USING PBL TO DEVELOP SOFTWARE TEST ENGINEERS

Caliane de O. Figueredo, Simone C. dos Santos, Paulo H. M. Borba
CIn - Centro de Informática da UFPE., Universidade Federal de Pernambuco
Prof. Luís Freire Avenue, Recife, Brazil
cof@cin.ufpe.br, scs@cin.ufpe.br, phmb@cin.ufpe.br

Gustavo H.S. Alexandre
C.E.S.A.R., Centro de Estudos e Sistemas Avançados do Recife
220, Bione Street, Recife, Brazil
gugahenrique@gmail.com

ABSTRACT
The growing presence of the software in the products and services consumed daily by the society demands a level of completely dependent quality not only of technology, but of its development process and of the involved professionals. By focusing on the professionals responsible for quality assurance, as the Test Engineer, the skills and competences of these need to be developed on basis of a vision very critical and detailed of the problem. The Test Engineer needs to be an “explorer” of the solution, discovering hidden bugs and looking to elimination of defects of the applications. In this context, this article proposes an approach of teaching focuses on training of “test” discipline that make use of problem-based learning to develop real skills required, supported by processes of planning and continuous assessment, in a computer aided software factory. To prove the applicability of this proposal, an empirical study was developed with positive results in teaching the discipline of “exploratory testing”.

KEY WORDS
Problem-Based Learning, Software Engineering, Software Testing.

1. Introduction
Currently, there is a growing demand for trained professionals in software testing because of the demands for quality products, usually associated with the elimination of defects, and the quality of software processes, related to the preservation of consistency of requirements defined by customer throughout the stages of software development.

From the academic standpoint, generally the disciplines of software testing in Computation degree courses follow the traditional process of forming of knowledge with focus in presentation of concepts, techniques and simple practices driven by a teacher, this one as principal agent. However, this approach focused on the teacher transforms the student into a passive agent.

Additionally, the practical activities are normally defined based on scenarios and situations designed by the teacher, almost always distant from the complexity of real-world problems. Finally, in this teaching and learning process, the evaluations of the students also finish focused only in the theoretical aspects of the knowledge area by checking the learning of concepts and fundamentals of software testing without considering the skill for the practical application of these concepts, as well as the process of creating solutions to problems.

As an alternative to traditional methods of teaching, the teaching method PBL - Problem Based Learning [1] has been applied in different market areas, since the medical field is linked to its origin, to the areas of engineering and technology [2]. In [3], the author defines the problem based learning (PBL) as a strategy where students work in teams in order to solve problems and also encourage the development of skills and attitudes, including teamwork, self initiative, and cooperation. In this model, the student changes his role in the learning process, passing of passive receiver for active, responsible for their learning. Furthermore, this methodology prescribes an environment based on software factories (roles, tools and standardized processes), where students are immersed in practice, interacting with friends and peers, inserted into a functional learning process for solution of real problems that match current real-world needs.

Despite the obvious benefits of PBL, it is important to note that adoption of this methodology is often confused with practical experiments in which students are left to fend for themselves, with little interaction with the teachers and low content support originating from disciplines. An effective PBL methodology is strongly process-oriented, since the approach needs to be planned to ensure that theory and practice will walk together and aligned. Moreover, learning needs to be accompanied by instruments that can assess its effectiveness [4], [5], [6].
In this context, this article describes a teaching approach based on PBL to train professionals in software testing, here “test engineers”. The adoption of PBL is accomplished through the creation of a software factory, supported by teaching processes, actual projects, multidirectional interactions between students and teachers and continuous assessment. An experiment conducted in a Training Program in Exploratory Testing (TPET) is ongoing, with some results that indicate important contributions both in the construction of the learning process more effective as the monitoring and validation of this process.

2. Planning the Problem-based Learning

Whereas PBL is essentially process-oriented, dumping practice in which students are subjected requires a teaching planning that involves defining the structure of the practical environment of educational goals and objectives, roles of human capital involved (students and teachers), computer tools support, and evaluation of results.

To realize this plan, the main elements of empowerment were identified, with reference to the work in [4].

2.1 Training Plan

In planning the training, targets and an action plan to achieve the proposed objectives are defined, as information from the profile of the class, student numbers and course proposal. This action plan included the following:

- **Training objectives**: defining the skills and abilities will be developed, according to the profile that you want to build and market demand;

- **Project Selection**: consists in selecting projects for software testing with real focus on functional learning from real problems, involving the participation of real customers.

- **Environment and infrastructure**: activities related to purchase and installation of infrastructure and tools needed to create an environment similar to a software factory focused on software testing stage.

2.2 Roles and Test Teams

The project execution involve the participation of five key players: the **client**, responsible for the demand of the test services and validation of the results; the **student**, who compose the testing team and is responsible for the execution of the projects of tests and achieving the results agreed with the clients; the **tutor**, responsible for teaching and preparing content of disciplines, performing an evaluator and facilitator role of student learning; the **consultant**, who acts like a specialist of tests of software helping in the projection of the activities, techniques and necessary trainings to the execution of the activities of the project; the **manager**, responsible for managing the software factory, performing activities of monitoring and control of projects.

The formation of the teams consists of the division of students into small and heterogeneous teams. According to the methodology of dividing students into teams described in Santos et al. [4], it is important consider affinities, skills and abilities of each student in order to avoid concentration and / or lack of skills / abilities in the same team. For each new project, the allocations of the roles of each student inside the team of tests are performed, such as: manager of tests, analyst of tests and executioner of tests. In spite of this allocation of papers, each integrant is involved in all the stages of the tests process.

2.3 Content Plan

In this activity, learning modules are defined as guides and support for the human capital involved in the training. A **Learning Module** (LM) is a set of practical activities, training, techniques and tools aimed at the theoretical and practical learning of specific content. The setting up of the LM involves the following steps: (1) define educational objectives of the module (2) define a test strategy from the educational objectives of the module (3) define a test process (activities, artifacts, roles and responsibilities) aligned to the strategy of testing the module, (4) choose the tools necessary to implement the activities planned for the module, (5) list the training previously identified as necessary for understanding and practical application of content covered in the module.

Figure 1 illustrates an example of a learning module that addresses specific software test content. Being an independent module, can be applied to any project of tests that use the test strategy defined in the module.

![Figure 1. Learning Module](image)

Each learning module has a version, a context and an associated level of maturity, which may evolve according to these characteristics and the software project of which is related, as illustrated in Figure 2.
The points represent the learning modules, the line T indicates the progress of the module over time (e.g., 1.0, 2.0, associated with the improvement or modification of its content), the C line indicates the evolution of the module in relation to the context (For example, the type of application that can be Desktop, Web or Mobile), and the line M indicates the progress of the module in relation to the maturity level of the organization (e.g., basic, intermediate or advanced, depending on the tools and techniques associated). The points surrounded by squares indicate the learning modules which have been performed.

By being specific, by version, context, and maturity, the modules allow monitoring the evolution of student learning during the implementation of project activities, enabling corrective actions or improvements might be realized.

### 2.4 Learning Assessment

In [7], the author describes the teaching/learning process as a cyclical process that begins with the definition of educational objectives and continues with the choice of methods and criteria of the assessment process.

In this circular process is performed three methods of assessment defined by Bloom [8], cited in [7]: diagnostic, used to determine if the student has the necessary prerequisites for the acquisition of new expertise; formative, held in order to verify that the student is achieving the objectives established during the course and; summative, used to classify a student, carried out to the end of a course, school period or unity of teaching.

In the context of training in software testing via learning modules and implementation of real projects, this process is divided into well defined phases, as illustrated in Figure 3.

In the step Preparation, the criteria and methods of assessment for each learning module are defined, aligned to the educational objectives initially planned:
- **Criteria**: it consists of the definition of clear form and lens of what it will be evaluated;
- **Assessment methods**: Choose the method (technique) and instruments that are best suited to assess what has been defined.

The Diagnostic step consists in preparing and applying evaluations what they make possible to identify the level of current knowledge of the student related to the content of the learning module, besides detect prerequisites for the acquisition of new expertise.

The step of Monitoring is dedicated to the elaboration and application of formative assessments. Continuous assessments are made, in order to identify gaps in learning with respect the aspects described in [9]:
- **Content**: evaluation based on the content taught in training, noting the theoretical knowledge acquired;
- **Procedure**: evaluation based on monitoring the process activities and the meeting deadlines, noting the oral language, posture, strong points and improvement;
- **Deliveries**: evaluation of the artifacts produced and status reports, noting criteria such as standardization, organization, accuracy, timeliness, oratory, among others.

Finally, in the step of Classifications are developed and applied character summative assessments, to verify the learning outcomes achieved by students, in accordance with the established criteria.

The final concept for students includes notes of formative assessments (content, process, and deliveries) and summative. This final concept is calculated by:

$$\text{Final concept} = (0.6 \times \text{formative assessment}) + (0.4 \times \text{summative evaluation})$$

According to Alexandre in [7], this evaluation methodology is not only concerned with the condition of pass/fail the student, but worry, especially in monitoring the behavior of the student before an evaluation, also providing resources to enable it to deepen and improve their knowledge on weaknesses identified by the evaluation.

### 3. Empirical Study

This section describes a case study conducted in a pilot project for training software testing, titled "Exploratory Testing Training – ETT", defined as part of a broader training program in software engineering. This training program was implemented by the Laboratory of Productivity of Software (LabPS) of INES (National Institute of Science and Technology for Software
Engineering) at Center of Informatics Federal University of Pernambuco (UFPE-CIN), in collaboration with partners of the institute innovation C.E.S.A.R (Recife Center for Advanced Studies and Systems) and employees of Software Industry in Pernambuco, Brazil.

The ETT project was designed for undergraduate students in computer science and related fields, and aims to offer to the students an opportunity to practice the key concepts and processes of software testing in order to provide the field and dissemination of technical knowledge in the area.

This project was conducted from May/2010 to September/2010, initially with 10 undergraduate students divided into two testing teams. Both teams were tasked to test a CASE tool (a tool to support functional testing), maintaining the same functionalities. The challenge was to find bugs in the tool using the testing exploratory technique, but in a controlled, efficient and creative mode, using a testing tool as object of study to learn software test. It’s important to emphasize that the exploratory testing stimulates the creativity and investigative characteristics, challenging the tester in test design of better and more effective quality.

3.1 Training Plan

The action plan consisted in the definition of educational objectives, selection of actual projects for training and preparation of test environment, as is detailed following:

- **Objectives of training**: to train test engineers in key concepts and techniques of exploratory testing;
- **Selection of actual projects**: this activity consisted in the selection of the tool named TaRGeT (Tool for Generation of Test and Requirement) as a tool to be tested by students in the ETT project. The main feature of this tool is the generation of test cases in an automated mode. Therefore, besides being the object of testing the project, TaRGeT is also a tool to support testing, being used on previous projects of this training program;
- **Environment and infrastructure**: to implement this project, there was no need to purchase equipment, since the LabPS already had the infrastructure needed for training and execution of tests. To prepare the test environment was necessary to install and configure the TaRGeT tool at the 5 computers available for the teams, considering that two team composed by 5 members were sharing these computers.

3.2 Test Teams and Roles Definition

In the ETT project, the students were organized into two teams of five members, one staff assigned in the morning (Scan team) and the other in the afternoon (Cambio team). These teams were heterogeneous, composed of students from different institutions and different periods of undergraduate Computing course. Each student took a role on the team, however, actively participating in all activities and phases of the testing process that was set. Worth remembering that due to the focus of this program was exploratory testing, there is no distinction between the designer and tester, and therefore, each team was consisted of four testers and a test manager.

Employees of the LabPS and others partners were assigned the roles of tutors and consultants. The calls were directed to professional with academic and practical experiences, often engaged in graduate programs. A dedicated professional took the role of project manager, leading the implementation of the methodology and the process of assessing students.

3.3 Content Plan

The Learning Module for the ETT project was set by the project manager in conjunction with collaborators of LabPS (tutors and consultants involved in the program). The content of the module is arranged as follows:

- **Educational Objectives**: on the conclusion of the learning module, it is expected that the student is able to know, understand and apply the concepts, processes and techniques of exploratory testing, conducted in conjunction with the modeling techniques of Black Box testing. In addition, it’s necessary to develop oral and written communication and teamwork skills among the students;
- **Test Strategy**: Second Kaner, Bach and Pettichord in [10], test strategy refers to the set of ideas that guides the test designer throughout the project. In this module, the test strategy discussed the techniques used and the level of testing. A test technique is about how to test. May be static (does not involve running tests), such as reviews and static analysis; or dynamic (involving execution). Examples of dynamic techniques are Black Box testing based on specifications (or behaviors); White box test, performed upon how the software was built and; the Exploratory tests, based on the skills, insights and experiences of the tester, without the need for scripts of test [11]. For this project, the learning module has focused on learning and applying the following techniques:
  - **Session Based & the Charter**: it is an exploratory test technique used to make exploratory testing more effective and clearer goals. In this approach, exploratory testing is conducted in sessions (time slots) of about 90 minutes, ranging from 60 to 120 minutes. Each session should perform a task previously defined (charter), describing the intent of what should be tested instead how testing should be performed.
• Modeling Techniques of the Black Box Testing: techniques that help the tester to select test scenarios more accurately and is effective in detecting certain types of errors. The term Black box means that no requires knowledge of the internal structure of the product.

• Levels or stages of tests: concern when testing, or at what stage of the development process given test should be done. Examples of levels are most commonly exploited are the Unit tests, applied to the code by programmers; System test, run the system as a whole to validate the execution of its functions; test of acceptance, performed before implantation of the software. This module has focused on testing the exploratory level of System testing.

• Test process: addresses the roles, phases and artifacts related to the exploratory testing activity.
  
  o The Roles: the roles defined for this module are the Test Manager and the Tester. The test manager is responsible for planning, managing and controlling the activities, schedule and team tests, as well as evaluating the test results. In the context of exploratory testing, the tester is the designer and executor of tests at the same time, it is he who creates and runs the test.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Test plan</td>
</tr>
<tr>
<td>Design</td>
<td>Charters</td>
</tr>
<tr>
<td>Session</td>
<td>Test notes</td>
</tr>
<tr>
<td>Debrief</td>
<td>Ideas for new testing sessions</td>
</tr>
<tr>
<td></td>
<td>Evaluation report of the tests</td>
</tr>
<tr>
<td>Bug Tracking</td>
<td>Incident report</td>
</tr>
</tbody>
</table>

Figure 4. Artifacts delivered in each activity

  o Phases and the artifacts: (1) Planning, involving the definition of the scope of testing and preparation of test plan; (2) Design, concern the creation of charters in which the test ideas and planning are registered by the tester, (3) Session is the implementation of the charter, where each session is based on a charter, including the time of preparing the testing environment, investigation, execution and recording of faults (bugs); (4) Debrief is a quick conversation between the test manager and tester about the results of the session in order to verify if what was inside the plan was actually tested, and to plan new test sessions, (5) Bug Tracking, relate to the follow-up phase of recorded bugs, until they reach a conclude state. The Figure 4 illustrates the artifacts delivered to each stage of this process.

• Tools: the TestLink (managing and running tests) and Mantis (incident management) tools were defined as tools to support testing and; the tool DotProject was used to manage the test project.

• Training plan: they were given a set of 4 training designed to boost student learning and support to project activities:
  
  o DotProject: project management tool;
  o Exploratory Testing: key concepts about exploratory testing;
  o Exploratory Testing SBC (Session Based & Charter);
  o Modeling Techniques of the Black Box Testing: Techniques for selection and development of scenarios for black box testing (equivalence classes, limits analysis, decision tables, use cases).

It is noteworthy that the Learning Module of the ETT project was performed as the second module training program. The first module, entitled "Functional Testing and Black Box, via GUI", addressed the basics concepts of software testing in a project called “Flip”, but the implementation of this module did not include mechanisms for evaluating the learning that assessed their performance, keeping out of this case study.

3.4 The Assessment Process

The assessment process defined in the methodology was applied in the ETT project, allowing a more concrete evaluation of the results in relation to the learning objectives proposed for this module. However, although the project has started with the participation of 10 students, only 6 completed the module in totally. The motives of these four students to let out this program were entirely personals: the first one due to conclusion of the undergraduate course, another decided to live in the other city and the other two were selected in curricular programs.

Considering that, at the end of the module, the students should be able to know, understand and correctly apply the concepts and techniques taught in the exploratory testing, in the Preparation stage the following criteria and methods of assessment were defined:

• Criteria: the criteria for evaluation of artifacts and status report (presentation) were defined. The artifacts generated by students were evaluated for the standardization, organization, completeness, correctness, creativity, appropriateness and timeliness. In presentations, the criteria considered were orality, posture, appearance, slides, monitoring of activities, strengths and improvement points.
• **Assessment methods:** individual or group assessment made use of evaluation instruments such as the objective tests for diagnostic and formative assessments, subjective tests, exercises or oral presentation to formative assessments.

In the beginning of the Learning Module was applied a diagnostic evaluation (step *Diagnostic*), with the objective of verifying the level of students’ prior knowledge regarding the content of Learning Module. A percentage of 50% of grades of these students were below 7.0, indicating little knowledge of the content to be addressed in the module.

After the diagnostic evaluation and initiation of activities, the *Formative* assessments (content, process and supply) were also applied, initiating the step of Monitoring. For each formative assessment conducted, the students received feedback from tutors in person, which is essential for the development of students, enabling the identification of gaps in learning.

Analyzing the data obtained from the formative assessment of the content aspect giving rise to the graph in Figure 5, one can observe that, except for assessments about the DotProject tool and Black Box Testing Techniques trainings, both teams evolve their notes throughout the process, converging for the same level of learning in the last formative assessment. Still, the grades related to the tools are above 7.0 which was the average required for the program.

In Figure 5, you can also see a drop in grades of the two teams in the evaluation of Black Box Testing Techniques, both with similar efficiency. This decrease can be explained by the complexity of the content measured and the fact it was more difficult to evaluate the module. Usually this content is fixed from practical activities. However, it was also observed that after the feedback of this assessment, students were able to apply the black box testing techniques more easily and quickly, since they could better understand the techniques from the questions they had during the assessment and during the practical activity. The same performance was also observed in other evaluations.

At the end of the Learning Module, the students answered a summative objective test, with issues related to the content seen in the module as whole, completing the evaluation process with the step of *Classification*. Of the six students evaluated, only one took a note below recommended average.

It was also observed in Table 1 that the notes in formative assessments of all students were above 8.0, while in the summative assessment, 66% of students scored below 7.0 (recommend average).

![Figure 6](image)

**Figure 5.** Formative assessment for the content aspect

**Figure 6.** Notes per student (formative and summative evaluation)

Even with a small sample, it was possible to evaluate some interesting points from the ETT project, such as the
importance of feedback from evaluations to students, but also the results of testing performed and quality of artifacts produced by them. According to the metrics collected regarding the amount of bugs found and the observations reported by the client and tutors, the results were very satisfactory.

It is noteworthy that although the two teams have conducted tests on the same tool, the bugs were found by Scan team were different bugs found by Cambio team. Compared to the bugs found during the tests of the TaRGeT tool performed early of this project, these were more critical and in greater quantity. This reinforces the importance of exploratory test strategy to expand the imagination of the test engineer, also helping to identify gaps that are usually ignored or difficult to find.

With respect to Learning Module, the results do not indicate a need to evolve in this module version and level of maturity, but rather an evolution in the level of context with other types of applications, such as web applications.

Beyond these results was also observed that students did good work in teams, demonstrating the acquisition of theoretical knowledge and testing practices, beside the advances in communication and orality, observed through the review of artifacts and presentation of the project status report, and the motivation to continue the training program.

4. Conclusion

This The aim of this study was to describe an approach for training in software testing using the instructional method Problem-Based Learning (PBL), involving the practice of real projects and training targeted to the content covered in the projects. To validate this approach, a pilot project was carried out by defining as challenge the test of a tool to support the testing process through the application of the exploratory testing technique, fully aligned with the creative character and discoverer of the test engineer.

The approach used enabled engagement with concrete situations and challenging, allowing students to experience real situations of software testing at the same time that technical and nontechnical roles were exercised, as decision making, work in team, and oral presentation of project results. Additionally, the approach provided tools that enabled managers to plan and oversee the training program for a much easier way and controlled.

It is important to emphasize that the use of this methodology in new experiments is essential for to validate its applicability.

As next steps, the LabPS is planning new training modules in the area of Software Testing and in the near future in other disciplines of the development process, such as software engineering and project management. Although not yet proven, it is believed that the methodology proposed in this paper can be used for new subjects with minor adjustments.

Acknowledgements

This work was [partially] supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

References