Is Pair Programing in Higher Education a Good Strategy?

Sónia Rolland Sobral

***Abstract*—Pair programming is by definition two-person programming on the same computer. The technique has been used in many higher education institutions and has been reported in some scientific articles, usually for introductory to programming courses.**

The aim of this article is to make a situation report analyzing the scientific production on pair programming for curricular units of introduction to programming in higher education, measuring the advantages and disadvantages of the strategy. The sample was composed by 153 articles indexed in Elsevier's Scopus. The results obtained by bibliometric analysis showed the publication rates, authors, in which journals they are published, which are the organizations and countries that publish the most, which are the most cited articles and what their purpose. The benefits reported are generally better code, improved programming and group skills, advantages for women and reducing the work of instructors. The problems are group compatibility: there are studies that randomly distribute pairs, while other use personality tests or knowledge self-assessment.

***Index Terms*—Pair programming, CS1, introduction to programming, higher education.**

I. Introduction

Pair programming is a technique that uses a computer, a mouse and a keyboard for a group of two programmers. Their positions change but one of them is the driver and the other is the navigator: both work as a team in which the driver's function is to write code, while the navigator's function is correcting the errors and monitoring the process. Programming in pairs is done synchronously but can be carried out by a pair located in different places, maintaining the positions of the driver and navigator (distributed programming in pairs). Pair programming is one of the techniques pertaining to extreme programming that is included in agile software techniques, whose primary objective is to write better code in the least possible time. Pair programming initially used in the industry has often been used as collaborative learning especially in introductory programming courses, but not only. The benefits of this strategy have often been defended by the academic community, reporting improvements in the quality of the code, faster execution of tasks, more confidence and satisfaction of the stakeholders. Others report problems in the groups and in the compatibility of the pairs, seeing no advantage in the strategy. The way pairs are formed is the concern of others, some who consider that it should be done randomly and in a rotating way, others who study the possibility of creating pairs based on their personalities. There are several studies and experiences with different objectives: improvement quality and time, increase students' confidence, their satisfaction in the tasks of learning to program, decrease the work of teachers and instructors, try to decrease the dropout rate or even gender concerns and the rise of women in computing. All studies that are related to education, and higher education, have a common focus: improving teaching / learning. The literature review shows that the results depend a lot on how the methodologies are applied.

This study aims to verify what type of publications have been made on the use of peer programming for introductory courses in higher education programming. Using bibliometric methods (and not only) we propose to understand what the objectives and concerns of those are who investigate this issue. Bibliometric analysis [1] is the quantitative study of bibliographic material: it provides a general picture of a research field that can be classified by papers, authors, and journals. Bibliometric methods employ a quantitative approach for the description, evaluation, and monitoring of published research. These methods have the potential to introduce a systematic, transparent, and reproducible review process and thus improve the quality of reviews [2]. Bibliometric analysis provides objective criteria that can assess the research development in a field and act as a valuable tool for measuring scholarship quality and productivity [3]. Bibliometric methods offer systematization and replication processes that can improve understanding of the dissemination of knowledge in a field and can highlight gaps and opportunities that may contribute to the advancement of the discipline [4]. The sample was composed by 153 articles indexed in Elsevier's Scopus. The results obtained by bibliometric analysis showed the publication rates, authors, in which journals they are published, which are the organizations and countries that publish the most, which are the most cited articles and what is the purpose of the most cited articles

This article is divided into five sections: we start by reviewing the literature, then we present our research questions. The following section presents the methodology and how the data is collected, the results are presented and at the end the conclusions are presented.

II. Literature Review

Pair programming is a practice in which two programmers work collaboratively at one computer, on the same design, algorithm, or code [5], [6]. All programming tasks are done in pairs at one display, keyboard, and mouse [7], [8]. The tasks include all phases of the development process (design, debugging, testing, etc.) not just coding [9].

One is the driver, who controls the keyboard and mouse

Manuscript received March 3, 2020; revised October 7, 2020.

Sónia Rolland Sobral is with the REMIT – Universidade Portucalense, Porto, 4200 Portugal (e-mail: sonia@upt.pt).

doi: 10.18178/ijiet.2020.10.12.1478

and is responsible for entering program code. The other is the navigator and sits next to the driver and watches for errors, discusses alternative design approaches, offers suggestions [10], [11]. He continuously assuring quality, trying to understand, asking questions, looking for alternative approaches, helping to avoid defect alone [12]. The programmers regularly trade roles while pairing creator becomes quality assurer and vice versa [13]. Code written by only one member of the pair is reviewed by both partners together before it is officially accepted as part of the program. The driver and navigator can brainstorm on demand at any time, communicating at least every 45 seconds to a minute [14]. One version is the Distributed pair programming, where the pair is not sitting side by side on the same computer: they synchronously collaborate on the same design or code, but from different locations [14]. To be able to do this, they require technological support for sharing desktops and verbal conversation or even capability of video conferencing with Web cams if required [14]. Pair-think refers to the pair's enhanced ability to generate and evaluate alternatives and pair relaying is the effect of having two people working to resolve a problem together [15].

In 1995, Larry Constantine [16] reported observing dynamic duos at Whitesmiths, Ltd. producing code faster and more bug-free than ever before: "Two programmers in tandem is not redundancy; it's a direct route to greater efficiency and better quality". Then in 1999, Extreme Programming (XP), a collection of well-known software engineering practices, was conceived and developed to address the specific needs of software development, producing all software in pairs, two programmers at one screen [9]. XP is one of the agile software process paradigms. Pair programming is one of the key practices in XP [11], which operates on 12 core principles: the planning game, continuous testing, on-site customer, small releases, refactoring, a 40-h work week, system metaphor, continuous integration, simple design, collective code ownership, coding standards, and pair programming [17]. XP uses short iterations with small releases and rapid feedback, close customer participation, constant communication and coordination, continuous refactoring, continuous integration and testing, collective code ownership and pair programming [18]. The first empirical study was published in 1998 by Professor Nosek [19] from Temple University. He reported on his study 15 programmers working for 45 minutes on a challenging problem, important to their organization, in their own environment, and with their own equipment. Results showed that pair programming improved both their performance and their enjoyment of the problem-solving process. The groups completed the task 40% more quickly and effectively by producing better algorithms and code in less time. Generally, programmers are skeptical about working in pairs: usually a programmer's job is lonely and silent, but after trying it out they become very fond and most programmers grow to prefer pair programming. They admit to working harder and smarter on programs because they do not want to let their partner down [15]. Many benefits are described in the literature, such as increased productivity, improved code quality, enhanced job satisfaction, confidence [11], and less time to solve the problem than individuals [19].

Pair programming has become widely accepted as an alternative to solo programming: when they pair off they find solutions which none of them would have found alone [12] and started to be used in teaching as a collaborative teaching strategy, especially in the introduction to programming courses, but not only. The first experiences were carried out at North Carolina State University [5], [20], University of Karlsruhe [7], [12] and University of California at Santa Cruz [21], [22] among others. In these cases, the courses were always CS1, computer science one. Findings included more confident students, greater course completion and pass rates [17], a more likely to persist in computer-related majors, reduced workload for the teaching staff [22]. There are studies that prove the speed of getting the tasks done: assignments 40 – 50% faster than solo developers [23]. It often benefits women and can be a solution for more women in computer science courses [10], [24].

But there are those who hate this strategy using various arguments, the first of which is a huge expenditure of time, money and resources: when two people are doing the same task, the spent effort is doubled [11], pairs spend almost twice as much total programmer effort as solo programmers [13]. Others doubt the benefits and say there is a need for more rigorous studies to compare the effectiveness of pair programming with reviewing techniques [7], [12]. Other studies find reasons that make the findings unclear: students with lower self-reported programming skill enjoy pair programming more than students with higher self-reported programming skill, work their best when the pair is at their own level and don't like working with peers who think they have a lower level of knowledge [25]. However, educators cannot predict this perception, nor can pairs be formed based on similar technical competence [20].

The most critical aspect of creating an effective pair programming implementation is to minimize the potential scheduling conflicts between partners [21]. Many of the studies that have been done on pair programming are related to compatibility like personality type, learning style, skill level, programming self-esteem, work ethic or time management preference [26]. The team's success depends on how effective they work as a team, despite their skills and abilities [27]. Students notably preferred to pair with a partner of similar or higher skill level, pairs comprised of a sensor and an intuited learning style seem to be compatible, and pairs with differing work ethic are generally not compatible [26]. Some studies investigated personality type using the Myers-Briggs Type Indicator to measure an individual's personality based on extroversion vs. introversion, sensing vs. intuition, thinking vs. feeling and judging vs. perceiving [9]. Other studies are based the five-factor NEO Personality Inventory: Agreeableness, Conscientiousness, Extraversion, Neuroticism, and Openness to experience [27]. But there are studies that find as a result that personality type had no effect on the results, so these pairings can be treated as random [28].

III. The Research Question

The question, along with the purpose of the review, the intended deliverables, and the intended audience, determines

how the data are identified, collected, and presented [29]. The questions that we want to answer in this paper are

- How has the evolution of the publication of articles related to pair programming in higher education been? Where were they published? What is the focus of these articles? Who published them?
- What are the most cited articles? Who writes them? What is the objective of those papers?

IV. Methodology and Data

The term bibliometrics was first used in 1969 by Alan Pritchard, hoping that the term would be used explicitly in all studies which seek to quantify the processes of written communication and would quickly gain acceptance in the field of information science [30]. Moed mentioned the potential of this type of study that reveals the enormous potential of quantitative, bibliometric analyses of the scholarly literature for a deeper understanding of scholarly activity and performance and highlights their policy relevance [31]. In scientific research, it is important to get a wider perspective of research already being conducted concerning a relevant subject matter [32] and a bibliometric analysis profile on the research trajectory and dynamics of the research activities across the globe [33]. This is a bibliometric study that systematically analyses the literature using articles indexed at Elsevier's Scopus (Scopus) database. This study conducts a bibliometric analysis that we expect provides a useful reference for future research. The search strategy was

TITLE-ABS-KEY ("pair programing" OR "programming in pairs" OR "paired programming") AND

TITLE-ABS-KEY ("higher education" OR "CS1" OR "university").

V. Results

A. Annual Evolution

A set of 153 published papers were collected. The first article in Scopus was published in 2000. Growth does not have a clear order, as can be seen in the following figure (Fig. 1). The year with the highest number of publications is 2008 (n=14).

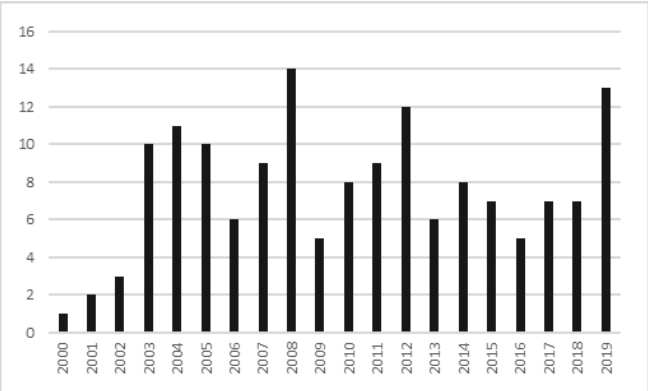

Fig. 1. Annual evolution.

B. Where

66% are conference papers: 102 conference papers, 34 journal articles, 14 conference review and three journal reviews.

There are 92 different publishing locations: 15 ACM Technical Symposium On Computer Science Education, nine in Lecture Notes in Computer Science, six from Americas Conference On Information Systems, five from Proceedings Frontiers In Education Conference, from Proceedings International Conference On Software Engineering and SIGCSE Bulletin Association For Computing Machinery Special Interest Group On Computer Science Education.

C. Focus

There are 160 different keywords. The most frequent are: Students, Pair Programming, Pair-programming, Computer Programming, Teaching, Software Engineering, Curricula, Engineering Education and Computer Science (Fig 2).

There are three clusters: cluster 1 with six items (Computer Science education, distributed pair programming, empirical software engineering, extreme programming, gender and pair programming), cluster 2 with four items (active learning, collaboration, collaborative learning, pair-programming) and cluster 3 with two items (cs1 and software engineering).

D. Who

39 articles have one or three authors. There are 31 articles written by two, 28 by four and 16 by five or more authors.

Williams, Laurie A. from Carolina A&T State University, Carolina, United States is the author with the most articles (9). Then Hanks, Brian F. from BFH Educational Consulting, Seattle, United States (10 papers) and Mendes, Emilia from Blekinge Tekniska Högskola, Karlskrona, Sweden (with six articles). There are authors from 31 countries: 49% are from the United States. All articles are written in English. 20 authors are from North Carolina State University. There are three clusters: c1 (Australia, Malaysia, New Zealand and Sweden), C2 (Canada and United States) and C3 (United States and Taiwan) (Fig. 3).

E. The Twenty Most Cited Documents

The 20 most cited documents are mainly from the beginning of the century (Fig 4). The first [5] was cited 201 times and the 20th [28] was cited 37 times.

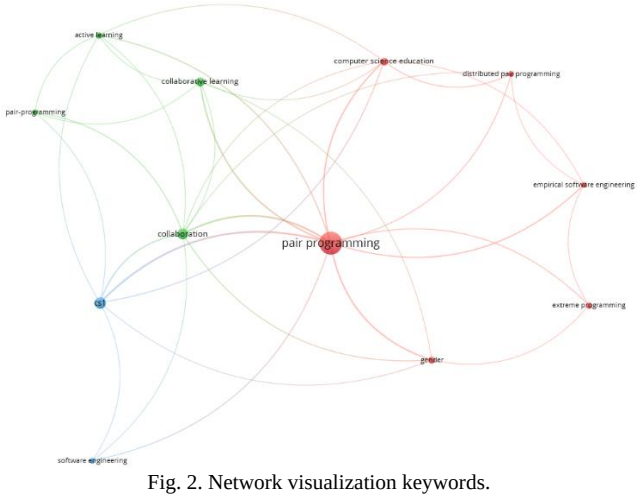

Fig. 2. Network visualization keywords.

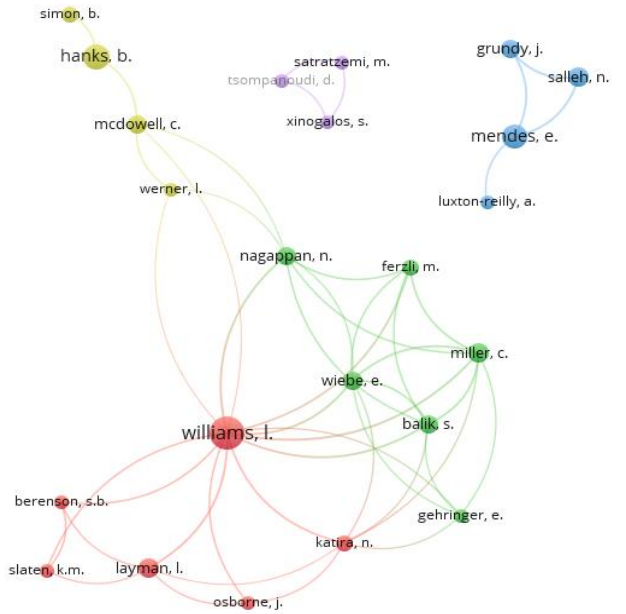

Fig. 3. Network visualization authors.

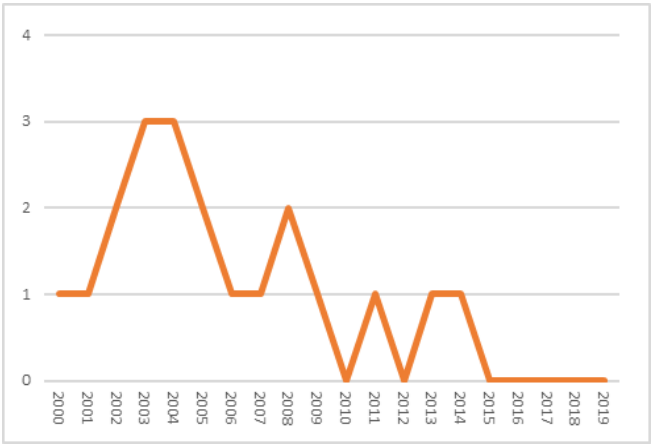

Fig. 4. Annual evolution top20.

Six are journal articles, 13 are Conference Paper and one is a review.

The affiliation of the authors of the four articles with more than 100 citations is North Carolina State University [5], International Islamic University Malaysia + University of Auckland + Swinburne University of Technology [27], Singapore Management University [34] and University of California, Santa Cruz [10].

Eight of the ten most cited papers report experiments, one is a review [27] and the other is a summary of the existing empirical knowledge on pair programming [11].

The experiences are diverse:

[5] An experiment in an introductory Computer Science course at North Carolina State University. In the fall 2001, 112 students were in the solo section and 87 were in the paired section, whereas in the spring 2002, 156 students worked solo and 346 students worked in pairs. Solo lab sessions were quiet and appeared to be very frustrating for the students. Alternately, paired labs were vocal and interactive. Results indicate that pairing helped the non-CS majors but did not cause any significant improvement among the CS majors. Student pair programmers were more self-sufficient, generally perform better on projects and exams, and were more likely to complete the class with a grade of C or better than their solo counterparts. Results indicate that pair programming creates a laboratory environment conducive to more advanced, active learning than traditional labs; students and lab instructors report labs to be more productive and less frustrating.

[34] A trial of the flipped classroom model for a programming course with pair programming as the predominant in-class active learning activity at Singapore Management University, with 46 Information Systems (IS) undergraduates during a special term in 2013. Student feedback on this pedagogy was generally very positive with many respondents considering it effective and helpful for learning. One of the biggest advantages mentioned by students is that they had the option to watch each video lecture as many times as required to be prepared for class. The author also observed that students were more engaged and empowered to take on more ownership for their learning.

[10] In the 2000-2001 academic year, 555 students (141 women, 413 men, and 1 whose gender was not reported) participated in a study on pair-programming, introductory programming course at UCSC. They studied four sections: three of the sections students pair-programmed; in the fourth they worked individually. Pair-programming is shown to be beneficial to all students, particularly beneficial for women. The collaborative nature of pair-programming teaches women students that software development is not the competitive, socially isolating activity that they imagined. It encourages women to pursue computer science as a major and as a potential career.

[7] An experiment at University of Karlsruhe, summer of 2000: 12 participants were computer science graduate students who needed to take a practical training course as part of their degree requirements. Project teams consisted of six students (three pairs). The students were asked to pair with different partners of their own choosing for each exercise and the project. Findings include it is unclear how to reap the potential benefits of pair programming, although pair programming produces high quality code. Designing in small increments appears problematic but ensures rapid feedback about the code. Writing test cases before coding is a challenge. It is difficult to implement XP without coaching.

[35] Experiment in UC San Diego, 1011 students, fall 2008, CS1, incorporating a trio of best practices: Media Computation, Pair Programming and Peer Instruction. Results: fewer students dropping, more students passing, and more passing students retained.

[22] Compares two experiments: 1200 students, two US universities, North Carolina State University and University of California Santa Cruz, to assess the efficacy of pair programming as an alternative learning technique in introductory programming courses. Students who used the pair programming technique were at least as likely to complete the introductory course with a grade of C or better when compared with students who used the solo programming technique. Paired students earned exam and project scores equal to or better than solo students. Paired students had a positive attitude toward collaboration and were significantly more likely to be registered as computer science-related majors one year later. Students in paired classes continue to be successful in subsequent programming

classes continue to be successful in subsequent programming classes that require solo programming.

[36] A web programming course taught at the University of Utah in Summer Semester 1999. 20 students worked in pairs, continuously collaborating on all programming assignments. It proved beneficial to the quality of their work products, allowed them to learn new languages faster and better than they had experienced with solitary learning. 'Pair-learning' also reduced the workload of the teaching because the students no longer relied primarily on them for technical support and advise.

[25] University of Wales, 60 students. Three groups by the self-placement of the students as expressed on the 'Programming Attitude Questionnaire'. Students with less self-confidence seem to enjoy pair programming the most. There is some evidence that warriors like pair programming even less when they are paired with phoebes and that student's produce their best work when paired with students of similar, or not very different, levels of confidence.

The systematic literature review of seventy-four papers of empirical studies that investigated factors affecting the effectiveness of PP for CS/SE students and studies that measured the effectiveness of PP for CS/SE students [27] suggest that PP was rarely employed in courses where students were exposed to software design/modeling tasks; paired students achieve productivity similar or better than solo students; and indicate that implementing PP in the classroom or lab does not lead to any detrimental effect on students' academic performance.

VI. Conclusions

Pair programming is by definition two-person programming on the same computer. The technique has been used in many higher education institutions and has been reported in some scientific articles, usually for introductory to programming courses. In this article, we reviewed the existing literature in this area, analyzing all publications in the Scopus database, a set of 153 published papers published from 2000.

The results show that there is no continuous evolution of publications, and the year in which most articles were published was 2008 (n = 14). 66% are conference papers. The location is different: there is no journal or conference where most articles are published. There are three keyword clusters and the focus is clearly students, Pair Programming and computer programming. Almost half of the authors (49%) have affiliation in the United States. Laurie A. Williams, from Carolina A&T State University, Carolina, United States is the author with the most articles (9). All articles are written in English. Regarding the most cited articles: there are four articles that have more than 100 citations and the 20th article in this list has been cited 37 times. Eight of the 10 most cited articles report experiences using pair programming. The results of the experiments show the benefits of using peer programming, showing that the compatibility of the two members is important as it may or may not dictate the success of the strategy.

Most articles show the benefits of pair programing reporting experiments for introductory programming courses. The benefits are generally better code, improved programming and group skills, advantages for women (and consequently a way of increasing the number of women in the IT area) and reducing the work of instructors. The reported problems are group compatibility: there are studies that randomly distribute pairs, while other studies use personality tests or knowledge self-assessment. Very important: almost all publications report that students enjoyed the experience, which makes it a good strategy for teaching programming to higher education students.

Conflict of Interest

This study was carried out without a conflict of interest.

Author Contributions

The author did a literature review, defined the methodology, research and data treatment, data analysis and conclusions, having written the entire document.

References

[1] J. Merigo and J. Yang, "A bibliometric analysis of operations research and management science," *Omega*, vol. 73, pp. 37-48, 2017.

[2] I. Zupic and T. Čater, "Bibliometric methods in management and organization," *Organizational Research Methods*, vol. 8, no. 3, 2015.

[3] M. Cobo, A. López-Herrera, and E. Herrera-Viedma, "SciMAT: A new science mapping analysis software tool," *Journal of the American Society for Information Science*, vol. 3, no. 8, pp. 1609-1630, 2012.

[4] G. Aparicio, T. Iturralde, and A. Maseda, "Conceptual structure and perspectives on entrepreneurship education research: A bibliometric review," *European Research on Management and Business Economics*, vol. 5, no. 3, pp. 105-113, 2019.

[5] N. Nagappan *et al.*, "Improving the CS1 experience with pair programming," *ACM SIGCSE Bulletin*, vol. 35, no. 1, pp. 359-362, 2003.

[6] S. Berenson *et al.*, "Voices of women in a software engineering course: Reflections on collaboration," *Educational Resources in Computing*, vol. 4, 2004.

[7] M. Muller and W. Tichy, "Case study: Extreme programming in a university environment," presented at International Conference on Software Engineering, 2001.

[8] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley , 1999.

[9] K. D. F. I. I. Choi, "Exploring the underlying aspects of pair programming: The impact of personality," *Information and Software Technology*, vol. 50, no. 11, pp. 1114-1126, 2008.

[10] L. Werner, C. McDowell, and B. Hanks, "Pair-programming helps female computer science students," *ACM Journal on Educational Resources in Computing*, vol. 11, no. 4, p. 4, 2004.

[11] H. Hulkko and P. Abrahamsson, "A multiple case study on the impact of pair programming on product quality," presented at International Conference on Software Engineering, 2005.

[12] M. Muller, "Two controlled experiments concerning the comparison of pair programming to peer review," *Journal of Systems and Software*, vol. 78, no. 2, pp. 166-179, 2005.

[13] J. Nawrocki and A. Wojciechowski, "Experimental evaluation of pair programming," *European Software Control and Metrics*, 2001.

[14] P. G. E. S. D. Baheti, "Exploring the efficacy of distributed pair programming," *Lecture Notes in Computer Science*, pp. 208-220, 2002.

[15] L. Williams, "Integrating pair programming into a software development process," presented at Conference on Software Engineering Education and Training, in Search of a Software Engineering Profession, Charlotte, 2001.

[16] L. Constantine, *Constantine on Peopleware*, Prentice Hall, 1995.

[17] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "Pair programming improves student retention, confidence, and program quality," *Communications of the ACM*, vol. 49, no. 8, pp. 90-95, 2006.

[18] P. Abrahamssona, J. Warstab, M. Siponenb, and J. Ronkainen, "New directions on agile methods: A comparative analysis," presented at International Conference on Software Engineering, 2003.

[19] J. Nosek, "The case for collaborative programming," *Communications of the ACM*, vol. 41, no. 3, 1998.

[20] N. Katira, L. Williams, E. Wiebe, S. Balik, and E. Gehringer, "On understanding compatibility of student pair programmers," *SIGCSE*, 2004.
[21] J. W. L. M. C. Bevan, "Guidelines for the use of pair programming in a freshman programming class," presented at Software Engineering Education Conference, 2002.
[22] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner, "Building pair programming knowledge through a family of experiments," presented at International Symposium on Empirical Software Engineering, 2003.
[23] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE Software*, vol. 17, pp. 19-25, 2000.
[24] S. S. K. W. L. H. C.-W. Berenson, "Voices of women in a software engineering course: Reflections on collaboration," *Journal on Educational Resources in Computing*, vol. 4, no. 1, 2004.
[25] L. Thomas, M. Ratcliffe, and A. Robertson, "Code warriors and code-a-phobes: a study in attitude and pair programming," *SIGCSE*, 2003.
[26] L. L. L. O. J. K. N. Williams, "Examining the compatibility of student pair programmers," presented at AGILE Conference, 2006.
[27] N. M. E. G. J. B. G. Salleh, "An empirical study of the effects of personality in pair programming using the five-factor model," presented at International Symposium on Empirical Software Engineering and Measurement, 2009.
[28] J. H. L. H. L. H. J. R. D. Carver, "Increased retention of early computer science and software engineering students using pair programming," presented at Software Engineering Education Conference, 2007.
[29] A. Booth, A. Sutton, and D. Papaioannou, *Systematic Approaches to a Successful Literature Review*, 2md ed, SAGE Publications Ltd, 2016.
[30] A. Pritchard, "Statistical bibliography or bibliometrics," *Journal of Documentation*, vol. 25, pp. 348-349, 1969.
[31] H. F. Moed, *Citation Analysis in Research Evaluation*, vol. 9, Springer, 2005, p. 348.
[32] S. M. R. P. Z. E. A. Bojović, "An overview of forestry journals in the period 2006–2010 as basis for ascertaining research trends," *Scientometrics*, vol. 8, pp. 1331–1346, 2014.
[33] W. G. M. H. G. E. A. Liu, "Profile of developments in biomass-based bioenergy research: A 20-year perspective," *Scientometrics*, vol. 99, pp. 507–521, 2014.
[34] H. Mok, "Teaching tip: The flipped classroom," *Journal of Information Systems Education*, vol. 25, no. 1, pp. 7-11, 2014.
[35] L. Porter and B. Simon, "Retaining nearly one-third more majors with a trio of instructional best practices in CS1," presented at ACM Technical Symposium on Computer Science Education, Denver, 2013.
[36] L. Williams and R. Kessler, "Effects of 'pair-pressure' and 'pair-learning' on software engineering education," presented at Conference on Software Engineering Education, Austin, 2000.

Sónia Rolland Sobral is a professor at Universidade Portucalense since 1993 and currently she is a researcher at REMIT. Her research is in economics, management, and information technologies. She is an aggregate (Dr. Habil) in information sciences, doctorate (PhD) in information systems and technologies, the master (MSc) in electrical and computer engineering and degree in management informatics. She has more than 100 scientific publications and her focus is the distance education, serious games, computer programming and higher education policies. She is addicted to sports, and seriously passionate about technology and travel.