CIÊNCIAS SOCIALMENTE APLICÁVEIS:

INTEGRANDO SABERES E ABRINDO CAMINHOS

JORGE JOSÉ MARTINS RODRIGUES
MARIA AMÉLIA MARQUES

(Organizadores)

VOL VIII

EDITORA ARTEMIS
2023

CIÊNCIAS SOCIALMENTE APLICÁVEIS:

INTEGRANDO SABERES E ABRINDO CAMINHOS

JORGE JOSÉ MARTINS RODRIGUES
MARIA AMÉLIA MARQUES
(Organizadores)

VOL VIII

EDITORA ARTEMIS
2023

Editora Chefe	Profª Drª Antonella Carvalho de Oliveira
Editora Executiva	M.ª Viviane Carvalho Mocellin
Direção de Arte	M.ª Bruna Bejarano
Diagramação	Elisangela Abreu
Organizadores	Prof. Dr. Jorge José Martins Rodrigues Prof.ª Dr.ª Maria Amélia Marques
Imagem da Capa	cienpies
Bibliotecário	Maurício Amormino Júnior – CRB6/2422

Editora Artemis
Curitiba-PR Brasil
www.editoraartemis.com.br
e-mail:publicar@editoraartemis.com.br

Editora Artemis
Curitiba-PR Brasil
www.editoraartemis.com.br
e-mail:publicar@editoraartemis.com.br

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

C569 Ciências socialmente aplicáveis [livro eletrônico] : integrando saberes e abrindo caminhos: vol. VIII / Organizadores Jorge Rodrigues, Maria Amélia Marques. – Curitiba, PR: Artemis, 2023.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
Edição bilíngue
ISBN 978-65-87396-81-1
DOI 10.37572/EdArt_300523811

1. Ciências sociais aplicadas – Pesquisa – Brasil. 2. Abordagem interdisciplinar do conhecimento. I. Rodrigues, Jorge José Martins. II. Marques, Maria Amélia.

CDD 307

Elaborado por Maurício Amormino Júnior – CRB6/2422

EDITORA ARTEMIS
2023

Editora Artemis
Curitiba-PR Brasil
www.editoraartemis.com.br
e-mail:publicar@editoraartemis.com.br

APRESENTAÇÃO

O oitavo volume desta coleção segue a lógica dos livros anteriores. Procura apresentar ao leitor uma coletânea de artigos sobre problemáticas que são transversais ao campo das ciências sociais aplicadas.

Sendo discutível, na metodologia seguida na organização dos vários volumes procurou-se privilegiar artigos que abordassem novas tendências e/ou problemáticas transversais relevantes, adotassem metodologias mais holísticas e/ou modelos de investigação aplicada, apresentassem estudos de caso nacionais e/ou internacionais e procurassem ser reflexivos. Nesse contexto, o presente volume está organizado em três grandes eixos – Programação, Sustentabilidade, Educação e redes sociais.

Na construção da estrutura de cada eixo procurou-se seguir uma lógica em que cada artigo possa contribuir para uma melhor compreensão do artigo seguinte, gerando-se um fluxo de conhecimento acumulado que se pretende fluido e em espiral crescente.

Assim, o eixo Programação é constituído por um conjunto de oito artigos. A programação pode ser entendida como um conjunto de actividades que visam transformar tarefas repetitivas e monótonas em rotinas cooperativas e colaborativas. Estas rotinas são algoritmos e modelos matemáticos geradores de informação estruturada e eficiente que, apesar da sua racionalidade limitada, é útil para a tomada de decisões, sejam individuais ou de grupo.

O eixo Sustentabilidade junta um conjunto de sete artigos que, em comum, contribuem para a construção da responsabilidade social. As mudanças climáticas estão a perturbar a vida de milhões de pessoas no planeta, com especial ênfase nas regiões rurais mais pobres e com impacto negativo na economia. Assim, exigem-se políticas públicas inclusivas que incentivem o uso de materiais multiusos, amigos do ambiente. Os resíduos sólidos urbanos necessitam de ser melhor geridos e as empresas deverão ser incentivadas a incorporar aquelas políticas nas suas estratégias, para reforço dos seus valores, conforto e bem-estar dos seus constituintes.

O eixo Educação e redes sociais tem seis artigos. As principais teorias de liderança parecem apontar para que esta seja contingencial, podendo ser ensinada e as respectivas competências treinadas e melhoradas. Todo o ensino, presencial ou a distância, tem os seus pontos fortes e pontos fracos. Exigem-se comportamentos éticos, nomeadamente em ambiente de redes sociais, para evitar fraudes quer com os conteúdos quer com a respectiva avaliação, com eventuais traumas psicológicos em quem é visado.

Com a disponibilização deste livro e seus artigos esperamos que os mesmos gerem inquietude intelectual e curiosidade científica, procurando a satisfação de novas necessidades e descobertas, motor de todas as fontes de inovação.

Jorge Rodrigues, ISCAL/IPL, Portugal
Maria Amélia Marques, IPS/ESCE, Portugal

SUMÁRIO

PROGRAMAÇÃO

CAPÍTULO 1

NUMERICAL CALCULATION BASED ON AGILE PROGRAMMING DEVELOPMENT TRAINING

Data de submissão: 20/04/2023
Data de aceite: 05/05/2023

Ángel Rubén Barberis
Faculty of Exact Sciences of the National University of Salta
Doctor from the University of Buenos Aires in Computer Science
Master of the University of Buenos Aires in Numerical Simulation and Control
Graduate in Systems Analysis
Salta-Capital, Argentina

Lorena Elizabeth Del Moral Sachetti
Orán Regional Headquarters of the National University of Salta
Specialist in Institutional Psychopedagogy from the National University of Salta
Graduate in Systems Analysis
San Ramón de la Nueva Orán-Salta
Argentina

Jorge Alberto Silvera
Faculty of Exact Sciences of the National University of Salta
Graduate in Systems Analysis
Salta-Capital, Argentina

ABSTRACT: Computer programming is a set of activities that make it difficult to teach and learn. Consequently, learning to program is one of the first and most challenging task that computer students face in the first years of career. Difficulties extend and impact strongly on other subjects that contemplate programming as one of their objectives in pedagogical practice. The reality of university studies in Computer Science is not alien to that of other careers, which can be seen in the high failure rates and high dropout rates in professional training. This paper presents a methodological strategy that allows students to train in agile software development as an approach that goes beyond the mere practice of computer application programming. The strategy promotes cooperative and collaborative actions in teamwork, which facilitates the student's acquisition of professional programmer's skills.
KEYWORDS: Agile training. Agile programming. Agile programming. Programming training. Scrum. Programming with Scrum.

1 INTRODUCTION

Currently, Computer Science careers are strongly impacted by the high dropout rates of students in subjects related to computer programming. The effect is transmitted from subjects that introduce the first programming

concepts (Bennedsen & Caspersen, 2019) to those that do not teach programming, but whose academic objectives include the implementation of software development (Lahtinen et al., 2005). The art of computer programming is a complex and difficult task to approach academically (Sarpong et al., 2013). The complexity of the Educational Process of Programming lies in the fact that it demands the interaction of skills from both the teacher and the students, and requires the guarantee that the educator provides a cooperative and collaborative environment to develop in the student other skills such as the psycho-cognitive and teamwork among others, necessary for the approach of multidisciplinary problems (Roberts, 2011). Therefore, it is of utmost importance to have a methodological teaching and learning strategy that fosters a group work environment, in which both social and communication skills can be fostered, making the habit of helping, sharing and cooperating an inexcusable rule in the classroom. In this sense, several researchers visualise software development as a cooperative activity, where the main characteristic is teamwork (Lewandowski & Bourguin, 2006). In the age of technological development, it is not only the availability of information, knowledge and the means to communicate it that matters, but also the way in which it can be applied in real practices. The development of skills inherent to computer programming (creativity, self-efficacy, problem solving, reasoning, teamwork, etc.) develops in the student the multifaceted capacities that allow him to face interdisciplinary problems of different degrees of difficulty, which added to a good training, he/she acquires the experience of a good programmer. The cognitive stimulations inherent to problem solving (exploration, analysis and solution-seeking activities) (Kotovsky, 2003), stimulate a learning process, which favours mental development, foregrounds research skills, trains them in solution generation, and with it, students move towards the greater challenge of mastering computer programming skills. The problem-solving activity in programming needs in addition to the technical ability to synthesise or summarise a solution (López-Cruz et al., 2017). This skill along with teamwork can be developed through constant practice or training in programming.

The main contribution of our work is to share the results of the implementation of an experimental methodology based on agile software development using Scrum as a tool in a cooperative framework, focused on project-based learning, which provides an environment in which students are trained in the programming of computer applications. The methodology allowed students to become active protagonists in a social framework in which socially isolated or shy students are integrated into the group-class, thus improving academic performance and significantly reducing the dropout rate in the subject Numerical Programming in the degree course in Systems Analysis at the National University of Salta, Argentina.

2 CONCEPTUAL REFERENCE FRAMEWORK

2.1 COOPERATIVE LEARNING

One of the central axes of the methodology implemented was to foster a cooperative environment in which students can develop teamwork skills, specifically in the context of computer application programming.

Cooperation is working together to achieve shared goals. Within cooperative activities, individuals seek outcomes that are beneficial not only to themselves, but also to all other members of the group. Cooperative learning is the educational use of small groups in which students work together to maximise their own learning and that of others (Johnson et al., 1999).

Cooperative learning increases motivation and participation thanks to the interaction between teachers and students; enabling a continuous exchange of ideas, the development of communication and social skills, and the overcoming of negative attitudes. Students, feeling supported and confident, are able to consolidate their own learning style (García et al., 2019).

In summary, cooperative work is a classroom management strategy that favours the organisation of students into heterogeneous groups to carry out learning tasks and activities. This involves grouping students into small teams in order to promote the development of each member.

2.2 PAIR PROGRAMMING IN A COOPERATIVE ENVIRONMENT

An educational technique that has elements in common with cooperative learning is pair programming (Faja, 2014). In this form of cooperation, two programmers work together on a computer. At any particular time, one member of the team (the “driver”) may be working at the computer: either writing a program or modelling a design. The other member (the “navigator”) may be actively observing the work of the “driver”, helping to work out possible bugs, analysing alternative solutions, inquiring about the required knowledge, etc. The roles of “driver” and “navigator” are periodically exchanged between the two team members. Pair programming was originally popularised as part of the Extreme Programming software development methodology (Back & Andres, 2005). Research in the literature reports that pair programmers produce higher quality code in half the time compared to programmers working alone (Williams et al., 2002). The technique has also been found to be effective for programming learners, leading to improved student learning and satisfaction and reduced frustration in cognitive development (Mentz et al., 2008).

Cooperative learning uses methods similar to peer programming to help students learn about programming and problem-solving processes. However, in an environment that promotes cooperative learning, guides students through different levels of cooperation. Thus, for example, in the early stages, the full group can make a brainstorm to solve a problem. At a later stage, students could work in pairs to solve the problem and then compare their solutions with those developed by another couple of the same group. Later, other exercises provide students with the opportunity to work on problems for themselves, with the assistance of other group members if necessary. With this incremental approach, even more advantages are offered than strict peer programming. At first, everyone in the group are learning to address a programming task. Therefore, it is useful to have as many different points of view as possible. As their programming and problem-solving skills develop, students' progress to work in pairs. Finally, they have the opportunity to develop confidence solving individually (even with the support of the group).

2.3 PROBLEM-BASED COOPERATIVE LEARNING

Other cooperative learning experiences are designed to separate and highlight important aspects of programming and problem solving. This is the case of Problem Based Learning (PBL), whose learning results from the process of working on understanding and problem solving, where the problem is an important element in the learning process. The theoretical framework states that the characteristics of PBL are (Barrows, 1996): *Learning is learner-centred*; *cooperative learning occurs in small groups of students*; *teachers are facilitators or guides*; *problems are the tool for the development of computer problem solving skills*; *new information is acquired through self-directed learning*.

Thus, the PBL marks an absolutely different position from learning based on traditional teaching. In an PBL environment, students receive the description of a problem and they themselves identify what they need to know, study it and apply it to specify the solution. While, in traditional teaching, students receive the description of the problem, the teacher shows what they need to know, and induces them to determine a solution.

Cooperative problem-solving groups in an PBL environment are usually formed from two to four members.

Problem-based learning is well suited to engineering (as it is to medicine, where it is currently used) because it helps students develop the skills and confidence to explore, analyse and specify the appropriate solution to a given problem. The process of building models together in interpersonal, face-to-face interaction results in learning that is difficult to achieve otherwise.

2.4 AGILE COOPERATIVE FRAMEWORK

Promoting a cooperative environment based on an PBL is not everything, unless it is accompanied by artefacts that facilitate its development. In this context, it was decided to use agile techniques such as Scrum for the development of software applications, which allows the Numerical Methods Programming chair to train its students in programming, more than a mere practice of curricular content.

Scrum (Alaimo, 2013) is a framework that allows to strengthen and consolidate the relationships of the human team that interacts cooperatively in agile software development. It proposes a set of practices and artefacts that make it possible to transform a complex problem into simple activities of immediate and progressive resolution, generating at the same time a relational, interactive and cooperative context of constant inspection and adaptation so that those involved can create their own work style. In this way, an environment is created in which the team is provided with the necessary mechanisms to develop good working practices in a complex context.

In summary, the Scrum development framework seeks to focus the team on producing quality end-value, collaborating and cooperating in interactive communication to achieve continuous learning improvements.

3 METHODOLOGICAL STRATEGY

The characteristics and principles of the agile development of Scrum provide the central dynamics of pedagogical innovation. This not only provides the ideal mechanisms to simulate the work reality in the classroom, but also facilitates the combination of the best guidelines of each active methodology used.

From the experience, a set of activities were delineated, organised in stages that may differ or vary according to the type of students, the focus of the subject and the amount of programming used in the academic practices.

The stages and moments that make up the proposal are developed according to a temporal sequence that does not occur in a linear way, but by successive approximations. The stages are: *Diagnosis and enquiry*; *Design and planning*; *Implementation - Methodological action*; and *Evaluation - Reflection*.

3.1 DIAGNOSIS AND ENQUIRY

The teacher performs an analysis of educational reality to develop the pedagogical proposal. This implies knowing the students in depth for an adequate implementation of

the following stages. The main activity of this stage is the initial diagnostic evaluation to identify the students' psycho-cognitive state of the students. The evaluation is structured in three blocks. The first block includes conceptual exercises necessary to address the curricular contents of the subject.

The second block is oriented towards simple numerical problems. The third block is directed towards identifying enthusiastic programmers or students with particular skills related to programming. The information obtained in this block contributes to the formation of better criteria for pedagogical adaptation both in the teaching-learning process and in knowledge consolidation practices.

3.2 DESIGN AND PLANNING

Once the teacher defines the educational reality in which the learning and consolidation process of knowledge will be developed, the design of the contextualized methodological proposal is carried out. The development of an academic practice based on the agile training of programming requires adequate planning by teachers, based on the elements that make up the academic training process (objectives, contents, times, evaluation instruments, among others) , that is, the curriculum. To do this, it is necessary to contemplate the didactic purposes (objectives and competences) and the contents. In addition, more effort must be dedicated to the design of methodological strategies (groupings, activities, times, materials) and the evaluation system. It should be noted that the adaptation of content to the students of a cycle develops in parallel and in tune with the methodology and evaluation.

In turn, the contents, methodological aspects and evaluative modality depend on the objectives and competencies to be achieved. Among the tasks that are developed at this stage, the training of study groups, the design of activities for the consolidation of knowledge, and temporary planning, stands out.

3.3 IMPLEMENTATION - METHODOLOGICAL ACTION

Temporalization is fundamental in the design of practices and training in agile programming, but it is not easy to determine a priori how much time the different individual learning activities take. The implementation stage is the one that demands the most time, because the previously defined design is launched. Teachers fulfill the roles of Product Owner and Scrum Master, which are vital for programming training in a Scrum environment. The pedagogical action of the innovative proposal will succeed as students are instructed in particular roles. The enthusiastic programmer coordinates

the cooperative activities of the group and the teacher of the practice instructs the enthusiast to a balanced, participatory, multifaceted and varied actions, without implying a decrease in their academic performance. This includes peer tutoring, work dynamics, self -organization and autonomous decision. From this, a simulation of labor reality in the classroom is developed.

3.4 EVALUATION - REFLECTION

The evaluation design is closely related to the teaching methodology used. Depending on how the evaluation is considered when designing the process, it can be perceived as a judgment or as an occasion to learn. In the methodological proposal, the evaluation is seen as a systemic process. In it, the teacher reviews the pedagogical model that supports the training activity, and selects the strategies and tools that allow verifying the evolution and real progress achieved by the students. They must coexist group evaluations as a whole, while individual evaluations. This will allow the evaluation to be consistent in its entire dimension. The first decision to make teachers is the importance that will give to each of these two types of evaluation. Empirically, it is known that a better option is that the proposed activities are mostly evaluated (that is, more than 50%) with group evaluation. Therefore, individual evaluation may not be necessary or have less relevance, but never more important than the group. Above all, if active methodologies are implemented, supported by collaborative and cooperative learning with study groups. The importance of group and individual evaluation depends, obviously, the professional criteria of the teacher.

4 RESULTS

The new methodology was implemented during the year 2022 in the context of the Numerical Calculus subject taught in the second year of the degree curriculum. The course does not teach programming, but uses it as a resource in the academic practices for the implementation of numerical applications. The course develops the practice with eight problem guides. Each guide incorporates approximately 80% of problems related to algorithmic programming on computers, and 20% with concept consolidation exercises.

During the years 2014 and 2015, the first experiments were carried out in which only the agile Scrum methodology was used in the practical classes of the subject. The experimental results showed a positive impact on the students compared to previous years, when traditional lecture-based teaching was used (Barberis & Del Moral, 2016). The core activities of the agile methodology were delineated, which resulted in higher

motivation for students' learning. Adaptive mechanisms in teaching were specified and a pedagogical evaluation of the strategy was conducted in 2015, which included improvements over the experiment conducted in 2014 (Del Moral & Barberis, 2016). The results of the methodological evaluation were very encouraging. The experiment was repeated during 2016, and a variant of the previous year was implemented in 2017. The conclusive results at the end of 2016 and 2017 were that the teaching staff put a lot of effort into implementing the methodology, as a consequence of weak training in agile development with Scrum. Even so, the academic performance rate increased from 55% to an average of 85% of students who passed the course. The dropout rate decreased to an average of 5%. In 2018 and 2019, the traditional lecture-based teaching system was resumed. During these years, the teaching staff furthered their training in agile development with Scrum, and the current pedagogical proposal was outlined, which will be put into practice from 2022 onwards. In 2020 and 2021 the methodology was not applied as a consequence of the pandemic that affected the whole world.

In the year 2022, the resulting statistics were: 89.87% of the students enrolled regularised the subject, while 6.33% were left free, and 3.8% dropped out of the course. These statistics imply a significant improvement in academic performance compared to previous years (see table 1).

Table 1. Annual records from 2011 to 2022 of students enrolled in Numerical Calculus, and the numbers of regularised, free and dropouts.

Year	C. Registered	C. Regularised	C. Leisure	C. Desertion
2011	66	31	9	26
2012	58	33	9	16
2013	38	18	7	13
2014	74	56	14	4
2015	85	71	9	5
2016	78	67	7	4
2017	82	37	23	22
2018	67	38	9	20
2019	70	44	12	14
2022	79	71	5	3

From a reflective analysis of the diagnosis and evaluations of the process, it is possible to note a reserved and uninvolved attitude on the part of the students in the initial stages of the course. The assignment of roles, division of tasks and organisation were the factors with the greatest impact on group work. The appropriate management of the students' psychological behaviour, the implementation of peer tutoring, and practices

centred on cooperative dynamics, allowed the development of new skills, not only as a programmer but also as a human being. Thus, an environment was generated in which greater dialogue between peers, trust, a reflective, active and participatory attitude was fostered. All these actions enabled the department to quickly detect cognitive and social problems and to implement corrective and adaptive actions.

5 CONCLUSIONS

The implementation of a methodology as described necessarily requires the prior training of the teaching body in the agile development of software with Scrum. You must have clear knowledge about problems-based learning and project-based strategies. As well as dominate the pedagogical activities of collaborative and cooperative learning. The importance of the previous training of teachers lies in the need to have teaching profiles that react rapidly in the instrumentation of pedagogical solutions in a non -planned situation. The speed with which the methodological dynamics are developed, and the limited times for the curricular development of the subject, make that, in some situations, they cannot be detected in the mediateness psycho-cultural positions of students who prevent them from developing social and collaborative skills In teamwork. Passing these scenarios implies a greater risk in the abandonment of the course by the protagonist. Major inconveniences of marginality or integration can be raised to a study group, which require the advice of a psycho-pedagogical professional. Capture the attention of students, generate in them the sense of learning in learning, create spaces for reflection and discussion of knowledge, plan activities that induce the development of critical thinking, are just some of the tasks demanded by professionalized teachers, not only in the issues of the subject curriculum, but also, in the pedagogical. The stage of diagnosis and inquiry is of the utmost importance for the development of the methodological proposal. This provides sufficient information to plan the strategy that increases educational quality. The adaptive and contextualized characteristics of the design stage ensure the success of the following stages. The reaction rate and adaptive changes in the march of the implementation stage, favor the academic development of labor reality in the aulic context.

REFERENCES

Alaimo, M. (2013). *Proyectos Ágiles con Scrum* (E. Kleer, Ed. 1 ed.). Kleer Agile Coaching & Training. http://www.kleer.la/es/publicamos/scrum

Back, K., & Andres, C. (2005). *Extreme Programming Explained: embrace change* (2ª ed.). Addison-Wesley Professional.

Barberis, A. R., & Del Moral, L. E. (2016). Scrum como Herramienta Metodológica en el Entrenamiento Cooperativo de la Programación: De la Teoría a la Práctica. XI Congreso de Tecnología en Educación y Educación en Tecnología 2016 (TE&ET 2016), Universidad de Morón, Argentina.

Barrows, H. S. (1996, 1996/12/01). Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning, 1996*(68), 3-12. https://doi.org/10.1002/tl.37219966804

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads, 10*(2), 30–36. https://doi.org/10.1145/3324888

Del Moral, L. E., & Barberis, A. R. (2016). Evaluación de una Propuesta Metodológica para el Entrenamiento de la Programación. 4to Congreso Nacional de Ingeniería en Informática / Sistema de Información (CoNaIISI 2016), Universidad Católica de Salta, Argentina.

Faja, S. (2014). Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses. *Information Systems Education Journal (ISEDJ), 12*(6), 36-45. https://files.eric.ed.gov/fulltext/EJ1140923.pdf

García, R., Traver, J. A., & Candela, I. (2019). *Aprendizaje cooperativo: Fundamentos, características y técnicas* (E. CCS, Ed. 2 ed.). ICCE (Instituto Calasanz de Ciencias de la Educación).

Johnson, D. W., Johnson, R. T., & Holubec, E. J. (1999). *El aprendizaje cooperativo en el aula*. Paidos.

Kotovsky, K. (2003). Problem Solving – Large/Small, Hard/Easy, Conscious/Nonconscious, Problem-Space/Problem-Solver: The Issue of Dichotomization. In J. E. Davidson & R. J. Sternberg (Eds.), *The Psychology of Problem Solving* (pp. 373-384). Cambridge University Press. https://doi.org/10.1017/CBO9780511615771.013

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14-18. https://doi.org/10.1145/1151954.1067453

Lewandowski, A., & Bourguin, G. (2006). A New Framework for the Support of Software Development Cooperative Activities.

López-Cruz, O., Mora, A. L., Sandoval-Parra, M., & Espejo-Gavilán, D. L. (2017). Teaching Computer Programing as Knowledge Transfer: Some Impacts on Software Engineering Productivity. Trends and Applications in Software Engineering, Cham.

Mentz, E., van der Walt, J. L., & Goosen, L. (2008, 2008/12/01). The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education, 18*(4), 247-260. https://doi.org/10.1080/08993400802461396

Roberts, F. S. (2011, 2011/07/01). The Challenges of Multidisciplinary Education in Computer Science. *Journal of Computer Science and Technology, 26*(4), 636-642. https://doi.org/10.1007/s11390-011-1164-1

Sarpong, K. A.-m., Arthur, J. K., & Owusu Amoako, P. Y. (2013). Causes of Failure of Students in Computer Programming Courses: The Teacher - Learner Perspective. *International Journal of Computer Applications (IJCA), 77*(12), 27-32. https://doi.org/10.5120/13448-1311

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002, 2002/09/01). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education, 12*(3), 197-212. https://doi.org/10.1076/csed.12.3.197.8618

SOBRE OS ORGANIZADORES

Jorge Rodrigues é economista. Licenciado, mestre e doutor em Gestão (ISCTE-IUL), com Agregação (UEuropeia). Mestre e pós-doutorado em Sociologia – ramo sociologia económica das organizações (FCSH NOVA). Professor coordenador com agregação no ISCAL – *Lisbon Accounting and Business School* / Instituto Politécnico de Lisboa, Portugal. Exerceu funções de direção em gestão (planeamento, marketing, comercial, finanças) no setor privado, público e cooperativo. Contabilista certificado. É investigador integrado no Instituto Jurídico Portucalense. Ensina e publica nas áreas de empresa familiar e família empresária, estratégia e finanças empresariais, gestão global, governabilidade organizacional, marketing, planeamento e controlo de gestão, responsabilidade social e ética das organizações.
https://orcid.org/0000-0001-7904-0061

Maria Amélia Marques, Doutora em Sociologia Económica das Organizações (ISEG/ULisboa), Mestre em Sistemas sócio-organizacionais da atividade económica - Sociologia da Empresa (ISEG/ULisboa), Licenciada (FPCE/UCoimbra), Professora Coordenadora no Departamento de Comportamento Organizacional e Gestão de Recursos Humanos (DCOGRH) da Escola Superior de Ciências Empresariais, do Instituto Politécnico de Setúbal (IPS/ESCE), Portugal. Membro efetivo do CICE/IPS – Centro Interdisciplinar em Ciências Empresariais da ESCE/IPS. Membro e Chairman (desde 2019 da ISO-TC260 HRM Portugal. Tem várias publicações sobre a problemática da gestão de recursos humanos, a conciliação da vida pessoal, familiar e profissional, os novos modelos de organização do trabalho, as motivações e expectativas dos estudantes Erasmus e a configuração e dinâmica das empresas familiares. Pertence a vários grupos de trabalho nas suas áreas de interesses.
https://orcid.org/0000-0002-7196-3838

ÍNDICE REMISSIVO

A

B

C

E

F

G

H

I

J

K

L

M

O

P

Q

R

S

T

U

V