A Facilitator's Guide to Create and Consolidate a Teenage Coding Camp

By Ilenia Fronza, *Free University of Bozen/Bolzano* and Luis Corral, *ITESM Campus*

We look back on a journey of more than 10 years of teaching coding skills to high school students using block-based programming platforms. Eleven years of evolution, capabilities, and technical content impact teenage participants in problem-solving, creativity, collaboration, and coding skills. We discuss how we plan and execute each instance of a coding camp, the progression of the topics and tools, and the main insight collected working with more than 500 participants. We summarize our experience in four main aspects: Learning strategy, Enabling Technology, Teamwork, and Partnerships. We also list recommendations to facilitators and share a series of reflection points.

INTRODUCTION

In current education programs, it is common to find the first experience with software development in the early years of elementary education or even preschool, in many ways, thanks to the dissemination of tools that allow the development of software programs without the need to become proficient in a programming language. The distribution of these software tools is an essential enabler of the popularization and democratization of software development at all levels. Nowadays, it is easy to find friendly software development tools in toys, video games, "do it yourself" kits, and educational environments. However, the power and scope of these friendly tools go way beyond children or K12 students—some of these tools have facilitated the development of professional software products by non-expert personnel who create software for productivity or entrepreneurial purposes. Behind this exciting and empowering possibility lie the so-called block-based programming languages, which are tools that allow the user to create working software products with little knowledge of the structure and syntax of a standard programming language. Block-based programming languages commonly use a programming primitive as a puzzle piece metaphor, providing visual cues about how and where commands may be used, how they should be constructed, the required parameters, and so on [12]. Examples of widely spread block-based programming environments are Scratch and Code.org. Ubiquitous in educational settings are Lego® Mindstorms® Software Development Kit for robot programming and MIT's AppInventor or Thunkable for developing mobile applications. A recent systematic mapping study [10] identified block programming tools, emerging technologies, audiences, and learning spaces where block programming is being worked on.

When block-based programming languages are the first approach to coding and software development, students benefit from being guided in a learning process from the basics of problem-solving through implementing solutions in the form of software applications without focusing on syntactical issues. The systematic approach to problem-solving and Computational Thinking skills [13] to decompose the problem and implement solutions are essential foundations for students to understand problems and then think of software as a tool to solve them. Indeed, block-based programming helps to work on aspects such as problem-solving and algorithmic thinking, among other skills [10].

Whether they are called camps, hackathons, or anything else with the same basic meaning (i.e., "Short time collaborative innovation activity focusing on some use of computer skills'" [11]), these intensive events bring together individuals from diverse backgrounds and collaboratively tackle complex challenges within a limited timeframe. Coding camps that consider both problem-solving and software development tools boost the potential of students to leverage coding as an effective way to implement a solution to a real need. The role of an expe-

Whether they are called camps, hackathons, or anything else with the same basic meaning, these intensive events bring together individuals from diverse backgrounds and collaboratively tackle complex challenges within a limited timeframe.

rienced instructor as a learning leader and the collaboration that emerges among participants enriches the learning experience, gives structure, promotes teamwork, and makes it enjoyable. With this in mind, we, researchers in Computing Education with a strong Software Engineering background, have organized and facilitated a coding camp on mobile software development at the Free University of Bozen/Bolzano (Italy) since 2012. The preparation of the course curriculum covered the foundational aspects of problem-solving and coding practice using block-based programming tools; moreover, it made a purposeful attempt to stimulate teamwork and delivery in a professional, collaborative setting.

Coding camps are getting increasingly popular and already in 2016, a survey found a 175% growth rate in 2014 alone [1], while two recent systematic literature reviews found a substantial number of research papers dedicated to the topic. The first review, published in 2019, found 51 relevant papers (most published after 2012) and found that although these events are highly praised, their links to actual educational activities are still very scarce [11]. In 2021, a second review retrieved 46 papers and highlighted that contemporary academic literature gives little guidance to organize hackathons, and more research works are needed to provide best practices, means, and tools [9].

Some 11 years have passed, and our coding camp is still offered summer after summer, with a clear progression over the years on topics, tools, and educational environment. Maintaining the viability of the school, particularly keeping up with the attention and interest of groups of teenagers, has not been easy. For this reason, following the suggestions in recent reviews of the literature [9,11], we have taken a break in our journey to discuss and share how we plan and execute the course, the evolution of the topics and tools we use, and the main insight collected working with more than 10 classes that sum more than 500 students.

GOAL AND STRUCTURE OF THE CODING CAMP

Originally, the purpose of the *MobileDev* coding camp (https://mobiledev.inf.unibz.it) was to showcase Computer Science as a career option for high school students and attract talent to the available bachelor programs such as Computer Science and Information Technology. Students came from various schools and backgrounds, so the course could not assume that there was any foundation or previous knowledge of coding or software development. For this reason, the coding camp was structured as a 20-hour course run over one week (4 hours per day) and organized in three parts [7]:

- first, explaining general concepts about problem understanding and problem-solving, on top of Computational Thinking skills;
- second, teaching the fundamentals of computer programming using a block-based programming language to avoid the need to learn programming language syntax; and
- third, proposing a challenge or final project in which students, working in teams, apply the acquired knowledge and skills with a hands-on approach.

The desired learning goal of the coding camp is that, by its conclusion, students should be able to understand and decompose a problem, prototype the solution as a simple software product, and execute it in a given target platform.

Since its original design, the coding camp leveraged the use of cell phones as a strategic feature, as it is common for teenage students to own and use mobile devices intensively; moreover, by that time, there were already block-based programming environments focused on mobile software development. These resources paved the road to setting up a short training plan, learning by example, and producing simple projects until the essential knowledge of the tool is attained. In order to ensure that the coding camp accomplishes its goal not only from the point of view of instructors but also of participants, a survey of self-assessment of skills and overall satisfaction with the course is conducted on the last day of the coding camp.

EVOLUTION OF THE CODING CAMP

The timeline in Figure 1 represents the evolution of the *MobileDev* coding camp, its milestones, major changes, and pivotal initiatives.

Table 1 summarizes the scientific publications that describe our approach to organizing coding camps as an educational strategy to teach software engineering and coding abilities. Articulating all the specifics of the coding camp's educational strategy is out of the scope of this paper. Instead, we aim to learn from them and complement our vision after more than 10 years of organizing our activity to identify key aspects and best practices. In this way, we build up a set of solid advice for organizers and facilitators while validating such advice on top of several peer-reviewed scientific contributions that discuss the strategy, execution, results, and feedback of these coding camps.

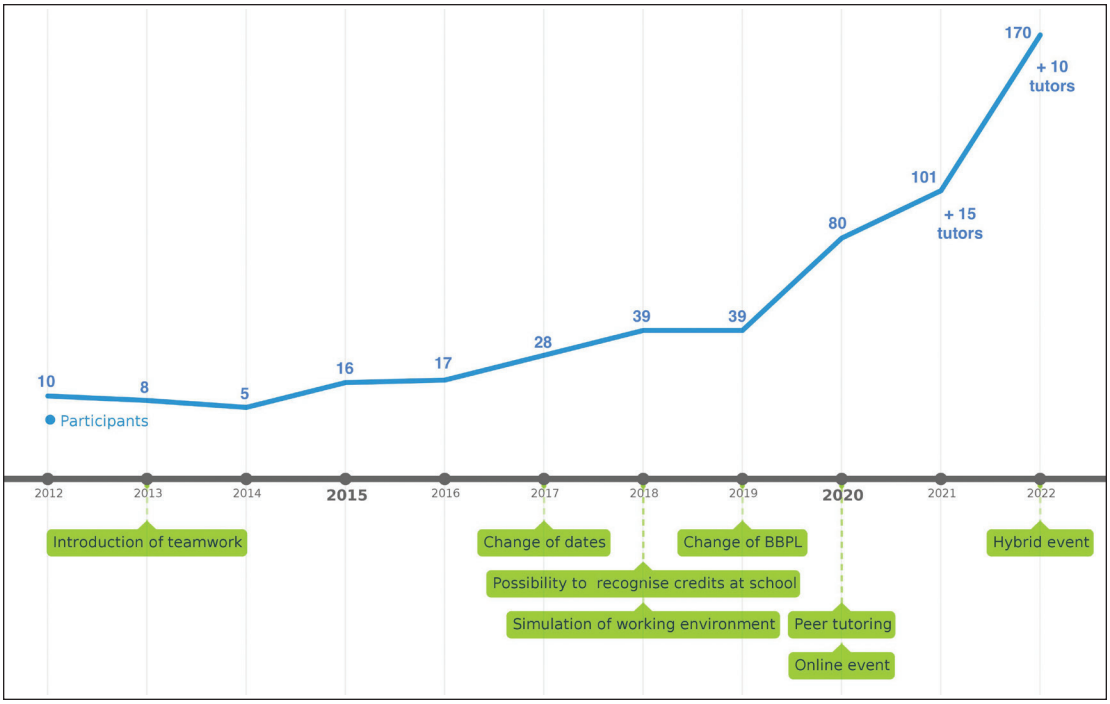

Figure 1: Coding camp timeline: Milestones and Number of Participants per Year.

Table 1: Scientific publications focusing on several aspects of the coding camp's educational strategy.

Reference	Contribution
[2]	The paper presents a strategy to leverage each phase of the software development process to foster specific computational thinking skills.
[3]	The paper presents a strategy to foster computational thinking skills during the coding camp by demonstrating the function of a simple algorithm using a simple hardware prototype.
[4]	The results of a follow-up project show the camp's effectiveness in delivering technical foundations on fundamental aspects of Software Engineering and concepts of product design and teamwork.
[5]	The paper presents the instructional strategy of the coding camp and an assessment framework to evaluate its outcome. The results show the instructional strategy's effectiveness and find games (with Software Engineering take-away messages) to be a cornerstone of a successful outcome.
[6]	The paper proposes and validates the inclusion in the assessment framework of the coding camp of a metric that leverages "When", a condition typically found in block-based programming languages.
[7]	The paper analyzes how a practical approach to convey CT skills motivates and effectively prepares student tutors. Moreover, comparing the camp's previous and subsequent editions identifies the tutors' specific contributions.
[8]	The paper reports the experience of designing and running a fully remote edition of the coding camp. The comparison with a face-to-face edition shows that we succeeded in keeping the fun alive (confirming the importance of the proposed games) in the online edition; participants produced the results at the same level of quality in terms of product and process as in the face-to-face edition.

The first edition of *MobileDev* was run in the summer of 2012 with 10 students who attended by invitation (Figure 2). After 11 editions, in 2022, the number of enrolled students reached a record of 170 participants (Figure 3). In the following sections, we will discuss what happened in between.

TEACHING/LEARNING STRATEGY

As a coding camp, the main driver of the experience should be practical. However, as discussed, it is difficult to assume that students come to the course already equipped with knowledge or expertise in structured problem-solving or software coding. For this reason, our learning plan lays its foundations on the basics of problem decomposition and problem-solving strategies on top of Computational Thinking. A group of high school students already faces several daily challenges that can be interpreted as "problems to solve." Elemental situations like choosing what bus to ride, how to reach a place in the town, or how to prepare a sandwich provide an excellent atmosphere to under-

Figure 2: First edition of *MobileDev*, 2012.

Figure 3: Last edition of *MobileDev*, 2022.

stand a high-level problem, decompose sub-problems of limited scope, and propose solutions that deliver an efficient solution to the challenge at hand. Problems proposed by instructors should be timely and congruent with the environment of the participants, and solutions should be represented in the form of pen-and-paper explanations or verbally in front of other peers.

When transitioning to code examples, problems and challenges evolve to elements that require knowledge of logic, mathematics, or general culture that the group already has. Instructors' focus changes to explain the basics of views (user interfaces), controls (commands, execution flow, operators, parameters, control structures), and user interface events. These principles provide a sound foundation of concepts; as the lessons advance, more complex concepts are built on top of basic ones. After covering the core of the body of knowledge offered in the course, participants identify a problem to solve and develop a "capstone project" that prototypes a solution to the problem. Summary projects should encompass the skills gained during the course. Final projects were produced individually only in the first edition (2012), but from the second class (2013) until today, final projects are developed in teams of two or three students.

Starting in 2017 (Figure 1) and with the purpose of fostering solid collaboration between students, instructors introduced games [5] as an educational strategy to illustrate development practices, cultivate communication and collaboration skills, and improve the overall experience of participating in the camp. With this strategy, games and fun were found to be a cornerstone of a successful outcome. With the rise of the COVID-19 emergency in 2020, when the coding camp transitioned to an online format, we aimed to keep the same "level of fun" in the online coding camp. The class is gathered in collaborative games or sent out on scavenger hunts. Participants are generally invited to participate in games that foster a sense of belonging, engage participants, and reduce fatigue due to prolonged computer use, moving around and releasing energy before focusing again [8]. For example, during the online editions of *MobileDev*, teams competed to create color wheels (Figure 4) using the highest number of colors and objects (found in the surroundings) in 15 minutes. Being in an online setting, one team member built the wheel while the others provided suggestions. The takeaway message of this activity was about the importance of working together toward a solution by identifying small steps. Moreover, the game introduced an element of fun when observing the type and the number of objects placed on the wheels.

Figure 4: A color wheel created by the *MobileDev* participants.

ENABLING TECHNOLOGY

In the first edition of the coding camp, we used MIT AppInventor as a block-based programming environment. Back in 2012, MIT block-based tools like Scratch and AppInventor were very well known in academia, widely available, and free of charge. For these reasons, we selected MIT AppInventor as a development environment for the coding camp. The effective puzzle metaphor of MIT AppInventor, as well as the extensive documentation and numerous examples available, permitted the creation of a learning journey that allowed students to implement a simple application in a very short time and boosted the feeling of accomplishment thanks to the possibility of running the produced application in an actual cellular phone. At the time, the disadvantage of the MIT AppInventor platform was that it was bound only to the Android operating system, and several students owned cellphones operated by a different

platform (typically iOS). In order to overcome this situation, at that time, the coding camp distributed university-owned cell phones to students needing a real execution platform

However, we overcame this restriction in the 2019 edition (see the timeline in Figure 1) by introducing Thunkable as the block-based programming language of choice. Back in 2016, Thunkable emerged as a spin-off company of some team members who developed MIT AppInventor. The advantage brought by Thunkable was the possibility of developing applications that, following an identical block-based programming paradigm, could now be installed both in Android and iOS-operated devices. This change in the programming technology brought no change to the strategy and other structural aspects of the coding camp (for instance, problem-solving or puzzle-based coding primitives). The reception of Thunkable by students was very positive since the operating system of the owned cellphone was no longer a factor in determining whether they could use the device they owned. This possibility permitted organizers to open the call for applications to the school as a *Bring Your Own Device* (BOYD) coding camp. It demonstrated a higher feeling of accomplishment in students who could develop an application and then install and try it on their own mobile phones.

Much as in other educational and professional settings, the emergency conditions brought by the COVID-19 pandemic posed a fundamental challenge for organizers, sponsors, and participants of the coding camp. On this front, the transition to two fully remote editions (2020 and 2021, Figure 1) was made possible thanks to the rapid adoption of communication and participation technologies like Zoom to set up the sessions, Mentimeter to allow for crowdsourced participation, and Google Documents, through which students shared problems, wrote simple specifications, or drew diagrams collaboratively. In 2022 (Figure 1), the coding camp transitioned again to permit face-to-face interaction. Nonetheless, the school format was set to hybrid due to the possibilities placed by collaboration tools and to accommodate a rising number of participants meeting health and safety restrictions. Moreover, we chose the hybrid setting to feature an authentic experience during the coding camp. Indeed, many companies transitioned toward hybrid work as a "new normal" way of working after the COVID-19 pandemic. Half of the class meets in a classroom while the other half attends via MS Teams (Figure 5), accessing the same course content in synchronous interaction with instructors and peers.

TEAMWORK AND COLLABORATION

As discussed above, in the first edition of the coding camp, we favored the course's technical foundation and the individual acquisition of concepts. However, starting in the second edition, we fostered teamwork in a twofold strategy: (1) Work Alone, since each participant is expected to work independently, and the learning and skills gained in the course are a personal benefit; and (2) Work in Teams, because in a real-world setting, teamwork is not only beneficial but a true conditional for success. Participants develop coursework exercises individually but work on projects in teams. In this way, participants are motivated to learn individually as much as possible while ensuring they contribute, communicate, and collaborate with peers.

Figure 5: When working remotely, teams must organize themselves to collaborate and obtain the best outcome. This picture shows the development of a game that requires building a paper tower. The team in the background decided to work locally. The team in the laptop display is working from home.

As a framework for collaborative development, students are invited to embrace agile software development practices such as describing requirements as user stories and metaphor systems, taking roles of customers or project leaders, producing prototypes (including pencil-and-paper sketches), and working in iterations to produce incremental releases.

In the last editions of the school, one of the most effective techniques leveraged was the incorporation of peer-to-peer mentorship [6]. Thanks to our collaboration with local education authorities, in 2020 (Figure 1), we proposed that coding camp "alumni," that is, students who completed the coding camp in past years, participate in the next edition, taking the role of technical tutors. These tutors support the learning leaders (teaching staff) by making themselves available to their peers, accompanying them in learning the basic principles, and using the tools with their previous knowledge and peer-to-peer connection as a powerful teaching resource. With the development of student tutors, the coding camp helps to create not only additional technical capacity that remains exclusive for the individual profit of

After reflecting on the evolution of the coding camp, the planning effort, and the results obtained, we divided the insight gained into independent points that may be useful for instructors and enthusiasts to design a learning plan, promote it to students, and keep it current, evolving together with the changes in technology and unexpected changes in training design and delivery.

each student, but it also creates a seed effect that positively impacts the learning experience of less experienced students. Also, as they are part of a similar age segment, student tutors bring additional elements of confidence, communication, openness, and point of view to the coding camp. It is worth mentioning that, in the online setting, breakout rooms challenge facilitators to observe participants' behavior directly; thus, tutors can support facilitators in observing teams by using an ad-hoc protocol.

PARTNERSHIP WITH SPONSORS

We promoted the coding camp to high schools in the city in partnership with the local Education Authority of the Autonomous Province of Bolzano (Italy). This partnership was a true enabler for the strategy and success of the school after several editions. In 2017 (Figure 1), in agreement with the local Education Authority, we shifted the dates of the coding camp to the week the regular school year starts instead of being offered as a summer course during the summer holiday. With this adjustment, we secured two key elements for the coding camp's success: On the one hand, students were available in the city and not out of town on vacation. On the other hand, should the coding camp schedule overlap one or more hours of regular schoolwork, the time invested in the activity could be recognized with a value of credits for the benefit of students. This initiative is known in Italy as the "path for transversal skills and orientation" (PCTO), a government initiative aimed at helping students develop their professional skills and career orientation by providing them with work-related experiences. Under the PCTO, students must complete several hours of work-related experiences, for instance, internships, apprenticeships, or practical training.

To deepen its impact and recognition as a PCTO activity, starting in 2018 (Figure 1), *MobileDev* promoted the development of socially-relevant software projects within an atmosphere that emulates a professional setting, including instructors taking the role of customers, teams organized with clear roles and responsibilities, rolling out a development schedule, and implementing an agile software development life cycle. The produced software applications aim to solve a problem relevant to society (healthcare, water preservation, garbage management, and other similar topics), adding a relevant notion of usefulness, value, or noble purpose to the work. In 2019, the coding camp was recognized as an activity that fostered learning and practice in a professional setting and was named a "distinguished project" nationwide. This recognition ramped up the number of participants in the following years (Figure 1). With the transition to a remote format during the pandemic emergency, even more students could be accommodated, including those from other cities and regions in the country.

RECOMMENDATIONS TO FACILITATORS

After reflecting on the evolution of the coding camp, the planning effort, and the results obtained, we divided the insight gained into independent points that may be useful for instructors and enthusiasts to design a learning plan, promote it to students, and keep it current, evolving together with the changes in technology and unexpected changes in training design and delivery.

1. Find the Right Methodological Approach. Through all the editions of the coding camp, the teaching staff has been the same, ensuring continuity on the learning plan and teaching style, and following-up on the experience and feedback of previous years. Instructors implement a method that builds from a strong foundation of problem-solving and then software coding. The thirst for learning of young students can make them impatient to start the "hands-on" parts of the course; however, it is fundamental to convey that, to produce a good software solution, it is first essential to create a human solution. Embracing the "solve the problem first, then write the code" philosophy, students understand the importance of growing first as problem solvers or designers of solutions, and once the solution exists, prototype it in the form of a software development tool. A major lesson learned from this approach is that former participants have voiced is that, even though vocationally they will not choose computer science as their major, participating in the coding camp gives them skills to approach better to problem-solving regardless of the application domain.

2. Incorporate Games and Fun activities. Let us not forget that our target audience is a group of spirited teenagers who are certainly curious and want to learn, but most probably, they do want to have fun, too. Thus, facilitators should leverage games and entertaining activities as an opportunity to deliver key learning that goes beyond the technical aspects [5,8]. Fine-grained soft skills and professional awareness like communication, empathy, resourcefulness, listening skills, collaboration, and inclusion

can be taught through games and similar experiences. Indeed, a game carefully chosen energizes the class and delivers key messages. After the game is completed, a quick download discussion helps to share the lessons learned during the game and sets the stage to resume the learning journey with new topics.

3. Carefully Choose the Enabling Technology. As technical training, the selected development tools can determine the success of the learning journey. We took advantage of the availability and easy comprehension of block-based programming languages and also from the enormous power that such platforms display today. Tools like Thunkable permit the development not only of educational but also professional software applications. Students learn the basics of the tool and quickly discover the potential of the software tool in a larger scope. It is crucial to keep an eye on the evolution of technology due to the continuous evolution of tools and technologies, and as instructors, be early adopters of emerging tools to avoid becoming outdated or obsolete. Also, thanks to the technology, the coding camp was viable even in the most challenging times. The incorporation of communication and distributed collaboration tools permitted remote attendance when it was mandatory but opened doors as well to extend the camp's outreach by allowing a hybrid format that was key to increasing the number of participants every year while simulating a remote and distributed work environment that is currently normal after the COVID-19 emergency.

4. Cultivate Teamwork. By stimulating the work in an emulated professional setting, teamwork is the keystone for project success. Participants share the same learning goal and, working towards it, actively assist others in learning and, in turn, benefit from an effective learning environment. A course strategy in which participants are required to work in teams gives them exposure to working with other participants of different backgrounds, priorities, perspectives, and approaches to work. The experience of remote and hybrid work also gives a taste of contemporary professional work. In software development companies (as well as in other domains), companies are distributed geographically, and colleagues, partners, and customers can sit in different regions or countries. Finally, being exposed to the execution of an agile-inspired software development process also gives participants a look at how collaboration works when aiming for a common goal, looking to fulfill a customer's requirements, or taking different roles or responsibilities.

5. Promote Students as Teaching Partners. We mentioned the relevance of incorporating student tutors into the knowledge-transfer strategy of the coding camp [7]. Adding student tutors was a major milestone in the camp's history, as it represented the extension of the learning process, not only as a participant of the coding camp but also reinforcing the skills gained serving as a tutor. We strongly believe that a coding camp that can create a seed effect in young students motivates the learning process and sustains the viability of the camp by continuously preparing student tutors. On top of that, we observed that camp participants take clear advantage of the tutors' participation as they grow motivation, trust, and communication among peers. As course "alumni" become tutors, we stimulate a multiplier effect to benefit an increasing number of students in the years to come. Finally, collecting feedback from participants is paramount to reviewing the learning goals of a course, assessing the overall experience, and iterating in the planning, content, and delivery to maintain a continuous improvement attitude in which the next coding camp could always be better than the previous one.

6. Take the opportunity for diversity. The *MobileDev* coding camp takes pride in being a safe environment for learning. The diversity of backgrounds, academic specializations, and learning styles is vast, and very distinct profiles converge in the coding camp to learn how to work in teams in harmony. The teaching staff underscores that no one knows all the answers (including instructors themselves). This affirmation promotes a safe environment to raise questions, express your own point of view, and establish a continuous communication pace, always with respect and professionalism. The participation of female talents in the coding camp has been remarkable. In the first edition in 2012, with 10 participants, two were women. In the most recent 2022 edition, the school accounted for a female gender representation metric of 30%. Fostering gender diversity is not limited to promoting female participation in the course, but it is as well empowered in the configuration of teams, where participants are commonly sorted into teams of two women and one male. In this way, students of different genders, backgrounds, and approaches contribute to the team's objectives, accomplish a goal together, and value the participation of each one.

7. Look for Sponsors. The coding camp was born as a vocational strategy to attract talent to the Free University of Bozen/Bolzano (Italy). Without the university's support and sponsorship, instructors would lack a proper environment, including facilities (classroom, equipment, communication channels), tools (laptop computers, mobile phones), and advertisements. In parallel, the support that the coding camp has received from the local education authority has been of utmost importance in spreading the word to schools, reaching the target audience, and convincing teachers and students that participating in this coding camp is not only formative but also convenient. With the support of key partners like the Education Authority, organizers can ensure that relevant stakeholders endorse the project and bring a boost of trust to other actors like parents or guardians. The recognition the *MobileDev* coding camp has gained through the years has helped to create a solid and recognized brand that builds trust and reputation. Thanks to the support of local Education institutions, the coding camp's success is undeniable.

We are optimistic that several of these guidelines will hold when designing a learning journey in fields other than Software Engineering. However, further research and experiences should be conducted and reported before coming to a solid conclusion on this front.

TAKE AWAY AND CONCLUSION

We have reviewed the evolution for over 10 years, offering a coding camp for teenage students using block-based programming languages. Although the impact of the coding camp on students is yet to be analyzed in detail, we can summarize our experience in major takeaway points for other facilitators, instructors, or enthusiasts of *Computer Science for All* and similar initiatives.

1. Focus on the method: A structured approach to problem-solving, powered by a process like Computational Thinking, ensures that participants understand the importance of framing a problem, decomposing it, finding a solution, and only after that, thinking of software as a tool to implement solutions.

2. Empower learners and foster collaboration: Promoting a diverse and collaborative environment fosters teamwork and makes the coding camp not only technical but a fun and enjoyable social experience. Instructors are expected to serve as learning leaders. However, the experience evolves to the next level when students are empowered to help themselves solve the challenges, or, as we observed in later editions of the school, a peer-to-peer tutoring program is formalized.

3. Keep up with technology: A decade may be relatively long in terms of available technology, both software and hardware tools. Facilitators should take care of incorporating current software development tools and execution platforms to ensure the timeliness of the transferred knowledge and engage participants with the technology they actually use and aspire to master.

4. Build a brand: After ten years of promoting a coding camp, people (especially teachers and parents) start identifying it, and the experience of previous editions helps build a reputation. With this, we cultivate expectations in parents, teachers, and students who wait for the call and apply early to secure a place.

5. Connect with stakeholders: Running and sustaining a code camp without help can be game-lost from the outset. Identifying sponsors and working hand in hand with them gives the support, endorsement, and dissemination needed to attract the best talent to the course.

6. Do not forget purpose: There was a clear evolution in the attractiveness, number of participants, and interests of sponsor schools when organizers placed a notion of purpose on the outcome products of the coding camp. Going beyond example apps, undertaking a short challenge to produce a tool with a meaningful purpose attracts, focuses, and motivates participants, instructors, and sponsors.

Coding camps are an unparalleled environment to transfer knowledge and empower and enable young students to acquire new skills. After 11 years of running a coding camp, we have witnessed the continuous motivation of teenagers to learn, share knowledge, create tutoring relationships, promote collaboration, and, through a sense of purpose, leverage their new knowledge to accomplish results that positively impact their communities. ❖

References

1. Champagne, J. (2016). *Are coding bootcamps worth it?* https://blog.capterra.com/are-coding-bootcamps-worth-it/
2. Fronza, I., El Ioini, N., & Corral, L. (2015). Students Want to Create Apps: Leveraging Computational Thinking to Teach Mobile Software Development. In *Proceedings of the 16th Annual Conference on Information Technology Education (SIGITE '15)*. Association for Computing Machinery, New York, NY, USA, 21–26. https://doi.org/10.1145/2808006.2808033
3. Fronza, I., Corral, L., & Pahl, C. (2019). Combining Block-Based Programming and Hardware Prototyping to Foster Computational Thinking. In *Proceedings of the 20th Annual SIG Conference on Information Technology Education (SIGITE '19)*. Association for Computing Machinery, New York, NY, USA, 55–60. https://doi.org/10.1145/3349266.3351410
4. Fronza, I., Corral, L., Pahl, C., & Iaccarino, G. (2020). Evaluating the Effectiveness of a Coding Camp through the Analysis of a Follow-up Project. In *Proceedings of the 21st Annual Conference on Information Technology Education (SIGITE '20)*. Association for Computing Machinery, New York, NY, USA, 248–253. https://doi.org/10.1145/3368308.3415391
5. Fronza, I., Corral, L., & Pahl, C. (2020). End-User Software Development: Effectiveness of a Software Engineering-Centric Instructional Strategy. *JOURNAL OF INFORMATION TECHNOLOGY EDUCATION*, vol. 19, p. 367-393, ISSN: 1539-3585, doi: https://doi.org/10.28945/4580
6. Fronza, I., Corral, L., & Pahl, C. (2020). An Approach to Evaluate the Complexity of Block-Based Software Products. *INFORMATICS IN EDUCATION*, vol. 19, p. 15-32, ISSN: 1648-5831, doi: 10.15388/infedu.2020.02
7. Fronza, I., Corral, L., Iaccarino, G., & Pahl, C. (2021). Enabling Peer-Led Coding Camps by Creating a Seed Effect in Young Students. In *Proceedings of the 22nd Annual Conference on Information Technology Education (SIGITE '21)*. Association for Computing Machinery, New York, NY, USA, 117–122. https://doi.org/10.1145/3450329.3476860
8. Fronza, I., Corral, L., Wang, X., & Pahl, C. (2022). Keeping fun alive: an experience report on running online coding camps. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22)*. Association for Computing Machinery, New York, NY, USA, 165–175. https://doi.org/10.1145/3510456.3514153
9. Happonen, A., Tikka, M., & Usmani, U. A. (2021, November). A systematic review for organizing hackathons and code camps in Covid-19 like times: Literature in demand to understand online hackathons and event result continuation. In 2021 *International conference on data and software engineering (ICoDSE)* (pp. 1-6). IEEE.
10. Perin, A. P. J., dos S Silva, D. E., & Valentim, N. (2023). Investigating block programming tools in high school to support Education 4.0: A Systematic Mapping Study. *Informatics in Education*, 22 (3), 463-498.
11. Porras, J., Knutas, A., Ikonen, J., Happonen, A., Khakurel, J., & Herala, A. (2019). Code camps and hackathons in education-literature review and lessons learned. *Proceedings of the 52nd Hawaii International Conference on System Sciences*.
12. Weintrop, D. Block-based programming in computer science education, Communications of the ACM 62, 8 (2019):22–25. https://doi.org/10.1145/3341221
13. Wing, J. M. Computational thinking, *Communications of the ACM* 49, 3 (2006): 33–35. https://doi.org/10.1145/1118178.1118215

Ilenia Fronza
Faculty of Engineering
Free University of Bozen/Bolzano
Piazza Domenicani 3
Bolzano, Italy 39100
ilenia.fronza@unibz.it

Luis Corral
ITESM Campus
Epigmenio González 500 Fracc. San Pablo
Queretaro, Mexico 76130
lrcorralv@tec.mx

DOI: 10.1145/3643726