Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 00 (2023) 1–13
Procedia Computer Science

Automated Assessment in Programming Education: A Bibliometric Analysis of the Literature Over the Last Decade

José Carlos Paiva[a,*], Álvaro Figueira[a], José Paulo Leal[a]

[a]*CRACS - INESC TEC & DCC - FCUP, Porto, Portugal*

Abstract

Learning to program requires a lot of practice, creating room for discovery, trial and error, debugging, and concept mapping. The learner must walk this long road herself, supported by appropriate and timely feedback. Providing individualized, accurate, rich, and timely feedback in programming exercises to meet these requirements is not a humanly feasible task. Therefore, the early and steadily growing interest of Computer Science educators in automated assessment of programming exercises is not surprising. The research area of automated assessment has existed for more than half a century, and interest in it continues to grow as it adapts to new developments in Computer Science and the resulting changes in educational requirements. Over the years, several systematic literature reviews have been published reporting advances in tools and techniques. However, there is not yet a major bibliometric study that examines the relationships and influence of publications, authors, and journals to make these research trends visible. This paper presents a bibliometric study of automated assessment of programming exercises, including a descriptive analysis using various bibliometric measures and data visualizations. The data were collected from the Web of Science Core Collection using a keyword-based search and some additional filters. The results obtained allow us to identify the most influential authors and their affiliations, monitor the evolution of publications and citations, establish relationships between emerging themes in publications, discover research trends, and more. This paper provides a deeper knowledge of the literature and facilitates future researchers to start research in this field.

Keywords: automated assessment, programming education, programming exercises, computer science, bibliometrics, data visualizations

1. Introduction

Practice is the key to learning how to program. Practical programming skills can only be acquired through extensive and varied experience in solving programming challenges. Such an experience should provide the learner with room for discovery, trial and error, debugging, and concept generation, while supporting the learner with individualized, accurate, rich, and rapid feedback to unlock their progress. Obviously, feedback that meets these requirements cannot be guaranteed by a human instructor.

Automated assessment tools for programming tasks have emerged as a solution to this problem. They have been part of Computer Science (CS) education almost since learners began asking about software development, and their

*Corresponding author

Email addresses: up201200272@fc.up.pt (José Carlos Paiva), arfiguei@fc.up.pt (Álvaro Figueira), zp@dcc.fc.up.pt (José Paulo Leal) value is already unanimously recognized by practitioners. Nevertheless, interest in assessing various program properties (e.g., quality, behavior, readability, and security), in adapting feedback, and in developing better and more powerful tools has not waned since then [1].

Over the years, several studies have been conducted to summarize the new developments in this field, at their respective times [2, 3, 4, 5, 6, 1]. All of these studies focus either on comparing the features of the tools [6] or on exploring the methods and techniques for some facets of the automated assessment tools [2, 3, 4, 5] or both [1]. These studies have been very important in understanding what has already been done in this area and what resources are available for reuse. However, to the best of authors' knowledge, no bibliometric study has been conducted to examine the quantitative aspects of scientific publications and their relationships [7]. Such a study would contribute to a deeper knowledge of the literature and facilitate future researchers' entry into research in this field. For example, it can provide information on the authors currently worth following, an indication of authors' affiliations, temporal evolution of publications and citations, relationships between emerging topics in publications, co-occurrence of topics and corresponding clustering, citation networks with (and without) temporal evolution, and research trends.

In this paper we aim to present a comprehensive bibliometric study of the literature on automated assessment in programming education, considering the Web of Science Core Collection. In particular, our goal is to answer the following research questions for the decade 2010-2020.

RQ1 Summarizing the collected data ...

- **RQ1-1** what has been the annual scientific production?
- **RQ1-2** what has been the average time interval for a new publication to get the first citation?
- **RQ1-3** which are the main journals/conferences to find literature in the area?

RQ2 Regarding authors ...

- **RQ2-1** who are the most productive? Who is more active lately? Who has more impact?
- **RQ2-2** do the most productive authors publish alone or as a group?
- **RQ2-3** what are the corresponding main affiliations?

RQ3 Regarding citations ...

- **RQ3-1** which are the most influential?
- **RQ3-2** which are the most relevant co-citations?

RQ4 Regarding the topics discussed ...

- **RQ4-1** what are the basic, niche, motor, and emerging?
- **RQ4-2** how did they evolve during the past decade?
- **RQ4-3** what is mostly discussed?
- **RQ4-4** are there significant differences if we see their yearly frequency?

The remainder of this paper is organized as follows. Section 2 presents the most relevant past surveys on automated assessment. Section 3 presents the methodology used to conduct this study. Section 4 presents the results of the bibliometric analysis and answers each of the research questions. Section 5 discusses the results, and compares them to recent literature review [1]. Finally, Section 6 summarizes the major contributions of this study.

2. Background

Tools to automate the assessment of programming assignments have been developed for more than sixty years [8]. Over the years, there have been several literature reviews that addresses advances in automated assessment of programming assignments, in particular, regarding the existing tools. In 2005, Ala-Mutka [3] published a survey of the characteristics of programming assignments that were automatically assessed to the date, distinguishing between the different types of techniques: dynamic analysis – which requires the execution of the program; and static analysis – in which the source code is evaluated without executing the program. In the same year, Douce et al. [2] conducted a review of automated assessment systems developed from the 1960s to the date, identifying three generations of automated assessment systems: (I) early attempts to automate assessment; (II) systems managed via a command-line or local graphical user interface; and (III) web-based tools.

Five years after the previous reviews, Ihantola et al. [4] conducted a systematic literature review on the advances in automated assessment of programming exercises, filling the existing gap. Our review takes a similar approach to the review by Ala-Mutka [3], in that it discusses the technical and pedagogical features rather than the specific tools. In 2016, Souza et al. [6] examined assessment tools for programming assignments, selecting 30 tools and categorizing them according to several dimensions to facilitate the choice of the adequate tool by instructors. Recently, Paiva et al. [1] performed a ten year comprehensive review of developments in the field of automated assessment in Computer Science, focusing not only on aspects such as testing techniques, sandboxing, feedback generation, and learning analytics. Furthermore, the review also includes a revision of the tools that shaped the decade.

No bibliometric study of the literature on automated assessment of programming assignments has been found in our research. Such a study would allow for a better understanding of the literature, particularly the most influential sources, authors, and publications, the relations between, trending topics of research, among other bibliometric and sociometric findings.

3. Methodology

The data for this study has been collected from the Web of Science (WoS) Core Collection, during the third week of September 2022. For that, a query[1] has been built to search all publication fields for a combination of keywords, as presented below

ALL=((automatic OR automated) AND (assessment OR evaluation OR grading OR marking) AND (programming OR computer science OR program))

The query includes a filter to limit results to those published in the decade 2010-2020. In addition to that, two refinements were needed. The first to narrow down search results to the adequate WoS categories for this area, namely: Computer Science Information Systems, Computer Science Artificial Intelligence, Computer Science Interdisciplinary Applications, Computer Science Software Engineering, Education Educational Research, Education Scientific Disciplines, Multidisciplinary Sciences, and Education Special. Even though some of these categories may still include out-of-scope publications, excluding them could result in the loss of important publications.

The result was a set of 11789 publications. The full record and cited references from these publications have been retrieved. A total of twenty-four BibTeX exports were necessary to obtain the data from all the 11789 publications, due to the limitations of WoS on the number of records allowed to be exported in a single request (in these conditions, the limit is 500). Finally, the twenty-four BibTeX files obtained have been merged into a single BibTeX file.

The collected set of 11789 publications was subject to a pre-processing phase, aiming to identify the relevant publications for analysis. For this phase, we have read the titles and abstracts of the papers to apply the following inclusion/exclusion criteria (as in Paiva et al. [1]):

IN if it presents either an automated assessment system or tool for CS education.

IN if it presents either an automated assessment technique or method for CS education.

IN if it presents an experience on the use of automated assessment for CS education.

IN if it presents a review on the use of automated assessment for CS education.

OUT if the automated assessment approach described is only applicable in the industry (based on authors' opinion, if in doubt).

OUT if it is a tutoring system or other system that does not automatically assess CS tasks other than quiz-based tasks.

[1] `https://www.webofscience.com/wos/woscc/summary/f75398e2-c55c-4b98-b5c0-103c1ebcb3cc-53a79dff/relevance/1`

OUT if it describes a general-purpose automated assessment approach, such as typical quizzes.
OUT if not written in English.
OUT if only abstract is available.

The outcome of this phase encompasses a set of 592 publications, which were selected for further analysis.

To answer the research questions that led to this study (see Subsection 1), several graphical representations were obtained using R and bibliometrix [9] – an open-source R-tool for quantitative research in scientometrics and bibliometrics. Bibliometrix provides methods for importing bibliographic data from SCOPUS, Clarivate Analytics' Web of Science, PubMed, Digital Science Dimensions and Cochrane databases, and performing bibliometric analysis, including co-citation, coupling, scientific collaboration analysis and co-word analysis. Some of the research questions, however, required a more customized analysis using R and traditional packages.

4. Results

This section presents the results of the analysis, answering each research question presented in Section 1. Subsection 4.1 provides a summary of the data used in the analysis, including answers to **RQ1**. Subsection 4.2 encompasses the results related to the authors' analysis (i.e., **RQ2**). Subsection 4.3 demonstrates the results regarding the analysis of citations (i.e., **RQ3**). Subsection 4.4 presents answers to **RQ4**, which pertains to topics and keywords.

4.1. Data Summary

The literature on automated assessment of programming assignments demonstrates the increasing research interest in this area, reflected by a growing rate of approximately 6.57 in annual scientific production during the decade 2010-2020. However, there was a slight decrease in the number of publications in the last two years, an exceptional situation that can be associated with the COVID-19 pandemic crisis. Therefore, 2018 was the peak year with the highest number (99) of publications. Figure 1 shows a visualization of the number of publications per year, with a linear trend and the associated confidence interval responding to **RQ1-1**.

Each of the collected documents was cited by an average of 8.81 other publications, with an average rate of 1.36 per year. Thus, in response to **RQ1-2**, it takes an average of 8.82 months to receive the first citation. Figure 2 shows the average and median citations of a document per year of publication with vertical error bars representing the corresponding variability. For example, a publication of 2010 (i.e., with 10 years) has an average of 8.11 citations, while a 5-year old publication has an average of 8.49 citations.

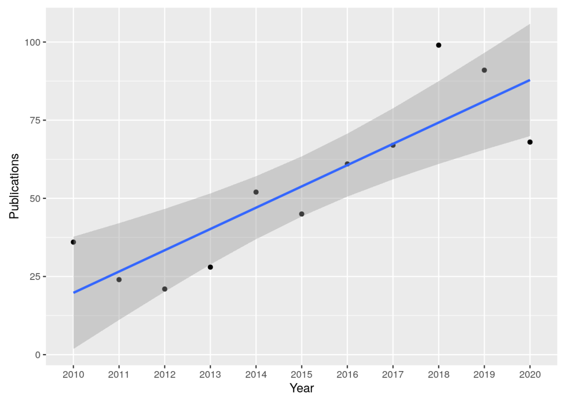

Figure 1. Number of publications per year with linear trend and its confidence interval

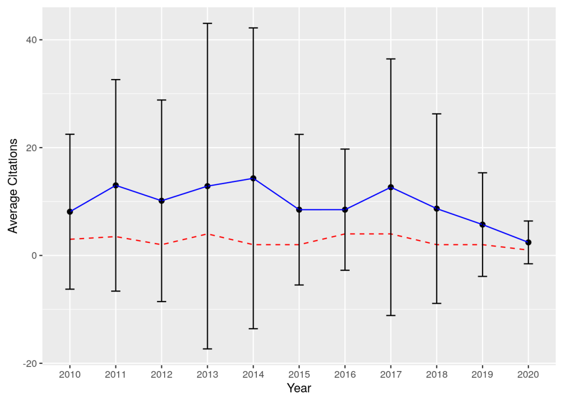

Figure 2. Average (blue line) and media (red dashed line) citations per year of publication with vertical error bars

Concerning the sources of the publications, they are distributed among 361 different sources. The top 25 publication sources (**RQ1-3**) are shown in Figure 3, and account for more than a fourth of the total publications. The Proceedings of the 51st ACM Technical Symposium on Computer Science Education is the source with the highest

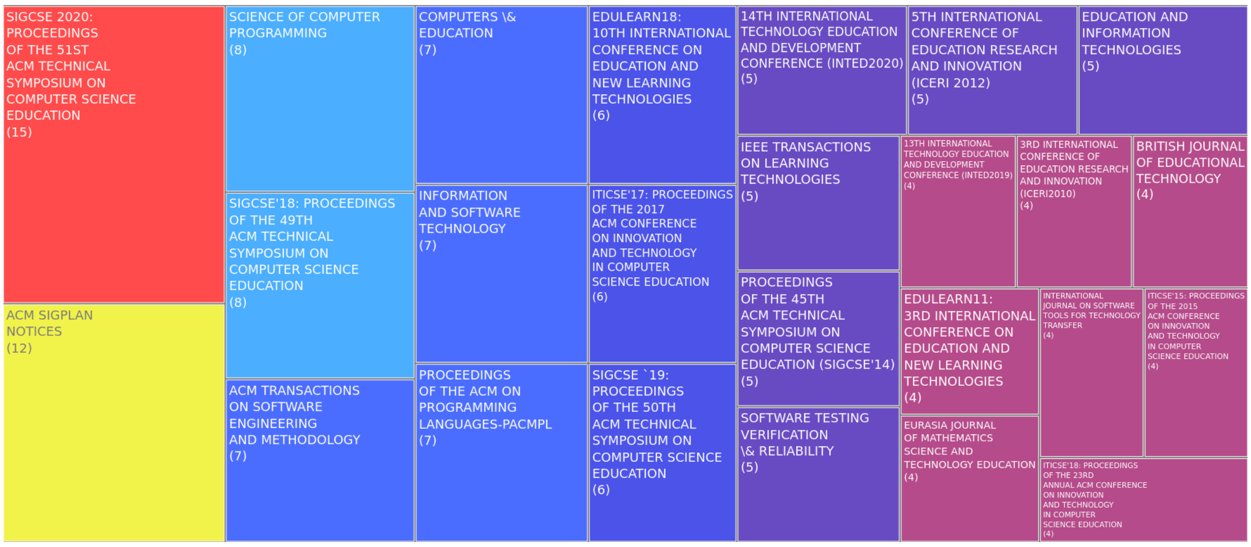

Figure 3. Top-25 sources of publications in a tree-map

number of articles collected (15), followed by ACM Special Interest Group on Programming Languages (SIGPLAN) Notices with 12 publications. Science of Computer Programming and the Proceedings of the 49th ACM Technical Symposium on Computer Science Education come up tied in third place, each with 8 publications. ACM Transactions on Software Engineering and Methodology, Computers & Education, Information and Software Technology, and the Proceedings of the ACM on Programming Languages, with 7 publications each, complete the top-5 sources.

4.2. Authors

There are a total of 1618 authors on the selected publications. Of these, 46 are authors of documents with only one author, while 1572 are authors of documents with multiple authors. On average, there are 3.26 authors and 3.47 co-authors per document (i.e., excluding single-author publications).

With respect to **RQ2-1**, by "most prolific authors" we mean authors who have made more publications. Figure 4 shows the top-10 authors (sorted in descending order, from top to bottom) who have made more contributions to the field, and for them the number of publications and citations per year. From this perspective, the authors who are more active recently, such as Fraser G. and Edwards S. H., and those who were more active at the beginning of the decade, such as Xie T., Queirós R., and Leal J. P., are easier to identify. Nevertheless, the most impactful works are that of Fraser G., which concentrates on software testing techniques, and Kim M., who investigates fault localization and program repairing techniques. Finally, Kim D., who works mostly in techniques for automated generation of feedback, completes the podium regarding authors' impact. This can be confirmed by measuring the authors' h-index (5, 4, and 4, respectively).

To answer **RQ2-2**, we collect all publications from the most prolific authors and construct a histogram of the number of authors per publication for each of them separately. Figure 5 illustrates the result. The only author who has worked alone is Ricardo Q. (1), while all others have no single-authored publications. Nevertheless, Edwards S. H. publishes mainly in small groups of two or three. Bey A. has only publications with two co-authors. Interestingly, Sade M. and Tonisson E. have only worked in large groups of 6 or more authors.

Regarding the authors' affiliations (**RQ2-3**), there are 636 distinct identified affiliations within the collected publications. Note that a publication can count to more than one affiliation, if it involves either authors with multiple affiliations or documents with multiple authors resulting from a collaboration between different institutions. The top-20 most prolific affiliations are presented in Figure 6, which alone account for more than 39% of the identified affiliations. The Carnegie Mellon University is the institution with the most publications (21), followed by the University of Porto (17). The Nanjing University, the University of Illinois, and the University of Tartu, both appearing with 16 publications, occupy the third position.

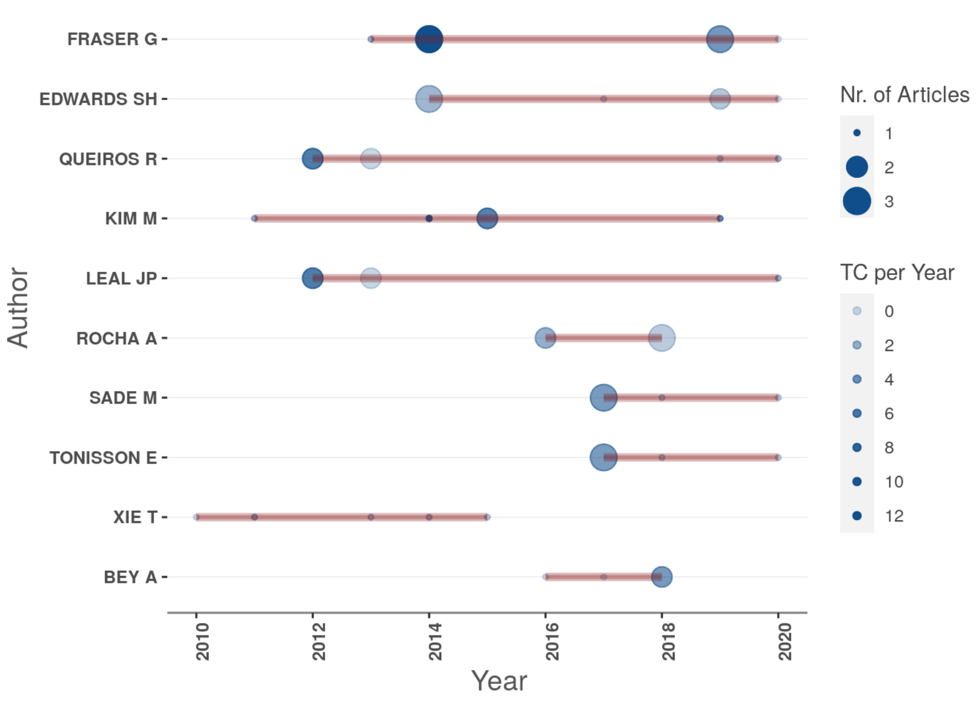

Figure 4. Productivity of authors over time (TC stands for Times Cited)

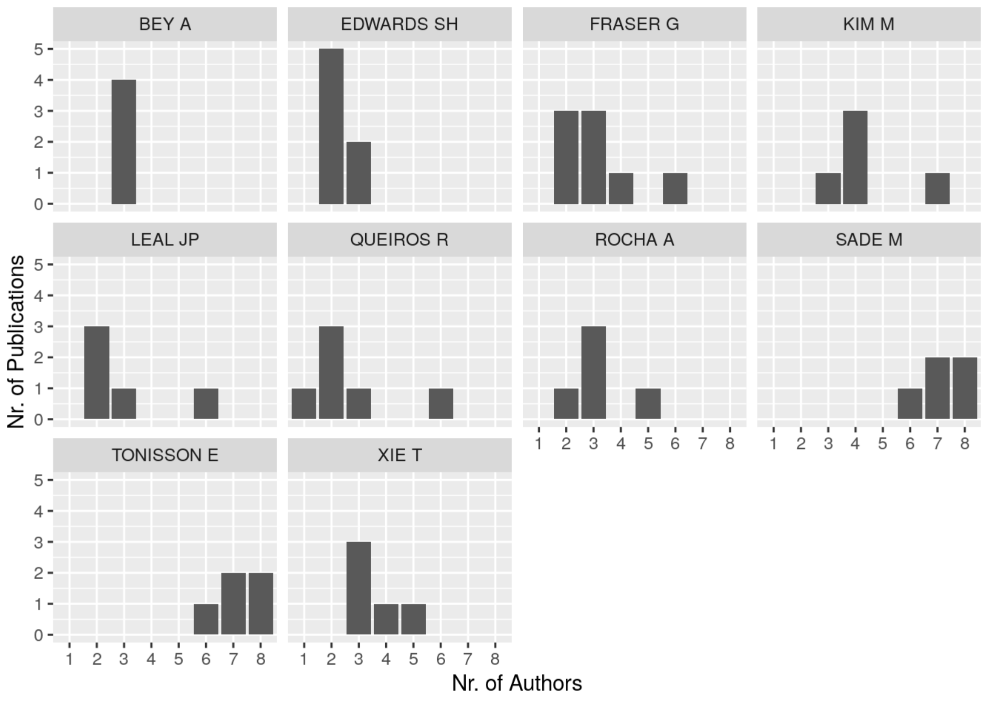

Figure 5. Number of authors per publication for the most productive authors

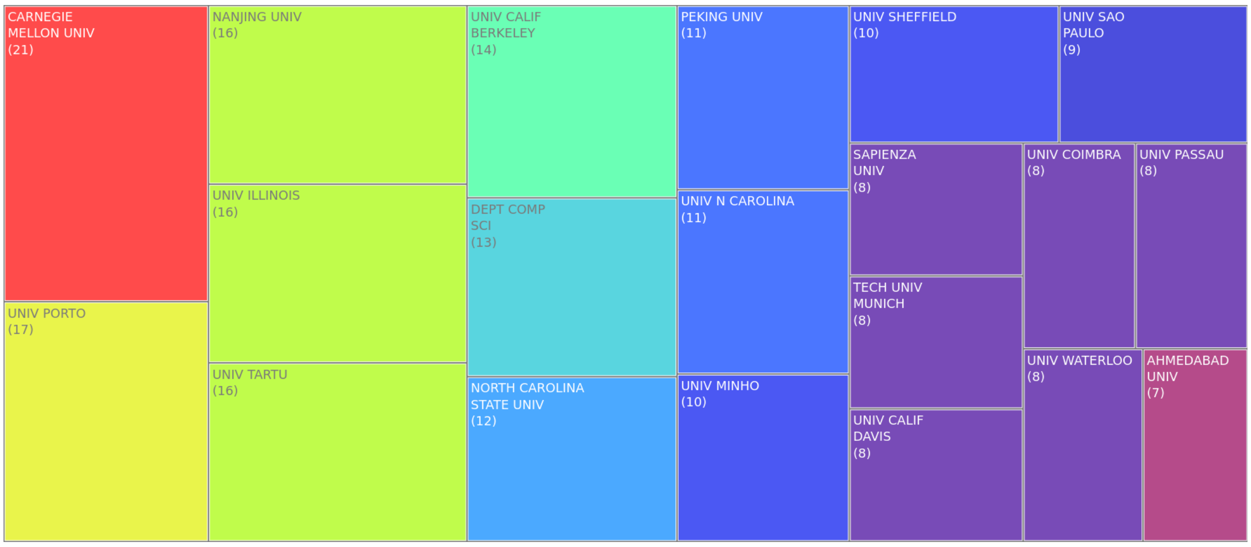

Figure 6. Top-20 authors' affiliations by productivity in a tree-map

4.3. *Citations*

In selecting the most influential publications (**RQ3-1**), it is important to have a measure that takes into account not only the number of citations but also the year of publication. For this purpose, we used the Normalized Citation Score (NCS) of a document, which is calculated by dividing the actual number of cited publications by the expected citation rate for publications of the same year. Furthermore, the answer to **RQ3-1** is twofold.

On the one hand, the local NCS (i.e., citations within the collected data) determines the most influential publications within the area. The top-5 publications under such conditions are: "A distributed system for learning programming on-line" by Verdú et al. [10]; "Marking student programs using graph similarity" by Naudé et al. [11]; "A Critical Review of Automatic Patch Generation Learned from Human-Written Patches: Essay on the Problem Statement and the Evaluation of Automatic Software Repair" by Monperrus [12]; "A system to grade computer programming skills using machine learning" by Srikant S. [13]; and "Comparing test quality measures for assessing student-written tests" by Edwards et al. [14].

On the other hand, looking at all the citations provides a global perspective on the most influential publications. The top-5 publications in this regard are: "Automated Feedback Generation for Introductory Programming from Assignments" by Singh et al. [15]; "Precise Condition Synthesis for Program Repair" by Xiong et al. [16]; "Ask the Mutants: Mutating Faulty Programs for Fault Localization" by Moon et al. [17]; "Context-Aware Patch Generation for Better Automated Program Repair" by Wen et al. [18]; and "Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming" by Blikstein et al. [19].

As for **RQ3-2**, the answer is provided in the historiographic map of Figure 7, a graph proposed by E. Garfield [20] which is a chronological network map of the most relevant co-citations from a bibliographic collection. This map identifies four separate groups corresponding to different topics, namely: **Group I (Light Blue)** encompasses works on automated feedback for CS projects [21, 22]; **Group II (Purple)** captures works exploring the automated assessment of the computational thinking skills of novice programmers [13, 23, 24]; **Group III (Green)** includes publications on automated program repair techniques and tools [12, 25]; **Group IV (Yellow)** includes automated assessment tools for assessing code and tests' quality [14, 26, 27, 28]; **Group V (Red)** includes works integrating automated assessment tools with other e-learning tools [10, 29]; **Group VI (Blue)** captures a group of works aiming to improve feedback on automated assessment [11, 30, 31].

4.4. *Topics and Keywords*

Keywords, either provided by the authors or extracted as n-grams from the title or abstract, can provide information about current issues, trends, and methods in the field. Therefore, for the group of research questions **RQ4** these are the properties that are the subject of analysis.

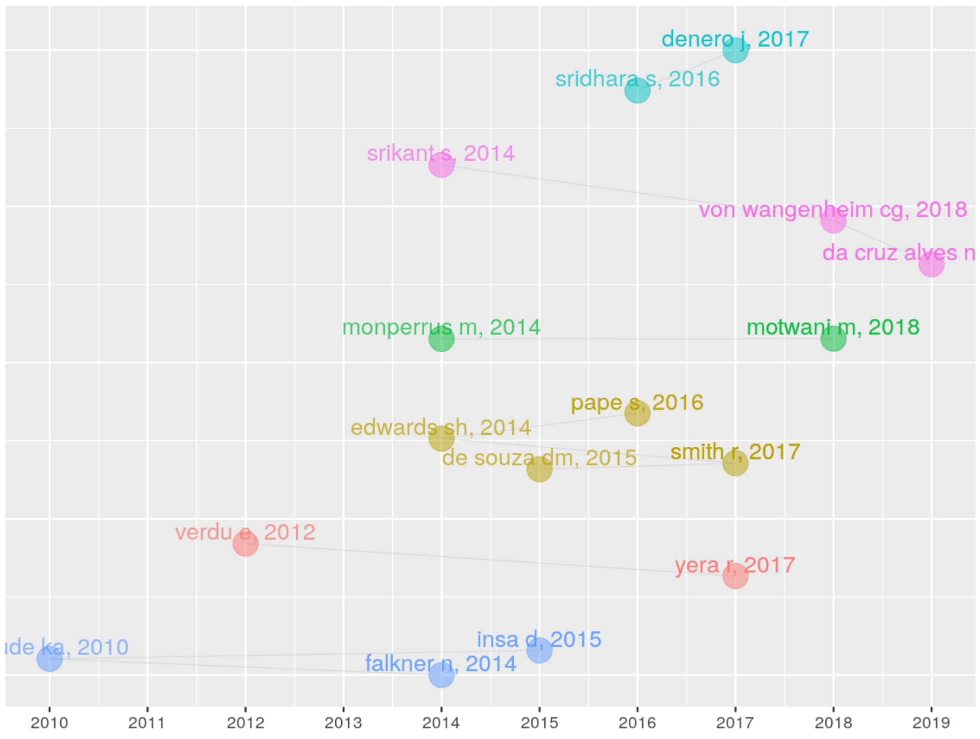

Figure 7. Historiographical representation of a chronological network map of most relevant co-citations

For the first question of the group (**RQ4-1**), the answer is provided in Figure 8, through a thematic map based on the analysis of co-word networks and clustering using the authors' keywords. This approach is similar to the proposal of Cobo et al. [32]. It identifies four types of topics (themes) based on density (i.e., degree of development) and centrality (i.e., degree of relevance), namely: emerging or declining (low centrality and low density), niche (low centrality and high density), motor (high centrality and high density), and basic (high centrality and low density) topics. Among the emerging or declining topics, a cluster involving Feedback – an important aspect of automated assessment – is notable, but so is another cluster involving Machine Learning – which unsurprisingly is also making inroads in this area – and Automated Program Repair – a technique used to automatically correct programs, which is being applied to generate feedback. Niche topics include Program Synthesis and Program Refactoring, which include techniques that can help assessing programs based on a set of constraints and/or generate accurate feedback. Motor themes focus on Static Analysis – analyzing source-code rather than its runtime behavior – while the other topics have to do with the domain itself (e.g., automated assessment, programming, and software testing). Finally, Symbolic Execution – a method of abstractly executing a program to find out what inputs cause the execution of each part of a program – is the only identified keyword that can be classified as a topic.

As for **RQ4-2**, Figure 9 divides the decade into three sections (2010-2013, 2014-2017, and 2018-2020) and shows the thematic evolution between the three sections, based on analysis of the co-word network and the clustering of the authors' keywords [32]. Some interesting outcomes of this analysis are: the evolution of Static Analysis and its later ramifications to Testing, Tools, and Machine Learning; the rising of Machine Learning approaches that rapidly penetrate different domains and provide better results than existing techniques; and the disclosure of techniques important for feedback purposes, such as Fault Localization, Automated Test Generation, and Automated Program Repair.

The visualizations so far already provide a good introspective on the topics that have been addressed in the last decade. However, to answer **RQ4-3**, the analysis focuses on 2-grams extracted from the abstract. To this end, Figure 10 presents a conceptual structure map created using Multiple Correspondence Analysis (MCA) – a data analysis method to measure the association between two or more qualitative variables – and Clustering of a bipartite network of the extracted terms. Using this approach, 2-grams are divided into four clusters, which can be described

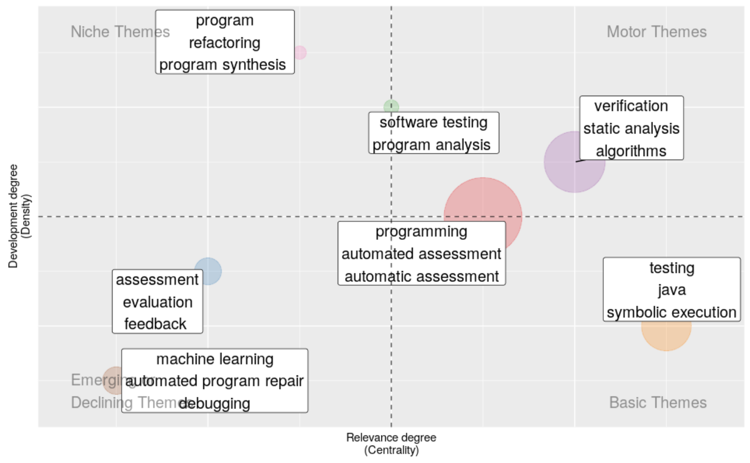

Figure 8. Thematic map based on authors' keywords

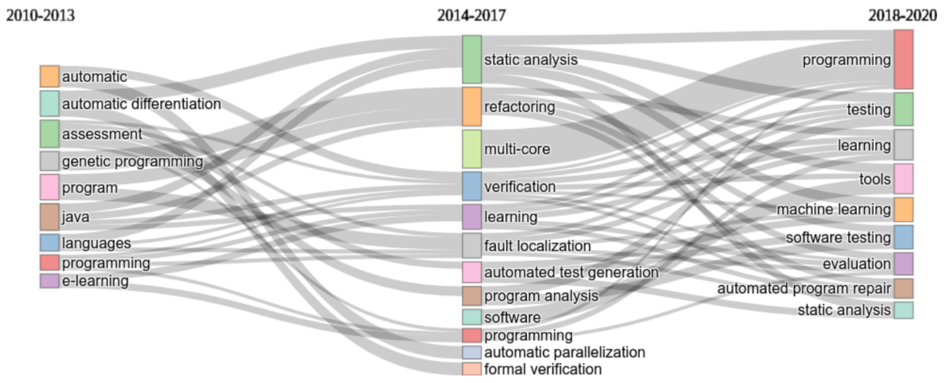

Figure 9. Thematic evolution based on authors' keywords

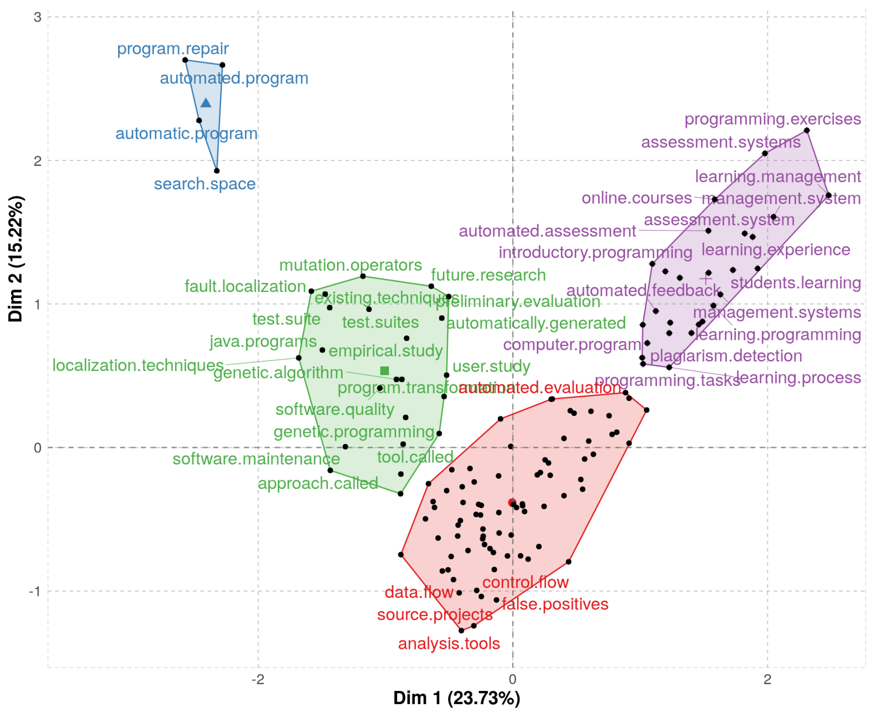

Figure 10. Conceptual structure map of abstract 2-grams obtained through Multiple Correspondence Analysis (MCA)

as follows: **Group I (Blue)** captures terms related to automated program repair; **Group II (Green)** includes terms related to testing techniques and evaluated facets of a program; **Group III (Red)** contains 2-grams related to static analysis techniques; and **Group IV (Purple)** whose terms are more related to tools and systems.

With respect to **RQ4-4**, Figure 11 shows the ten most frequent abstract 2-grams by year. When looking at the frequency of these 2-grams by year, the increasing interest in Static Analysis, Automated Program Repair, and Automated Test Generation is readily apparent. Although less visible, Machine Learning and Learning Analytics have also increased slightly over the years. This indicates a large growing interest in improving automated feedback generation, as most topics gaining popularity are related to source code analysis (Static Analysis and Machine Learning – in the current context) and fixing (Automated Program Repair, Automated Test Generation – including counter-example –, and Machine Learning) techniques. Nevertheless, dynamic analysis-based assessment using test suites is still highly frequent. Moreover, feedback for teachers, through Learning Analytics, seems to be now a topic of interest within the area Automated Assessment. Note that, for this visualization, the set of generated 2-grams has been preprocessed to remove common terms (e.g., science, introductory, programming, paper, work, result, etc) and match synonyms (e.g., apr tool and repair tool).

5. Discussion

The results demonstrate that Automated Assessment is still an area of increasing research interest, with a significant and growing number of publications and consequently authors and citations. The only exception coincides with (and can be justified by) the COVID-19 pandemic situation, which occurred between the start of 2020 and the start of 2022. The number of citations has maintained a nearly constant rate over the years (see Figure 2). Most of these publications appear in journals and conference proceedings, with shares of nearly 30% and 70%, respectively.

The most closely related and recent systematic literature review on automated assessment is the one by Paiva et al. [1]. This review identified a new era of automated assessment in Computer Science, the era of containerization, among other interesting findings. In particular, the growing interest in static analysis techniques to assess not only the correct functionality of a program, but also the code quality and presence of plagiarism. Furthermore, it notices

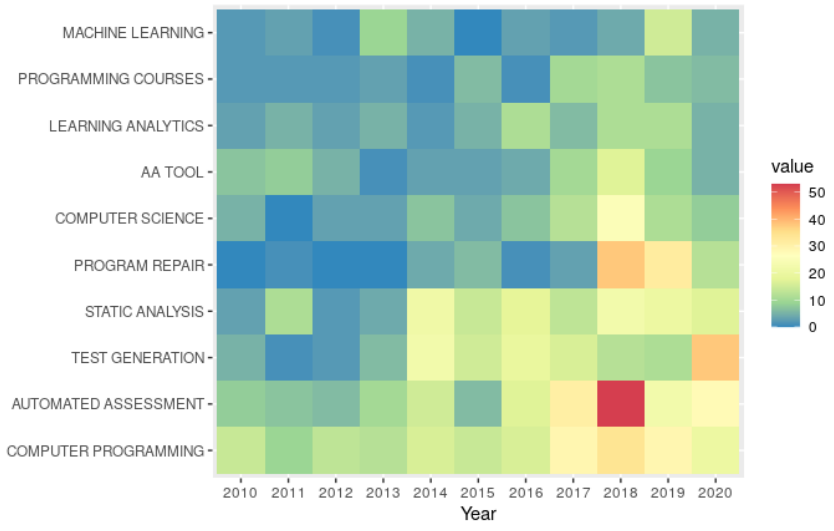

Figure 11. Top-10 most frequent abstract 2-grams by year

the efforts towards better feedback primarily by introducing techniques from other research areas, such as automated program repair, fault localization, symbolic execution, and machine learning. Regarding automated assessment tools, more than half of the mentioned tools are open source. Finally, the increasing interest in incorporating Learning Analytics into automated assessment tools to help teachers understand student difficulties is also mentioned.

A technical report by Porfirio et al. [33] presents a systematic literature mapping of the research literature on automatic source code evaluation until 2019, which also had similar findings. In particular, it (1) shows the increasing number of publications during the years; (2) notices a few attempts to extract knowledge and visualize information about students from data produced during the automated assessment of source code (i.e., first attempts on Learning Analytics); and (3) demonstrates that functional correctness is the aspect receiving most attention while dynamic test cases-based strategies are predominating, but these are slowly losing space to static analysis techniques in favor of assessing the source code itself rather than its functionality and improving feedback.

Results from this paper concerning authors (see Subsection 4.2) and citations (see Subsection 4.3) are novel. The responses given in subsection 4.4 to research questions of group **RQ4** confirm most of the findings of previous works, namely the recent focus on static analysis approaches and the introduction of techniques from other research areas, such as automated program repair, fault localization, and machine learning. Traditional automated assessment based on running the program against a set of test cases is still the dominating strategy. Moreover, the high frequency of some keywords related to Learning Analytics corroborates the interest in integrating outcomes from this research area into automated assessment tools. Nevertheless, this research could not capture enough information to confirm the trend of containerization of automated assessment. As the conducted analysis had minimal human interference, if "docker" (or a related term) was neither a frequent keyword nor part of a frequent abstract 2-gram, then it was not identified. In contrast, in the aforementioned review [1], a number of publications were manually annotated with a predetermined set of tags after reading.

6. Conclusion

This paper presents a bibliometric study of publications on automatic assessment in Computer Science from the decade 2010-2020, based on the Web of Science (WoS) Core Collection. The analysis shows that this is still a research area of growing interest, where there is still much room for improvement of current solutions, especially through static analysis and source code analysis techniques used in other research areas. Therefore, it will be worthwhile to continue pursuing this topic in the coming years. The analysis performed allowed us to answer all the research questions posed at the beginning of this study and presented in section 1. In addition, part of the results are identical to a recently published systematic literature review on automated assessment in computer science.

Admittedly, this study has some limitations. First, the Web of Science (WoS) Core Collection does not include publications from all sources. Second, the names of some authors and affiliations appear in different forms over the decade, which may introduce some bias into the analysis. In this case, the work of a database such as the Web of Science (WoS) Core Collection to standardize affiliations and authors is important.

Acknowledgements

J.C.P. wishes to acknowledge the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology), Portugal – for the Ph.D. Grant 2020.04430.BD.

References

[1] J. C. Paiva, J. P. Leal, A. Figueira, Automated assessment in computer science education: A state-of-the-art review, ACM Trans. Comput. Educ. 22 (3). doi:10.1145/3513140.
URL `https://doi.org/10.1145/3513140`

[2] C. Douce, D. Livingstone, J. Orwell, Automatic test-based assessment of programming: A review, Journal on Educational Resources in Computing 5 (3) (2005) 4. doi:10.1145/1163405.1163409.
URL `https://dl.acm.org/doi/10.1145/1163405.1163409`

[3] K. M. Ala-Mutka, A Survey of Automated Assessment Approaches for Programming Assignments, Computer Science Education 15 (2) (2005) 83–102. doi:10.1080/08993400500150747.
URL `http://www.tandfonline.com/doi/abs/10.1080/08993400500150747`

[4] P. Ihantola, T. Ahoniemi, V. Karavirta, O. Seppälä, Review of recent systems for automatic assessment of programming assignments, in: Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10, ACM Press, Berlin, Germany, 2010, pp. 86–93. doi:10.1145/1930464.1930480.
URL `http://portal.acm.org/citation.cfm?doid=1930464.1930480`

[5] R. Romli, S. Sulaiman, K. Z. Zamli, Automatic programming assessment and test data generation a review on its approaches, in: 2010 International Symposium on Information Technology, Vol. 3, IEEE, Kuala Lumpur, Malaysia, 2010, pp. 1186–1192. doi:10.1109/ITSIM.2010.5561488.

[6] D. M. Souza, K. R. Felizardo, E. F. Barbosa, A Systematic Literature Review of Assessment Tools for Programming Assignments, in: 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), IEEE, Dallas, TX, USA, 2016, pp. 147–156. doi:10.1109/CSEET.2016.48.
URL `http://ieeexplore.ieee.org/document/7474479/`

[7] A. Andrés, Measuring Academic Research, Chandos Publishing (Oxford), Witney, England, 2009.

[8] J. Hollingsworth, Automatic graders for programming classes, Commun. ACM 3 (10) (1960) 528–529. doi:10.1145/367415.367422.
URL `https://doi.org/10.1145/367415.367422`

[9] M. Aria, C. Cuccurullo, bibliometrix: An r-tool for comprehensive science mapping analysis, Journal of Informetrics 11 (4) (2017) 959–975. doi:https://doi.org/10.1016/j.joi.2017.08.007.
URL `https://www.sciencedirect.com/science/article/pii/S1751157717300500`

[10] E. Verdú, L. M. Regueras, M. J. Verdú, J. P. Leal, J. P. de Castro, R. Queirós, A distributed system for learning programming on-line, Computers & Education 58 (1) (2012) 1–10. doi:https://doi.org/10.1016/j.compedu.2011.08.015.
URL `https://www.sciencedirect.com/science/article/pii/S036013151100193X`

[11] K. A. Naudé, J. H. Greyling, D. Vogts, Marking student programs using graph similarity, Computers & Education 54 (2) (2010) 545–561. doi:https://doi.org/10.1016/j.compedu.2009.09.005.
URL `https://www.sciencedirect.com/science/article/pii/S0360131509002450`

[12] M. Monperrus, A critical review of "automatic patch generation learned from human-written patches": Essay on the problem statement and the evaluation of automatic software repair, in: Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 234–242. doi:10.1145/2568225.2568324.
URL `https://doi.org/10.1145/2568225.2568324`

[13] S. Srikant, V. Aggarwal, A system to grade computer programming skills using machine learning, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 1887–1896. doi:10.1145/2623330.2623377.
URL `https://doi.org/10.1145/2623330.2623377`

[14] S. H. Edwards, Z. Shams, Comparing test quality measures for assessing student-written tests, in: Companion Proceedings of the 36th International Conference on Software Engineering, ICSE Companion 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 354–363. doi:10.1145/2591062.2591164.
URL `https://doi.org/10.1145/2591062.2591164`

[15] R. Singh, S. Gulwani, A. Solar-Lezama, Automated feedback generation for introductory programming assignments, in: Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 15–26. doi:10.1145/2491956.2462195.
URL `https://doi.org/10.1145/2491956.2462195`

[16] Y. Xiong, J. Wang, R. Yan, J. Zhang, S. Han, G. Huang, L. Zhang, Precise condition synthesis for program repair, in: Proceedings of the 39th International Conference on Software Engineering, ICSE '17, IEEE Press, 2017, p. 416–426. doi:10.1109/ICSE.2017.45.
URL `https://doi.org/10.1109/ICSE.2017.45`

[17] S. Moon, Y. Kim, M. Kim, S. Yoo, Ask the mutants: Mutating faulty programs for fault localization, in: 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation, IEEE, 2014, pp. 153–162. doi:10.1109/ICST.2014.28.

[18] M. Wen, J. Chen, R. Wu, D. Hao, S.-C. Cheung, Context-aware patch generation for better automated program repair, in: Proceedings of the 40th International Conference on Software Engineering, ICSE '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1–11. doi:10.1145/3180155.3180233.
URL `https://doi.org/10.1145/3180155.3180233`

[19] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, D. Koller, Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming, Journal of the Learning Sciences 23 (4) (2014) 561–599. arXiv:https://doi.org/10.1080/10508406.2014.954750, doi:10.1080/10508406.2014.954750.
URL `https://doi.org/10.1080/10508406.2014.954750`

[20] E. Garfield, Historiographic mapping of knowledge domains literature, Journal of Information Science 30 (2) (2004) 119–145. arXiv:https://doi.org/10.1177/0165551504042802, doi:10.1177/0165551504042802.
URL `https://doi.org/10.1177/0165551504042802`

[21] J. DeNero, S. Sridhara, M. Pérez-Quiñones, A. Nayak, B. Leong, Beyond autograding: Advances in student feedback platforms, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 651–652. doi:10.1145/3017680.3017686.
URL `https://doi.org/10.1145/3017680.3017686`

[22] S. Sridhara, B. Hou, J. Lu, J. DeNero, Fuzz testing projects in massive courses, in: Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 361–367. doi:10.1145/2876034.2876050.
URL `https://doi.org/10.1145/2876034.2876050`

[23] C. G. von Wangenheim, J. C. R. Hauck, M. F. Demetrio, R. Pelle, N. da Cruz Alves, H. Barbosa, L. F. Azevedo, Codemaster - automatic assessment and grading of app inventor and snap! programs, Informatics in Education 17 (1) (2018) 117–150. doi:10.15388/infedu.2018.08.

[24] N. da Cruz Alves, C. G. V. Wangenheim, J. C. R. Hauck, Approaches to assess computational thinking competences based on code analysis in k-12 education: A systematic mapping study, Informatics in Education 18 (1) (2019) 17–39. doi:10.15388/infedu.2019.02.

[25] M. Motwani, S. Sankaranarayanan, R. Just, Y. Brun, Do automated program repair techniques repair hard and important bugs?, Empirical Softw. Engg. 23 (5) (2018) 2901–2947. doi:10.1007/s10664-017-9550-0.
URL `https://doi.org/10.1007/s10664-017-9550-0`

[26] D. M. D. Souza, S. Isotani, E. F. Barbosa, Teaching novice programmers using progtest, International Journal of Knowledge and Learning 10 (1) (2015) 60–77. arXiv:https://www.inderscienceonline.com/doi/pdf/10.1504/IJKL.2015.071054, doi:10.1504/IJKL.2015.071054.
URL `https://www.inderscienceonline.com/doi/abs/10.1504/IJKL.2015.071054`

[27] S. Pape, J. Flake, A. Beckmann, J. Jürjens, Stage: A software tool for automatic grading of testing exercises: Case study paper, in: Proceedings of the 38th International Conference on Software Engineering Companion, ICSE '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 491–500. doi:10.1145/2889160.2889203.
URL `https://doi.org/10.1145/2889160.2889203`

[28] R. Smith, T. Tang, J. Warren, S. Rixner, An automated system for interactively learning software testing, in: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 98–103. doi:10.1145/3059009.3059022.
URL `https://doi.org/10.1145/3059009.3059022`

[29] R. Yera, L. Martínez, A recommendation approach for programming online judges supported by data preprocessing techniques, Applied Intelligence 47 (2) (2017) 277–290. doi:10.1007/s10489-016-0892-x.
URL `https://doi.org/10.1007/s10489-016-0892-x`

[30] N. Falkner, R. Vivian, D. Piper, K. Falkner, Increasing the effectiveness of automated assessment by increasing marking granularity and feedback units, in: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 9–14. doi:10.1145/2538862.2538896.
URL `https://doi.org/10.1145/2538862.2538896`

[31] D. Insa, J. Silva, Semi-automatic assessment of unrestrained java code: A library, a dsl, and a workbench to assess exams and exercises, in: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 39–44. doi:10.1145/2729094.2742615.
URL `https://doi.org/10.1145/2729094.2742615`

[32] M. Cobo, A. López-Herrera, E. Herrera-Viedma, F. Herrera, An approach for detecting, quantifying, and visualizing the evolution of a research field: A practical application to the fuzzy sets theory field, Journal of Informetrics 5 (1) (2011) 146–166. doi:https://doi.org/10.1016/j.joi.2010.10.002.
URL `https://www.sciencedirect.com/science/article/pii/S1751157710000891`

[33] A. Porfirio, R. Pereira, E. Maschio, Automatic source code evaluation: a systematic mapping, Tech. rep., Federal University of Technology, Paraná, Brazil (UTFPR) (09 2021). doi:10.13140/RG.2.2.36112.33287.