Applying Acceptance Test Driven Development to a Problem Based Learning Academic Real-Time System

Luiz Felipe Simoes Hoffmann, Luiz Eduardo Guarino de Vasconcelos, Etienne Lamas, Adilson Marques da Cunha, Luiz Alberto Vieira Dias
Computer Science Division
Aeronautics Institute of Technology, ITA
Sao Jose dos Campos, Brazil
{lfelipe.sh, du.guarino, etienelamas}@gmail.com, {cunha, vdias}@ita.br

Abstract—This paper focuses on the importance of test driven development, in an interdisciplinary academic case study, involving unauthorized access and fraud. In it, Problem Based Learning (PBL) enabled the organization of an environment for diverse knowledge and skills, favorable to experimentation and prototyping of new models, methods, techniques, and tools. The application of the principles of agile methods aimed at overcoming difficulties related to the computer systems development, such as requirement changes, uncertainties, and risks, in order to improve quality. Unlike traditional methodologies, tests were used since the beginning of the development lifecycle with Acceptance Test Driven Development (ATDD). The agile testing quadrants guided the categorized elaboration of tests during unit testing, integration testing, system testing, and acceptance testing phases. The main contribution of this article was the integrated application of PBL, ATDD, and agile development in the case study of a Real-Time Embedded System for Unauthorized Access Control and Fraud Detection (Sistema Embarcado de Tempo Real para Controle de Acessos Indevidos e Deteccao de Fraudes – SETRAIF).

Keywords—unauthorized access control and fraud detection; problem based learning; agile methods; acceptance test driven development; agile testing quadrants

I. INTRODUCTION

Financial institutions lose annually billions of dollars resulting from unauthorized access and fraud on electronic commerce transactions. Fraudsters and cybercriminals are often ahead of these companies’ technological security layers. The challenge arises in developing preventive solutions and forms of authentication without hindering access to the offered services [1].

Computer systems are susceptible to defects that when identified belatedly, increase development and maintenance costs, as well as threaten projects’ integrities [2].

Agile methods have test driven development as one of their principles [3]. It keeps technical debt under control, improves return on investment, and reduces risks to users and companies [4].

Problem Based Learning (PBL) has been object of study in recent years, with the purpose to verify its acceptability. Results suggest that the practice of PBL reveals unseen potentials and provides a dynamic learning process [5].

In this context, this paper presents a proposal for a test driven development applied to a computer system prototype using PBL in an interdisciplinary environment. Section II introduces the case study. Section III describes the methodology used in the development. Sections IV to VII emphasize the importance of test driven development. Finally, section VIII summarizes the main conclusions.

II. CASE STUDY – UNAUTHORIZED ACCESS AND FRAUD

To academically enable the scope of the domain knowledge of this case study, involving unauthorized access and fraud, a prototype of a Real-Time Embedded System for Unauthorized Access Control and Fraud Detection (Sistema Embarcado de Tempo Real para Controle de Acessos Indevidos e Deteccao de Fraudes – SETRAIF) was designed and developed.

The SETRAIF is an electronic commercial transactions authentication system. It consists of four devices, as shown in Fig. 1:

i) Transactions Mobile Device (Dispositivo Movel de Transacoes – DMT);
ii) Cloud Communication Device (Dispositivo de Comunicacao na Nuvem – DCN);
iii) Access Control Device (Dispositivo de Controle de Acesso – DCA); and
iv) Fraud Control Device (Dispositivo de Controle de Fraudes – DCF).

The DMT provides electronic transactions, in a practical and reliable way, with the users’ mobile devices (e.g., cell phones and tablets). It comprises two modules, the customer module (DMT-C) and the vendor module (DMT-V).

First, the customer gets the product’s information by its Quick Response Code (QR Code) [6] and electronically communicates his or her buying interest to the vendor.

Subsequently, the DMT-V module acquires the data of the product, the customer, and the vendor, as well as their geolocation. The DMT-V asks the DCN for authorization (via token) to sell the product. If it is confirmed, the order is released to the customer.
The DCN, which is a communication layer with the Internet (cloud) on the customer device, provides the secure exchange (encrypted) of information between the DMT and the DCA.

From this point, the DCA uses the information from the vendor and customer devices to validate or block access. On success, the information is sent to the DCF and later to a credit card company.

Completing the process, the DCF verifies the legitimacy of the transaction, quickly and reliably, based on the customer’s behavior and geographic history, to reduce the risks from fraud for those involved.

The development of each device took into consideration: testability, best practices of software engineering, and a quality assurance plan to ensure the quality of the product, the process, and the service.

The completion of the case study occurred in seventeen weeks on the second half of 2012, at Aeronautics Institute of Technology – ITA – Brazil.

Professors, researchers, students of the undergraduate program in computer engineering, and students of the graduate program in electrical engineering and computer engineering were involved.

Four courses contributed to the accomplishment of this project: CES-63 Embedded Systems (undergraduate course); CE-235 Real-Time Embedded Systems; CE-230 Software Quality, Reliability, and Safety; and CE-237 Advanced Topics in Software Testing.

III. DEVELOPMENT METHODOLOGY

The following methodologies and tools were used for the development of the SETRAIF prototype: the Scrum methodology [7]; adaptations of the Rational Unified Process (RUP) [8]; the Unified Modeling Language (UML) [9]; and Computer-Aided Software Engineering (CASE) [8].

Scrum is based on a framework for iterative and incremental development [10]. The concept behind it is the simplification of project management [2].

It provides a structure of roles, meetings (ceremonies), rules, and artifacts in which teams are responsible for defining and adapting their process [10].

According to Stober and Hansmann [2], Scrum has three roles: product owner, scrum master, and team.

A student developer team was established for each one of the SETRAIF’s devices. Each team had a professor as a product owner and a scrum master, preferably experienced, represented by a student from the CE-235 Real-Time Embedded Systems discipline.

In Scrum, sprint characterizes the development in small and manageable iterations [2]. To complete the SETRAIF prototype in seventeen weeks, its scope was reduced and two sprints of four weeks each were defined.

As with traditional projects, requirements must be gathered, prioritized, and documented [2]. Product Backlog is the Scrum document that represents a prioritized list of requirements describing, in the customer’s terminology, what he or she wants [7].


Following, the planning poker technique [12] conducted the story point estimation, which measures the development effort, allowing the distribution of the user stories in two sprint backlogs.

Scrum meetings overcame the clarification of doubts, usually pertinent of interdisciplinary activities, such as: sprint planning; daily scrum; sprint review; and sprint retrospective meetings [2].

Since some of the team members were geographically dispersed, the use of collaborative tools for virtual meetings, like e-mail and Google Hangout [13], became indispensable.

The monitoring of the two sprints’ progress took place with the sprint burndown technique [7].

IV. TEST DRIVEN DEVELOPMENT

Quality is not equal to test. Quality is achieved by appropriately mixing the development and testing activities until one is indistinguishable from the other [14].

In this case, testing must be an unavoidable aspect of development. From the integration of development and testing, quality is improved [14].
Therefore, testing is a cross-functional activity that involves the whole team and should be continuously done from the beginning of the project [15].

In agile development, two kinds of teams are usually identified: customer team and developer team [3].

The customer team includes business experts, product owners, product managers, and other persons related to the business side of the project. This team communicates and collaborates with the developer team throughout iterations, writing stories, drawing examples, and reviewing finished stories [3].

Everyone involved in delivering code is a developer and is part of the developer team. Agile principles encourage team members to take multiple activities. However, each team needs to decide what expertise its projects require [3].

Testers are integral members of the customer team, gathering requirements, helping the customers to express their needs, and advocating for quality on their behalf. They are also part of the developer team, collaborating with developers to automate tests and assisting them in delivering the maximum business value [3][15].

Nevertheless, testing should not be subservient to development. Testers must have technical and business knowledge, as well as acting autonomously, based on the priorities, complexities, and product needs [14].

As a result, the “Advanced Topics in Software Testing” and the “Software Quality, Reliability, and Safety” disciplines played an important role for the SETRAIF prototype quality assurance.

Testing activities occurred in both sprints, as presented in details in sections V to VII.

V. TEST PLANNING

According to Whittaker et al. [14], the value of the test plan is questionable, because this tends to get outdated quickly. The real value lies in the activity of developing the test plan, which enables the analysis of the problems and their possible solutions. The test plan should be directed to test cases.

The development of a test plan, objective and concise, for each sprint took into account only the specific points of the current release.

The test plans encompassed the following items: introduction; objectives; scope; test approach; test environment; risks; and test cases [3].

Two experiments were carried out: one using the test matrix [3][16] technique and the other with the ACC (Attribute Component Capability) [14] technique.

A test matrix can be used as an alternative to the test plan, because it gives a quick review of the testing required. On one side, it lists what is to be tested; on the other, it indicates which test is to be performed, or how will the software be tested.

ACC is a risk-based test planning that can be expressed as follows:

\[ \text{Risk} = \text{Impact} \times \text{Frequency}. \]  

(1)

In this technique, attributes represent the adjectives of the system. They are the qualities and characteristics that promote and distinguish the product from the competition.

Components are the nouns of the system describing the building blocks that together constitute the system in question.

Capabilities are the verbs of the system. They represent the actions the system performs at the command user. Capabilities are responses to inputs, answers to queries, and activities accomplished on behalf of the user.

The most important aspect of capabilities is that they are testable.

ACC is not about completeness. The main point is to identify, quickly and iteratively, the risks (Fig. 2 adapted from Far [17]) of the system and the strategy needed to address them.

VI. TESTING STRATEGY

The design of a testing strategy is, essentially, a process of identifying and prioritizing project risks and deciding what actions to take to mitigate them [15].

Software quality has many dimensions, each requiring a different testing approach [3].

The identification of the testing strategy and, consequently, the tests cases for the SETRAIF prototype was based on the agile testing quadrants [3], presented in Fig. 3.

They have arisen initially with Brian Marick [3][15] and are widely used to model the various types of tests in order to deliver a high quality application.

The “y” axis divides the matrix into tests that support the team and the tests that critique the product. The “x” axis categorizes the business-facing and technology-facing tests.

The order in which these quadrants are numbered has no relationship to when the different types of testing are done. This order depends on each project’s risks. However, agile development starts with customer tests (e.g., Acceptance Test Driven Development [11][18] approach), which tell the team what to code [3].

Figure 2. Risk.
According to Crispin and Gregory [3], about every development team, agile or not, struggles with requirements elicitation. Traditional teams might invest months in requirements analysis and specification, to find out later that they are incorrect or outdated. Teams in chaos might have no requirements at all, with programmers making their best guess as to how a feature should be developed.

Agile development embraces change, but what happens when requirements change during an iteration? The overcoming of these difficulties has been object of study by professors and students of ITA and achieved, progressively, with the practice of PBL.

New features usually begin with high-level user stories without implementation details [3]. Nonetheless, later these details should be specified.

ATDD is a collaborative requirements discovery approach, based on lean and agile principles, in which acceptance tests are created by the customer, developer and tester. This strategy is called triad and performed before the implementation of the requirements [11].

Fig. 4 [19] presents the ATDD steps mapped to Scrum iteration.

This step progress has benefited from various ATDD’s artifacts and techniques, such as: charter, persona, scenarios, and use cases. More than actual testing, the purpose was the requirements clarification [11].

Once reached the requirements clarification, the Theme Screening and the Theme Scoring [20] techniques enabled the prioritization of those considered most important for testing.

Following, it started the preparation of the acceptance tests. Automating an acceptance test with its respective requirement is known as executable specification [11][18].

From this point, other testing techniques guided the coverage of the remaining testing quadrants. They are: Boundary Value Testing; Decision Table Testing; Pairwise Testing; State Transition Testing; Domain Analysis Testing [21]; Performance Testing [22]; Load Testing [23]; and Security Testing [24].

Not enough time to test properly, difficulty in determining the expected results of each test, and the impossibility of testing everything represent some of the challenges in the testing activity [21].

Those techniques helped in the identification and documentation of complex business rules, in the view of the system as a whole, and in a significant reduction of test cases to achieve more efficient results.

Finally, during the preparation of the testing strategy for the SETRAIF prototype, it was noted the importance and challenge in the definition of the automation tools, the test data design, and the testing environment.

VII. RUNNING TESTS AND TESTS RESULTS

Automating and running test cases for the SETRAIF prototype took place in both sprints, during the unit testing, integration testing, system testing, and acceptance testing phases [3], in order to increase the product’s quality.

Among the various tools used for this purpose, the prominent were: Robot Framework [19], Selenium, and Coverage [23].

Robot Framework is a generic test automation framework for acceptance testing and ATDD. It utilizes keyword driven testing approach, as well as data driven and Behavior Driven Development (BDD) [18] approaches. Pekka Klärk at Nokia Siemens created Robot Framework in 2005.

Fig. 5, 6, and 7 illustrate, respectively, test cases with BDD approach, a running test case with Robot Framework and Selenium tools, and its Robot Framework tests report.

Selenium is a browser automation tool, primarily used for testing purposes. It supports some of the largest browsers and operating systems vendors, as well as integrates with Robot Framework.

Coverage is a tool that performs coverage analysis (i.e., measures which lines in a program are executed and which lines are not). It helps in checking the adequacy and completeness of test cases.
Coding style facilitates common understanding of code and eases the integration of newcomers in the projects, and for this reason it is considered an important quality measure [25]. Companies like Google take coding style and code reviews seriously, having a committee to guarantee its readability [14].

It is common that, during the coding phase, developers in search of a solution or in the integration process encounter themselves with code duplication. Automating tests for the identification of code duplication helps in the system maintenance and improves coding habits [25].

Performance and load testing are crucial for real-time embedded systems design and, unfortunately, are usually often ignored [22].

Performance testing is related to the amount of time that a function takes to execute [22].

On the other hand, load testing focuses on the scalability of the system. Normally, the load conditions are varied, as well as the duration of the tests or any other conditions the system may be subjected [23].

The usability testing on the SETRAIF prototype relate to the third agile testing quadrant, that is, are intended to criticize the product. Tools like persona or the developers’ intuition help looking at the product with the end user in mind [3].

With the exception of usability testing, all others were automated.

Table 1 and Fig. 8 present the results of a battery of tests performed on a code fragment during development. The techniques used were:

i) Coding Style (CS);
ii) Duplicated Code (DC);
iii) Unit Testing (UT);
iv) Coverage Analysis (CA);
v) Acceptance Testing (AT);
vi) Load Testing (LT); and
vii) Usability Testing (US).

Table 1 and Fig. 8 present the results of a battery of tests performed on a code fragment during development. The techniques used were:

<table>
<thead>
<tr>
<th>Test Approach</th>
<th>Quantity</th>
<th>Pass</th>
<th>Fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>4</td>
<td>0 (0%)</td>
<td>4 (100%)</td>
</tr>
<tr>
<td>DC</td>
<td>1</td>
<td>1 (100%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>UT</td>
<td>21</td>
<td>17 (80.95%)</td>
<td>4 (19.05%)</td>
</tr>
<tr>
<td>CA</td>
<td>4</td>
<td>4 (100%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>AT</td>
<td>14</td>
<td>9 (64.29%)</td>
<td>5 (35.71%)</td>
</tr>
<tr>
<td>LT</td>
<td>1</td>
<td>1 (100%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>US</td>
<td>7</td>
<td>5 (71.43%)</td>
<td>2 (28.57%)</td>
</tr>
</tbody>
</table>
This paper has described a proposal for a test driven development of a computer system prototype using PBL, in an interdisciplinary environment.

Developing a computer system prototype with PBL requires dedication and commitment to overcome the various obstacles presented. The interdisciplinary approach provides that students, who do not know each other, exchange different knowledge and priorities.

ATDD proved to be essential to the project, facilitating the requirements specification, but mainly raising awareness, to those involved, of the importance of testing as an auxiliary mechanism for quality assurance.

The authors recommend the continuation of this research in the following semesters for validating its suitability.

ACKNOWLEDGMENT

The authors of this paper thank to the Brazilian Aeronautics Institute of Technology (ITA), the Casimiro Montenegro Filho Foundation, and the 2RP Company for supporting this challenge.

REFERENCES

[17] I. E. Far, “Google Test Tools,” Google Test Automation Conference [Online]. Available: <https://docs.google.com/presentation/d/1IZWHqK8f_fudh_kxU3oTYs968d0qmn6Y_0WFVDof6ms/edit?pli=1#slide=id.g4ecbad_0_0> 05.25.2013.