An environment for teaching enterprise software evolution: using playful techniques with robotics

João Ricardo Moreno Camilo, Alexandre L'Erario, José Augusto Fabri
Universidade Tecnológica Federal do Paraná - UTFPR
PPGI - Programa de Pós Graduação em Informática
Cornélio Procópio – PR - Brasil
{jrmcamilo91}@gmail.com, {alerario, fabri}@utfpr.edu.br

***Abstract*—This full paper presents a Research-to-Practice approach to teaching software evolution in undergraduate courses. There are many private enterprise systems running today. The evolution of these systems is based on many different stakeholders: software producer, external partners and others. Besides that, there are many approaches to security, requirements, software process. An enterprise system can't stop and needs to upgrade many times from many stakeholders. The undergraduate students don't learn in their courses how to keep the integrity of this product. The objective of this study is to show an environment simulating these scenarios in the classroom by using LEGO Mindstorms. The scope of this environment is for technological teaching: software engineering and system analysis. The result is an environment for training students for enterprise systems software evolution. This scenario requires a highly cooperative and coordination capacity. For this reason, students must understand and apply software development techniques mutually to governance and accordance. Two experiments were applied in two undergraduate courses with the purpose of measuring the learning ability related to software evolution. The results indicated that the use of play techniques associated with robotics in the teaching process was relevant for the consolidation of concepts related to software evolution.**

***Index Terms*—software evolution, LeJOS, Mindstorms**

I. Introduction

Since the emergence of software engineering, researchers and practitioners have struggled to deal with constant demands for changing requirements to evolve software systems. The changes can be driven by internal reasons, such as a new and more efficient policy of the organization, or by external factors, such as suppliers, technology providers or regulatory agencies [1]. These continuous changes consist of the software evolution. Over the time and with excessive modifications, software becomes more complicated and tracking changes is not an easy task [2]. This evolution poses several research challenges, especially in large and complex systems [3].

The software evolution has gained importance and has recently moved to the attention of software developers. As a consequence, most software development now takes place at the software evolution stage. This event has changed the face of software engineering [4]. Following the evolution of software developments, enterprise systems customization can be driven by internal reasons. Such as a new and more efficient logistics management policy suggested by the customer company that should be reflected in the enterprise system; or by external factors, such as suppliers, technology providers or regulatory agencies [1]. For example, an enterprise system may require updates to accommodate new tax regulations by financial authorities to address emerging money laundering practices [2].

In this context, enterprise systems are one of the most common information system solutions adopted by organizations around the world. Enterprise systems are large and complex because they allow the complete integration of all the processes, interactions and financial transactions of a company on the same platform through a database and accessibility through a unified interface [5] [6]. It is a modularized or packaged software [7], with the possibility to customize or add modules or packages by third parties [8]. Enterprise systems projects differ from other types of software projects because they cover thousands of business activities, require different configuration and modification activities to reflect immediate business requirements [9]. Enterprise systems inevitably evolve align their functionality with changing business requirements [10] [11].

For this reason, it is salutary to include knowledge of software evolution in IT students training. This work aims to present the application of ludic techniques with the use of robotics in the teaching of software evolution. This work also aims to contribute about the importance of creating a scenario of software evolution in the graduation, concept that helps to consolidate the formation of the student as a professional.

In the area of computing, there are several initiatives that propose the use of robots as a learning object [12], [13]. The purpose of these objects is to increase students' motivation and commitment to the development of the subject taught, as well as to stimulate students' creativity due to their dynamic, interactive and even fun nature.

In order to validate the effectiveness of the technique presented in this work, two experiments were applied in undergraduate courses (Software Engineering / Technology in Analysis and Development of Computer Systems) of the Federal Technological University of Parana (UTFPR) - Campus Cornelio Procopio, PR, Brazil.

The objective of the experiments was to measure the learning ability of teams to apply processes and activities related to software evolution. In the experiments, the students solved exercises related to software evolution, whose objective

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

was to make the LEGO Mindstorms robot perform new activities, using LeJOS (Java for LEGO Mindstorms). In this work, LEGO Mindstorms is used to enable the achievement of simulation scenarios.

The results indicated that the use of gaming techniques associated with robotics in the teaching process was relevant to the consolidation of concepts related to software evolution. All the results were obtained from students with no software evolution knowledge.

The structure of this text includes a brief theoretical foundation, presented in section II. In section III, the methodology applied to perform the experiments, in section IV, the analysis of the results and in section V presents the conclusion, followed by the future works in section VI.

II. THEORETICAL FOUNDATION

This section presents the theoretical basis used to provide a theoretical basis of the main concepts used in this work.

A. Software Evolution

There are several definitions for software evolution and software maintenance. Some authors refer to software evolution when a feature or some other aspect of the software is improved, and refer to software maintenance primarily to the adaptations and fixes the essential to keep the software running [14]. Since maintenance objectives aim at reduced goals, maintenance processes are usually simpler and more predictable than the evolution processes [4]. In this study, we adopted evolution and maintenance of software through these concepts.

Practitioners involved in software evolution are likely to consciously or unconsciously confront some of the constraints imposed by the laws of software evolution that Meir M. Lehman introduced in the 1970s. These laws or, rather, empirical hypotheses, suggest that any actively used software system should continually change to satisfy its stakeholders [15] [16]. Therefore, these must be continually improved, adapted and corrected if they are to remain effective within a constantly evolving application environment. Thus, the evolution of such systems is a complex phenomenon characterized by feedback multi-level, multi-loop and multi-agent [14].

Studies evaluate that development and evolution can be spread across multiple teams to leverage different knowledge, experience, or capabilities. In this way, breaking a task into smaller, more manageable pieces is often an effective way to deal with the kind of complexity. The artifacts developed or modified separately need to be assembled as efficiently as possible into a consistent whole in which the pieces still functioning as specified [17].

In the context of software evolution, Camilo et al. [18] identified, through a single case study, the main stakeholders involved in demands for changes in enterprise systems. The main findings were highlighted in Figure 1.

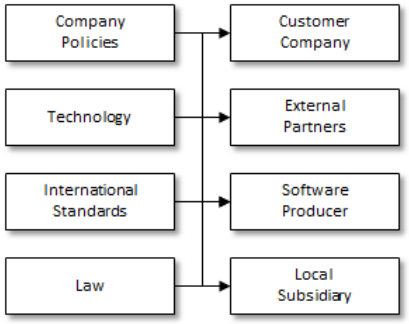

Fig. 1. Identified demands and stakeholders.

B. Robotics an object of the learning

Robotics as a technical discipline has often appeared in school curriculum. According to Leska [19], and Sullivan [20], there are indications that robotics was an efficient learning object for teaching. The following are some works of different natures that represent the state of the art without the use of robotics in education.

Lew [12] state that the use of robots helps students better understand fundamental concepts and increases enthusiasm in teaching and learning activity. Vallim [13] uses robots and says that this is a proper approach to the introduction of engineering concepts and provides professional skills.

Brandt and Colton [21] used Mindstorms to teach programming, mechanics, and control to first-year mechanical engineering students at Brigham University. The objective of this work was to present the detection of an interface. The authors conclude that Mindstorms is a versatile platform widely accepted by students. With their use, the authors improved mechanical learning, sensor calibration, programming language and the principles of physics.

Fabri [22] applied Mindstorms to the teaching and learning of software processes and project management. To validate the effectiveness of the work, they carried out 8 experiments, including university courses and companies in the software production sector. Experimental results have proven that the process of knowledge transfer in processes and software management using Mindstorms has led to greater consolidation, motivation, and satisfaction of those involved.

Serodio [23] applied Mindstorms to the teaching and learning of software maintenance. With reduced approaches and goals, the software maintenance was evaluated by 4 experiments carried out including undergraduate courses. Experimental results have proven that the software maintenance activity is poorly understood in undergraduate courses.

However, no work has been found that directs the use of robots specifically in teaching software evolution.

III. RESEARCH METHODOLOGY

The experimental method is one of the main ones for conducting trials in software engineering [24]. According to

Wohlin et al. [24], the experimental research consists of: determine a study object, select the variables that could influence it, define the forms of control and observe the effects that the variable produces on the object.

The experiments were performed based on an execution plan divided into the following steps: Environment Definition, Subject Definition, Sample Definition and Experience Execution. The research question, outlined by the work, that provided the mapping of the experimental method was:

- Is it possible to teach Software Evolution through ludic techniques in robotics?

A. Setting the Environment

The experiments were carried out in an academic environment, in undergraduate courses (Software Engineering / Technology in Analysis and Development of Computer Systems), at the same campus of the Federal Technological University of Parana (UTFPR) - Campus Cornelio Procopio, PR, Brazil.

Students were randomly selected from the undergraduate courses. The experiments were followed without interference during execution.

The groups were divided as follows:

1- Experiment performed with a group of 14 students (Experiment 1); 2- Experiment performed with a group of 16 students (Experiment 2);

Each participant had access to a characterization form. This form aimed at minimally identifying the student who participated in the experiment. The form used in this experiment can be obtained through: https://goo.gl/eCEzPD.

B. Definition of subjects

The total number of participants was 30 students, divided in two experiments. Each experiment was divided in 3 groups, each group representing a stakeholder (Customer Company, External Partner and Software Producer).

C. Sample definition

It is the amount of subjects who participated in the experiment. This information is entirely tied to the number of students. A total of 30 students participated in the experiment.

D. Experiment Execution

The execution of the experiment aims to characterize the steps that the researcher will follow to carry out the experiment. The steps characterized for the accomplishment of these experiments are:

1) Bring all students together in a classroom.
2) Ask each participant to complete the characterization form, containing Name, Participating Group, Age, membership of the course, time of experience in Software Engineering, development in Java and far.
3) Provide the written consent form for signing, available at https://goo.gl/NEaJPb.
4) Presenting the concepts of software evolution and the LEGO Mindstorms robot, presentation available at https://goo.gl/QtS77a.
5) Ask the class to be divided into 3 groups (Customer Company, External Partner and Software Producer) and run the environment setup, available at https://goo.gl/TN5WtL.
6) After the environment has been configured, each group should download the first version of the project available at GitHub:
   https://github.com/lejosgrupo1/lejosgrupo1 (Group 1)
   https://github.com/lejosgrupo2/lejosgrupo2 (Group 2)
   https://github.com/lejosgrupo3/lejosgrupo3 (Group 3)
7) Each group received a document containing the fictional change in law. This document contained the new Mindstorms paths to be implemented by each group and other instructions.
8) Provide each group a robot for the execution of software evolution, in order to make the robot perform the new paths.
9) Set an estimated time to perform the proposed activity.
10) Each group must implement the source code from its responsibility and make it available in the GitHub repository. Using Mindstorms color sensor, Group 1 must implement the path from START position to the first YELLOW color identified. Group 2 must implement the path from YELLOW color to the fourth RED color identified. Group 3 must implement the path from fourth RED color to the first BLUE color identified.
11) After carrying out the evolution process and execution of the proposed activities, it will be necessary to complete another questionnaire to evaluate the use of Mindstorms in software evolution. This form will be available at https://goo.gl/kuiBvs.

In each experiment, the participants were divided into 3 groups: Customer Company (Group 1), External Partner (Group 2) and Software Producer (Group 3). The proposal for this experiment was to meet a new demand for evolution of the LEGO Mindstorms, as a consequence of a fictional change in law. Figure 3 highlights the demand and the stakeholders involved in the experiments. The fictional law was change that the robot could execute three new routes following the path of a track (game board). Each of the three new routes had to be developed by a different group (stakeholder). Each of them had a different level of access and limited knowledge about the proposed enterprise system as shown in Figure 2. The full final version of the route must be merged into the Customer Company (Group 1).

It is important to note that most of the students had no knowledge about LEGO Mindstorms and LeJOS technology. Given this fact, one of the activities of this experiment was precisely mapping this information to the performance analysis in the indicated problem. Figure 4 illustrates the path to be performed by the robot in the experiments performed. The robot was on a pathway (game board) and should travel the path identified in the dashed line.

The questionnaire applied after the execution of experiment is composed by 18 questions, according to the Table I. The questionnaire were defined taking into account aspects that

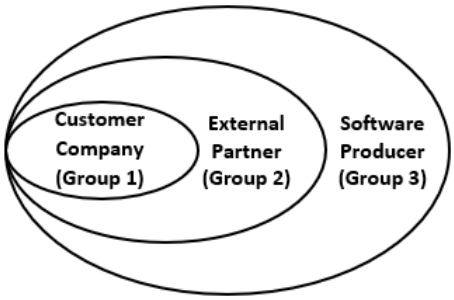

Fig. 2. Access and knowledge level of the system.

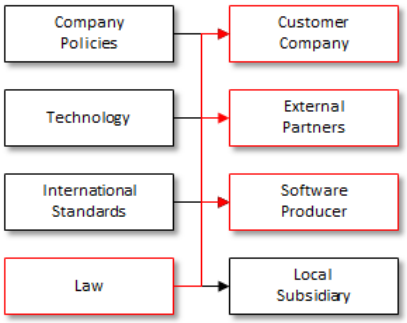

Fig. 3. Selected demand and stakeholders.

involve both a general evaluation of the experiment and a measurement of the level of learning in order to answer to the research question.

IV. ANALYSIS OF RESULTS

At the undergraduate level, two experiments were carried out with a total of 30 students. The first experiment contained 14 students, and the second experiment contained 16 students. As the groups had homogeneous characteristics, the results were unified, as presented in the characterization form. The software engineering experience of the teams ranged from less than 1 year and 1 to 3 years as shown in Figure 5.

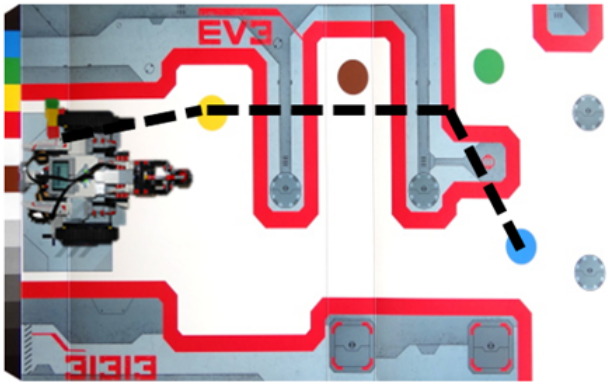

Fig. 4. Path to be performed by the robot.

TABLE I
QUESTIONS

Q1	Was the evolution process a simple task?
Q2	Do you consider that your team had the ability to perform evolution?
Q3	Did you have the ability to relate / know the stakeholders?
Q4	Was the time allotted enough to implement evolution?
Q5	Did you have to do a training?
Q6	Was there an identification of the problem to be solved?
Q7	Did you consider it important to know the product before performing the evolution?
Q8	Was stakeholder knowledge necessary to maintain the product?
Q9	Did you have difficulty setting up the evolving development environment you need?
Q10	Was the timing of the development of evolution adequate?
Q11	Within the stipulated deadline, do you agree that you did not spend enough time knowing the new requirements?
Q12	Did you perform evolution activity as stipulated?
Q13	Did you consider it important to know the software architecture before performing the evolution?
Q14	Did you consider it important to know the libraries used by the product before performing the evolution?
Q15	Did you consider it important to know the business rules of the product before performing the evolution?
Q16	Did you consider it important to know the FDI before performing the evolution?
Q17	At some point did you need to contact any stakeholder / applicator of the experiment for clarification?
Q18	Did the experiment / lesson provide notions of work involving the software evolution in enterprise systems?

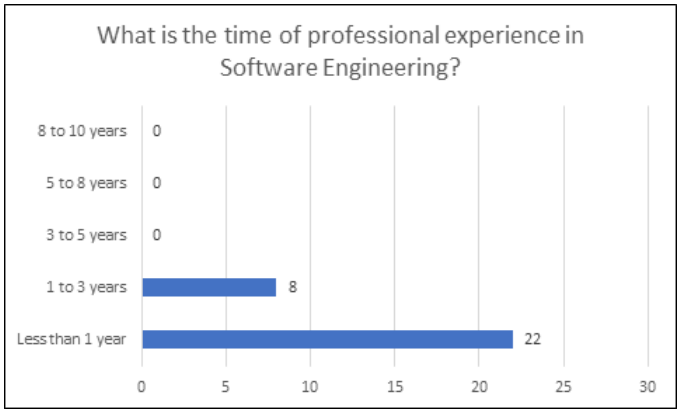

Fig. 5. Experience in Software Engineering.

Analyzing the Figure 6, it is possible to notice that the Q1 to Q18 legends of the X axis represent the questions answered in the software evolution questionnaire. In the Y axis are presented the total percentage answered by all the participating students, represented with the options of answers based on the Likert scale.

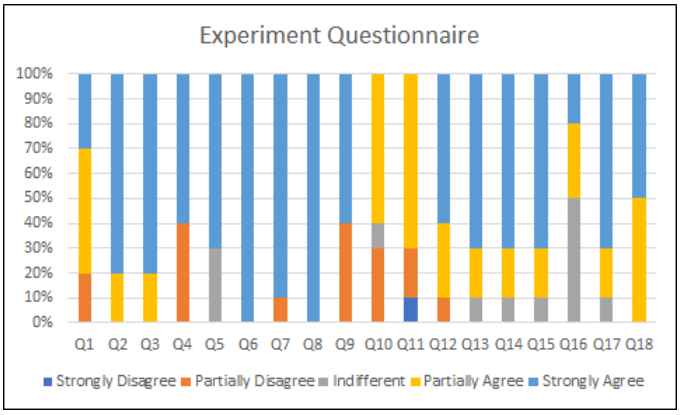

Fig. 6. Experiment questionnaire.

Taking into account that in the two experiments carried out the classes had the same characteristics, the analysis of the questionnaires was unified. Therefore, it is concluded that in Question 1 (Q1), 30% fully agree that the evolution process was a simple task, 50% agree partially and 20% disagree partially. 80% fully agree that their team had the ability to perform the evolution (Q2) and 20% partially agree. 80% partially agree that they had the ability to relate to stakeholders (Q3) and 20% fully agree. 60% partially agree that the time allotted was sufficient to implement evolution, while 40% disagree partially (Q4). In (Q5) 70% agreed fully to have to train with the colleague to be able to develop the evolution, while 30% partially agreed. All agreed fully that there was identification of the problem to be solved (Q6). In (Q7), 90% fully agreed to be important to know the product before performing the evolution and 10% disagreed partially. In (Q8), everyone agreed that stakeholder participation was necessary to maintain the product. In (Q9), 60% had difficulty setting the developmental evolution environment required, and 40% disagreed partially that they had difficulty. 60% agree partially that it was the moment to start the development of evolution (Q10) was adequate, 10% believed to be indifferent and 30% disagreed partially. Within the stipulated deadline, 70% agree that they did not spend enough time knowing the new requirements (Q11), 20% disagree partially and 10% disagree completely. 60% fully agree that they performed the evolution activity as stipulated (Q12), 30% agree partially and 10% disagree partially. Regarding the importance of knowing the architecture of the software before performing the evolution (Q13), 70% fully agreed, 20% partially agreed and 10% found it indifferent. Regarding the importance of knowing the libraries used by the product before performing the evolution (Q14), 70% agreed fully, 20% partially agreed and 10% found it indifferent. Regarding considering it important to know the business rules of the product before performing the evolution (Q15), 70% fully agreed, 20% partially agreed and 10% found it indifferent. Regarding the importance of knowing the IDE before performing evolution (Q16), 20% fully agreed, 30% partially agreed and 50% found it indifferent. Regarding finding it important to contact some stakeholder for clarification (Q17), 70% fully agreed, 20% partially agreed and 10% found it indifferent. All agreed that the experiment provided notions of work involving the software evolution in enterprise systems (Q18).

TABLE II
QUESTIONNAIRE AVERAGE

Q1	4
Q2	5
Q3	5
Q4	4
Q5	4
Q6	5
Q7	5
Q8	5
Q9	4
Q10	3
Q11	3
Q12	4
Q13	5
Q14	5
Q15	5
Q16	4
Q17	5
Q18	5

With the average of responses based on Likert, it is possible to observe from Table II that the most of the answers were positive after the execution of the experiment.

Analyzing the results of the four experiments shown in Figure 6, it is possible to conclude in both experiments that:

- For most participants, carrying out the evolution process was a simple task. For other participants, it was a rather complex task. This characteristic was a reflection of the experience of some of the participants. Some of them already had experience using Mindstorms. (Q1)
- Although the most of respondents had the ability to perform the evolution, it was observed that because of the time, the merge between the group codes could not be completely realized. (Q2)
- Although most respondents could relate / know the stakeholders, it was observed that, due to the time, the expected relationship was not fully accomplished. (Q3)
- Although most of the participants stated that the time allotted to implement the evolution was enough, the students had difficulties in configuring the environment and this partially affected the deadline. (Q4) (Q9) (Q12)
- Although most students claim that they had to undergo training, some students did not need to, since they stated prior knowledge in working with LEGO Mindstorms. (Q5)

- Everyone agreed that it was possible to identify the problem by changing legislation. (Q6)
- It was the first time they had contact with the product. Although it was a simple matter to be resolved, they were afraid to change the code for fearing the impact of changes on other stakeholders (Q7).
- Most of the participants found it indifferent or felt that it was not the time to begin the development of evolution. It was observed that the excess of news at one time was detrimental to the development of evolution. (Q10)
- For some participants that have stated prior knowledge and work with LEGO Mindstorms, found it immaterial to know the software architecture, libraries used, business rules, and IDE. (Q13) (Q14) (Q15) (Q16)
- Students performed the evolution without worrying about the other characteristics of the system, only focusing on what was proposed.
- Finally, participants agreed that the experiment provided notions of work involving the software evolution in enterprise systems, which helps to fulfill the main objective of this work. (Q18)

In this experiment, the LEGO Mindstorm robot proved to be an object of teaching and learning relevant in teaching software evolution and, in this case, also application, flexible and practical. In the literature, mentalities are applied in the teaching and learning of computer programming. Also in the processes of teaching, learning, management of software projects [22] and also motivating the teaching of software evolution. The students presented a quick and fun demonstration of results. These qualities encouraged the teams to evolve and, when analyzing the answers applied to each question. It was possible to verify if the use of such an approach generated motivation and interest, being these essential factors for the success of teaching any subject. One of the main opportunities in this study was the attempt to simulate a situation that happens in the industry, which motivates greater interest in undergraduate students.

V. Conclusions

Software evolution activity is also poorly understood and easily confused with software development and maintenance activity. The success of the operations of several organizations is directly linked to the attendance of enterprise systems to the requirements demanded by the stakeholders, as changes in technology, law, company policies and international standards.

The lack of professional knowledge in software evolution approaches makes many enterprise systems become obsolete. Therefore, it is necessary to guarantee the efficiency and availability of systems. The software evolution scenario is not commonly addressed in undergraduate courses. And this study allowed the transmission of knowledge from a context very present in the industry, thus consolidating the formation of the student as a professional.

This work presented the results of 2 experiments applied in undergraduate courses, the objective of the experiments was to measure the learning ability of the teams in the application of processes and activities related to software evolution. In the experiments, the students solved exercises related to this concept, whose objective was to make the LEGO Mindstorms robot perform new activities.

The results of this study differ from other studies by the type of scenario addressed. Despite being strongly present in the industry, the software evolution in corporate systems is a little discussed in undergraduate activities.

Therefore, through the experiment performed and the analysis of the answers obtained through the applied questionnaire, it can be concluded that it is possible to teach software evolution through the use of ludic techniques in robotics. However, it is clear that this is a scenario that addresses a complexity of relationship and communication between stakeholders, a complexity not yet experienced by undergraduates. Therefore, the contribution of this work was consolidated through the results obtained.

It is also noticed that the context of enterprise systems applied in the experiment was of relevant difficulty for the participants. The difficulty was not only greater because some participants had experience with Mindstorms, which provided a further advance in the implementations of the experiment.

An explicit point of the software evolution applied to enterprise systems identified in this study is related to the context of access to the new requirements demanded by new functionalities. Another point is communication between stakeholders. New features bring many questions to stakeholders, such as the publication of a new Law in the experiment, where the parties do not have a solid knowledge of the changes. In terms of communication, involving different stakeholders in solving a problem can bring numerous problems related to access, knowledge, responsibilities, among others.

Although Serodio [23] presents a context of changes in software, the reduced purpose of the paper does not involve the complex context of software evolution in the context of enterprise system.

Finally, we can highlight that the main contribution of this work is in the teaching of software evolution through a context that simulates a situation that occurs in the labor market: a context that is of a great importance that is not commonly presented to undergraduate students. It is important that this context is reinforced, mainly by the aspects of communication between stakeholders.

The scope of this work was limited to the experiments described here. Although there are indications that the results are positive, the experiments will be replicated to confirm the truth of the facts.

VI. Future Works

As a proposal of future work, it is intended to perform more experiments with the same game environment in a more complex software and perform this comparison of results. It is also intended to improve the software and compare the experience with simple software without the robot.

Although not the objective of this study, it is also recommended as future work a study aimed at understanding the program in the exchange of source codes between stakeholders.

REFERENCES

[1] Y. M. Ha and H. J. Ahn, "Factors affecting the performance of enterprise resource planning (erp) systems in the post-implementation stage," *Behav. Inf. Technol.*, vol. 33, pp. 1065–1081, Oct. 2014.

[2] M. Parhizkar and M. Comuzzi, "Impact analysis of erp post-implementation modifications: Design, tool support and evaluation," *Computers in Industry*, vol. 84, pp. 25 – 38, 2017.

[3] A. Talai and Z. E. Bouras, "Software evolution based activity diagrams," in *2017 8th International Conference on Information Technology (ICIT)*, pp. 82–88, May 2017.

[4] V. Rajlich, "Software evolution and maintenance," in *Proceedings of the on Future of Software Engineering*, FOSE 2014, (New York, NY, USA), pp. 133–144, ACM, 2014.

[5] N. K. Ömüral and O. Demirörs, "Effort estimation for erp projects: A systematic review," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 96–103, Aug 2017.

[6] R. Addo-Tenkorang and P. Helo, "Enterprise resource planning (erp): A review literature report," 01 2011.

[7] C. S.-P. Ng and G. G. Gable, "Maintaining erp packaged software – a revelatory case study," *Journal of Information Technology*, vol. 25, pp. 65–90, Mar 2010.

[8] C. Lopez and J. L. Salmeron, "A framework for classifying risks in erp maintenance projects," in *Proceedings of the International Conference on e-Business*, pp. 1–4, July 2011.

[9] M. Daneva and R. Wieringa, "Cost estimation for cross-organizational erp projects: Research perspectives," *Software Quality Journal*, vol. 16, pp. 459–481, Sept. 2008.

[10] T. Oseni, M. M. Rahim, S. P. Smith, and S. Foster, "An initial empirical evaluation of the influence of erp post-implementation modifications on business process optimisation," 2014.

[11] M. Themistocleous, Z. Irani, R. M. O'Keefe, and R. Paul, "Erp problems and application integration issues: an empirical survey," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 10 pp.–, Jan 2001.

[12] M. W. Lew, T. B. Horton, and M. S. Sherriff, "Using lego mindstorms nxt and lejos in an advanced software engineering course," in *2010 23rd IEEE Conference on Software Engineering Education and Training*, pp. 121–128, March 2010.

[13] M. B. R. Vallim, J. M. Farines, and J. E. R. Cury, "Practicing engineering in a freshman introductory course," *IEEE Transactions on Education*, vol. 49, pp. 74–79, Feb 2006.

[14] T. Mens, Y. G. Guehéneuc, J. Fernández-Ramil, and M. D'Hondt, "Guest editors' introduction: Software evolution," *IEEE Software*, vol. 27, pp. 22–25, July 2010.

[15] I. Herraiz, D. Rodriguez, G. Robles, and J. M. Gonzalez-Barahona, "The evolution of the laws of software evolution: A discussion based on a systematic literature review," *ACM Comput. Surv.*, vol. 46, pp. 28:1–28:28, Dec. 2013.

[16] T. Mens, *Introduction and Roadmap: History and Challenges of Software Evolution*, pp. 1–11. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[17] P. Brosch, G. Kappel, P. Langer, M. Seidl, K. Wieland, and M. Wimmer, *An Introduction to Model Versioning*, pp. 336–398. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[18] J. R. M. Camilo, A. L'Erario, J. A. Fabri, and T. Pagotto, "A process for distributed software evolution," in *Proceedings of the 13th International Conference on Global Software Engineering*, ICGSE '18, (Piscataway, NJ, USA), IEEE Press, 2018.

[19] C. Leska, "Introducing undergraduates to programming using robots in the general education curriculum," in *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '04, (New York, NY, USA), pp. 263–263, ACM, 2004.

[20] A. Sullivan and M. U. Bers, "Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade," *International Journal of Technology and Design Education*, vol. 26, no. 1, pp. 3–20, 2016.

[21] A. M. Brandt and M. B. Colton, "Toys in the classroom: Lego mindstorms as an educational haptics platform," in *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 389–395, March 2008.

[22] J. A. Fabri, A. L'Erario, R. H. C. Palácios, and W. Godoy, "Applying mindstorm in teaching and learning process and software project management," in *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8, Oct 2015.

[23] H. C. S. Thomazinho, A. L'Erario, and J. A. Fabri, "Teaching software maintenance with ludic techniques supported by robotics," in *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8, Oct 2017.

[24] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.