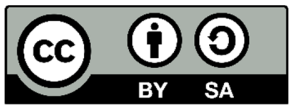


https://revistas.udistrital.edu.co/ojs/index.php/Tecnura/issue/view/1136

DOI: https://doi.org/10.14483/22487638.19637

Clasificación del artículo: Revisión de tema

Artificial Intelligence and Computer-Supported Collaborative Learning in Programming: A Systematic Mapping Study

Inteligencia artificial y aprendizaje colaborativo asistido por computadora en la programación: un estudio de mapeo sistemático

Cómo citar: **Hidalgo**., C.G. **Bucheli-Guerrero**., V.A. y **Ordóñez-Eraso**., H.A. (2023). *Artificial Intelligence and Computer-Supported Collaborative Learning in Programming: A Systematic Mapping Study. Tecnura*, *27*(75). https://doi.org/ 10.14483/22487638.19637

Carlos Giovanny Hidalgo

Systems engineer, Master of Engineering, PhD student. Assistant professor at Universidad del Valle, Cali, Colombia. Email: carlos.hidalgo@correounivalle.edu.co ORCID: https://orcid.org/0000-0003-2308-0720

Víctor Andrés Bucheli-Guerrero

Systems engineer, master's degree in Engineering and Computing, PhD in engineering. Full professor at Universidad del Valle, Cali, Colombia. Email: victor.bucheli@correounivalle.edu.co ORCID: https://orcid.org/0000-0002-0885-8699

Hugo Armando Ordóñez-Eraso

Systems engineer, master's degree in Engineering and Computing, PhD in Engineering. Full professor at Universidad del Cauca, Popayán, Colombia. Email: hugoordonez@unicauca.edu.co ORCID: https://orcid.org/ 0000-0002-3465-5617

ABSTRACT

Objective: The Computer-Supported Collaborative Learning (CSCL) approach integrates artificial intelligence (AI) to enhance the learning process through collaboration and information and communication technologies (ICTs). In this sense, innovative and effective strategies could be designed for learning computer programming. This paper presents a systematic mapping study from 2009 to 2021, which shows how the integration of CSCL and AI supports the learning process in programming courses.

Methodology: This study was conducted by reviewing data from different bibliographic sources such as Scopus, Web of Science (WoS), ScienceDirect, and repositories of the GitHub platform. It employs a quantitative methodological approach, where the results are represented through technological maps that show the following aspects: i) the programming

languages used for CSCL and AI software development; ii) CSCL software technology and the evolution of AI; and iii) the ACM classifications, research topics, artificial intelligence techniques, and CSCL strategies.

Results: The results of this research help to understand the benefits and challenges of using the CSCL and AI approach for learning computer programming, identifying some strategies and tools to improve the process in programming courses (*e.g.*, the implementation of the CSCL approach strategies used to form groups, others to evaluate, and others to provide feedback); as well as to control the process and measure student results, using virtual judges for automatic code evaluation, profile identification, code analysis, teacher simulation, active learning activities, and interactive environments, among others. However, for each process, there are still open research questions.

Conclusions: This work discusses the integration of CSCL and AI to enhance learning in programming courses and how it supports students' education process. No model integrates the CSCL approach with AI techniques, which allows implementing learning activities and, at the same time, observing and analyzing the evolution of the system and how its users (students) improve their learning skills with regard to programming. In addition, the different tools found in this paper could be explored by professors and institutions, or new technologies could be developed from them.

Keywords: artificial intelligence, computer programming, computer-supported collaborative learning, learning computer programming

RESUMEN

Objetivo: El enfoque de aprendizaje colaborativo asistido por computadora (CSCL) integra la inteligencia artificial (IA) para mejorar el proceso de aprendizaje a través de la colaboración y las tecnologías de la información y la comunicación (TICs). En este sentido,

se podrían diseñar estrategias innovadoras y efectivas para el aprendizaje de la programación de computadoras. Este artículo presenta un estudio sistemático de mapeo de los años 2009 a 2021, el cual muestra cómo la integración del CSCL y la IA apoya el proceso de aprendizaje en cursos de programación.

Metodología: Este estudio se realizó mediante una revisión de datos proveniente de distintas fuentes bibliográficas como Scopus, Web of Science (WoS), ScienceDirect y repositorios de la plataforma GitHub. El trabajo emplea un enfoque metodológico cuantitativo, en el cual los resultados se representan a través de mapas tecnológicos que muestran los siguientes aspectos: i) los lenguajes de programación utilizados para el desarrollo de software de CSCL e IA; ii) la tecnología de software CSCL y la evolución de la IA; y iii) las clasificaciones, los temas de investigación, las técnicas de inteligencia artificial y las estrategias de CSCL de la ACM.

Resultados: Los resultados de esta investigación ayudan a entender los beneficios y retos de usar el enfoque de CSCL e IA para el aprendizaje de la programación de computadoras, identificando algunas estrategias y herramientas para mejorar el proceso en cursos de programación (*e.g.*, La implementación de estrategias del enfoque CSCL utilizadas para formar grupos, de otras para evaluar y de otras para brindar retroalimentación); así como para monitorear el proceso y medir los resultados de los estudiantes utilizando jueces virtuales para la evaluación automática del código, identificación de perfiles, análisis de código, simulación de profesores, actividades de aprendizaje activo y entornos interactivos, entre otros. Sin embargo, aún hay preguntas investigación por resolver para cada proceso.

Conclusiones: Este trabajo discute la integración del CSCL y la IA para mejorar el aprendizaje en cursos de programación y cómo esta apoya el proceso educativo de los estudiantes. Ningún modelo integra el enfoque CSCL con técnicas de IA, lo cual permite

implementar actividades de aprendizaje y, al mismo tiempo, observar y analizar la evolución del sistema y de la manera en que sus usuarios (estudiantes) mejoran sus habilidades de aprendizaje con respecto a la programación. Adicionalmente, las diferentes herramientas encontradas en este artículo podrían ser exploradas por profesores e instituciones, o podrían desarrollarse nuevas tecnologías a partir de ellas.

Palabras clave: inteligencia artificial, programación de computadoras, aprendizaje colaborativo asistido por computadora, aprendizaje de programación

INTRODUCTION

In Computer Science (CS), programming courses have a higher attrition rate compared with other courses (Figueiredo & García-Peñalvo, 2018; Munson & Zitovsky, 2018; Zingaro *et al.*, 2018). The use of collaborative approaches in programming courses has shown satisfactory results by providing skills, aptitudes, and good practices, among other benefits (Suarez *et al.*, 2021). The Computer Supported Collaborative Learning (CSCL) approach integrates artificial intelligence (AI) to improve the learning process through collaboration and information and communication technology (ICT) (Kozma, 2000; Loizzo & Ertmer, 2016; Docq & Daele, 2001; Solarte-Pabón & Machuca-Villegas, 2019; Thomas *et al.*, 2002). According to Magnisalis *et al.* (2011), the CSCL approach is based on the following processes: a) Group Formation (GF support): forming student groups to obtain the best results from collaboration; b) Learning Domain (LD support): evaluating student learning activities with the help computers; c) Path of Interaction (PI support): the student receives comments on learning outcomes. The literature shows some projects that integrate CSCL processes with artificial intelligence (AI) techniques, such as Machine Learning to automatically identify groups of students according to their profiles or natural language processing for the automatic

assessment of code (Casamayor *et al.*, 2009; Costaguta & de los Angeles Menini, 2014; Black & Wiliam, 1998; Varier *et al.*, 2017).

However, in the literature, there is no state-of-the-art technique that shows how the integration of CSCL and AI impacts and improves the learning process in programming courses. This paper presents a systematic mapping study based on tech mining (Mohammadi, 2012), which was conducted by reviewing data from scientific documents such as journals and conference proceedings from Scopus, WoS, and ScienceDirect, in addition to data extraction and analysis from repositories in the GitHub platform. The final corpus contains 316 records: 90 papers and 226 repositories. This, with the aim to understand the evolution of programming languages, technologies, tools, and strategies based on CSCL and AI to support teaching in programming courses.

This paper has three sections. The first section presents the methodology in three parts related to the elaboration of the mapping study: research questions and expected results, data protocol, and the construction of technological map visualizations. The second section shows the expected results, classified into four aspects: the commonly used programming languages in the development of technologies and learning tools based on CSCL and AI; the timeline of the evolution of projects, tools, and technologies; the strategies for CSCL processes based on AI; and the classification of the reference corpus according to the ACM Computing Classification System (CCS) (ACM, n.d.). Finally, based on the results, the third section discusses and concludes what is needed to improve the learning process in programming courses.

METHODOLOGY

This section details the process carried out to elaborate the mapping study of the learning programming process as supported by CSCL and AI. It includes the sources of information,

the search functions, and the selection criteria used to construct the final reference corpus and the procedures for analysis and technological mapping.

This mapping study is based on tech mining, which aims to generate practical intelligence using analytical and visualization applications for data analysis (Choi *et al.*, 2012; Mohammadi, 2012; Barab *et al.*, 1997; Antonenko *et al.*, 2012). Thus, future technologies and innovations can be anticipated, ensuring long-term competitiveness and supporting decision-making processes (Jonassen, 2012; Capelo & Dias, 2009; Fadde, 2009).

This systematic review aims to find the technological wave and cutting-edge research of CSCL and AI in the programming learning process by solving four questions of interest, as shown in Table 1.

Table 1
Research questions

ID	Research questions
1	What programming languages are the most used for software development based on CSCL and AI?
2	What is the timeline of projects, tools, and technologies based on CSCL and AI?
3	How are strategies classified according to CSCL processes and software types?
4	How does the integration of CSCL and AI improve the learning process in programming courses?

Source: Authors

Protocol data

Selection of data sources. Two sources of information were used to cover research and technological information, where the following was considered: i) documents from journals and conferences published on the Internet, books, and websites from reliable sources; and ii)

data from the GitHub software repositories, which is the most popular open-source code platform among developers. GitHub contains information on more than 3,5 million software developers and more than 23 million repositories since 2008 (GitHub, 2018).

Search and data filters. Table 2 shows ten search functions specific to the Scopus, ScienceDirect, Web of Science (WoS), and GitHub data sources. The search functions included keywords taken from experts in this field, as well as from researchers and programming professors. In addition, the Table shows how many documents were obtained for each data source and the total records per query.

Table 2
Search functions, data source, and total records per query

ID	Queries	Source	Total records
1	(”cscl AND (”programming course OR introduction to programming courses”))	Scopus	23
2	(“cscl” AND “programming course”) AND (LIMIT-TO (SUBJAREA, ”COMP”) OR LIMIT-TO (SUBJAREA, ENGI”))	Scopus	16
3	(”learning to programming”)) AND (cscl) AND (programming course) AND (LIMIT-TO (SUBJAREA,” COMP”))	Scopus	20
4	(”cscl” AND ”programming course”)) AND (groups) AND (LIMIT-TO (SUBJAREA, ”COMP”) OR LIMIT-TO (SUBJAREA, SSOCI”))	ScienceDirect	13
5	((”cscl” AND ”programming course”) AND assessment))	ScienceDirect	13
6	TS=(learning programming AND cscl)	Web of Science	11
7	TI=(programming course) AND TS = computer supported collaborative learning	Web of Science	26
8	(platforms to learn to program+cscl)	GitHub	93

9	(programming course+artificial intelligence+cscl)	GitHub	145
10	(platforms to programming course+cscl)	GitHub	39

Source: Authors

The above queries resulted in 122 papers and 277 repositories, out of which 83 pieces of data were excluded based on four criteria such as research or project, incomplete information, missing name of publication, conference or journal, low number of citations, stars, copies, and forks. A PRISMA diagram showing the search and filters is shown in Figure 1. After filtering the information, the final corpus contained 316 records: 90 papers and 226 repositories. Table 3 shows the records filtered by each data source.

Table 3
Final corpus of references to carry out the mapping study

Equation ID	Source	Excluded	Final score	Total records
1	Scopus	4	19	48
2	Scopus	3	13	
3	Scopus	4	16	
4	ScienceDirect	6	7	17
5	ScienceDirect	1	10	
6	Web of Science	10	16	25
7	Web of Science	2	9	
8	GitHub	14	79	224
9	GitHub	20	125	
10	GitHub	19	20	

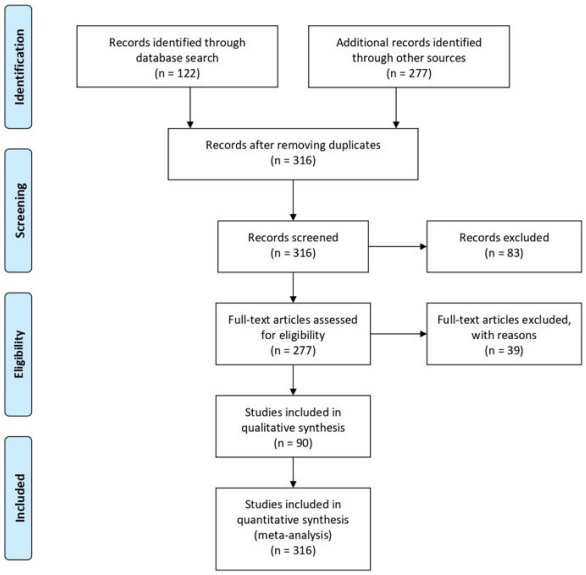
Source: Authors

As seen in Figure 1, the implementation of PRIMA is divided into four sections, as follows:

- *Identification*: this allows to identify how many records have been obtained from the queries of the different data sources.
- *Screening*: all the information is collected, forming a data corpus.
- *Eligibility*: according to the exclusion criteria mentioned above, the data are filtered to obtain a final corpus. In this section, the records that are not qualitative are also identified.
- *The item included*: a dataset with structured information is obtained which can be processed for analysis.

Figure 1

Flow diagram for a review of CSCL and AI in programming learning through repositories and scientific documents



Source: Authors

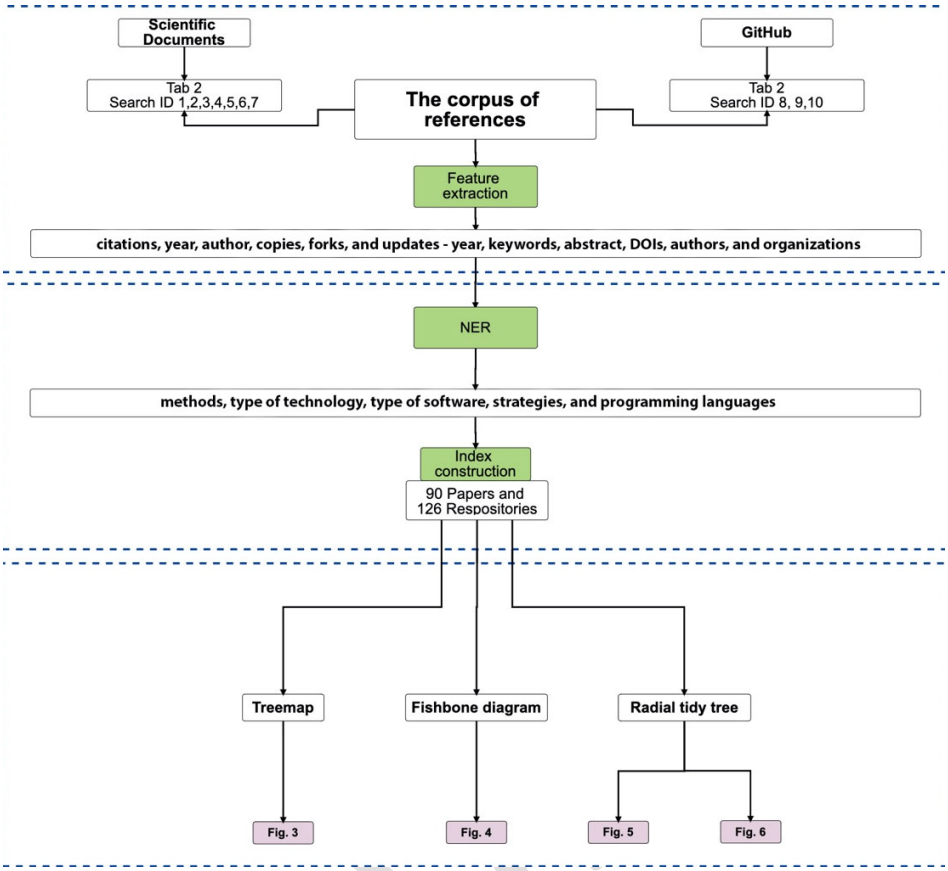
Technological maps and visualizations. The systematic review of the mapping was supported by technical maps generated semi-automatically through tools and computational techniques. Technological maps were constructed based on information from the reference corpus. The systematic mapping study is represented by four technological maps that are displayed using the following visualizations: treemap (Google Inc., n.d.), Radial Tidy Tree (vega, n.d.), and fishbone (D3, n.d.).

Figure 2 shows the workflow of this study, which involved three processes. The first one, the corpus of references, contains information from repositories and scientific documents. On the one hand, the features of the repositories were extracted: citations, year, author, copies, forks, stars, and updates. On the other hand, the features of the scientific documents were extracted: year, keywords, abstract, DOI, authors, and organizations. In the second process,

the extracted features were processed by a NER (name entity recognition), a technique of natural language processing (NLP) that extracts specific entities from the text (*e.g.*, cities, names, and others). NER-Spacy (Vasiliev, 2020) was used to extract the following features from the text fields of the corpus of references: methods, type of technology, type of software, strategies, and programming languages. The index construction task aimed to divide and organize the reference corpus information by years, types of software, and software categories. The third process involved constructing technological maps using three types of visualizations: treemap, radial tidy, and fishbone. The treemap is used to display large amounts of hierarchically structured (tree structured) data. The space in the visualization is divided into rectangles that are sized and sorted by a quantitative variable. The levels in the hierarchy are displayed as rectangles that contain other rectangles. Each set of rectangles on the same level in the hierarchy represents a column or an expression in a data table. Each individual rectangle at a level in the hierarchy represents a category in a column. The radial tidy tree is a node link tree diagram of classes in a package hierarchy, positioned in polar coordinates using Vega's tree transform. Adjusting the parameters shows layouts suitable for general trees or cluster dendrograms. Fishbone is a representation tool for categorizing the potential causes of a problem or evolution of a problem in order to identify its root causes. Typically used to identify progress analysis, a fishbone diagram combines the practice of organization charts with a mind map template.

Figure 2

Workflow for the construction of technological maps from the reference corpus



Source: Authors

Construction of technological maps

The technological map of programming languages represents the most used programming languages and the development trends in technologies for learning programming. This technological map was constructed by means of the cooccurrence process, in which the repositories that belong to the same programming language are grouped. In the process, the TAGs or labels of the NER are included in a vector. The set of vectors constitutes a matrix of programming language cooccurrences. Thus, if the label contains information about the programming language, a value of 1 is assigned to the vector; otherwise, it is 0. The cooccurrence matrix is transformed into a square matrix, and it becomes the input of the treemap. Each rectangle that represents a programming language consists of rectangles that

represent repositories. The largest rectangle at the top left corner represents the most important programming language, and the smallest rectangle at the bottom right corner represents the least important one. The orange tone represents a repository in the upper hierarchy, while a green tone represents a repository in the lower hierarchy, based on the most cited, copied, and cloned studies, as well as on forks.

The evolution timeline depicts the technological evolution of the subject under study. It presents the dynamics of the most relevant projects, of the type of technologies, and of the programming languages used in each period. To summarize, the map shows the evolution of the technologies for learning programming in one decade. The technological map is elaborated by performing the following tasks:

a) The reference corpus is organized by the number of stars, cites, and forks in the repositories, as well as by the number of citations in the papers. To this effect, the fishbone presents the most relevant projects over time.
b) The reference corpus is grouped using a data ontology that contains categories and subcategories of the Computing Classification System (CCS) (ACM, n.d.). The fishbone shows the type of technologies extracted from the CCS.
c) The cooccurrence of the programming languages is determined. For each year, the languages are grouped into three categories: more relevant, relevant, and less relevant. The relevance is associated with the number of repositories that use a programming language for a given year.
d) Finally, the extracted information is manually organized into the fishbone.

As for the technological map according to the Computing Classification System (CCS), the reference corpus was grouped using a data ontology of the Association for Computing Machinery (ACM, n.d.). The cooccurrence was determined in order to identify which

repository or paper belonged to a category of the CCS. A vector represents one reference, in which each category or subcategory appears or not. Thus, the set of vectors constitutes a matrix of CSS cooccurrences, where, if the reference contains information about the category or subcategory, a value of 1 is assigned to the vector; otherwise, it is 0. The cooccurrence matrix is transformed into a square matrix, and it becomes the input of the radial tidy tree graph. Thus, the references are shown by CCS categories and subcategories.

For the technological map of AI-based CSCL strategies, the reference corpus was grouped by CSCL strategies, methods, or algorithms. In addition, it was determined whether the reference was related to artificial intelligence. While elaborating this technological map, the references of the corpus were classified according to the following criteria:

a) Is the reference a strategy, a method, or an algorithm?
b) Does the strategy, method, or algorithm refer more than four times to the same paper or repository?
c) Is the strategy, method, or algorithm based on CSCL and AI?

If a reference meets the three criteria, it is stored in a segment of a vector. Then, its cooccurrence is determined. Thus, the vector is represented as a coincidence matrix, which is then transformed into a square matrix, which will in turn be the input for the radial tidy tree graph.

RESULTS

This study found the most used programming languages in the development of technologies and learning tools based on CSCL and AI (Figure 3), and it aided in the elaboration of a timeline of the evolution of the projects, the types of technologies used, and the most relevant programming languages from 2009 to 2019 (Figure 4). It also classified the information according to the categories and subcategories of the ACM Computing Classification System

(CCS) (Figure 5), as well as the strategies based on CSCL processes and AI techniques (Figure 6).

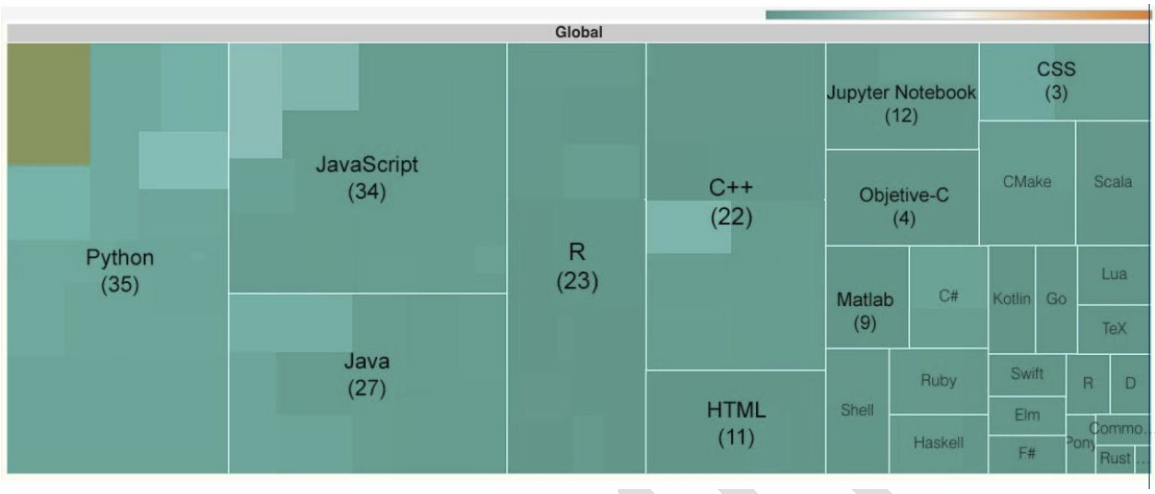
Technological map of programming languages

The review and analysis of the programming languages used in the development of tools based on CSCL and AI in programming courses aided that of the technological trends and specifications of the new technologies. From 2007 to 2021, 31 programming languages were used to develop 226 software projects to support the programming learning process. The most used languages were Python with 15,6%, JavaScript with 15,0%, and Java with 12,1%. Other programming languages complete the remaining 53%, such as Jupyter Notebook, R, C ++, Matlab, Objective-C, and Rust, among others.

In the development and research carried out to improve the learning of programming, there are issues related to the correct identification of the programming paradigm. Through a review of historical projects, these paradigms could be identified and grouped in order to provide a correct approach for new projects in this regard. In this review, the programming languages were classified by paradigm: 56% belong to the object-oriented paradigm, 27% to the functional paradigm, 9% to the logical paradigm, and 8% to the imperative paradigm. It is important to mention that, out of the 31 languages, 19,35% are multi-paradigm, such as Python, JavaScript, C++, Jupyter Notebook, Objetive-c, and Rust (Figure 3).

Figure 3

Programming languages used in the development of technologies based on CSCL and AI for improving the learning process in programming courses



Source: Authors

In the repositories, it was found that there is affinity between programming languages for a specific development. For Python, JavaScript, and Java, most of the projects found are focused on online learning platforms (nsoojin, n.d.; Leocardoso94, n.d.; Haghighatlari *et al.*, 2020) and standalone platforms such as EVEA (vieiraeduardos, n.d.), Muse (sainuguri, n.d.), and Classroom (Karanval, n.d.). R and C++ complement each other in data processing and low-level programming tasks, where the software exchanges information directly with the hardware. Other languages such as Objective-C, Matlab, and Julia are integrated with Jupyter Notebook and design languages such as CSS in order to process large amounts of data and build interactive visualizations. Languages such as Python are combined with R and Java and used for technical implementations such as machine learning, natural language processing, and deep learning. JavaScript, R, and C++ are integrated to build projects that implement techniques based on neural networks, knowledge discovery in databases, and data mining.

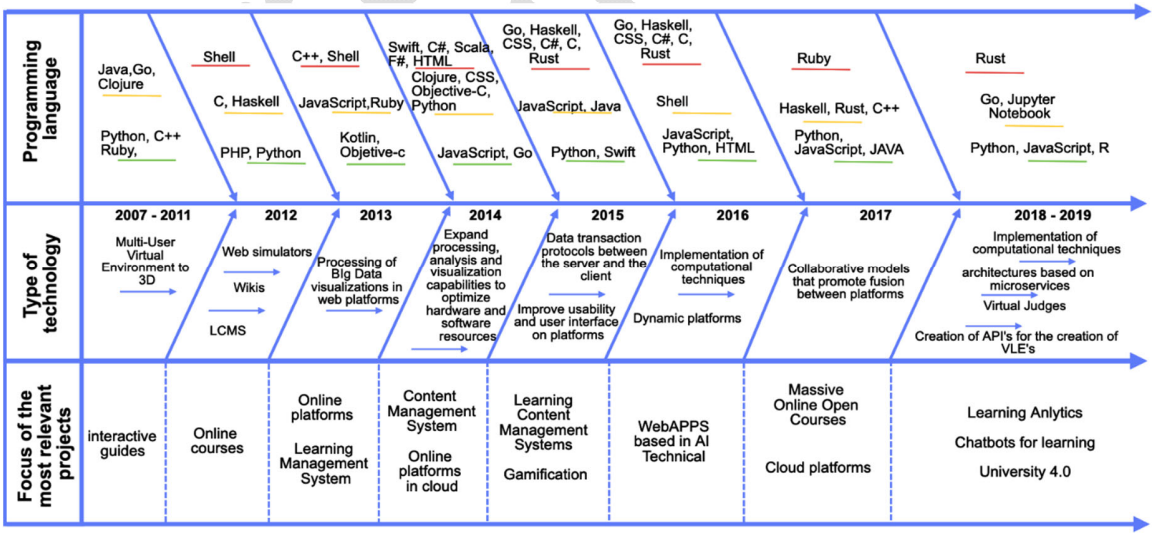
The languages were grouped according to the computational tasks in which they are most used. That is to say, in data processing: R, Python, Node, and Java; for scaling applications: R, C++, Ruby, and Python; for data visualization: JavaScript and Python with HTML and CSS; for multiple processes (threads): Python, Ruby, and Node; and for adaptive and responsive tasks: JavaScript and Python.

Evolution timeline

The technological evolution of programming learning from 2009 to 2021was studied in the corpus of repositories. For each year, the most relevant repositories, programming languages, and technologies were identified. Figure 4 describes the projects over one decade. In the first years, the projects were related to online platforms, online courses, content management systems, web apps for learning, and massive online open courses (MOOC). In the last years, the projects focused on cloud platforms and intelligent learning management systems.

Figure 4

Evolution of the best projects, types of technologies, and most relevant programming languages for improving the learning process in programming courses



Source: Authors

From 2009 to 2011, there were web application developments based on the client server architecture, where processes are executed on the machine and displayed in the web browser. The programming languages used to process data in the backend were C ++ and Ruby; to display in the frontend, Python, JavaScript, Java, Go, and Clojure were used. The most relevant projects involve interactive guides that use multi-user interaction and basic 3D environments to learn a programming language, such as ruby-kickstart (Cheek, 2019), go-book (Yuuta, 2019), html5-gamebook (J. Williams, 2019), and the Stanford Machine Learning Course (Pathrabe, 2019). Interactive guides are resources designed to reinforce student learning in a virtual environment through audio and video. Some of the strategies used were the intelligent tutoring system (Triantafillou *et al.*, 2002a) and the rule-based system (Magnisalis *et al.*, 2011).

In 2012, the improvements were made to interactive guides via games with 3D graphics, wikis, and learning assistants, where the interaction between teacher and student is supported by the computer. Programming languages such as PHP, Scala, C, and Haskell pioneered the development of platforms that support group learning in UML (unified modeling language) design problems such as jgltut (integeruser, n.d.), ElixirSchool (philss, n.d.), and ZeroBraneEduPack (pkulchenko, n.d.). These types of platforms are considered to be the basis of current projects such as Repl.it, Pastebin, and CodePen (Lafleur, 2017). On the other hand, the Learning Content Management System (LCMS) appears in Moodle (Moodle, n.d.), Chamilo (chamilo, n.d.), and Evolcampus (S.L., n.d.), which supports user management and is designed to provide educators, administrators, and learners with a single robust, secure, and integrated system to create personalized learning environments. The term *e-learning* became popular with virtual teaching methodologies. Some of the CSCL strategies created

are adaptive hypermedia systems (AHS) (Yang & Luo, 2007) and e-learning systems (Debdi *et al.*, 2015).

The high-performance computing processes used in online learning platforms forces companies and developers to build new architectures that allow optimizing resources. In 2013, exploration began on frameworks such as CakePHP (PHP5.3, n.d.), Kotlin (Kotlin, n.d.), and Django (django, n.d.) using the MVC model (model view template). This model allows improving the speed of platforms. The first platforms to integrate this model were Minerva (dmlc, 2019), LetsCode (Drupal, 2014), and Modern-JOGL (jvm, n.d.). These platforms support programming courses through the teaching of block programming. Moreover, new libraries were created to visualize large amounts of data as the first steps in the use of artificial intelligence techniques such as Machine Learning were taken (Pea *et al.*, 1999).

In 2014, the common use of the Python and JavaScript frameworks significantly improved the development of analysis tools, source code tests, tools with visualizations that allow interpreting data, and integration with visualization libraries such as bootstrap, jQuery, and CSS styles. 2014 was characterized by the development of platforms for video conferences and augmented reality projects such as tparisi (2012) and Edgarjcfn (2014). Similar projects have been developed in academies and are currently integrated into Google, Coursera, and Platzi (Coursera, 2014). In this vein, some CSCL strategies include JigSaw learning (Gutwin *et al.*, 2013; Magnisalis *et al.*, 2011) and *Predict student behavior* (Desmarais & Baker, 2012).

Between 2015 and 2016, educational platforms aided by audio and video were developed under the concept of gamification (*i.e.*, platforms that support learning through games which foster soft skills and knowledge management). The methods used in the teaching tools allow

achieving goals by means of collaboration. The main developments were based on languages such as Python, Swift, JavaScript, and Java. The platforms that stand out are Free Code Camp (Porras *et al.*, 2007), ClassroomWiki (Khandaker & Soh, 2010), and JavaStud (yrojha4ever, 2015). Other platforms, such as tpot (EpistasisLab, n.d.) and Dex (johnlee175, n.d.), use machine learning to improve students' writing style in a programming language. For these years, different uses of the strategies were reported: the Coalition Formation Algorithm (Yang & Luo, 2007) and the Team Syntegrity Model (TSM) (hnshhslsh, 2016).

Between 2017 and 2018, it became necessary for the platforms developed in different programming languages to run in any software and hardware environment. JavaScript is a pioneer in the implementation of microservices (MS) architectures, using lightweight and portable containers for applications that work in any environment. The MS-based repositories reported for these years are Judge (Aglio, 2016), an online open-source judge API to compile and execute code with given test data; and judge (hnshhslsh, 2016), GreedExCol (Debdi *et al.*, 2015), and uoj (vfleaking, 2016), which rely on containers with Mobi technology (formerly Docker) to optimize data I/O processes. The CSCL strategies for this period were based on two processes: evaluation with *Adaptive hypermedia systems* (AHS) (Triantafillou *et al.*, 2002) and feedback with *Predict student behavior* (Desmarais & Baker, 2012).

Finally, from 2019 to 2021, languages such as Python, Objetive-C, and Java were updated to execute their processes under MS-based architectures through frameworks such as Angular 6, Swift 8, and Java 9. This change caused the development of learning platforms for programming to increase. However, in this context, each participant learns individually, and the professor does not have direct interaction with the students, which is why the learning approach is not appropriate in online platforms. The CSCL approach appears with platforms that integrate practice and theory for students and professors without having to be present in

a classroom. This includes Sabo (tokers, 2016), Judger (University, 2015), and LeetCode (wty21cn, 2016) by Apple. Since 2018, platforms have focused on the specific processes of CSCL. For GF support, there are platforms such as: SunnyJudge (yudazilian, 2017) and timus (Soh *et al.*, 2005): for LD support: INGInious (luvoain, n.d.), UNcode (Restrepo-Calle *et al.*, 2018), SunnyJudge (0xyd, n.d.), and Trakla (trakla, n.d.); and, for PI support: PutongOJ and collaborative-online-judger (cqlzx, 2017). Currently, languages such as Python support CSCL processes through computational AI techniques based on prediction (Camisole 046), deep learning (RankFace) (Entropyxcy, 2017), and classification (Sun-Judge) (yudazilian, 2017). These projects, rather than being judges of a code, are able to evaluate the code of an application to know if it is optimal.

By reviewing the most relevant projects within the timeline, the projects based on CSCL and AI to improve the learning process in programming courses were identified. According to GitHub, these projects are in six software categories: software methodologies, compilation errors, software design, help with the style of the source code, artificial intelligence platforms, and virtual judges. Table 4 shows the most important projects of this review, grouped by the categories of the GitHub software, computational techniques, and CSCL processes. The relevance of a project in GitHub depends on the number of stars, copies, contributors, and forks.

Table 4

Most important projects between 2009 to 2021 grouped by the categories of the GitHub software, computational techniques, and CSCL processes

Project	Categories	ML	DL	NLP	DV	DM	CSCL process
ElixirSchool	LMS					X	GF

Minerva	Software methodologies	X			X	X	GF
CodeBuddies	Compilation errors				X	X	LD
GreedExCol	Software design	X	X		X		LD
ClassroomWiki	AI platform			X	X		GF
I-MINDS	Virtual judge	X			X	X	PI
UNCode INGInious	Virtual judge	X			X	X	PI

Source: Authors

Below are some works found in the review of the GitHub repositories. These projects apply some strategies based on CSCL (Table 4).

ElixirSchool (philss, n.d.) is the premier destination for people looking to learn and master the Elixir programming language. This platform aims for people with little knowledge to join people with a lot of experience. They have an algorithm that identifies which people should work in teams according to their abilities.

Minerva (dmlc, 2019) forecasts code and, based on this, it plans codes to keep students at an optimal level. To maintain this manually can take a long time and be surprisingly complex, so Minerva makes people with an optimal level periodically review their codes.

CodeBuddies (codebuddies, n.d.): community-organized hangouts for learning programming together.

GreedExCol (Debdi *et al.*, 2015): a CSCL system designed to support collaboration in experimental optimality results for greedy algorithms. GreedExCol supports the discussion between the members of a small group of up to four students. The methodology for working with GreedExCol is as follows: first, each student performs their individual work

(experimental research); then, the results obtained by each one are shared and discussed, so that they can propose the functions that are considered to be optimal.

ClassroomWiki (Khandaker & Soh, 2010): a collaborative web-based Wiki writing platform. For students, ClassroomWiki provides a web interface to write and review their group's Wiki, as well as and a thematic forum to discuss their ideas during collaboration. When students collaborate, ClassroomWiki tracks all student activities and builds detailed models of the students who present their contributions to their groups. ClassroomWiki is based on a multiagent framework that uses student models to form groups of students and improve collaborative learning. Through CSCL processes, ClassroomWiki supports the formation of active learning groups and rubrics.

I-MINDS (Soh *et al.*, 2008a) is a software solution for the intelligent management of classrooms or virtual groups in order to support activities both in real-time and offline, facilitating group work, adapting to the needs and background of individual users, and assisting moderation and group management. Its technology is based on intelligent software agents that autonomously interact on behalf of users. A useful tool to provide better support in CSCL processes, I-MINDS supports group formation and feedback. It does not have a code evaluator, but it does evaluate content.

UNCode (Restrepo-Calle *et al.*, 2018): this educational environment offers summative and formative comments by evaluating the functionality of computer programs. When faced with a programming problem, students must write a solution and validate it through the automatic grading tool of the educational environment. Summative feedback is offered through verdicts of the programs, which indicate whether the syntax, semantics, and efficiency of the program are correct or have some type of error. Based on these verdicts, a rating is assigned to the solutions proposed by the students. In addition, the environment also offers formative

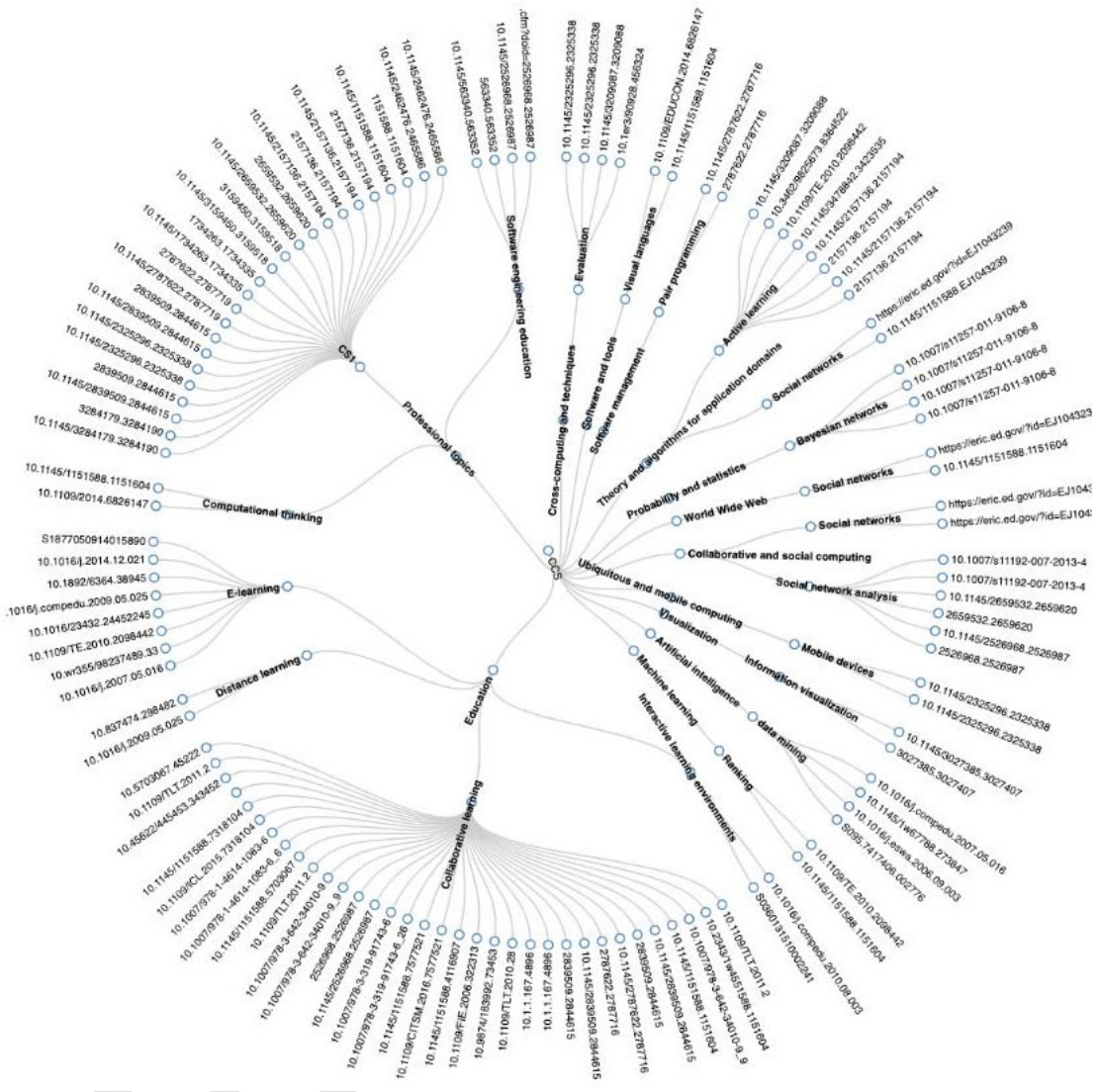
comments through different mechanisms that support the student in the improvement of the proposed programming solutions.

Technological map according to the ACM Computing Classification System (CCS)

In the classification of the corpus of references, three results were found. First, according to the categories and subcategories of the CCS (Figure 5), 20% of the repositories and papers focus on collaborative learning with multiple agents based on cooperative methods, virtual education, and virtual classes. 16% focus on data analysis through tools, social software, and network analysis. 15,3% are e-learning and b-learning systems. For the remaining percentage, the projects are divided into the formation of groups, roles, and wikis. Second, in the analysis of projects and documents based on CSCL processes with artificial intelligence support, 15% are data mining, 12% involve natural language processing, 20% use neural networks, and 7% use predictive and statistical methods. Third, the references were classified as follows: 23% are academic courses, 12% are Application Programming Interfaces (APIs), 9% are software platforms, 13% are learning games, and 36% are platforms and learning tools.

Figure 5

Corpus of references classified by the categories and subcategories of the Computing Classification System (CCS)



Source: Authors

Classification of strategies according to CSCL processes and types of software

10 strategies based on CSCL and AI were found. Each of the strategies has a specific purpose within GF, LD, and PI support. Figure 6 shows the grouping between the corpus of references and the strategies found. Each strategy is detailed below:

i) **VALCAM algorithm.** This algorithm makes use of a virtual currency (V) in the following manner. The system agent works as the provider and accountant of the

virtual currency. Every time the user agents form a coalition and perform the required task, their individual and group performances are evaluated by the system and group agents, respectively. After the evaluation, the system agent rewards each user agent's individual performance, while each group agent rewards each user agent's performance as a group member (Soh *et al.*, 2006a, 2008).

ii) **JigSaw learning.** The JigSaw technique is a method for organizing classroom activities that makes students dependent on each other to succeed. It breaks classes into groups and breaks assignments into pieces that the group assembles to complete (a jigsaw puzzle) (Gutwin *et al.*, 2013; Magnisalis *et al.*, 2011).

iii) **Adaptive educational system (AES).** An AES is mainly a system that aims to adapt some of its key functional characteristics (for example, content presentation and/or navigation support) to the learner's needs and preferences. Thus, an adaptive system operates differently for different learners, considering information accumulated in individual or group learner models (Debdi *et al*., 2015; Triantafillou *et al.*, 2002).

iv) **Intelligent tutoring system (ITS).** It aims to provide learner tailored support during the problem-solving process, as a human tutor would do. To achieve this, ITS designers apply techniques from the broader field of Artificial Intelligence (AI) and implement extensive modeling of the problem-solving process in the specific domain of application (Triantafillou *et al.*, 2002).

v) **Adaptive hypermedia systems (AHS).** It builds a user model of the goals, preferences, and knowledge of the individual user, and it uses this model to adapt the content of pages and the links between them to his/her needs. The variables

that user models include can be classified as *user-dependent*, which includes those directly related to the user and define him/her as an individual, and as *user-independent,* which affect the user indirectly and are mainly related to the context of a user's work with a hypermedia application (Yang & Luo, 2007).

vi) **Rule-based system.** It uses the Constraint-Based Modeling (CBM) approach (*i.e.*, it represents the domain knowledge as a set of constraints and a rule-based system) and offers adaptive/intelligent support by providing learners with hints during individual and group problem-solving processes, as well as feedback on peer interaction based on individual student contributions. Results show that CBM is an effective technique for both modeling and supporting students in developing collaboration skills (the participants acquired both declarative knowledge about good collaboration and did collaborate more effectively) (Magnisalis *et al.*, 2011).

vii) **Coalition Formation Algorithm.** In the initial state, all agents are mutually independent and not cooperative. Hereafter, as the agents acquire unceasingly more knowledge from the system and the environment, every agent may form some coalition on the basis of certain principles by consulting and comparing. Each coalition is considered as an independent entirety. All members in the coalition cooperate fully, so the coalition will be allowed to draw support from the abilities and resources of other members to complete tasks more efficiently than a single agent (Yang & Luo, 2007).

viii) **E-learning systems.** E-learning can be thought of as the learning process created by interaction with digitally delivered content, services, and support. It involves

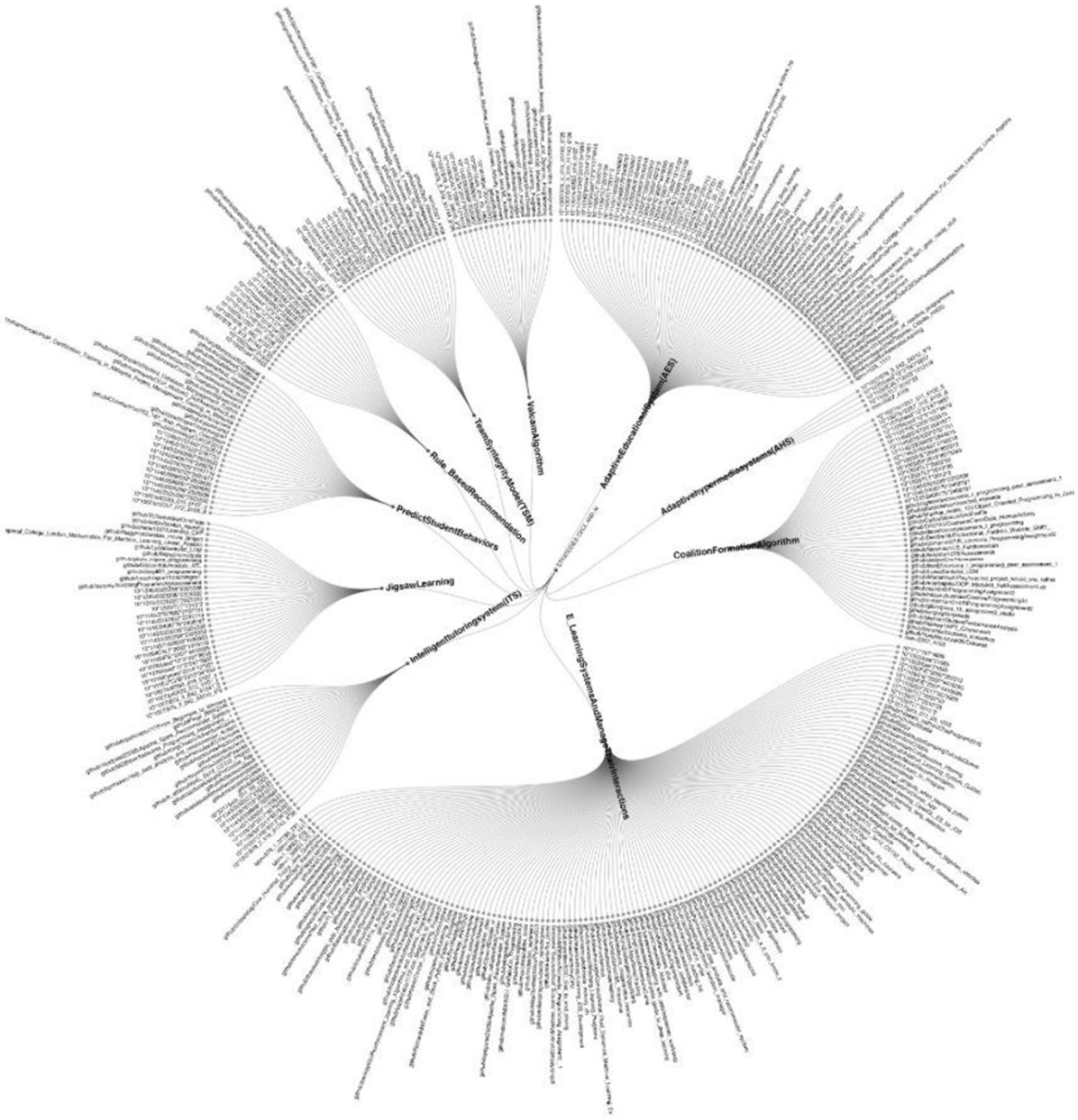
the intensive use of information and communication technologies (ICTs) to serve, facilitate, and revolutionize the learning process (Debdi *et al.*, 2015).

ix) **Predict student behavior.** A prediction model whose main objectives are automatically predicting students' performance and helping to measure and improve their goals (Abdulwahhab & Abdulwahab, 2017; Qiu *et al.*, 2016).

x) **Team Syntegrity Model (TSM)** is a new process developed by Stafford Beer to allow groups to work together in a democratic, nonhierarchical fashion in order to capture their best thinking. It is a particularly appropriate process to use when groups are characterized by high levels of diversity (Asproth *et al.*, 2011; Leonard, 2011).

Figure 6

The corpus of references classified by AI-based CSCL strategies



Source: Authors

Table 5 groups each strategy according to the type of implementation (manual or technological), the specific process of the CSCL to which it points, and the number of papers that explain its documentation or implementation.

Table 5
Final corpus of references to carry out the mapping study

Strategy	CSCL process	Tools found
Valcam algorithm	GF	8
Predict student behavior	GF	12
Team syntegrity model	GF	9
JigSaw learning	LD	14
Adaptive education system	LD	18
Intelligent tutoring system	LD	19
Adaptive hypermedia system	LD	5
Rule-based system	PI	15
Coalition formation algorithm	PI	18
E-learning systems	PI	15

Source: Authors

CONCLUSIONS

Based on the technological mining methodology, a reference corpus was constructed with 316 documents and repositories from 2009 to 2019. In the corpus analysis, this study obtained relevant information on how the integration of CSCL and AI improves the process of learning programming through four technological maps that show the programming languages and paradigms for software development, the evolution of the tools that support programming learning, and the classification of the corpus according to CCS categories, subcategories, and strategies based on CSCL and AI. In addition, the proposed methodology can support other

research works in the construction of the state of the art. However, it is necessary to improve the process of semiautomatic construction of technological maps, as well as to implement new techniques for data extraction, storage, and analysis.

The results of this research help to understand the benefits and challenges of using the CSCL and AI approach in the programming learning process. Some strategies and tools have been identified, indicating that there are multiple and varied ways to implement this approach, such as the implementation of CSCL strategies used to form groups, to evaluate, and to provide feedback, as well as the use of AI computational techniques to control the process and measure student results using virtual judges for automatic code assessment, profile identification, code analysis, teacher simulation, active learning activities, and interactive environments, among others. However, there are still open research questions.

Regarding the GF support process, strategies are currently being implemented to group students by numerical categories, abilities and skills, student profile, and weighted maxima and minima. Moreover, these strategies have been supported by AI to automatically identify and group using Machine Learning methods (Thomas *et al.*, 2002; Bennedsen *et al.*, 2008; Wiggins *et al.*, 2015; Soh *et al.*, 2006a, 2006b; Costa *et al.*, 2017) and probability and statistical methods (Salcedo & Idrobo, 2011; Hazzan & Dubinsky, 2003). However, this study did not find a CSCL strategy or AI-supported tool that groups computer programming students. Thus, the following question arises: How should CSCL and AI strategies be implemented to form groups in programming courses?

As for the LD support process, automatic code assessment appears in the form of virtual judges that are used in programming competitions, encouraging their integration into academic programming courses. However, virtual judges implemented as a method of teaching programming are not enough to foster logic and abstraction skills. Therefore,

different learning platforms have appeared, such as I-MIND (Khandaker *et al.*, 2006), UNCode (Restrepo-Calle *et al.*, 2018), and INGInious (luvoain, n.d.), among others, which allow automatically assessing code, controlling the learning process, and obtaining an analysis of the results. However, this tool falls short in the evaluation of syntactic and semantic code, code style, multiparadigm compilers, and plagiarism identification. In this paper, some strategies supported by AI which could improve the code assessment process, but there are still questions to be answered: What would be the best method for automatic code assessment? How can students achieve programming proficiency through automatic code assessment?

In the case of PI support, programming content feedback is a difficult question to answer, since each student has a different programming style, but there are currently tools such as GreedExCol (Debdi *et al.*, 2015) and ClassroomWiki (Khandaker & Soh, 2010), and I-MINDS (Soh *et al.*, 2008a) which identify the syntactic structure tree of a programming language in order to compare it with the student's code. This form of feedback has proven to work well. However, when evaluating by competencies or with a numerical system, this feedback is not so effective, as it needs to be more precise. AI and the CSCL could improve this process, helping to solve questions such as: How does AI improve the process of providing feedback from the source code? How could CSCL strategies be implemented in programming learning tools?

In the development and documentation of tools that support the learning of programming, there is still work to be done. On the one hand, there are no tools that implement all CSCL processes. With respect to the use of AI techniques, out of the tools found, 7% implement AI techniques, but only 1% is documented. On the other hand, there is no model that integrates the CSCL approach with AI techniques, thus allowing to implement learning activities and

to observe and analyze the evolution of the system and how its users (students) improve their skills. In addition, the different tools found in this paper could be explored by professors and institutions, or new technologies could be developed from them.

In further studies, new alternatives will be explored to improve the process of learning programming, including aspects related to the implementation of tools, statistical methods, and learning analyses based on CSCL and AI that enrich and allow monitoring students' training process, improving good programming practices, soft skills, and fostering greater collaboration and individual and group abstraction.

REFERENCES

Abirami, A. M., & Kiruthiga, P. (2018). Collaborative learning tools for data structures. *Journal of Engineering Education Transformations*, *31*(3), 79-83. https://doi.org/10.16920/jeet/2018/v31i3/120763

Abdulwahhab, R. S., & Abdulwahab, S. S. (2017). *Integrating learning analytics to predict student performance behavior* [Conference presentation]. 2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA), Muscat (pp. 1–6). IEEE. https://doi.org/10.1109/ICTA.2017.8336060

ACM (n.d.). *CCS 2012.* https://dl.acm.org/ccs/ccs.cfm

Aglio (2016). *Judge0 ap.* https://api.judge0.com/

Agredo-Delgado, V., Ruiz, P. H., Collazos, C. A., Alghazzawi, D. M., & Fardoun, H. M. (2018). Towards a framework definition to increase collaboration and achieve group cognition. In P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies: Design, Development, and Technological Innovation* (pp. 337-349). Springer. https://doi.org/10.1007/978-3-319-91743-6_26

Antonenko, P. D., Toy, S., & Niederhauser, D. S. (2012). Using cluster analysis for data mining in educational technology research. *Educational Technology Research and Development, 60*(3), 383-398. https://doi.org/10.1007/s11423-012-9235-8

Asproth, V., Nyström, C. A., Olsson, H., & Oberg, L.-M. (2011). Team syntegrity in a triple loop learning model for course development. *Issues in Information Science and Information Technology, 8*, 1-11. https://doi.org/10.28945/1400

Bandrowski, A., Brush, M., Grethe, J. S., Haendel, M. A., Kennedy, D. N., Hill, S., Hof, P. R., Martone, M. E., Pol, M., Tan, S. C., Washington, N., Zudilova-Seinstra, E., & Vasilevsky, N. (2016). The resource identification initiative: A cultural shift in publishing. *Journal of Comparative Neurology, 524*(1), 8-22. https://doi.org/10.1002/cne.23913

Barab, S. A., Bowdish, B. E., & Lawless, K. A. (1997). Hypermedia navigation: Profiles of hypermedia users. *Educational Technology Research and Development, 45*(3), 23-41. https://doi.org/10.1007/BF02299727

Bennedsen, J., Caspersen, M. E., & Kolling, M. (Eds.) (2008). *Reflections on the teaching of programming*. Springer. http://link.springer.com/10.1007/978-3-540-77934-6

Bevan, J., Werner, L., & McDowell, C. (2002). Guidelines for the use of pair programming in a freshman programming class. In IEEE (Eds.), *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEE&T 2002)* (pp. 100-107). IEEE. https://doi.org/10.1109/CSEE.2002.995202

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *International Journal of Phytoremediation, 21*(1), 7-74. https://doi.org/10.1080/0969595980050102

Blank, D., Kay, J. S., Marshall, J. B., O'Hara, K., & Russo, M. (2012). Calico: A multi-programming-language, multi-context framework designed for computer science education. In ACM (Eds.), Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (pp. 63-68). https://doi.org/10.1145/2157136.2157158

Bratitsis, T., & Demetriadis, S. (2012). Perspectives on tools for computer-supported collaborative learning. *International Journal of e-Collaboration, 8*(4), 73653. https://doi.org/10.4018/jec.2012100101

Bravo, C., Marcelino, M. J., Gomes, A., Esteves, M., & Mendes, A. J. (2005). Integrating educational tools for collaborative. *Journal of Universal Computer Science, 11*(9), 1505-1517. https://lib.jucs.org/article/28475/

Burch, C. (2009). Jigsaw, a programming environment for java in CS1. *Journal of Computing Sciences in Colleges, 24*(5), 37-43. https://dl.acm.org/doi/10.5555/1516595.1516604

Capelo, C., & Dias, J. F. (2009). A feedback learning and mental models perspective on strategic decision making. *Educational Technology Research and Development, 57*(5), 629-644. https://doi.org/10.1007/s11423-009-9123-z

Casamayor, A., Amandi, A., & Campo, M. (2009). Intelligent assistance for teachers in collaborative e-learning environments. *Computers and Education, 53*(4), 1147-1154. https://doi.org/10.1016/j.compedu.2009.05.025

chamilo (n.d.). *chamilo-lms: Chamilo is a learning management system focused on ease of use and accessibility*. https://github.com/chamilo/chamilo-lms

Cheek, J. (2019). *JoshCheek/ruby-kickstart*. https://github.com/JoshCheek/ruby-kickstart

Choi, S., Park, H., Kang, D., Lee, J. Y., & Kim, K. (2012). An SAO-based text mining approach to building a technology tree for technology planning. *Expert Systems with Applications, 39*(13), 11443-11455. https://doi.org/10.1016/j.eswa.2012.04.014

codebuddies (n.d.). *codebuddies/codebuddies: CodeBuddies.org: Community-organized hangouts for learning programming together – community-built using MeteorJS.* https://github.com/codebuddies/

Costa, E. B., Fonseca, B., Santana, M. A., de Araújo, F. F., & Rego, J. (2017). Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior, 73*(C), 247-256. https://doi.org/10.1016/j.chb.2017.01.047

Costaguta, R., & de los Angeles Menini, M. (2014). An assistant agent for group formation in CSCL based on student learning styles. In ACM (Eds.), *EATIS '14: Proceedings of the 7th Euro American Conference on Telematics and Information Systems* (art. 24). ACM. https://doi.org/10.1145/2590651.2590674

Coursera (2014). *An introduction to interactive programming in Python (part 1).* https://www.coursera.org/learn/interactive-python-1

cqlzx (2017). *Collaborative online judger.* https://github.com/cqlzx/collaborative-online-judger

D3 (n.d.). *d3 fishbone.* http://bl.ocks.org/bollwyvl/9239214

Damasevicius, R. (2009). Analysis of academic results for informatics course improvement using association rule mining. In G. A Papadopoulos, W. Wojtkowski, G. Wojtkowski, S. Wrycza, & J. Zupancic (Eds.), *Information Systems Development* (pp. 357–363). Springer. https://doi.org/10.1007/b137171_37

Debdi, O., Paredes-Velasco, M., & Velázquez-Iturbide, J. A. (2015). GreedExCol, A CSCL tool for experimenting with greedy algorithms. *Computer Applications in Engineering Education, 23*(5), 790-804. https://doi.org/10.1002/cae.21655

Desmarais, M. C., & Baker, R. S. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction, 22*, 9-38. https://doi.org/10.1007/s11257-011-9106-8

django (n.d.). *The Web framework for perfectionists with deadlines*. https://www.djangoproject.com/

dmlc (2019, November). *minerva. Distributed (Deep) Machine Learning Community.* https://github.com/dmlc/minerva

Docq, F., & Daele, A. (2001). *Uses of ICT tools for CSCL: How do students make as their's own the designed environment?* https://dial.uclouvain.be/pr/boreal/object/boreal:75948

Drupal (2014). *letscode.* http://www.letscode.com/

Echeverría, L., Cobos, R., Machuca, L., & Claros, I. (2017). Using collaborative learning scenarios to teach programming to non-CS majors. *Computer Applications in Engineering Education, 25*(5), 719-731. https://doi.org/10.1002/cae.21832

Edgarjcfn (2014). *Weblet importer.* http://edgarjcfn.github.io/pylearn/#level01

Entropy-xcy. (2017). Rankface. https://github.com/Entropy-xcy/RankFace

EpistasisLab (n.d.). *tpot: A Python automated Machine Learning tool that optimizes machine learning pipelines using genetic programming*. https://github.com/EpistasisLab/tpot

Fadde, P. J. (2009). Instructional design for advanced learners: Training recognition skills to hasten expertise. *Educational Technology Research and Development, 57*(3), 359-376. https://doi.org/10.1007/s11423-007-9046-5

Figueiredo, J., & García-Peñalvo, F. J. (2018). Building skills in introductory programming. In F. J. García-Peñalvo (Ed.) *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality – TEEM'18* (pp. 46-50). ACM. https://doi.org/10.1145/3284179.3284190

GitHub (2018). *GitHub Octoverse.* https://octoverse.github.com/

Google Inc. (n.d.). Treemaps | Charts. Retrieved 2019-10-17, from https://developers.google.com/chart/interactive/docs/gallery/treemap

Gutwin, C., Ochoa, S. F., Vassileva, J., & Inoue, T. (Eds.). (2013). *Collaboration and technology* (vol. 8224). Springer. http://link.springer.com/10.1007/978-3-319-63874-4

Haghighatlari, M., Vishwakarma, G., Altarawy, D., Subramanian, R., Kota, B. U., Sonpal, A., & Hachmann, J. (2020). ChemML: A machine learning and informatics program package for the analysis, mining, and modeling of chemical and materials data. *WIREs, Computational Molecular Science, 10*(4), e1458. https://doi.org/10.1002/wcms.1458

Hazzan, O., & Dubinsky, Y. (2003). Teaching a software development methodology: The case of extreme programming. In IEEE (Eds.), *Proceedings 16th Conference on Software Engineering Education and Training, 2003. (CSEE&T 2003)* (pp. 176-184). IEEE. https://doi.org/10.1109/CSEE.2003.1191375

hnshhslsh (2016). *virtual-judge.* https://github.com/hnshhslsh/virtual-judge

integeruser (n.d.). *jgltut: Learning modern 3D graphics programming with LWJGL 3 and JOML.* https://github.com/integeruser/jgltut

johnlee175 (n.d.). *dex.* https://github.com/johnlee175/dex

Jonassen, D. H. (2012). Designing for decision making. *Educational Technology Research and Development, 60*(2), 341-359. https://doi.org/10.1007/s11423-011-9230-5

jvm (n.d.). *modern-jogl-examples*. https://github.com/jvm-graphics-labs/modern-jogl-examples

Karanval (n.d.). *EVEA: Virtual environment for teaching and learning.* https://github.com/Karanval/EVEA

Khandaker, N., Soh, L.-K., & Jiang, H. (2006). *Student learning and team formation in a structured CSCL environment.* https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.2501&rep=rep1&type=pdf

Khandaker, N., & Soh, L.-K. (2010). ClassroomWiki: A collaborative Wiki for instructional use with multiagent group formation. *IEEE Transactions on Learning Technologies, 3*(3), 190-202. https://doi.org/10.1109/TLT.2009.50

Kotlin (n.d.). *Kotlin programming language*. https://kotlinlang.org/

Kozma, R. (2000). Reflections on the state of educational technology research and development. *Educational Technology Research and Development, 48*(1), 5-15. https://doi.org/10.1007/BF02313481

Lafleur, J. (2017). *How to share code and make it shine*. codeburst. https://codeburst.io/how-to-share-code-and-make-it-shine-f5ffcea1794f

Leocardoso94 (n.d.). *Free-Courses: A collection of free courses about programming.* https://github.com/Leocardoso94/Free-Courses

Leonard, A. (2011). Team syntegrity: A new methodology for group work. *European Management Journal, 14*(4), 407-413. https://doi.org/10.1016/0263-2373(96)00028-X

Loizzo, J., & Ertmer, P. A. (2016). MOOCocracy: The learning culture of massive open online courses. *Educational Technology Research and Development, 64*(6), 1013-1032. https://doi.org/10.1007/s11423-016-9444-7

luvoain (n.d.). *Installation and deployment — INGInious 0.5.dev0 documentation*. https://docs.inginious.org/en/v0.5/install_doc/installation.html

Magnisalis, I., Demetriadis, S., & Karakostas, A. (2011). Adaptive and intelligent systems for collaborative learning support: A review of the field. *IEEE Transactions on Learning Technologies, 4*(1), 5-20. https://doi.org/10.1109/TLT.2011.2

Mansilla, P. S., Costaguta, R., & Schiaffino, S. (2014). Multi agent model for skills training of CSCL e-tutors. In ACM (Eds.), *EATIS '14: Proceedings of the 7th Euro American Conference on Telematics and Information Systems* (art. 30). ACM Press. https://doi.org/10.1145/2590651.2590680

Mohammadi, E. (2012). Knowledge mapping of the Iranian nanoscience and technology: A text mining approach. *Scientometrics, 92*(3), 593-608. https://doi.org/10.1007/s11192-012-0644-6

Moodle (n.d.). *Moodle - Open-source learning platform*. https://moodle.org/?lang=es

Munson, J. P., & Zitovsky, J. P. (2018). Models for early identification of struggling novice programmers. In ACM (Eds.), *SIGCSE '18: Proceedings of the 49th ACM Technical*

Symposium on Computer Science Education (pp. 699-704). ACM. https://doi.org/10.1145/3159450.3159476

nsoojin (n.d.). *coursera-ml-py: Python programming assignments for Machine Learning by Prof. Andrew Ng in Coursera*. https://github.com/nsoojin/coursera-ml-py

Pathrabe, U. A. (2019). *UtkarshPathrabe/Machine-Learning-Stanford-University-Coursera*. https://github.com/UtkarshPathrabe/Machine-Learning-Stanford-University-Coursera

Pea, R. D., Tinker, R., Linn, M., Means, B., Bransford, J., Roschelle, J., Hsi, S., Brophy, S., & Songer, N. (1999). Toward a learning technologies knowledge network. *Educational Technology Research and Development, 47*(2), 19-38. https://doi.org/10.1007/BF02299463

philss (n.d.). *Elixir School*. https://elixirschool.com/es/

PHP5.3 (n.d.). *CakePHP – Build fast, grow solid – PHP Framework – Home*. https://cakephp.org/

pkulchenko (n.d.). *ZeroBraneEduPack: A collection of simple lessons, scripts, and demos in Lua, suitable for learning programming concepts*. https://github.com/pkulchenko/ZeroBraneEduPack

Porras, J., Heikkinen, K., & Ikonen, J. (2007). Code camp: A setting for collaborative learning of programming. *Advanced Technology for Learning, 4*(1), 43-52. https://doi.org/10.2316/Journal.208.2007.1.208-0906

Qiu, J., Tang, J., Liu, T. X., Gong, J., Zhang, C., Zhang, Q., & Xue, Y. (2016). Modeling and predicting learning behavior in MOOCs. In ACM (Eds.), *WSDM '16: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (pp. 93-102). ACM Press. https://doi.org/10.1145/2835776.2835842

Rahwan, T. (2007). *Algorithms for coalition formation in multi-agent systems* [Unpublished doctoral dissertation, University of Southampton].

Restrepo-Calle, F., Ramírez-Echeverry, J. J., & González, F. A. (2018, July 2-4). *UNCODE: Interactive system for learning and automatic evaluation of computer programming skills* [Conference presentation]. 10th International Conference on Education and New Learning Technologies, Palma, Spain. https://doi.org/10.21125/edulearn.2018.1632

sainuguri (n.d.). *Muse*. https://github.com/sainuguri/Muse

Salcedo, S. L., & Idrobo, A. M. O. (2011, October 12-15). *New tools and methodologies for programming languages learning using the scribbler robot and Alice* [Conference presentation]. 2011 Frontiers in Education Conference (FIE), Rapid City, SD, USA. https://doi.org/10.1109/FIE.2011.6142923

Soh, L.-K., Khandaker, N., Liu, X., & Jiang, H. (2005). Computer-supported structured cooperative learning. In C.-K. Looi, D. Jonassen, & M. Ikeda (Eds.), *Proceedings of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation* (pp. 428-435). ACM. https://dl.acm.org/doi/10.5555/1563334.1563390

Soh, L.-K., Khandaker, N., Liu, X., & Jiang, H. (2006a). A computer-supported cooperative learning system with multiagent intelligence. In ACM (Eds), *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 1556-1563). ACM Press. https://doi.org/10.1145/1160633.1160933

Soh, L.-K., Khandaker, N., Liu, X., & Jiang, H. (2006b). Multiagent coalition formation for computer-supported cooperative learning. *IAAI'06: Proceedings of the 18th conference on Innovative applications of artificial intelligence, 2*, 1844-1851. http://dl.acm.org/citation.cfm?id=1597122.1597146

Soh, L.-K., Khandaker, N., Liu, X., & Jiang, H. (2008). I-MINDS: A multiagent system for intelligent computer- supported collaborative learning and classroom management. *International Journal of Artificial Intelligence in Education, 18*(2), 119-151. http://digitalcommons.unl.edu/csearticles/61

Solarte-Pabón, O., & Machuca-Villegas, L. (2019). Fostering motivation and improving student performance in an introductory programming course: An integrated teaching approach. *Revista EIA, 16*(31), 65. https://doi.org/10.24050/reia.v16i31.1230

Suárez, C. G. H., Guerrero, V. A. B., Calle, F. R., & Osorio, F. A. G. (2021). Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1. *Investigación e Innovación en Ingenierías, 9*(1), 50-60. https://doi.org/10.17081/invinno.9.1.4185

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. *ACM SIGCSE Bulletin, 34*(1), 33-37. https://doi.org/10.1145/563517.563352

tokers (2016). *SABO.* https://github.com/tokers/sabo

tparisi (2012). *WebVR.* https://github.com/tparisi/WebVR

trakla (n.d.). *WWW-TRAKLA.* http://www.cs.hut.fi/tred/WWW-TRAKLA/WWW-TRAKLA.html

Triantafillou, E., Pomportsis, A., & Georgiadou, E. (2002). *AES-CS: Adaptive educational system based on cognitive styles.* https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=5E362A162F2DFC505C4EB10D5E600A54?doi=10.1.1.2.5116&rep=rep1&type=pdf

University of Quebec (2015). *Onlinejudge*. https://onlinejudge.org/

van Gorp, J., & Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education, 11*(3), 247-260. https://doi.org/10.1076/csed.11.3.247.3837

Varier, D., Dumke, E. K., Abrams, L. M., Conklin, S. B., Barnes, J. S., & Hoover, N. R. (2017). Potential of one-to-one technologies in the classroom: Teachers and students weigh in. *Educational Technology Research and Development, 65*(4), 967–992. https://doi.org/10.1007/s11423-017-9509-2

Vasiliev, Y. (2020). *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press. https://www.overdrive.com/search?q=F7C72EAA-7BBD-4B45-8957-9B44182DF5B0

vega (n.d.). *Radial Tree Layout example*. https://vega.github.io/vega/examples/radial-tree-layout/

Vesin, B., Ivanović, M., Klašnja-Milićević, A., & Budimac, Z. (2011, October 14-16). *Rule-based reasoning for altering pattern navigation in programming tutoring system* [Conference presentation]. 15th International Conference on System Theory, Control and Computing, Sinaia, Romania.

vfleaking (2016). Uoj *(universal online judge)*. https://github.com/vfleaking/uoj

vieiraeduardos (n.d.). *Classroom: Virtual learning environment.* https://github.com/vieiraeduardos/classroom

Weber, G., & Brusilovsky, P. (2001). Elm-art: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education (IJAIED), 12*, 351-384. https://sites.pitt.edu/~peterb/papers/JAIEDFinal.pdf

Wiggins, J. B., Boyer, K. E., Baikadi, A., Ezen-Can, A., Grafsgaard, J. F., Ha, E. Y., Lester, J. C., Mitchell, C. M., & Wiebe, E. N. (2015). JavaTutor. In ACM (Eds.), *SIGCSE '15: Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (p. 599).ACM Press. https://doi.org/10.1145/2676723.2691877

Williams, J. (2019). *html5-game-book*. https://github.com/jwill/html5-game-book

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). 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

Yang, J., & Luo, Z. (2007). Coalition formation mechanism in multi-agent systems based on genetic algorithms. *Applied Soft Computing Journal, 7*(2), 561-568. https://doi.org/10.1016/j.asoc.2006.04.004

Yannibelli, V., & Amandi, A. (2012). A memetic algorithm for collaborative learning team formation in the context of software engineering courses. In F. Cipolla-Ficarra, K.

Veltman, D. Verber, M. Cipolla-Ficarra, & Florian Kammüller (Eds.), *Advances in New Technologies, Interactive Interfaces and Communicability* (pp. 92-103). Springer. https://doi.org/10.1007/978-3-642-34010-9_9

yrojha4ever (2015). *JavaStud.* https://github.com/yrojha4ever/JavaStud

yudazilian (2017). *Sunnyjudge.* https://github.com/yudazilian/SunnyJudge

Yuuta (2019, October). *go-book.* https://github.com/initpy/go-book

Zingaro, D., Taylor, C., Porter, L., Clancy, M., Lee, C., Nam Liao, S., & Webb, K. C. (2018). Identifying student difficulties with basic data structures. In ACM (Eds.), *ICER '18: Proceedings of the 2018 ACM Conference on International Computing Education Research* (pp. 169-177). ACM. https://doi.org/10.1145/3230977.3231005