Using Dojo as a Pedagogical Practice to Introduce Undergraduate Students to Programming

Fabio Gomes Rocha
Universidade Tiradentes
Sergipe, Brasil
gomesrocha@gmail.com

Rosimeri Ferraz Sabino
Universidade Federal de Sergippe
Sergipe, Brasil
rf.sabino@gmail.com

Guillermo Rodriguez
ISISTAN (UNICEN-CONICET)
Tandil, Argentina
guillermo.rodriguez@isistan.unicen.edu.ar

***Abstract*—The objective of this article is to analyze the contribution of the use of software for introducing programming, using Dojo as a pedagogical practice. As an empirical research, we present a case study in the discipline of Introduction to Programming of the Computer Technician Course of the Brazilian Service for Industrial Apprenticeship, Regional Department, Sergipe. Based on the verification of difficulties in students learning, a test was developed using Dojo as a teaching methodology. The analysis were performed with a qualitative approach, using and associating the data obtained from the pedagogical activity, framework and references adopted in the case study. The final results showed a reduction in the number of students rate of absences, higher rate of students satisfaction and higher grades in the course, showing the possibility of adopting the experiment due to success in learning.**

Keywords: Dojo, Introduction to Programming, Professional education, Pedagogical practices.

I. Introduction

The evolution of technology occurred between the 1980s and 1990s with the emergence of microcomputer and the Internet has created new jobs and boosted computer programming. Due to the steady growth and high applicability of the software, it can be widely used ranging from the management of nuclear power plants to using simple electronic cellphone games, thus providing the expansion of demand for low-skilled work for this job. The training of skilled workers has been widrespread since 1962, in which universities and technical courses [1] got accredited and were included as a category of knowledge in WordSkills International, a global event that started 64 years ago and has the partnership of 52 member countries [2].

It has been observed that the expectations of computer technicians about the acquisition of knowledge are covered with issues related to systematic thinking, data abstraction and focus on problem solving as a system. The aim of this discipline (Introduction to Programming) is to focus on the study of the laws and the validity criteria governing thought and demonstration" [3]. Being an essential course for computer programming and therefore to careers related to computing [4], the Introduction to Programming discipline is administered in the first phase of the course, where the students should learn to develop logical reasoning to solve problems.

This discipline involves mental operations involving perception and analysis geared towards the development of knowledge for solutions; however, this discipline is one of the main reasons for dropout and failure in the early stages of technical courses in Computer Science [5]. This scenario therefore highlights the importance of developing a pedagogical practice that enables better results and students interest in learning. The aim of this research is to analyze the contribution of the use of software for teaching logic, adopting Dojo as a pedagogical practice [6] [7] [8] [9]. Dojo is a training technique for programmers, based on tests, designed by David Thomas, who started this practice in Paris in 2003. It was brought to Brazil in 2007 by Ivan Sanchez, from the Dojo Floripa group [10]. The Dojo is considered place where developers gather to train and learn. Meetings are periodic (usually weekly) and focused on a programming challenge [11]. In the light of the above, this paper aims to corroborate the hypothesis that the application of Dojo could facilitate the learning experience of programming, thereby reducing the rate of failure and dropouts, and also to increase interaction between students and teacher, and an abstract view to solve problems. The remainder of the paper is organized as follows. Section 2 describes background and related works. Section 3 presents the proposed methodology to validate our hypothesis. Section 4 reports our case-study and results. Finally, Section 5 concludes our paper and identifies future lines of research.

II. Background and Related Work

Dojo is a periodic meeting organized around a programming challenge where people are encouraged to participate and share their coding skills with the audience while solving the problem. Coding by following Dojo aims at acting as a safe and non-competitive environment, in which everyone is allowed to learn and make mistakes and share their knowledge so that everyone can improve and become better at their craft. Coding dojos are typically coached by a dojo master, who facilitates the space and time, while the participants engage in solving a problem. Often at least part of the participants acts as an audience, whose task is to monitor the working process of other participants and give feedback and improvement suggestions [12]. There are some XP principles that fit Dojo programming. Firstly, Failure promotes to fail when learning something new. Secondly, Redundancy means that one can always gain new insights when tackling the same problem with different strategies. Thirdly, Baby Steps states that step towards the solution should be small enough so that everybody can comprehend and replicate it later. There are

some general rules to make Dojo an easygoing and productive learning environment. The meeting is held in a room with enough space for all the participants and usually requires only a projector and a computer or laptop. Having whiteboard space for sketching and designing discussions is also valuable. The participants are encouraged to develop the solution using Test-Driven Development (TDD) and are free to choose whichever programming language they prefer. Simultaneously, considerable attention is paid to agile methods as a means to improve management of software development processes. The widespread use of such methods in professional contexts has encouraged their integration into software engineering training and undergraduate courses [13]. Along this line, there has been a strong demand for talented ICT (Information and Communication Technology) graduates in the software industry in New Zealand. To meet this demand, in 2015, the government of New Zealand provided funding for three new ICT Graduate Schools. The challenge for the schools was twofold: to provide a qualification for students transitioning into ICT and to prepare those with an ICT education for the workforce [14]. Particularly, introducing students to Dojo programming has been widely reported, showing promising results and insights for improving the teaching of programming in undergraduate courses. Sato et al. describe the experience of creating and running an active Coding Dojo in Sao Paulo, Brazil, sharing the lessons learned from the experience. The role of the Dojo in the learning process is discussed, showing how it creates an environment for fostering and sharing Agile practices such as TDD, Refactoring and Pair Programming, among others [12]. Schoeffel et al. have proposed to assess Coding Dojo. As this technique has been poorly addressed in literature, the research has contributed for finding evidences that it was more effective than usual classes based on speeches. The paper reported an experiment that compared the performance of a group that was exposed to the dojo technique and a control group exposed to usual speech-based classes. Pre and post tests were applied to students from a class of technicians in electronics and informatics. The results showed that the average grade of the group that has participated at the dojo activities was significantly higher than the average of the control group, especially when regarding evaluation of practice [15]. In [16], the authors have proposed an agile approach for bringing Agile practices to the learning community by means of coding dojo. To test the approach, a group of participants was asked to solve a programming task together using TDD and pair programming. In their experiment, they embedded a coding dojo into the Agile practices part of their undergraduate software engineering course. The participating students considered the coding dojo a useful experience, and most of them would recommend participation in coding dojos for their fellow students, as well. In [17], the authors have observed that both Pair Programming and Coding Dojo are rarely used in different types of programming tasks such as front-end programming tasks. Consequently, the authors presented an empirical study comparing Pair Programming and Coding Dojo in the teaching of mockups development. The results showed that Pair Programming was well accepted by the students with positive results. Futhermore, despite the benefits of Coding Dojo in the learning process, students reported several challenges related to motivation and user experience. A gamification approach was reported in [18], in which ClassDojo was designed for increasing students motivation in learning programming. Being also an experimental study the research was designed according to the controlled grouped pretest, posttest research model. The authors highligthed student motivation and positive impact on their lessons. In [19], the authors have presented some Coding Dojo characteristics that help teaching agile development techniques. Experienced practitioners were interviewed to get qualitative information about their perception of the Coding Dojo practice. By means of a survey, the authors concluded that Coding Dojo is a teaching technique to help developers create software with higher test coverage rates. In [20], the authors described the tool AppInventor for allowing students to develop mobile applications in a relatively quickly and easily way using a visual block programming environment. Experiments by means of a survey showed that students were able to collaborate and have fun under this approach. In [6], the authors proposed to use reflective practice as a sense-making device to underpin the investigation and improvement of coding dojo for effective learning. By conducting a case-study with two Dojo sessions, the authors stated that the insights from the reflective practice and related theories can open new and interesting inquiries on coding dojo, and help to better understand the dynamics of coding Dojo, and improve the Dojo practice accordingly. In [21], the authors proposed coding Dojo as a teaching technique to help developers create software with higher test coverage rates. The results showed three mains aspects. Firstly, baby steps helps gradual solutions way and it simplify the process of find a solution to a problem. Secondly, Coding Dojo contributes to learn TDD. Thirdly, pair programming helps the leveling of the group and it is good for the Coding Dojo session.

III. Methodology

As an empirical research, this study was tuned to the observation of a specific group of students [22], using exploratory and descriptive tests to verify learning difficulties face by students. Besides, a trial test of the use of Dojo was utilized as a methodology for teaching. The analysis of the results was made under a qualitative approach, associating the data obtained in the pedagogical activity and frameworks adopted in the study. The participants consisted of students of Introduction to Programming discipline, and the course duration was 140 hours from a of total 1.100 hours of Computer Technician course of the Brazilian Service for Industrial Apprenticeship, Regional Department, Sergipe, in the period of March-April 2017. This course was called Linked Education, and it was developed by the Industry Social Service (SESI) and National Industrial Apprenticeship Service (SENAI), where students attend high school in SESI in concomitant to technical education in the institute. Classes have annual enrollment. The

class analyzed is in its second year and has an average age of students between 14 and 16 years. The students were selected from the morning and afternoon study shifts, and were students of both genres, as shown in Table 1 below: The

TABLE I
DISTRIBUTION OF CLASS AND GENDER OF THE STUDENTS

Shift	**Men**	**Women**	**Total**
Morning	20	05	25
Afternoon	19	10	29
Total	39	15	54

teaching of the discipline started with 8 hours of theoretical approach on Applied Logic to Programming, contextualizing on the formation of the Computer Technician and identifying the relevance of the development of relevant activities in this profession. In continuation according to the teaching program, it was explored the relationship between logic with situations experienced in the daily lives of students. We seek knowledge implied, for example the simple fact of decision-making at the time when an individual crosses a street. This action requires the evaluation of a traffic light signaling impediment or release to the pedestrian.

In these introductory classes we adopted the Scratch tool as an incentive to the performances of classes in groups of four students for troubleshooting. Scratch is a software created by Massachusettes Institute of Technology (MIT) for teaching logic. It allows students to interact in a simple form, and also create virtual animations based on the application logic.

After three classes, totaling 12 hours, we continued with the presentation of the concept, the objectives and rules for the use of Dojo in finding solutions to problems. All students were involved in solving a single problem in the VisualG software. In this dynamic, we proposed a challenge. The purpose of the activity was not related to the resolution of the challenge, but to learn from the experiences from the group. After the definition and explanation of the problem, one of the groups started the coding work from a computer. This activity was exposed to the large group that is for all to follow. One group of students assumed the role of the ”pilot”, developing coding, while the second group acted as ”co-pilot”, observing and assisting the pilot.

Each group had from five to seven minutes to develop the activity. At the conclusion of the time limit, the pilot should join the large group, while the co-pilot takes over the position of the pilot, taking the first place or position. This was done to all the groups in the proposed solution. After this activity, we started with the practice of coding cycle in order to improve the code initially developed. The pedagogical intention is to helo students to understand that an initial solution can be optimized by other contributions, until it becomes the best one. The following lessons were interspersed with theoretical content, individual exercises and new propositions solutions using Dojo, leading to the practice of the contents worked on the lectures. Thus, every 4 hours of theoryl and practice, there were 4 hours of use of Dojo.

IV. CASE-STUDY AND RESULTS

The idea of using the Scratch software for pedagogical experience was based on the constructivist theory, which states that it attaches particular importance to the role of constructions in the world as support to what was thought of in the head mentally, becoming thus least one purely mentalist doctrine [23]. This technological tool has a simple interface, which allows the student to interact with virtual characters, also permitting the creation of interactive stories, games and animations. Its focus is to help young people think creatively, systematically and collaboratively [24]. The software of the target audience are young people between 8 and 16 years old, and thus the course was applied and investigated for students within the same age. The pedagogical proposals were established based on the stages of Piaget [25], putting the formal operational stage into consideration, which happens or occurs from the age of 11 or 12 to the adult stage of an individual. The conditions of human development stage were applied to the extent to which the student breaks down the problem in steps, starting from the general problem in a practical view to the complex thereby. This way, the practice aimed at developing a more abstract view to establish the required thought induced logic.

Using students as active participants in the troubleshooting process ranked them closer to the concrete, i.e, to their closest experiences in a process of middle man interaction as described by Bigge [26] about the perceived reality. The professor and his colleagues were also involved in this proposed dynamic interaction with a view to knowledge, since this movement implied a subject-subject-object relationship [27]. The classroom being the place of occurrence of the activity may take for themselves the prospect of interaction with knowledge and acts of the educational practice. The role of being the main place that develops collective intelligence is also assumed [28].

The act of encoding software results in the creation of an object with a specific significance to the students, as they see the completion of the application using their own knowledge. For this creation, students are encouraged to realize the problem individually, but also being led to consider the collaborative participation of the others in preparing the logical solution. This way, students had an active and interactive participation, which is more positive than the passive reception. Moreover active participation is encouraged by their readiness and learning standards [26].

In the proposed practice, the participation of the teacher is particularly relevant in times of transition from theoretical approach to the detailed explanation of the search for a solution to the proposed problem. In addition, it is essential that the teacher clarifies the possibility of the proposed solution to be reviewed by students and this brings about a continuous cycle of improvement in the initial solution. The process of teaching and learning thus becomes a reconstruction experience [29]. However, the hours of distribution of these moments of practice of the discipline should not lead students

to exhaustion. Daily practices should be avoided, as evidenced by the research spaced practice is more efficient than compact practice [26].

V. Conclusions

Since introduction to programming is essential to the study of programming, students are required to understand concepts and applications pertaining to their training and monitoring in their professional journey. The teacher in charge of this course is responsible for creating a motivating and attractive environment to help develop students cognition. The experiment conducted in this study, using the Dojo as an educational tool proved to be positive with surprising results among the zero rate of dropout and 6% of absences. In previous classes, with the same teacher and teaching practices but without the use of Dojo, class absences reached a high rate of 23%. At the end of the dynamic experiments, there was a report from students who claimed to be motivated, and therefore avoided skipping school. Though this experiment has been performed in only two classes, and the results are still preliminary, it is possible to observe that the interaction and creative application enables better assimilation of programming for the students. In the evaluations period prior to the use of Dojo, there were listed averages between six and seven, six being the minimum for approval.

After the experiment, the grades increased to the average of seven point five and eight, highlighting the students' opinions on the satisfaction gained in the proposed activity. Students have to be prepared for the logical processes necessary for the knowledge and skills to be worked on the course. Even though our preliminary positive results, the subsequent discipline called Programming Desktop requires experience in longer periods of observation, and in other groups. For this reaseon, we claim the adoption of Dojo as a pedagogical practice for the development of logical, abstract and systemic thinking. Therefore, the role of activities and tools used would help the teacher in making the teaching process more interactive and collaborative. Bearing in mind that the research does not hinder the possibilities of success with other practices, more case studies can be developed for the search of other tools to be applied to the teaching of logic, such as games and use of robotic lego.

Acknowledgment

Thank you for Tiradentes University (UNIT) support for this research.

References

[1] J. R. Rice, S. Rosen, History of the department of computer sciences at purdue university, Purdue University [ ]. : http://www. cs. purdue. edu/history/history. html [. .: 22 2010 .].

[2] Worldskills, WorldSkills International History, Ipsis Grfica e Editora, 2010.

[3] J. Mazano, J. d. Oliveira, Algoritmos: lógica para desenvolvimento de programação. 21a, São Paulo: Érica.

[4] E. P. Pimentel, V. F. de França, R. V. Noronha, N. Omar, Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores, in: Anais do Workshop de Informática na Escola, Vol. 1, 2003, pp. 533–544.

[5] Z. S. d. Silveira, Contradições entre capital e trabalho: concepções de educação tecnológica na reforma do ensino médio e técnico.

[6] J. Rooksby, J. Hunt, X. Wang, The theory and practice of randori coding dojos, in: International Conference on Agile Software Development, Springer, 2014, pp. 251–259.

[7] L. Bossavit, E. Gaillot, The coders dojo–a different way to teach and learn programming, in: International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer, 2005, pp. 290–291.

[8] M. Bravo, A. Goldman, Reinforcing the learning of agile practices using coding dojos, in: International Conference on Agile Software Development, Springer, 2010, pp. 379–380.

[9] D. T. Sato, H. Corbucci, M. V. Bravo, Coding dojo: An environment for learning and sharing agile practices, in: Agile, 2008. AGILE'08. Conference, IEEE, 2008, pp. 459–464.

[10] E. Bache, The coding dojo handbook, Emily Bache, Canada.

[11] Dojo.
URL http://apoie.org/Dojo.html

[12] D. T. Sato, H. Corbucci, M. V. Bravo, Coding dojo: An environment for learning and sharing agile practices, in: Agile, 2008. AGILE'08. Conference, IEEE, 2008, pp. 459–464.

[13] G. Rodríguez, Á. Soria, M. Campo, Measuring the impact of agile coaching on students performance, IEEE Transactions on Education 59 (3) (2016) 202–209.

[14] Y.-C. Tu, G. Dobbie, I. Warren, A. Meads, C. Grout, An experience report on a boot-camp style programming course, in: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, ACM, 2018, pp. 509–514.

[15] P. Schoeffel, D. F. Rosa, R. S. Waslawick, Um experimento do uso de coding dojo na aprendizagem de programação orientada a objetos, iSys-Revista Brasileira de Sistemas de Informação 9 (2).

[16] K. Heinonen, K. Hirvikoski, M. Luukkainen, A. Vihavainen, Learning agile software engineering practices using coding dojo, in: Proceedings of the 14th annual ACM SIGITE conference on Information technology education, ACM, 2013, pp. 97–102.

[17] B. Estácio, N. Valentim, L. Rivero, T. Conte, R. Prikladnicki, Evaluating the use of pair programming and coding dojo in teaching mockups development: An empirical study, in: System Sciences (HICSS), 2015 48th Hawaii International Conference on, IEEE, 2015, pp. 5084–5093.

[18] H. H. Özer, H. Bicen, Determining the effects of class dojo application on student success and perception, International Journal of Scientific Study.

[19] R. B. Da Luz, A. G. S. S. Neto, R. V. Noronha, Teaching tdd, the coding dojo style, in: Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 371–375.

[20] A. M. de Jesus, G. M. da Silva, I. F. Silveira, Using coding dojo with mobile game development to engage students to learn programing (2017) 50–58.

[21] R. B. da Luz, A. G. S. S. Neto, Coding dojo the social method to teach test driven development.
URL https://goo.gl/GJxqQ1

[22] R. K. Yin, Case study research: Design and methods (applied social research methods), London and Singapore: Sage.

[23] S. Papert, Mindstorms: Children, computers, and powerful ideas, Basic Books, Inc., 1980.

[24] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al., Scratch: programming for all, Communications of the ACM 52 (11) (2009) 60–67.

[25] J. Piaget, D. Elkind, Six psychological studies, Vol. 462, Vintage Books, 1968.

[26] M. L. Bigge, Learning theories for teachers, Harper & Row, 1982.

[27] L. Gamez, Psicologia da educao, LTC, 2013.

[28] V. M. Kenski, Educação e tecnologias, Papirus editora, 2007.

[29] J. Dewey, Experience and education, Kappa Delta Pi, 1998.