

# Teaching Software Engineering Principles in Middle Schools by Combining Robotics and Blogging

Ilenia Fronza<sup>a</sup>, Claus Pahl<sup>b</sup> and Boris Sušanj

Free University of Bozen/Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy

**Keywords:** Software Engineering, End-user Software Engineering, Computational Thinking, K-12.

**Abstract:** In the current labor market, a large number of people engage in programming activities, even without being trained developers. Fostering Software Engineering (SE) principles at the K-12 level can increase the quality of the code that students will write in their future careers. Middle school students usually learn Computer Science (CS) as part of other disciplines; thus, the challenge is achieving the CS learning objectives and foster SE principles while fulfilling the curricular objectives. In this work, we describe a didactic module and its assessment framework; moreover, we report the results of a classroom experience that shows the effectiveness of the proposed approach. This work provides educators with a practical example of how to cover several areas of technology in a way for middle school students to be engaged and to spark future interest. The results encourage us to work on the development of the next modules dedicated to middle schools.

## 1 INTRODUCTION

The number of unqualified people (i.e., end-users) who produce software in the labor market is considerable (Burnett and Myers, 2014) and includes secretaries, accountants, teachers, or anyone else who finds themselves writing programs to support their work (Ko et al., 2011). In 2015, among 26 million U.S. online job postings, there were as many as 7 million job openings in occupations that valued coding as a technical skill (Burning Glass Technologies, 2016). In 2017, Gartner Inc. predicted that by 2022, *citizen developers* (i.e., non-professional developers who build applications for use by other people) would be building more than a third of all web/mobile employee-facing apps delivered in organizations with mature citizen development initiatives (Gartner Inc., 2017).

One of the well-known issues of end-user-produced software is its overall low quality (Burnett, 2009), which is mainly explained by the lack of Software Engineering (SE) training (Scheubrein, 2003). To address this issue, End-User Software Engineering (EUSE) provides end-users with the knowledge of basic SE principles (Barricelli et al., 2019). A limited effort has been spent so far to facilitate the acquisition of software quality notions and SE principles (Mon-

teiro et al., 2017). In particular, there is a need to focus on middle schools to reach the highest number of students before they choose their careers. This endeavor sets several challenges. For example, the goal is not turning students into professional software engineers through ad-hoc activities (Burnett and Myers, 2014). Indeed, middle school students, in general, do not perceive the usefulness of SE principles. Moreover, in many contexts (including the one considered in this work) Computer Science (CS) is taught as part of other disciplines, which hinders the reservation of time for content that does not directly contribute to achieving the curricular objectives.

Based on these considerations, we propose a didactic module (for first-year middle school) that aims at achieving the CS learning objectives (through educational robotics) and foster SE principles, while guaranteeing the achievement of the curricular objectives. The proposed activities (and assessment framework) have been designed based on the learning objectives listed in a recent proposal of a core informatics curriculum (Forlizzi et al., 2018). To evaluate the effectiveness of the proposed module, we performed a classroom experience involving 11 students. The results show the effectiveness of the proposed module and encourage us to work on the development of the next modules for middle schools.

The remaining part of the paper is organized as follows. Section 2 describes the state of the art of

<sup>a</sup>  <https://orcid.org/0000-0003-0224-2452>

<sup>b</sup>  <https://orcid.org/0000-0002-9049-212X>

EUSE in K-12; Section 3 describes the rationale of the proposed module, and Section 4 details its structure and assessment framework. Section 5 describes the classroom experience, and Section 6 reports its results. Section 7 discusses our results and draws conclusions from this work, also proposing possible directions for future work.

## 2 STATE OF THE ART

In the last two decades, researchers proposed several approaches to support end-users in modifying or creating software artifacts to solve their professional or personal problems (Barricelli et al., 2019). In particular, the fields of End-User Programming (EUP), End-User Development (EUD), and End-User Software Engineering (EUSE) have been very active in this direction (Barricelli et al., 2019). In particular, EUSE focuses on systematic and disciplined activities to improve the quality of end-user-produced code (Barricelli et al., 2019; Burnett and Myers, 2014).

The large number of activities fostering coding and computational thinking at different levels of education contribute to increasing the number of end-users. Some students will pursue a CS career; many of them will write some code to support their work. Thus, there is a need to reach all the students before they choose their careers. Indeed, in 2000, M. Shaw indicated the need for treating software development from an engineering point of view to all students who learn software development, and not only to prospective software engineers (Shaw, 2000). Bollin et al. indicated SE as a valuable means to start exercising, at the K-12 level, a set of skills that are valuable in the labor market (Bollin et al., 2016). Rico and Sayani recommended introducing Agile methods as early as possible (Rico and Sayani, 2009). Few EUSE studies focus on K-12, where Agile methods are an excellent candidate (Fronza et al., 2019a). The existing studies focused on proposing a mentoring methodology on Agile (Meerbaum-Salant and Hazzan, 2010), achieving greater flexibility in development projects using Agile (Kastl et al., 2016), and understanding how to foster Agile in different contexts, including non-vocational (Fronza and Zanon, 2015; Fronza et al., 2016) and middle schools (Fronza et al., 2017). With this vision at hand, we explored further the possibility of fostering Agile principles in middle schools, while pursuing CS and curricular learning objectives. Following the EUSE guidelines (Burnett and Myers, 2014), we avoid introducing additional lectures on the topic.

## 3 RATIONALE

In the last years, a range of activities has been proposed to foster programming skills at different levels of education (Bocconi et al., 2016). Nevertheless, limited effort has been spent to explore the possibility of facilitating the acquisition of software quality notions and Software Engineering (SE) principles (Monteiro et al., 2017; Fronza et al., 2017).

In this work, we focus on middle schools to reach the highest number of students before they choose their future careers. Middle school students, including those considered in this work, usually learn CS as part of other disciplines. Therefore, the challenge is achieving the CS learning objectives and foster SE principles while also fulfilling the curricular learning objectives. Our long-term goal is providing schools with a set of progressive modules for this purpose. Our Research Questions are:

**RQ1:** Is it possible, in first-year middle school, to achieve CS learning objectives and foster SE principles, while fulfilling the existing curricular learning objectives?

**RQ2:** Is this approach effective?

We describe a first-year module that we have designed for this purpose. The module comprises a *robotics project* combined with an existing curricular activity, which is *blogging*. Together with a group of school teachers, we selected blogging to achieve the following curricular objectives: 1) create simple websites or blogs, 2) access the Internet with confidence, and 3) respect the copyright rules. Moreover, blogging is transversal to many disciplines (e.g., Languages); thus, it provides an example to students of their potential to use technology in any professional path of their choice. Blogging also taps into students' interests and engages them to learn the technical part, even if they are not particularly interested in it (Spires et al., 2012). Finally, this choice allows us to promote a competent/responsible use of devices through a Bring Your Own Device (BYOD) strategy by using smartphones to collect material for the blog.

We aim at providing students with the first experience of the entire SE process, from initial customer inception to product release. Therefore, our module consists in a *didactic transposition* (i.e., the process by which professional content (scientist, scholar, or expert knowledge) becomes school content, by adapting and shaping the professional knowledge to fit the school environment (Chevallard and Gilman, 1991; Hazzan et al., 2010)) of eXtreme Programming (XP). This approach fits the EUSE goal, which is fostering SE principles without turning students into professional software engineers (Burnett, 2009).

When teaching SE to undergraduates, it is essential to apply suitable practices according to students' goals instead of fulfilling the complete implementation of methods (Schneider and Johnston, 2005). Since teaching to even younger students, we followed this advice and selected a set of XP practices, which correspond to a set of Agile principles and are recommended to be adopted together (Fronza et al., 2019a; Kastl et al., 2016):

- *incremental development*, through small releases, frequent testing, and user stories;
- *customer involvement*, by having the instructor playing this role;
- *change*, through regular system releases, test-first, refactoring, and continuous integration.

We limited the difficulty of the programming task to let students focus more on the SE process (Fronza and Pahl, 2019). The triviality of the programming task supports our goal of starting a progressive set of activities, during which the didactic transposition of the Agile methods becomes closer and closer to the professional setting, also by increasing the difficulty of the programming task.

## 4 STRUCTURE OF THE MODULE

The proposed didactic module consists of the following four parts and covers a total of 20 hours, spread over ten weeks by having two hours per week:

1. introduction and team creation (4 hours);
2. blogs: introduction, Wordpress tutorial, first posts (6 hours);
3. unplugged introduction to robotics (4 hours);
4. robotics project (6 hours).

Module and assessment framework have been designed based on the learning objectives listed in a recent proposal of a core informatics curriculum (Forlizzi et al., 2018). Since we aimed at creating a progression of activities for middle schools, for the first year we selected from this proposal a subset of objectives of four areas (Table 1). Moreover, based on our research questions, we defined the *Area of SE*.

For assessment, under consideration of the underlying principles of Project-Based Learning (PBL) (Romeike and Göttel, 2012), we did not hand out tests, and we preferred *critique and revision*, supported by observation, interview, and code inspections (Fronza et al., 2017).

### 4.1 Introduction and Team Creation

We distribute a questionnaire to collect data about participants' backgrounds and habits; then, we open a discussion on the topic "what is a robot?", by showing pictures of different robots (e.g., humanoid and grass-cutter). Afterward, teams are formed. Working in teams is an essential task to master for software engineers (Zucconi, 1995), as it allows to exchange experiences and brings powerful opportunities for personal development. In particular, Agile teaches us that good products come from good teams (Martin, 2003). However, being part of an ineffective team may lead to extreme frustration and resentment (Oakley et al., 2004). Thus, we take the following steps to facilitate team building (Oakley et al., 2004): 1) instructors form teams, based on their knowledge of students' characteristics, trying to form teams whose members are diverse in ability level; 2) teams agree on a team name and logo.

### 4.2 Blogs: Introduction, Wordpress Tutorial, First Posts

We provide basic notions on the following topics:

- Basic architectural and functional concepts of the Internet and the Web.
- Basic architectural and functional concepts of computer-based systems and devices distinguishing between hardware and software.
- Copyright issues when publishing online, with Google Image Search (<https://images.google.com>) as an example to retrieve Creative Commons-licensed photos.
- Internet safety and the importance of protecting a blog by using a password and avoiding the publication of personal information.

Afterward, each team finds a blog online and describes it to the others. Each presentation is used to find some common characteristics of blogs. Finally, we provide a tutorial on Wordpress (<https://en.wordpress.com>). Table 2 details the assessment framework.

### 4.3 Unplugged Introduction to Robotics

During this part, participants work on unplugged activities, which get documented in the blog. Table 3 shows the assessment framework.

**Coding Game** (<http://bit.ly/2XqiQT1>). Four groups compete in a board game to reach specific points of the board, which requires to decide shared conventions for directions.

Table 1: Learning Objectives (Forlizzi et al., 2018) and Parts of the Module in Which They Are Pursued.

<b>Area of Algorithms</b>				
id	Knowledge and skills	P2	P3	P4
A1	To detect the potential ambiguities hidden in the description of an algorithm when natural language is used		✓	
A2	To describe an algorithm according to the capabilities of the automatic executors		✓	✓
A3	To write algorithms, even based on conventional notations, to describe simple processes inspired from the everyday life		✓	
A4	To detect/describe the conditions under which a process may terminate		✓	✓
<b>Area of Programming</b>				
P1	To try small/simple changes in a program to understand and modify its behavior, identify and fix its flaws			✓
P2	To write programs that use selections			✓
P3	To trace the progress of computation			✓
<b>Area of Digital Awareness</b>				
DA1	To understand the main architectural/functional concepts of the Internet and the Web	✓		
DA2	To understand the main architectural/functional concepts of computer-based systems and devices, distinguishing between hardware and software	✓		
DA3	To connect computer-based devices with each other and with peripheral devices	✓		✓
DA4	To recognize the value of personal data (not only sensitive data) and be aware of issues related to identity on the network	✓		✓
<b>Area of Digital Creativity</b>				
DC1	To experiment during the creation of digital content various digital tools and multiple processing methods, so as to express themselves at their best		✓	✓
DC2	To choose the most appropriate digital tools for their expressive goals		✓	✓
DC3	To select and organize digital content for an effective presentation		✓	✓
<b>Area of Software Engineering</b>				
SE1	To develop the solution in smaller portions at a time (i.e., incremental development)			✓
SE2	To take advantage of a customer's feedback (i.e., customer involvement)			✓
SE3	To be able to manage changes to requirements (i.e., change)		✓	✓

Table 2: Assessment in Part 2 (Ids Are Defined in Table 1).

id	Assessment criteria
DA1	Ability to summarize in the blog the main architectural and functional concepts of the Internet and the Web
DA2	Ability to summarize in the blog the main architectural/functional concepts of computer-based systems and devices distinguishing hardware and software
DA3	Ability to connect a robot/mobile phone via USB to the PC
DA4	Ability to ask relevant questions while using the Web, and basic awareness of some risks related to it

**Tell Me How You Make Toast** (<http://bit.ly/2V9kKK7>). Each student sketches a diagram of how to make toast, one step per post-it. Then, the solutions are combined in one single solution. The take-away message is about the importance of working together and identifying small steps to solve a problem.

#### 4.4 Robotics Project

After providing each team with a robot and a laptop, we introduce the programming environment (Studuino: <https://www.artec-kk.co.jp/studuino/>). Afterward, under consideration of the PBL principles, the same challenge is given to all the teams: *build a robot that can draw the boundary lines of a football field*. We chose this challenge for two main reasons: 1) geometric figures are a topic of first-year middle school; and 2) it does not require to use lights, sound, or sensors, which keeps the programming task to a basic level by avoiding conditions and input management. The modularity of the robot supports this choice, i.e., participants build an elementary robot that does not include sensors.

Teams write on a post-it each step to achieve the goal, and solve each task (e.g., “let the robot move forward”) during one *iteration*; when one step is complete, each team can show the *current version* (i.e., small releases) to the *customer* to get feedback. Teams are encouraged to frequently test and complete

Table 3: Assessment in Part 3 (Ids Are Defined in Table 1).

id	Assessment criteria
A1	Ability to define/follow a common set of conventions during the <i>coding game</i> , and to ask for clarification when some instructions are ambiguous
A2	Ability to adapt the description of an algorithm depending on the executor (i.e., peers during the <i>coding game</i> and the <i>Tell Me How You Make Toast</i> activity)
A3	Ability to define an algorithm to solve a real life example (i.e., preparing a toast)
A4	To describe the conditions to terminate the processes of the coding game and of the preparation of a toast
DC1	Ability to add effects to pictures when needed, and to write texts for the blog using various fonts and colors
DC2	Ability to identify the correct tool for each activity (e.g., writing, coding, etc.)
DC3	Overall quality of the blog
SE3	Ability to integrate changes to the solution

a task before starting the next one. At the beginning (and end) of each meeting, teams have a 5-minute stand-up meeting to recap: what have we done so far? What are we going to do today (next time)? Besides supporting the SE process (Section 3), these meetings serve to collect data for the blog, which is also done during the iterations by using smartphones (i.e., pictures, videos, short notes). Table 4 illustrates the assessment framework.

## 5 CLASSROOM EXPERIENCE

We performed a classroom experience involving 11 middle school students (7 M and 4 F); German was the mother-tongue of four students, and Italian of the other seven. The school teacher formed three teams (two teams of four and one of three students) and one/two German mother-tongue speakers were assigned to each team. The activities could be documented either in German or Italian, to make this module transversal to Languages.

Ten students (7 M and 3 F) completed the initial questionnaire. According to their answers, nine of them own both a computer and a smartphone, and seven use computers more frequently, mostly for internet browsing (eight respondents), writing text and preparing presentations (six), and listening to music (four). Smartphones are mostly used for listening to music (ten), internet browsing (ten), playing (eight), social networks/chats (seven). In most of the cases

Table 4: Assessment in Part 4 (Ids Are Defined in Table 1).

id	Assessment criteria
A2	Ability to adapt the description of an algorithm depending on the executor (i.e., the robot)
A4	Understand that the robot needs to stop when the entire square has been drawn
P1	Ability to describe the effect of small changes while programming the robot
P2	Ability to explain the presence of selection in the developed code
P3	Ability to tell what part of the code is being executed by the robot
DA3	Ability to connect a robot/mobile phone via USB to the PC
DA4	Ability to ask relevant questions while using the Web, and basic awareness of some risks related to it
DC1	Ability to add effects to pictures when needed, and to write texts for the blog using various fonts and colors
DC2	Ability to identify the correct tool for each activity (e.g., writing, coding, etc.)
DC3	Overall quality of the blog
SE1	Ability to organize small releases, based on user stories, and to test frequently to obtain a working prototype
SE2	Ability to listen to a customer’s feedback and use it during the following iteration
SE3	Ability to integrate changes to the solution

(seven), parents decide how long per day students can use a computer, and check what they are doing (seven); six students also enjoy showing to their parents what they have been doing. Parents represent the primary source of help (nine), but only six students enjoy using a computer together with their parents, usually for internet browsing. Four students defined a robot as “a *non-living-being* created by technology”; one student even defined it as “a *person* who could help with activities and research”. This strong association of robots with humanoids is also evident when students draw a robot: all the drawings represent humanoids (Figure 1 shows an example). When asked, “what do you think you should do to create a robot?”, only two participants mentioned *programming*, while all the others focused on the *building* aspect.

## 6 RESULTS

Figure 2 summarizes the assessment results, which show the effectiveness of the proposed approach (RQ2).

**Area of Algorithms.** The assessment of this area takes place during Part 3 and Part 4. During the *Tell Me How You Make Toast* exercise, one of the three teams provided a precise solution (objective A3 in Table 1) immediately because they were already aware of the need of being precise without taking some steps for granted (A2), such as “plug the toaster into a power outlet”. The other two teams presented correct (A3) but less precise solutions; however, they actively participated in the discussion during the review of the results and, in the end, demonstrated a satisfactory comprehension of the principles behind it (A2). In Part 4, all the teams understood that the algorithm needed to be even more precise than in Part 2, since a robot was going to execute it (A2). The message was that, for example, we could avoid telling a person to cut the bread to prepare a toast, while a robot can not guess omitted instructions.

As expected, the first part of the *Coding Game* generated misunderstandings due to the lack of clear conventions. Thus, the teams dedicated a large part of the game to successfully decide a shared set of conventions (A1) including, for example, the point of view taken for each instruction: when executing an arrow to the right, do we consider the right side of the pawn or of who is moving it? After finding an agreement on a set of shared rules, all the teams completed the game correctly (A2, A3), and all the participants were actively engaged in the activity. All the participants could identify end-conditions (A4) both during unplugged exercises (e.g., the toast is ready) and during the robotics project (i.e., the square is complete). Therefore, we achieved two-thirds of our objectives.

**Area of Programming.** All the teams delivered a simple yet working solution. The code is slightly different for the three teams: one team created a sequence of instructions; the second team found the *repeat* block and, after some explanations, refactored



Figure 1: One Answer to the Question “draw a Robot”.

the solution using a *repeat four times*. The third team also refactored the solution, but just inserted a *repeat three times* after drawing the first side of the square. None of the solutions implement a *selection* (P2). Favored also by the linearity of the solution, all the students could explain the robot behavior expected after a change in the code (P1). Moreover, they could tell what part of the code was executed by the robot (P3). Therefore, we achieved half of our objectives.

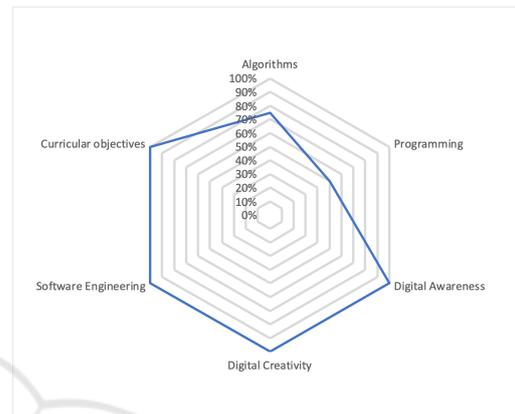


Figure 2: Summary of the Assessment Results.

**Area of Digital Awareness.** While populating the blog, all the participants demonstrated an excellent ability to connect different devices (DA3), for example, to transfer pictures. During Part 2, all the participants showed a good understanding of the provided notions about the Web, the Internet, and PC hardware, but summarizing these notions in a blog post was challenging for most of them. However, after an additional explanation, the participants could successfully write their posts (DA1, DA2). All the participants were already conscious of some internet security issues. Therefore, we introduced the concept of privacy and data tracking, and a short explanation of how e-mails and URLs work. During a group discussion to review these concepts, all the participants demonstrated a good yet basic understanding (DA4).

**Area of Digital Creativity.** All the participants could easily choose the software they needed to complete different tasks (DC2), such as writing posts and modifying pictures. Even though the blog structure imposed some restrictions, the final result shows a good level of creativity, especially in the choice of fonts, colors, and size (DC1). The presentation in the blog can be considered adequate, and the quality is acceptable (DC3) when considering that this was the very first blogging experience for all the participants.

**Area of Software Engineering.** All the participants got a first understanding of the XP practices, as they started (SE1) using their post-its (*user stories*) to

guide the production process and decide when a prototype (*small releases* and *testing*) was ready for the meeting with the *on-site customer*. Moreover, all the teams took advantage of the customer's feedback to improve the solution (SE2). However, we frequently had to solicit the teams to stop working with the robot to dedicate some time to other activities, such as stand-up meetings. Most of the time, reminding them that stand-up meetings represented an excellent chance to prepare some material for the blog worked as a motivating factor.

All the teams demonstrated an excellent ability to integrate changes to the solution. Thanks to the block-based programming environment, they integrated the new features on top of the existing ones (Fronza et al., 2019a) by making frequently sure that the changes did not negatively affect or break the existing code (SE3).

The proposed activity covered a limited number of hours, and the assigned task was rather trivial. These characteristics certainly did not recreate a real professional environment but supported our goal of starting a progressive set of activities, during which the didactic transposition of the Agile methods becomes closer and closer to the professional setting, also by increasing the difficulty of the programming task. Under consideration of this idea, we consider the results of this experience as an indicator that students reacted positively to this first time they were exposed to a SE approach. Indeed, although we did not provide additional lectures on SE, students adapted to the new organization of the work process, by taking advantage of frequent iterations and customer's feedback.

**Curricular Objectives.** As part of the curricular objectives, teachers had to ensure that students achieved the ability to: 1) create simple websites or blogs, 2) access the Internet with confidence, and 3) respect the Copyright rules. At the end of the module, the school teachers confirmed that all the participants achieved these objectives at the desired level.

## 7 CONCLUSION AND FUTURE WORK

This paper provides educators with a practical example of how to cover several areas of technology in a way for middle school students to be engaged and to spark future interest. Specifically, it shows how it is possible to achieve the CS learning objectives and foster SE principles while fulfilling the existing curricular objectives (RQ1). Even though the proposed activities have been designed by taking into account specific context factors, the fundamental design principles can generalize to other contexts. Indeed, the pro-

posed module does not introduce additional SE lectures, which is crucial to middle schools, where students would not perceive these additional lectures as useful. Moreover, we considered a context in which CS is taught as part of other disciplines, which happens frequently and hinders the reservation of time for other content that does not directly contribute to achieving the curricular learning objectives. Finally, the considered learning objectives (Table 1) can be reasonably present in other curricula.

The proposed assessment framework has been applied to analyze the outcome of the first classroom experience, and the obtained results show the effectiveness of the approach (RQ2). Further experiments are needed to generalize these results by involving other groups of participants. Moreover, the possible effect of language should be analyzed: in this experience, the official language was Italian, which could have increased the difficulty for German mother-tongue speakers. Additionally, the possible effects of participants' backgrounds should also be inspected.

The results of this paper are encouraging for working on the development of our next modules dedicated to middle schools, to progressively cover all the objectives included in the reference document for the informatics curriculum. Moreover, the proposed *Area of Software Engineering* (and its assessment framework) needs to be further extended by including more practices to support the development process.

This work, together with our previous ones (Fronza and Pahl, 2019; Fronza et al., 2019b), shows that it is possible to foster SE principles in K-12 with no or minimum programming tasks. It remains to be explored whether and how these activities will help to improve code quality when students write code to solve non-trivial tasks. Finally, performing a classroom experience next year with the same group of students would allow us to test for retention, and check if students are improving towards the achievement of the final objectives of the middle school curriculum.

## REFERENCES

- Barricelli, B. R., Cassano, F., Fogli, D., and Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137.
- Bocconi, S., Chiocciariello, A., Dettori, G., Ferrari, A., Engelhardt, K., et al. (2016). Developing computational thinking in compulsory education-implications for policy and practice. Technical report, Joint Research Centre (Seville site).
- Bollin, A., Pasterk, S., Antonitsch, P., and Sabitzer, B.

- (2016). Software engineering in primary and secondary schools-informatics education is more than programming. In *Software Engineering Education and Training (CSEET), 2016 IEEE 29th International Conference on*, pages 132–136. IEEE.
- Burnett, M. (2009). What is end-user software engineering and why does it matter? In *International Symposium on End User Development*, pages 15–28. Springer.
- Burnett, M. M. and Myers, B. A. (2014). Future of end-user software engineering: beyond the silos. In *Proceedings of the on Future of Software Engineering*, pages 201–211. ACM.
- Burning Glass Technologies (2016). Beyond point and click: the expanding demand for coding skill. <https://bit.ly/2YYL60A>. Accessed 18 Feb. 2020.
- Chevallard, Y. and Gilman, C. (1991). *La transposición didáctica: del saber sabio al saber enseñado*, volume 1997. Aique Buenos Aires.
- Forlizzi, L., Lodi, M., Lonati, V., Mirolo, C., Monga, M., Montresor, A., Morpurgo, A., and Nardelli, E. (2018). A core informatics curriculum for italian compulsory education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 141–153. Springer.
- Fronza, I., El Ioini, N., and Corral, L. (2016). Blending mobile programming and liberal education in a social-economic high school. In *Proceedings - International Conference on Mobile Software Engineering and Systems, MOBILESofT 2016*, pages 123–126.
- Fronza, I., El Ioini, N., Pahl, C., and Corral, L. (2019a). *Bringing the Benefits of Agile Techniques Inside the Classroom: A Practical Guide*, pages 133–152. Springer Singapore, Singapore.
- Fronza, I., Ioini, N. E., and Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education (TOCE)*, 17(4):19.
- Fronza, I. and Pahl, C. (2019). Teaching software engineering principles in non-vocational schools. In *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU)*, pages 252–261.
- Fronza, I., Pahl, C., and Susanj, B. (2019b). A didactic module to teach software engineering principles in middle schools. In *Proceedings of the 20th Annual SIG Conference on Information Technology Education, SIGITE '19*, pages 168–168, New York, NY, USA. ACM.
- Fronza, I. and Zanon, P. (2015). Introduction of computational thinking in a hotel management school [introduzione del computazionale thinking in un istituto alberghiero]. *Mondo Digitale*, 14(58):28–34.
- Gartner Inc. (2017). Market Guide for Rapid Mobile App Development Tools.
- Hazzan, O., Dubinsky, Y., and Meerbaum-Salant, O. (2010). Didactic transposition in computer science education. *ACM Inroads*, 1(4):33–37.
- Kastl, P., Kiesmüller, U., and Romeike, R. (2016). Starting out with projects: Experiences with agile software development in high schools. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, pages 60–65. ACM.
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., et al. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, 43(3):21.
- Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Meerbaum-Salant, O. and Hazzan, O. (2010). An agile constructionist mentoring methodology for software projects in the high school. *ACM Transactions on Computing Education*, 9(4):n4.
- Monteiro, I. T., de Castro Salgado, L. C., Mota, M. P., Sampaio, A. L., and de Souza, C. S. (2017). Signifying software engineering to computational thinking learners with agentsheets and polifacets. *Journal of Visual Languages & Computing*, 40:91–112.
- Oakley, B., Felder, R. M., Brent, R., and Elhadj, I. (2004). Turning student groups into effective teams. *Journal of student centered learning*, 2(1):9–34.
- Rico, D. F. and Sayani, H. H. (2009). Use of agile methods in software engineering education. In *2009 Agile Conference*, pages 174–179. IEEE.
- Romeike, R. and Göttel, T. (2012). Agile projects in high school computing education: emphasizing a learners' perspective. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, pages 48–57. ACM.
- Scheubrein, R. (2003). Elements of end-user software engineering. *INFORMS Transactions on Education*, 4(1):37–47.
- Schneider, J.-G. and Johnston, L. (2005). eXtreme Programming- helpful or harmful in educating undergraduates? *Journal of Systems and Software*, 74(2):121–132.
- Shaw, M. (2000). Software engineering education: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 371–380. ACM.
- Spire, H. A., Herve, L. G., Morris, G., and Stelpflug, C. (2012). Energizing project-based inquiry: Middle-grade students read, write, and create videos. *Journal of Adolescent & Adult Literacy*, 55(6):483–493.
- Zucconi, L. (1995). Essential knowledge for the practising software engineer and the responsibilities of university and industry for her education. In *Conference on Software Engineering Education*, pages 3–13. Springer.