Experimentation Package for Evaluation of Problems Applied to the Software Project Subject Using PBL

Jacson Rodrigues Barbosa*, Fabrizzio Alphonsus Alves de Melo Nunes Soares*, Auri Marcelo Rizzo Vincenzi*
*Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
E-mail: {jacsonbarbosa, fabrizzio, auri}@inf.ufg.br

***Abstract*—This paper presents an experimentation package that compares the traditional and problem-based learning (PBL) approaches in the Software Project academic subject. The package was applied in a controlled experiment in a Computer Science class of a higher education institution, having message-oriented middleware as a case study. The case study enabled us to validate the experimentation package and to collect initial data to investigate the advantages and disadvantages of PBL against traditional learning. Even though the statistical analysis failed to show differences between the two approaches in view of the data collected, students' answers to a questionnaire enable us to verify how PBL may be used to increase their motivation and interest in the subject.**

***Keywords*-Message-oriented middleware; software project teaching; problem-based learning.**

I. Introduction

In the teaching process of almost all fields of knowledge, a constant and crucial feature is problem solving or the preparation for problem solving. If the teacher provides facts and procedures to his/her students without giving them the chance to carry out investigations on their own and to formulate questions, they may memorize the subject but be unable to understand it in depth or to apply it [1].

Problem-based learning (PBL) offers a structure that helps students understand a given subject in more detail. According to this method, problems must challenge the students to reach higher levels of knowledge [2]. A possible way to verify the cognition levels that a certain problem must reach in PBL consists in applying Bloom's revised [3] taxonomy, which classifies the cognitive abilities of individuals according to six levels, as is shown in [4], and summarized below:

- Remember: to produce correct information from memory;
- Understand: to provide a meaning to educational material or experiences;
- Apply: to use a procedure;
- Analyze: to break down a concept into parts and report on their relation with the whole;
- Evaluate: to carry out inferences based on criteria and patterns;
- Create: to link pieces of data in order to create something new.

Therefore, this paper proposes an experimentation package which assesses the cognitive levels reached by students enrolled in the Software Project subject when instructed by PBL, as well as to identify the quality of the software they developed, based on standard OO metrics [5].

This paper is structured as follows: Section II presents the problems that were investigated in Software Project. Sections III and IV describe the experimentation package applied and the questionnaire given to the students, respectively. Section V displays the results obtained from applying the package. Finally, Section VI offers conclusions and suggestions for future research.

II. Problems applied to the Software Project subject

Message-oriented middleware (MOM) and software coupling were selected for the case study.

Message-oriented middleware is a communication method between software components used in distributed systems. A client may send and receive asynchronous messages to and from any other client, connected to a special agent that provides facilities to create, send, receive and read messages. Software coupling is a measure of the interconnection between theses classes or subsystems. Strong coupling means that the related classes need to know internal details from each other, that changes spread throughout the system and that the system is potentially more difficult to understand. Thus, loose coupling is linked to the considerable need for flexibility required by great distributed systems, in addition to failure tolerance. This means that the dependencies must be kept to a minimum as much as possible so that, in case of failure or unavailability of a system and/or service, the others remain available and working. MOM allows for loose coupling. Sender and receiver do not need to be synchronized or previously known. It is an

 ISBN: 978-1-61208-230-1

alternative to synchronized distributed methods which may resort to blocking during communication.

In the PBL approach used in this research, the notions of MOM and coupling are presented in three interdependent problems.

The first problem discusses the main concepts regarding MOM and coupling, as well as presents two message exchange models. The first model is based on producer/consumer problems and is also known as point-to-point. It follows the concept of queues that are fed with messages, which in turn are then removed by a single consumer. The producer sends messages to a queue and the consumer reads them. In this case, the producer knows the message's destination and sends it directly to the consumer's queue. This model comprises the following features:

- Only one consumer reads the message;
- The producer does not have to be executed while the consumer reads the message, just as the consumer does not need to be executed while the producer sends the message;
- Once a message is successfully read, the consumer acknowledges the message to the producer.

The second model selected is the publisher/subscriber model. It is based on the concept of topic, according to which messages are listed in topics that are received by one or more subscribers [6]. It supports publishing messages to a given topic of messages. The subscriber(s) may register interest in receiving ("subscribing") messages on a particular topic. According to this model, neither the publisher nor the subscriber knows about each other. Its features are:

- Several consumers may read the message;
- There is a timing dependency between publishers and subscribers of a given topic. A publisher must create a subscription for subscribers to receive messages. The topic subscriber must be continuously active in order to receive messages.

Once the problems were discussed with the students, the latter were asked to research two real-life problems that may be solved by using each of the models presented in class. They were given two days to solve this first problem.

As regards the second problem, two brief descriptions of systems were presented. The first is used for selling cinema tickets (Software 1), whereas the second records students' enrolments in the subjects of a given university (Software 2). Students were asked to define the most suitable architecture (publisher/subscriber or point-to-point) for both software, as well as create a class diagram and a prototype of graphical user interface. They were given two weeks to complete this task.

Finally, the third problem involved asking students to build Software 1. In view of the fact that they already had knowledge of Java™, they were asked to implement the software via Java 6, Java Message Service (JMS) - Version 1.1 and Oracle™GlassFish Server Open Source Edition 3.1.

The tasks were given in this order to promote more effective learning, taking into consideration the cognitive levels explored in each problem proposed. Further details regarding these levels are provided in the following section.

A. Problem analysis according to Bloom's taxonomy

Tables I, II and III show the cognitive levels for Problems 1, 2 and 3, respectively, in accordance with Bloom's taxonomy.

The first problem aims to assess the students' abstraction abilities, i.e. to verify their ability to choose real-life applications of the recently learnt tools. For instance, this problem verified their ability to choose real-life examples that are linked with each of the recently discussed MOM architectural models. Therefore, this problem intended to explore the most superficial levels of Bloom's taxonomy, having provided the students with an initial contact with MOM-related concepts. Hence, the concepts under study were those of remembering, understanding and applying.

Table I
Analysis of Problem 1

Cognitive level	**Characterization**
Remember	MOM software architectural concepts.
Understand	Understanding of MOM concepts.
Apply	Identification of real-life examples linked with both kinds of architecture.
Analyze	Not explored.
Evaluate	Not explored.
Create	Not explored.

The results of this stage are shown as a table of confusion, also known as a confusion matrix. It consists of a table with two rows and two columns that reports the number of false positives, false negatives, true positives and true negatives.

The second problem also aimed to assess the students' abstraction abilities, but unlike the previous problem, now they had to choose the tool i.e. the model most suited to solve the problem in question. Table II shows the cognitive levels explored in this problem.

The third problem aimed to assess the students' technical abilities. During this stage they had to show the extent of their ability to encode software by using the MOM models learnt. The cognitive levels explored are shown in Table III.

 ISBN: 978-1-61208-230-1

Table II
Analysis of Problem 2

Cognitive level	Characterization
Remember	Use of MOM concepts acquired in problem 1 and use of representational form of classes and associations in UML.
Understand	Understanding the representational form of classes and associations in UML in order to meet the corresponding functional requirements and architectural restrictions.
Apply	Use principles and techniques of software project modelling.
Analyze	Identification of parts of the problem that are modellable based on informal textual specification.
Evaluate	Design the software to meet the quality attribute of usability.
Create	Creation of conceptual model (UML class diagram) and graphical user interface project.

Table III
Analysis of Problem 3

Cognitive level	Characterization
Remember	Use of software project concepts acquired in problem 2.
Understand	Understanding of MOM to solve the problem.
Apply	Use of Java for error treatment, database access and message exchange.
Analyze	Identification of parts which must be implemented in the program based on the models.
Evaluate	Choose specific data structures to provide an effective solution.
Create	Development of software for selling/booking cinema tickets.

III. Experimentation package for assessing PBL in the Software Project subject

This section presents the experimentation package which was adopted during the experimental tasks and result analysis. Its detailed description makes it possible for this study to be replicated in future research. The experimentation package set for the analysis of PBL in the Software Project subject was organized following [7], Its stages were:

A. Definition of experiment

To analyze PBL in the teaching of Software Project.

With the purpose of assessing PBL in the teaching of Software Project.

As regards factors that contribute to the quality of teaching, such as the ability to remember, to understand, to apply, to analyze, to evaluate and to create.

In the context of undergraduate students of Computer Science enrolled in Software Project.

B. Selection of context

The context chosen for the experiment is the teaching and application of MOM-related concepts. The experiment involved the use of academic software developers whose time and tools were controlled for teaching and the designing of a software for cinema ticket selling via JMS.

C. Formulation of hypotheses

A two-way analysis of variance (ANOVA) was performed to verify statistical differences between PBL and traditional approaches in Software Project.

ANOVA defines the null hypothesis (H_0), according to which there are no statistical differences between the methods under analysis. When the probability (p) found is lower than 0.05, the null hypothesis may be rejected, otherwise it will not be possible to state that there are statistical differences between the methods.

D. Selection of variables

The variables analyzed in the experiment are divided into two types: dependent and independent. The former were the cognitive levels reached in the problems, and the latter were the problems' cyclomatic complexity and size.

E. Selection of participants

All students enrolled in the Software Project subject were selected ($N = 14$). Observe that, even though N is small, an experimentation package allows us to replicate this same experiment several times, and new collected data can be added to this one, thus increasing confidence on the obtained results. This first replication aimed at validating the proposed package.

F. Experimental project

Firstly, the teacher presented the major theoretical concepts regarding MOM and the two message exchange models: point-to-point (queue model) and publish/subscribe. No examples were given regarding the models.

G. Quality assessment

A control sample was also used to validate the experiment internally. This sample consisted of a task set to the students regarding the traditional learning approach (first part of the subject); according to the task, students had to define a class diagram based on a context previously specified by the teacher. The external validation, as the students' profiles in Table VII confirm, was based on academic professionals who were then enrolled in the third semester of the course and whose average work experience in the field amounted to 7.5 months.

 ISBN: 978-1-61208-230-1

H. Preparation

Students were not aware of the reasons behind the research being carried out, but were informed of the steps to be taken to solve each of the three problems proposed.

I. Application

The experimentation package was applied during one semester (five months) in the Software Project subject, part of the Computer Science course. The first part of the subject involved the application of traditional methodological strategies (first two months), and the second part involved PBL (last three months).

J. Descriptive statistics

Ordinal and interval scales were used for the statistical analysis of the experiment to verify students' performance under both traditional and PBL approaches. Once the data had been collected, the sample was assessed to establish whether it showed normal or non-normal distribution. In case of the former, parametric statistics was required; in case of the latter, non-parametric statistics was required [8].

K. Data reduction

The criteria for data reduction were only required in case of dropouts, i.e. students who did not carry out any of the tasks set. About 21.4% of data (three students dropped out of the university) was left out so as not to jeopardize the experiment results.

IV. Students' views on PBL

After having solved Problem 3, the students were given a questionnaire on the experience of problem-based learning. The following questions in Table VII were adapted from those found in [9].

V. Results and discussion

A. Characteristics of Problem 1

In Table IV each column represents the student's option, whereas each row yields the correct option. For instance, line 2 and column 2 show that 100% of the students managed to correctly identify real-life problems that may be solved through the point-to-point model.

As Table of Confusion shows, solving this problem proved quite easy for the students. Of the solutions provided for the Topic model, only 18.18% did not comply with its definitions.

Table IV
Table of confusion of Problem 1

	Topic	Point-to-point
Topic	81.82%	0%
Point-to-point	18.18%	100%

B. Characteristics of Problem 2

As Table V shows, all the students successfully selected the architecture model most suited to each of the software descriptions presented.

Table V
Selection of architecture model

Software	**Coherent**	**Incoherent**	**Total**
Software 1	100%	0%	100%
Software 2	100%	0%	100%

C. Characteristics of Problem 3

The estimate for the necessary effort to develop the software for this problem was calculated by the function point analysis (FPA) described in [10]: 1,949 lines of Java.

Table VI shows some metrics collected from the solutions given by the students with the aid of JaBUTi (Java Bytecode Understanding and Testing) [11]. The first metric regards the size of the programs in terms of non-comment source lines of code (NCLOC). Following are two metrics related to maximum and average cyclomatic complexity [12] of the methods from each program, CC-MAX and CC-AVG, respectively. The remaining metrics are part of C&K metrics [5], such as: weighted method count via cyclomatic complexity (WMC-CC), depth of inheritance tree (DIT), lack of cohesion in methods (LCOM), response for class (RFC) and coupling between objects (CBO).

Regarding the LOC metric, the mean size of implementations was 1,700 LOC. Half of them showed values above average and closer to the estimate given by FPA.

The analysis of metrics related to McCabe's cyclomatic complexity revealed that, even though CC-MAX shows methods with maximum cyclomatic complexity of about 30 (the recommended yield would not be greater than 10 [13]), this only occurs in some isolated methods which, despite showing several conditions, are simple from the standpoint of programming logic. Thus, in general, as the remaining complexity-related metrics (CC-AVG and WMC-CC) attest, class methods are described as simple and devoid of considerable risk [14], this is confirmed by AMZ-LOCM, which shows that the average size of the methods of each implementation ranged from 6.55 LOC in Implementation 1 to 2.59 in Implementation 5, which results in an overall size average of 3.95 LOC.

DIT shows that inheritance was little explored in all projects. The DIT limit was restricted to one, without considering the calculations of Java's API classes.

LCOM is important to estimate the degree of cohesion in a given software. For instance, in an object-oriented software, it may be used to measure the cohesion of each

 ISBN: 978-1-61208-230-1

software class. A high LCOM value may suggest that the class project is poor, as was proved in a class of Implementation 5 whose LCOM value was 366.

CBO is useful in showing the potential reuse of classes, given the fact that loosely-coupled classes are "more independent". However, strongly-coupled classes are also very complex and more sensitive to changes in a project, which makes maintenance difficult and requires more rigorous tests. But as Table VI shows, the solutions provided by the students generally showed low coupling.

D. Students' views

Table VII gathers information collected from the students after the academic subject had come to an end. All the students who answered Question 7 (62.5%) declared positive experiences with PBL. Some of the comments were "The method promoted opportunities to increase knowledge", "It makes the students responsible in the teaching-learning process" and "Motivation for students".

As regards Question 8, 71.43% of the students consider the use of PBL in Software Project a "good" opportunity to solve real-life problems. However, they also consider both methods (traditional and PBL) as important in the subject in question (refer to answers to Questions 9 and 10).

E. Statistical analysis

Table VIII and Figure 1 show statistical data that refer to students' performance in the tasks they were assigned, in accordance to the teaching methodology adopted. It is important to point out that, for both methodologies, the same group of students was used in all the problems.

The Lilliefors and Cochran test revealed the normality and homogeneity of the variances ($p < 0.05$); as the problems taken into account in the experiment are also independent, then ANOVA may be used to carry out data analysis.

ANOVA obtained $p = 0.75$ for the set of collected data. The result was higher than 0.05, therefore the null hypothesis (H_0) cannot be rejected i.e. from a statistical perspective and in view of the range of problems explored, there are no statistical differences between the traditional methodology and PBL.

Table VIII
DESCRIPTIVE STATISTICS

Teaching method	Mean	Median	Standard Deviation
Traditional	7.16	7	0.75
PBL	7.29	7.33	0.98

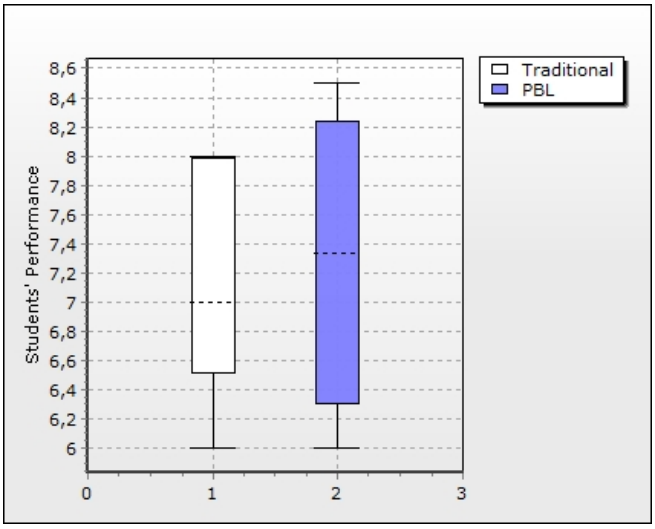

Figure 1. Boxplot of Students' Performance

VI. Conclusion

This paper presented the experimental package and results of a controlled experiment which aimed to compare students' performance in view of the application of two teaching methodologies (traditional and PBL) within the Software Project subject. However, this paper's main objective was to validate the experimentation package based on its first application in the Software Project subject.

The statistical analysis revealed that there are no differences between the teaching methods adopted. Nevertheless, the assessment of the mean differences showed that PBL's was 0.13 higher than the traditional methodology's; furthermore, the analysis of students' answers to the questionnaires revealed that they had positive experiences when using PBL.

In future research, we intend to apply this experimental package to other university classes in order to compare the differences. Furthermore, we intend to repeat the experiment with students enrolled in the following year. Further detailed information on this topic can be found in [15]. We also intend to carry out a statistical analysis (logistic regression) of the metrics collected by JaBUTi to identify patterns associated with cognitive levels.

Acknowledgments

The authors would like to thank the Brazilian Funding Agencies - CNPq and Capes - for their partial support to this work.

References

[1] R. Delisle, *How to use problem-based learning in the classroom.* Alexandria, Virgina, USA: ASCD, 1997.

 ISBN: 978-1-61208-230-1

Table VI
DIRECT METRICS OF SOLUTIONS PRESENTED

Implementation	LOC	CC-MAX	CC-AVG	WMC-CC	AMZ-LOC	DIT	LCOM	RFC	CBO
1	2,007	33	1.44	5.63	6.55	1	3.16	18.77	6.16
2	804	27	1.13	7.31	3.32	1	5.25	19.22	6.34
3	1,838	33	1.12	5.16	3.56	1	2.52	15.19	4.52
4	3,579	37	1.24	5.84	4.7	1.06	3.35	20.81	6.97
5	1,238	46	1.06	5.55	2.59	1	6.51	16.88	4.85
6	805	7	1,08	4.23	2.98	1	3.13	14.7	5.49
Average	1,711.83	30.50	1.18	5,62	3.95	1.01	3.99	17.60	5.72
Standard deviation	1,045.02	13.11	0.14	1.01	1.46	0.02	1.55	2.41	0.94

Table VII
INFORMATION OBTAINED AFTER QUESTIONNAIRE APPLICATION

Question	Answer	%
How old are you?	Average=20.26 years	-
Do you work with software development (internship, contract etc.)?	Yes	57.14
	No	42.86
If you answered "yes" to the previous question, how long have you worked in the field (specify the period in months or years, e.g. six months)?	Average=7.5 months	-
Why have you chosen this university to study Computer Science?	Quality of teaching at the university	12.5
	Visibility of the university	12.5
	Duration of the course	12.5
	No particular reason	12.5
When you made your decision, were you aware that the university's Computer Science Department had adopted a new teaching method?	Yes	0
	No	100
If you answered "yes" to the previous question, do you think this influenced your decision?	Yes	0
	No	0
The university decided to apply active teaching-learning methodologies in the Computer Science curriculum. How would you describe your experience as an active participant of such methodologies in this department?	Offered conditions to increase knowledge	37.5
	Makes student responsible in the teaching-learning process	12.5
	Students' motivation	12.5
	Question not answered	37.5
Based on your learning process, how do you rate the use of problem-based learning (PBL) in designing software projects to solve real-life problems?	Excellent	14.29
	Good	71.43
	Fair	14.29
	Poor	0
Do you prefer a traditional teaching method, in which the teacher presents the contents to be learnt and you study them?	Yes	0
	No	28.57
	Both	71.43
You have designed/implemented a software for cinema ticket selling as part of a PBL-based case study. Do you prefer this teaching-learning strategy to the traditional method?	Yes	42.86
	No	0
	Both	57.14

[2] J. A. M. Santos and M. F. Angelo, "Análise de problemas aplicados em um estudo integrado de programação utilizando pbl," *WEI - Workshop sobre Educação em Computação, Anais do XXIX Congresso da SBC*, pp. 519–522, 2009.

[3] B. J. Duch, S. E. Groh, and D. E. Allen, *The Power of Problem-Based Learning: a practical how to for reaching undergraduate courses in any discipline*. Virginia: Stylus Publishing, LLC, 2001.

[4] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C. Wittrock, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Complete Edition*. Allyn & Bacon, 2001.

[5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, June 1994.

[6] I. Gorton, *Essential Software Architecture*. Berlin, Germany: Springer, 2006.

[7] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Springer, 2000.

[8] D. C. Montgomery, *Design and Analysis of Experiments*, 5th ed. John Wiley & Sons, 2001.

[9] J. R. B. Costa, V. F. Romano, R. R. Costa, A. P. Gomes, and R. S. Batista, "Active teaching-learning methodologies: Medical students views of problem-based learning," *Revista Brasileira de Educação Médica*, pp. 13–19, 2011.

[10] E. J. Braude and M. E. Bernstein, *Software Engineering Modern Approaches*, 2nd ed. John Wiley Sons, 2011.

[11] A. M. R. Vincenzi, W. E. Wong, M. E. Delamaro, and J. C. Maldonado, "JaBUTi: A coverage analysis tool for java programs," in *XVII Simpósio Brasileiro*

 ISBN: 978-1-61208-230-1

de Engenharia de Software (SBES 2003), Manaus, AM, October 2003.

[12] T. J. McCabe, "A complexity measure," in *Proceedings of the 2nd international conference on Software engineering (ICSE '76)*, October 1976.

[13] R. S. Pressman, *Software Engineering – A Practitioner's Approach*, 7th ed. McGraw-Hill, 2009.

[14] E. Vandoren and K. Sciences, "Cyclomatic complexity," june 2000, available at: http://www.sei.cmu.edu/str/. Accessed on: 03/06/2012].

[15] J. R. Barbosa, F. A. A. de Melo Nunes Soares, and A. M. R. Vincenzi, "Problems applied to the software project subject using pbl," Web page, july 2012. [Online]. Available: http://www.inf.ufg.br/~auri/pbl-en/. Accessed on: [07/18/2012].

 ISBN: 978-1-61208-230-1