aumento da Efetividade da Construção de Procedimentos ETL em um Ambiente de Business Intelligence.* SBSI 2015, May 26–29, Goiânia, Goiás, Brazil.

DbUnit, (2016), http://DbUnit.sourceforge.net/

Deshpande, K. (2013), *Model Based Testing of Data Warehouse*, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 3.

Elgamal, N., Elbastawissy, A. and Galol-edeen, G. (2013), *Data Warehouse Testing,* EDBT/ICDT '13, Genoa, Italy.

Golfarelli, M. and Rizzi, S. A. (2009), *Comprehensive Approach to Data Warehouse Testing*, ACM 12th International Workshop on Data Warehousing and OLAP (DOLAP '09), Hong Kong, China.

Inmon, W. H. (2005), *Building the Data Warehouse*. 4th ed., Indianapolis, Indiana: Wiley Publishing Inc.

Kimball, R. (2004), *The Data Warehouse ETL Toolkit*. 1st ed., Wiley India (P) Ltd.

Kimball, R. and Ross, M. (2002), *The Data Warehouse toolkit: The complete Guide to Dimensional Modeling*, 2nd ed., John Wiley and Sons, Inc.

Kimball, R., Ross, R. M. and Thomthwaite, W. (2008), *The Data Warehouse lifecycle toolkit*, 2nd. ed., Indianapolis, Indiana: Wiley Publishing Inc.

Krawatzeck, R.; Tetzner, A. and Dinter, B. (2015), *An Evaluation Of Open Source Unit Testing Tools Suitable For Data Warehouse Testing,* The 19th Pacific Asia Conference on Information Systems (PACIS).

Myers, G. J., Badgett, T. and Sandler, C. (2012), *The Art Of Software Testing*, 3rd ed., New Jersey: Wiley.

Orne, M. T. (1962), *Sobre a psicologia social da experiência psicológica: Com referência particular para exigir características e suas implicações.*

Pressman, R. S. (2011), *Engenharia de software: Uma abordagem profissional,* 7th ed., São Paulo: AMGH Editora Ltda.

Ranjit S. and Kawaljeet, S. (2010), *A Descriptive Classification of Causes of Data Quality Problems in Data Warehousing*, 7 v. IJCSI International Journal Of Computer Science Issues.

Santos, I. P. O., Costa, J. K. G., Nascimento, A. V. R. P., Colaço Jr, M., (2012), *Desevolvimento e Avaliação de uma Ferramenta de Geração Automática de Código para Ambientes de Apoio à Decisão*. In: XII WTICG, XII ERBASE (2012).

Santos, I. P. O., Nascimento, A. V. R. P., Costa, J. K. G., Colaço Jr., M., Pereira, W. P. (2016), *Experimentation in the Industry for Automation of Unit Testing in a Business Intelligence Environment*. SEKE the 28th International Conference on Software Engineering and Knowledge Engineering. California, USA.

Santos, V. and Belo, O. (2011), *No Need to Type Slowly Changing Dimensions*, IADIS International Conference Information Systems.

Singh, R. and Singh, K. (2010), *A Descriptive Classification of Causes of Data Quality Problems in Data Warehouse*. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 2.

Sommerville, I. (2011), *Engenharia de Software*. 9th ed., São Paulo: Pearson.

SPSS, IBM Software, (1968), *Statistical Package for the Social Sciences,* http://goo.gl/eXfcT3.

Wohlin, C., et al. (2000), *Experimentation in Software Engineering: An introduction*. USA: Kluwer Academic Publishers.