

A Survey on the Effects of Working Conditions on Programming Efficiency in an Educational Environment

Mariia Charikova, Ananga Thapaliya, Susanna Gimaeva, Alexandr Grichshenko, Selina Varouqa, Luiz Jonatã Pires de Araújo(✉), and Giancarlo Succi

Innopolis University, Innopolis 420500, Russia
l.araujo@innopolis.university

Abstract. A recurrent concern of instructors and managers in learning and industrial sectors is how to organise the working environment to increase the productivity in tasks such as programming and software testing. Evidence of the increasing interest from different domains in this topic is the growing amount of research that has been published on physical factors (e.g., product, personnel, project and process), programming tasks (e.g., tests, questionnaires, programming, testing and debugging), and assessment methods (e.g., time, software metrics and academic grading). The objective of this paper is to survey the literature and to enable one to gain valuable insights into the relevance of physical factors to improve programming efficiency, especially in a learning environment. This study also makes recommendations on the techniques that can provide further experience for learners before joining the industrial sector. Finally, this survey suggests research directions, including an analysis of the correlation between physical factors and measurable productivity.

Keywords: Programming efficiency · Academic environment · Physical conditions · Productivity factors · Evaluation of productivity

1 Introduction

The term "working conditions" is often used to refer to the environment and surrounding circumstances in which employees or students work. Such conditions include a wide range of aspects, from physical components of the workspace or study space to the methodologies that the institution implements, which can influence the developer's behaviour and productivity. Working conditions can be divided into product, personnel, project and process related circumstances [1]. Examples of these factors include elements in the work or the study space such as light, noise, music, company culture, job satisfaction, satisfaction of the study process, relationship with colleagues or other students, technical training, different coding practices such as pair programming, and the complexity of a task.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 289–300, 2020.
https://doi.org/10.1007/978-3-030-58817-5_22

This topic plays an essential part in educational processes for students enrolled in university-level courses of programming and industrial companies providing internal training. A careful study of the working conditions' impact on programming efficiency can increase the effectiveness of learning activities, leading to higher standards of education and skilful specialists. This paper surveys the existing literature concerning the effects of working or learning conditions on developers' efficiency in the educational process, focusing on the main investigated physical factors, programming tasks and assessment methods. Moreover, by analysing the trends and gaps in the literature, this study enables better decision making regarding the surroundings for students and practitioners in the academia and the industrial sector.

The methodology for collecting relevant literature consists of data acquisition from scientific journals and conferences using search keywords and sentences including "working conditions", "programming efficiency", "physical factors". This was followed by an analysis of the relevance of individual studies, resulting in a summary covering papers published between 1984 and 2018 addressing different working and academic conditions' effects.

The remaining of this paper is organised as follows. Section 2 outlines the existing research on physical factors, typical programming tasks and methods for performance assessment. It also categorises the current body of study and introduces a taxonomy for such factors. In Sect. 3, we discuss the trends and the gaps in the literature and this analysis constitutes a useful resource for researchers and software project managers willing to improve the work environment in their companies. Section 4 presents overall observations and suggestions for further research.

2 State-of-the-Art

2.1 On the Physical Factors

Physical factors can be divided into four different categories according to their characteristics: product, personnel, project and process factors. Product factors refer to software qualities that are established throughout the stages of software development. They concern to artefacts (e.g. software code, specifications, documents) and system characteristics (e.g. complexity, scalability, instability) [1]. Product factors also include the application type, programming language, database management system, development model (agile/waterfall), hardware platform, graphical user interface and operating system [2]. An example of a study on product factors is presented by Delorey et al. [3], which explained how the choice of programming language affects productivity and the outcome. In another study, Mohapatra et al. [4] showed that product factors such as code reusability and convenient document management system could significantly impact productivity while maintaining high quality.

Personnel factors comprise developers' attributes that impact the productivity of the whole team during the development process. This type of factor is often related to the expertise level of individuals and their roles in the project.

For example, the study by Trendowicz et al. [1] investigated how skills of development team members (e.g. analysts, architects, engineers, project managers), programming suppliers, clients, maintainers and subcontractors affect the overall performance. It is possible to see that age and gender influence the quality of the code and the productivity of the development team [5]. Moreover, personnel factors like pair programming [6], the role of a participant in the project [7] and compatible soft skills [8] also affect the programmers' efficiency. A common limitation of the aforementioned studies is that they ignore the relationship between developers in a team, which is an important personnel factor.

Project factors include various aspects of setup and maintenance, such as the high turnover rate of the staff. Examples of project factors that impact developers' productivity are customer participation, development environment adequacy, staff availability, use standards, methods, tools, software's logical complexity, requirements volatility, quality requirements, efficiency requirements, installation requirements, staff's soft skills and domain knowledge [2]. Raziq et al. [9] also discussed how satisfaction and relationship with colleagues or classmates during a project could affect productivity.

Process factors are related to the characteristics of software processes including techniques, apparatuses and innovations that are implemented during the product life-cycle. These factors also include quality of evaluations, consistency of examinations, quality of the tools, nature of the executive procedures, or degree of customer cooperation [1]. As described by Sudhakar [10], different software engineering tools and techniques such as rapid prototyping, software testing, e-mail, document management systems, programming languages, and configuration management systems are examples of process factors that affect efficiency.

2.2 On the Programming Tasks

In this section, we consider tasks which can be used for evaluation of the programming efficiency in an educational context. They can be grouped in four macro-categories: tests, questionnaires and exams; implementing data structures and algorithms; working on a project; testing and debugging.

This survey focuses on programming tasks in an educational process, which are mostly assessed through exams and tests. For example, Hu et al. [11] investigated the impact of such evaluation techniques on productivity. There were no special assignments to estimate students' productivity; instead, they studied in different physical conditions and took general exams and tests. Other studies focused on the task of learning an extracurricular topic, with outcomes being evaluated according to questionnaires for participants [12]. Wiedenbeck et al. [13] proposed a different approach for assessing students' progress by providing them with simple tasks for approximately 1.5 min. Such a method stands as an alternative method for evaluating the learning outcome since short tasks can assess knowledge but not ingenuity nor the ability to produce a creative solution quickly.

Another type of activity which is important for computer science students is the implementation of algorithms and data structures. Nanz and Furia [7], and Bergersen et al. [14] mention simple tasks for algorithms implementation. The development of basic algorithms allows researchers to consider the minimum number of internal human factors and focus on the implementation. Another approach outlined by Darcy and Ma [15] is to provide a coding task as an exercise. Different difficulty levels of the proposed tasks help researchers to evaluate the influence of physical factors on the programmer's productivity. Not only standard algorithms can be used in the evaluation as indicated by Sprigle and Schaefer [16], who used tasks in LOGO programming language for drawing.

Although this survey focuses on the academic environment, it is noteworthy to mention studies that focus on programming tasks in industrial projects. For example, Raziq and Maulabakhsh [9] used different types of tasks from those commonly used in academia. The advantage of using industrial tasks is that they can be studied over a long time, as indicated in [3] and [10]. For instance, in [2], the employees of a large company were observed during the software project, and researchers investigated how different factors affect productivity. It is worth emphasizing that, unlike academic tasks, industrial ones are more narrowly focused. For example, Sullivan and Umashi Bers [5] discussed software development and robotics tasks. There are many projects in industry that are dependent on the specific domain, which complicates the process of evaluating the productivity of programmers.

Testing and debugging are here considered separately, although both are part of industrial and academic projects. Chaparro et al. [6] mentioned different tasks which are related to debugging, re-factoring and program comprehension. When evaluating productivity, the debugging skill can be critical since small errors or problems with external services can cause the inability to present a working code. Khan et al. [17] organized multiple-choice questionnaires to measure a programmer's performance on recognizing debugging problems. Such an approach is one of the simplest ways to give objective feedback on the productivity of testers and programmers. A more comprehensive method is presented in [18], where the project manager assigned to programmers a set of modules for testing in model-based development. Khan et al. [19] proposed a special system of providing tasks for evaluating the productivity of programmers. The training session consisted of watching a video clip, followed by a series of debugging questions that had to be answered within a fixed time. After completing the training session, participants continued with the actual test, which consisted of two cycles of watching clips and debugging test.

2.3 Constraints and Performance Assessment of Programming Tasks

This section examines two important elements related to programming tasks: the constraints and evaluation methods for assessing programming performance. Most of the literature focuses on combining time constraint and completeness evaluation, using software development metrics and academic grading.

One of the simplest methods for evaluating programming performance which has been employed is to assess whether the task was finished in its entirety. In this type of evaluation, the time to solve a task is used as a performance metric. The literature contains several examples of research in this direction [3,6,7,14,20]. However, most of the research addresses programming in the workplace with limited research being done in an educational context [21].

Using software metrics is a practical way to evaluate not only students' performance but also the quality of the generated code. Some metrics have been commonly used by the industry sector and can offer a good benchmark. For example, Mohapatra [22] uses function points and a person's monthly effort to estimate individual productivity. Sudhakar et al. [10] employed a more comprehensive set of metrics to evaluate programming performance: use function, use case, object and feature points. The literature also contains examples of not conventional software metrics for programming quality assessment, including the number of lines of code (LOC) [7,10] and memory usage [7]. Darcy and Ma [15] extended LOC by considering blank, comment and executable lines. Vihavainen [23] employed an alternative measurement by counting indentation errors. It is noteworthy that LOC cannot be the only one measure because it partially depends on the choice of programming language.

Several studies have used academic grading for assessing programming quality [11,13,17]. For example, Raschs et al. [12] surveyed developers to compare his or her overall performance with peers at the same level and speciality area. A nine-point scale was used, with labels ranging from "very inferior, the very worst" to "very superior, the very best". Mohapatra [22] focused on evaluating team productivity and demonstrated that there exists a strong relationship between programming quality and following a systematic project management plan. It is also important to consider the fact of possible subjectivity of people who put such evaluation, which affects academic grading as a metric.

3 Discussion

As evident in the analyzed studies, working conditions play an essential role in software developers' performance. For example, Malik et al. [24] conducted a study to demonstrate that the physical environment has a significant positive effect on employees' productivity. The study used multiple regression analysis to emphasize that the performance mostly depends on physical working conditions, training and development, and communication practices. In another study, Srivastava [25] demonstrated that there is a dependence between the working environment and the employees' behaviour by examining the resultant organizational effectiveness and performance.

Several studies have also confirmed that environmental conditions affect the emotional and psychological health of employees. A job usually occupies a considerable portion of a person's day, which makes relevant any discussions about the impact of a job on the emotional health [26]. According to McCraty et al. [27], there is a strong relationship between emotional and physical health

and the work-related indicators of workplace satisfaction as well as the perception of individual contribution. The study enabled an overall reduction of the blood pressure, improved emotional health and workplace-related measures of the employees by enhancing the work environment.

Working conditions have a significant effect not only on the emotional health of employees but also on their physical health [28]. One of the essential components of the workspace is its shape or design. Open workspaces might lead to different work collaboration and efficiency compared to small closed individual office design. The design of the workspace may influence the surrounding noise and have an impact on the employee's performance. One important aspect of choosing the office design is the type of work that is being done, whether it requires collaboration and teamwork, or it is an individual task. Danielsson and Bodin [29] showed that the lowest health status for employees was found in medium-sized and small open-plan offices, while the best health rates were found among employees in cell and flex offices.

A study by Estácio et al. [30] revealed that the work environment could also be affected by the style of work: individual, team-based, or pair-based, with collaborative practices outperforming solo programming with regard to acquiring programming skills. Furthermore, it was shown that enhancing the learning outcomes, improving students' satisfaction and reducing frustration in assignments could be achieved with pair programming [31] and more interactive learning environments [32]. Flor and Hutchins [33] noted that the "solutions space", i.e. all possible solutions for a problem, is maximized when two programmers concentrate on a task. There are other factors which may influence the effectiveness of collaborative learning. A study performed by Chaparro et al. [6] on postgraduate students reported that close skill level and personality compatibility are among the main elements that positively affect the success of pair programming. Another finding of the research is that collaborative programming can boost or undermine the performance depending on the type of task. According to the study, software design and implementation were demonstrated to be more efficient, while debugging was not significantly faster nor easier with this practice.

There has been a considerable increase in the number of publications on working conditions in the last decades. Appendix in Sect. 4 represents the summary of considered literature, namely physical factors, programming tasks and performance measure chosen for each study. Figure 1 shows an increasing amount of research on personnel physical factors and programmers' productivity. This growth is more noticeable on the number of studies about project and process factors such as satisfaction and relationship with colleagues. This interest is caused, among other factors, by an increasing concern with creating a comfortable atmosphere in offices and educational centres to improve the productivity of employees and students.

Regarding the number of publications on job-specific problems, there has been a considerable increase over the years (see Fig. 2a). This can be explained in part by the acknowledged relationship between profit and employees'

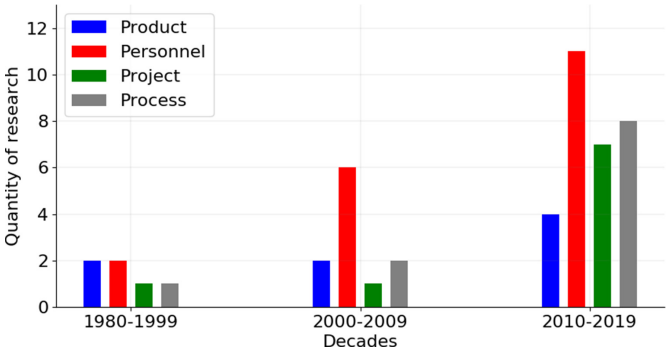
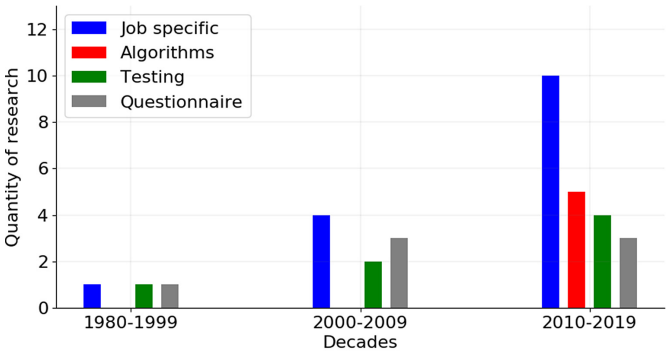
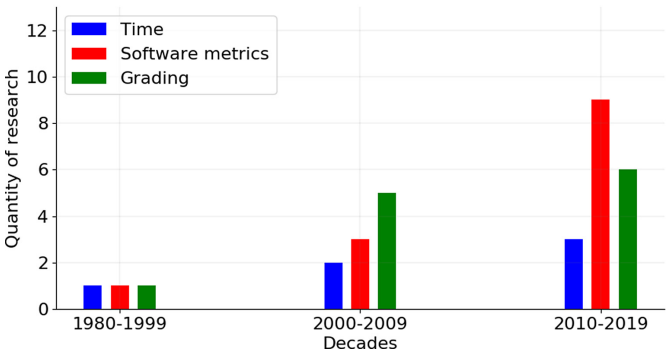


Fig. 1. Research of physical factors impacting performance in the last decades.



Research of programming tasks impacting performance.



Research of measures evaluating performance.

Fig. 2. Research of programming tasks impacting performance.

productivity. Moreover, the number of papers considering debugging and testing tasks, as well as exams and questionnaires has also increased. Interestingly, there has been no significant amount of research on programming tasks related to algorithms, to the best of the authors' knowledge. It is also possible to notice that most of the research has focused on performance assessment using software metrics such as source lines of code (SLOC) and software function points (SFP), as shown in Fig. 2b.

It is noteworthy that most of the conducted research has focused on the physical factors in an isolated manner. In other words, the effects of the synergy between different factors are often ignored. Future research should focus simultaneously on several factors as their interaction can impact the results of the software project. Moreover, the majority of studies has used software metrics for assessing programming productivity addressing the industrial environment [7,10,22]. Such an approach might bring additional complexities and a steeper learning curve to students compared to the usual academic grading evaluation. However, the use of software metrics in a learning environment can also provide practical experience which might be useful for future industrial jobs [11,13].

4 Conclusion and Future Research

The research on working conditions is relevant not only for corporations aiming to maximise the productivity of their employees but also for educational institutions interested in improving their curricula and improving students' overall qualification before they enter the job market. A deeper understanding of concepts like physical factors, programming tasks and methods of evaluating performance can enable better decision making. The body of research is, therefore, a useful resource for professors and university managers to improve the academic environment.

This survey is meant to serve as a guideline to future researchers, software project managers and specifically university faculty members. Moreover, a taxonomy for physical and programming factors has been introduced to assist the stakeholders to quickly recognise and solve potential issues that might affect the performance. The review of the extensive literature on programming metrics can be applied to evaluate the existing model of performance in a company or the educational institution, identifying the need for improvements in their processes.

One possible research direction is the study on the potential synergy between different working conditions, and how their combination affects the performance. Such an approach could provide more comprehensive insight into the operation and management of the entire project or the curriculum. Any conclusions that might emerge from the integration of industry and academia will provide a better understanding of factors impacting performance and consequently benefit both fields.

Appendix

References	Physical factors	Programming tasks	Performance measure
Thadhani (1984)	Programming experience, task complexity, a fraction of time spent directly working on the task	Industrial project	Time (constraint) and lines of code
Rasch and Tosi (1992)	Goal difficulty and clarity, high-achievement needs, self-esteem	Job-specific tasks	Self-evaluation, relative performance
Maxwell and Forselius (2000)	Application type, programming language, database management system, development model, hardware platform	Regular development work	Customers' participation, development environment adequacy, staff availability, standards use, methods use, tools use, software's logical complexity, requirements volatility
Hu and Kuh (2003)	College environment, the estimated effort put into study	Usual tests and exams in university	System of points of the university
Wiedenbeck et al. (2004)	Programming experience, self-efficacy, ability to construct mental models	Time specific simple and complex tasks	Course grading system
Chaparro et al. (2005)	Pair Programming	Debugging, refactoring and program comprehension	Amount of finished exercises
Darcy and Ma (2005)	GPA and age	Individual coding task (Simple payroll system)	Specification conformance, code structure complexity
Del (2007)	Programming language	Job specific tasks (working on project)	Speed
Khan et al. (2007)	Programmers' mood and emotions	Various debugging exercises	Completion of the debugging test
Trendowicz and Mnch (2009)	Product, process, personnel and project factors	Industrial level tasks	Overall development productivity
Paiva et al. (2010)	Commitment, communication, benefits, consistent requirements, experience, motivation, location, project and team size	Job specific tasks	Questionnaire with developers
Bergersen et al. (2011)	Individual expertise and skills	Simple programming tasks	Time and quality
Khan (2011)	Programmer's current mood and emotions	Actual test, which consisted of two cycles of the movie and debugging test	Completion of the debugging test

References	Physical factors	Programming tasks	Performance measure
Mohapatra (2011)	Application complexity, client support, availability of modules and testing tools, document management system and computational performance	Software project development	Following systematic steps as laid down in the project management plan
Mohapatra (2011)	Effective training, availability of skilled manpower in the technology domain, well-documented procedure	Software development, maintenance and testing industrial projects	Function points, number of defects
Sudhakar et al. (2012)	Size of the team, computing infrastructure and software engineering tools	Software project	Lines of code
Watson et al. (2013)	Logs of compilation errors and code snapshots across the semester	Coding tasks in an introductory programming course	Overall coursework mark
Kamma and Jalote (2013)	Techniques used by programmers to organise their work	Set of modules for testing in model-based development	Actual effort spent by the programmer in a task and the software size of that task
Vihavainen (2013)	Students' programming behaviour (eagerness to start new exercises, the time required to complete an exercise)	Regular assignments in an introductory programming course	Course grade
Dagiene et al. (2014)	Prior coding experience	Algorithmic thinking contest tasks	Grades
Mohapatra and Sreejesh (2014)	Application complexity, training, client support, reusing existing code and quality of document management system	Software projects	Cost estimation model
Nanz and Furia (2015)	Roles of participants	Algorithm Implementation and other 745 simple tasks	Lines of code, memory usage
Raziq and Maulabakhsh (2015)	Relationship with team members	Job specific tasks	Job satisfaction
Sullivan and Umashi Bers (2017)	Age and Gender	IT and Robotic tasks	Function points, object points, use case points and feature points
Busechian et al. (2018)	Pair Programming	Job specific tasks	Topological brain maps
Wagner and Ruhe (2018)	Organisational cultural factors, Team Culture factors, experience and work environment factors	Industrial level tasks	Effort per SLOC (source lines of code), function points

References

1. Trendowicz, A., Münch, J.: Factors influencing software development productivity - state-of-the-art and industrial experiences. Adv. Comput. **77**, 185–241 (2009)
2. Maxwell, K.D., Forselius, P.: Benchmarking software development productivity. IEEE Softw. **17**(1), 80–88 (2000)
3. Delorey, D.P., Knutson, C.D., Chun, S.: Do programming languages affect productivity? A case study using data from open source projects. In: First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS 2007, p. 4 (2007)
4. Mohapatra, S., Sreejesh, S.: Model for improving productivity without impacting quality of deliverables in it projects. Int. J. Inf. Technol. Project Manage. (IJITPM) **5**(2), 14–29 (2014)
5. Sullivan, A., Bers, M.U.: Girls, boys, and bots: gender differences in young children's performance on robotics and programming tasks. J. Inf. Technol. Educ. Innov. Pract. **15**, 145–165 (2017)
6. Chaparro, E.A., Yuksel, A., Romero, P., Bryant, S.: Factors affecting the perceived effectiveness of pair programming in higher education. In: 17th Workshop of the Psychology of Programming Interest Group, June 2005, pp. 5–18 (2005)
7. Nanz, S., Furia, C.A.: A comparative study of programming languages in rosetta code. In: Proceedings - International Conference on Software Engineering, vol. 1, pp. 778–788 (2015)
8. Fernández-Sanz, L., Villalba, M.T., Medina, J.A., Misra, S.: A study on the key soft skills for successful participation of students in multinational engineering education. Int. J. Eng. Educ. **33**(6), 2061–2070 (2017)
9. Raziq, A., Maulabakhsh, R.: Impact of working environment on job satisfaction. Procedia Econ. Financ. **23**, 717–725 (2015)
10. Sudhakar, P., Farooq, A., Patnaik, S.: Measuring productivity of software development teams. Serb. J. Manage. **7**(1), 65–75 (2012)
11. Hu, S., Kuh, G.D.: Maximizing what students get out of college: testing a learning productivity model. J. Coll. Student Dev. **44**(2), 185–203 (2003)
12. Rasch, R.H., Tosi, H.L.: Factors affecting software developers' performance: an integrated approach. MIS Q. **16**, 395–413 (1992)
13. Wiedenbeck, S., Labelle, D., Kain, V.N.: Factors affecting course outcomes in introductory programming. In: 16th Workshop of the Psychology of Programming Interest Group, pp. 97–110 (2004)
14. Bergersen, G.R., Hannay, J.E., Sjoberg, D.I., Dyba, T., Karahasanovic, A.: Inferring skill from tests of programming performance: combining time and quality. In: 2011 International Symposium on Empirical Software Engineering and Measurement, April 2014, pp. 305–314 (2011)
15. Darcy, D.P., Ma, M.: Exploring individual characteristics and programming performance: implications for programmer selection. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, p. 314a. IEEE (2005)
16. Sprigle, J.E., Schaefer, L.: Age, gender, and spatial knowledge influences on preschoolers' computer programming ability. Early Child Dev. Care **14**(3–4), 243–250 (1984)
17. Khan, I.A., Hierons, R.M., Brinkman, W.P.: Moods and programmers' performance. In: PPIG, p. 2 (2007)
18. Kamma, D., Jalote, P.: Effect of task processes on programmer productivity in model-based testing. In Proceedings of the 6th India Software Engineering Conference, pp. 23–28. ACM (2013)

19. Khan, I.A., Brinkman, W., Hierons, R.M.: Do moods affect programmers' debug performance? Cogn. Technol. Work **13**(4), 245–258 (2011). https://doi.org/10.1007/s10111-010-0164-1
20. Thadhani, A.J.: Factors affecting programmer productivity during application development. IBM Syst. J. **23**(1), 19–35 (1984)
21. Watson, C., Li, F.W.B., Godwin, J.L.: Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies, pp. 319–323. IEEE (2013)
22. Mohapatra, S.: Maximising productivity by controlling influencing factors in commercial software development. Int. J. Inf. Commun. Technol. **3**(2), 160–179 (2011)
23. Vihavainen, A.: Predicting students' performance in an introductory programming course using data from students' own programming process. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies, pp. 498–499. IEEE (2013)
24. Malik, M.I., Ahmad, A., Gomez, S.F., Ali, M.: A study of work environment and employees' performance in Pakistan. Afr. J. Bus. Manage. **5**(34), 13227 (2011)
25. Srivastava, A.K.: Effect of perceived work environment on employees' job behaviour and organizational effectiveness. J. Indian Acad. Appl. Psychol. **34**(1), 47–55 (2008)
26. Robone, S., Jones, A.M., Rice, N.: Contractual conditions, working conditions and their impact on health and well-being. Eur. J. Health Econ. **12**(5), 429–444 (2011). https://doi.org/10.1007/s10198-010-0256-0
27. McCraty, R., Atkinson, M., Tomasino, D.: Impact of a workplace stress reduction program on blood pressure and emotional health in hypertensive employees. J. Altern. Complement. Med. **9**(3), 355–369 (2003)
28. Arnold, K.A., Dupré, K.E.: Perceived organizational support, employee health and emotions. Int. J. Workplace Health Manage. **5**(2), 139–152 (2012)
29. Danielsson, C.B., Bodin, L.: Office type in relation to health, well-being, and job satisfaction among employees. Environ. Behav. **40**(5), 636–668 (2008)
30. Estácio, B., et al.: Evaluating collaborative practices in acquiring programming skills: findings of a controlled experiment. In: 2015 29th Brazilian Symposium on Software Engineering, pp. 150–159. IEEE (2015)
31. Nagappan, N., et al.: Improving the cs1 experience with pair programming. ACM SIGCSE Bull. **35**(1), 359–362 (2003)
32. Oyesiku, D., Adewumi, A., Misra, S., Ahuja, R., Damasevicius, R., Maskeliunas, R.: An educational math game for high school students in Sub-Saharan Africa. In: Florez, H., Diaz, C., Chavarriaga, J. (eds.) ICAI 2018. CCIS, vol. 942, pp. 228–238. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01535-0_17
33. Flor, N.V., Hutchins, E.L.: A case study of team programming during perfective software maintenance. In: Empirical Studies of Programmers: Fourth Workshop, p. 36. Intellect Books (1991)