Keeping Fun Alive: an Experience Report on Running Online Coding Camps

Ilenia Fronza
ilenia.fronza@unibz.it
Free University of Bozen/Bolzano
Bolzano, Italy

Xiaofeng Wang
xiaofeng.wang@unibz.it
Free University of Bozen/Bolzano
Bolzano, Italy

Luis Corral
lrcorralv@tec.mx
ITESM Campus Queretaro
Queretaro, Mexico

Claus Pahl
claus.pahl@unibz.it
Free University of Bozen/Bolzano
Bolzano, Italy

ABSTRACT

The outbreak of the COVID-19 pandemic prohibited radically the collocation and face-to-face interactions of participants in coding bootcamps and similar experiences, which are key characteristics that help participants to advance technical work. Several specific issues are faced and need to be solved when running online coding camps, which can achieve the same level of positive outcomes for participants. One of such issues is how to keep the same level of fun that participants obtained through physical activities and interactions in the face-to-face settings. In this paper, we report on our experience and insights gained from designing and running a fully remote coding camp that exposes high school students to Agile-based Software Engineering practices to enhance their ability to develop high-quality software. To design the online coding camp, we adapted the face-to-face version of the coding camp to keep the same "level of fun", i.e., adaptations aimed at increasing communication, engaging participants, and introducing fun items to reduce fatigue due to prolonged computer use, while preserving the technical curriculum that enables students to attain the learning goals originally planned. The comparison with the results of the face-to-face coding camp shows that we succeeded in keeping the fun alive in the online edition, and the participants of online camp were able to produce the results at the same level of quality in terms of product and process as in the face-to-face edition. From our experience, we synthesize lessons learned, and we sketch some guidelines for educators.

CCS CONCEPTS

• **Applied computing** → **Distance learning**; **Collaborative learning**; • **Software and its engineering** → *Collaboration in software development*.

KEYWORDS

Online coding camps, Distance learning, COVID-19, High school, Fun

1 INTRODUCTION

Coding bootcamps, hackathons, and coding-intensive learning experiences have been available for long, and they are continuously leveraged by students and practitioners to start, consolidate or deepen their knowledge on software development and enabling technologies. These learning environments are commonly space-bound, time-limited, and strongly collaborative, which makes them socially strong. Vicinity of participants, face-to-face interactions, and fun are key characteristics that help participants to advance their technical work, share best practices, and grow individually and collectively in expertise. Even though participants reported bootcamps to be more open and inclusive [59], research in computing education found several barriers bootcamp participants might face. For example, stereotypes of *nerdiness* and *intelligence* exist [59] as in other computing education contexts [36]; moreover, considerable perseverance and confidence [59] are needed to face intensive activities.

A forced transition to remote work due to the COVID-19 outbreak brought very important learning: working in a remote setting implies not only the implementation of collaboration tools or a most effective leverage of communication channels. It requires a complete mindset of autonomy, teamwork, collaboration, technological resources, and understanding of goals. Cultivating these traits in early-career students equips them with qualifications that may enable them to mesh in a global community. Future global workers will rely on an excellent command of these tools and an unprecedented development of these characteristics, so the earlier this effort is done, the easier it will be for students to become sooner highly effective global collaborators. Early exposure to a distributed work environment will prove valuable as an effective training field to embrace practices of online work, develop a good command of the use of the enabling technology, and fine-tune the human aspects that allow for remote collaboration, tremendously necessary in the productive world of today and the future. In the process of learning how to conduct and leverage online classes, virtual events, and remote hackathons, several conditions that are unlike to appear in face-to-face settings came up. Since this is a rather recent issue, the existing literature is scarce in reporting and discussing the adjustments or adaptations needed to effectively replicate the traditional highly interactive, fun, face-to-face dynamics into virtual, remote, or online coding camps.

In this paper, we report on our insights gained from designing and applying a fully remote coding camp that exposes high school students to agile-based Software Engineering practices to enhance their ability to develop high-quality software. We use as a baseline a Software Engineering-centric instructional strategy for intensive, face-to-face, project-based events for high school students [15], in which games and fun were found to be a cornerstone of a successful outcome. Therefore, we aim at keeping the same "level of fun" in the online coding camp. For this reason, adaptations for the transition to online aim at increasing communication [24] and a sense of belonging [42], engaging participants [51], and reducing fatigue due to prolonged computer use [68] by proposing unplugged activities that require participants to move around and release energy before focusing again. To evaluate the success of our approach, we extend the face-to-face assessment framework [15] to understand if the online coding camp successfully emulates the development process and achieves the same quality of the developed products with respect to the face-to-face setting, while keeping fun alive.

Based on our experience, we synthesize guidelines for educators. The paper is organized as follows: Section 2 provides background and an overview of related work. Section 3 describes the online coding camp design adapted from the face-to-face version; Section 4 compares the online coding camp with the face-to-face one. Section 5 discusses the experience and synthesizes guidelines. Finally, Section 6 concludes the paper.

2 BACKGROUND AND RELATED WORK

The body of research in Software Engineering (SE) education focuses on developing (in software engineers, computer scientists, and information technology experts) professional skills to abstract real-world problems and deliver solutions in the form of software products. Over the years, researchers proposed classroom experiences in which software processes are mapped efficiently to course sessions; moreover, they presented insightful discussions of practices, behaviors, and interactions among students [37]. In attempts to broaden participation in computing and engage end-users, a wide variety of outreach activities have been proposed [12], including intensive project-based experiences (e.g., bootcamps, hackathons, summer schools), which are increasingly popular [6, 11] and also attract K-12 students to increase their awareness of computing. Radical collocation of participants and face-to-face interactions [48] are key characteristics that help participants quickly advance technical work in coding camps and similar experiences [61]. For this reason, most of the research provided guidelines to organize face-to-face events [15, 18, 22, 34, 45] and investigated their educational advantages when teaching Software Engineering concepts [16, 18], also exploring participants' perspectives [59]. A recent study explored how these events can be harnessed within Software Engineering education to teach the necessary skills and competences for the students [49].

The COVID-19 pandemic made clear that flexible and resilient education systems are needed [1]. The emergency prompted the organizers of coding camps and similar experiences to think whether they could be moved online, not only while waiting for a return to "normal life", but as a possible solution for sustainable development [62]. Although studies found no difference in student performance in conventional remote learning versus in-person learning [30, 47, 57], the emergency remote teaching context goes in a different direction, with students having a very low performance [3, 13]. Moreover, research in the fields of Global Software Engineering [55] and computing education found several specific issues that need to be solved when moving coding camps online, including communication issues [24], lack of a sense of belonging [42], lack of engagement [51], and fatigue due to prolonged computer use [68]. In addition, facilitators need to be able to replicate the face-to-face dynamics and the typical hands-on approach that dedicates little time to explaining fundamental principles in favor of example-centric and copy-paste programming [27, 50].

From a Software Engineering process perspective, online intensive project-based experiences need to integrate different tools and strategies (tailored to the specific audience) to support the entire software development process with special attention to the phases that require strong team collaboration, such as software design [28]. Real-time collaborative code editors and compilers exist for different languages (such as CodeCollab[2]), but they are currently not available for block-based programming languages (BBPL) that proved to be very popular, even among small businesses and entrepreneurs. As a workaround for this problem, the BBPL *App Inventor* programming may leverage *AI2 Project Merger*, in which potentially two authors may produce two distinct functional areas of a project and eventually merge them in a single project[3]. Additionally, even though Agile methods accommodate the K-12 environment [17] and are suitable for development projects facing high uncertainty [8], they rely heavily on face-to-face communication [9]. When coding camps go fully online, questions arise naturally regarding whether agile methods could still offer the same level of accommodation and whether they could be implemented effectively to support the development of projects and teamwork. Finally, according to the recent learning theories, students prefer to have an active role in their learning process [40], and learning depends on reflection upon experiences [31]. In particular, games promote learning by merging educational content and entertainment activities that increase engagement, emotion, and motivation [56]. Research in SE education suggests as well to involve students in active learning experiences that provide *takeaway messages* [15, 54] by mimicking the "real-world" SE [32]. Although games are not the silver bullet [2], researchers proposed games to teach several SE topics, including programming [23], agile concepts [15], global software engineering [53], requirements engineering [32], cross-domain stakeholder-alignment [33], testing [38], and software quality assurance [43]. Moreover, games have been included also in bootcamps [15, 39]. Nevertheless, most of the proposed games have been conceived to be carried out in face-to-face settings because of their key characteristic (i.e., use of tangible objects).

Despite all the related challenges, online intensive project-based experiences have been recently proposed to target problems related to COVID-19 [4, 19, 26, 63]. However, literature is lacking reports on actions needed to take advantage of past face-to-face experiences to

---

[2]https://codecollab.io/#Welcome
[3]https://appinventor.mit.edu/explore/resources/ai2-project-merger

organize online intensive project-based events. In particular, there are limited insights on how to keep the same level of fun to engage participants and reduce their fatigue due to prolonged computer use. This experience report intends to offer the lessons we have learned through designing and running online coding camps.

3 ONLINE CODING CAMP DESIGN

We use as a baseline our Software Engineering-centric instructional strategy for intensive, face-to-face project-based events for high school students [15], which has the same target audience (i.e., highschoolers with little or no previous knowledge of software development) and goal (i.e., to expose participants to agile-based software engineering practices through the development of mobile apps). Games, activities, and team dynamics were the cornerstone of a successful face-to-face coding camp; participants consistently highlighted the importance of engaging games and activities, and mentioned this aspect among one of the reasons why they would suggest coding camp to friends [15]. Therefore, while transforming to a fully remote format, we introduced adaptations to replicate the face-to-face dynamics and the typical hands-on approach. In particular, we aimed at "keeping fun alive", i.e., the main goal of the adaptations was to increase communication [24] and a sense of belonging [42], engage participants [51], and reduce fatigue due to prolonged computer use [68]. This section describes the changes with respect to the face-to-face coding camp [15].

3.1 Instructional strategy

As in the face-to-face coding camp [15], the online version consists of twenty hours of activity (one four-hour session for five days) divided into five sessions as shown in Table 1.

Table 1: Timetable of the online coding camp.

Session	Hours	Activities
1	4	Foundations of logical thinking, structured sequencing, and data abstraction
2-4	12	Iterative development of mobile apps
5	4	Completion and presentation

Table 2 summarizes the instructional strategy in the online setting, which keeps one of the key characteristics of the face-to-face instructional strategy (i.e., learning-by-playing): specific activities let participants reason action courses when there is a need for planning, managing, or empowering. As shown in Table 2, each strategy fosters eXtreme Programming (XP) practices (which fit K-12 education [17, 29, 41]) that participants heuristically mix during their activity. While some strategies (marked with * in Table 2) required limited changes, new games (marked with ** in Table 2) substituted the face-to-face ones. For each new game, Table 2 shows the replaces of face-to-face games and provides a rationale for the replacement. Online games required participants moving around and releasing energy before focusing again. After each game, 15 minutes were reserved for reflections to share thoughts and collect students' opinions on the takeaway message of each game. Thus, 20-30 minutes of every session were dedicated to games.

The remaining part of this section focuses on the changes with respect to the face-to-face coding camp (i.e., * and ** in Table 2).

We are here to help (*). During all the sessions, participants can ask for the facilitator's support by first showing the current release of the app and describing the solutions that they already tested.

Changes: Participants ask for support via the dedicated button in Zoom. The use of breakout rooms places the challenge of obtaining face time with facilitators. For this reason, we opt for a *peer-led event*, i.e., advanced students serve as tutors [14, 35]. Each tutor is assigned three teams and visits the corresponding breakout rooms regularly: interactions regarding questions and answers are done directly, unless the technical need of the question requires the facilitator's opinion.

Block-based programming (*). We keep using a block-based programming environment in sessions 2-5, as it allows problem-driven learning [44] and fosters XP practices [17].

Changes: In an online setting, we can not provide the participants with mobile phones; thus, instead of App Inventor, we use Thunkable (https://thunkable.com), which builds applications both for iOS and Android, to make sure that most of the participants can use their own devices. Otherwise, live testing is also possible using the Thunkable emulator on PC.

Teamwork (*). Facilitators form teams [46] of three students attending three different school types; mixed teams include two females to prevent them from being in a minority within the team [20] and enhance collaboration [58]. Teams choose a logo/name, define the goal of the app, and can collaborate on the same code (by sharing the editor's screen) or develop software parts individually.

Changes: To ease communication and improve the learning experience, we encourage camera usage and explain why we are doing so in session 1 [5]. Participants are notified about this norm the week before so that they can prepare to be comfortable in front of the camera (e.g., room, clothes, etc.); moreover, *show-and-tell* games (e.g., color wheel game) encourage and motivate camera usage. When not in plenary, teams work autonomously in Zoom breakout rooms.

Paper tower (**). This activity substitutes the *marshmallow challenge* [66] of the face-to-face coding camp [15]. Indeed, specific material is needed for the marshmallow challenge and it would be complicated to ensure that all participants (in different locations) have the same type of material (e.g., marshmallows of the same size). For this reason, teams compete (during session 2) in building the tallest freestanding tower in 18 minutes using 20 A4 paper sheets, which are easily available at each location. Being in an online setting, one team member builds the tower, while the others provide suggestions. The takeaway messages of this activity are analogous to the ones of the marshmallow challenge: prototyping and iterating can help achieve success, the importance of collaborating very quickly, and the value of cross-functional teams.

Color wheel (**). This activity substitutes the *tell me how you make toast*, which heavily relies on drawing manually on post-its that are then grouped and analyzed by the team. During session 3, teams compete in creating a color wheel (Figure 1) using the highest number of colors and objects (found in the surroundings) in 15 minutes. Being in an online setting, one team member builds the wheel while the others provide suggestions. The takeaway message

Table 2: Elements of the online instructional strategy (adapted from [15]) and changes with respect to the face-to-face (F2F) coding camp (∗ adapted; ∗∗ new).

	Strategy	Session(s)	Length (min.)	XP Practice	Replaced F2F strategy	Motivation for replacement
	Manipulatable examples	1-5	–	User stories	–	–
	Focus on problem-solving	1-5	–	Small releases, testing	–	
	Alert without imposing	1-5		Refactoring, testing	–	–
∗	We are here to help	1-5	–	Small releases, teamwork, on-site customer (i.e., one of the facilitators took the role of final customers, provided feedback, and refined requirements)	–	–
∗	Block-Based Programming	2-5	–	Continuous integration, refactoring, testing	–	–
∗	Teamwork	1-5	–	Collective ownership, pair programming, metaphor and coding standard	–	–
∗∗	Game: Paper tower	2	18	Prototyping and iterating, quick collaboration, simple design, teamwork	Marshmallow challenge	1) The needed material (i.e., 20 A4 paper sheets) is more easily available at each participant location; 2) it introduces an element of fun when towers fall down
∗∗	Game: Color wheel	3	15	Simple design, teamwork, user stories	Tell me how you make toast	1) It does not rely on manually drawing on post-its; 2) it reduces fatigue due to prolonged computer use by requiring to move around, activating physically and releasing energy before focusing again; 3) it introduces an element of fun when observing the type and amount of objects placed in the wheels
∗∗	Game: Thirty items	4	15	Prototyping and iterating, quick collaboration, teamwork	Letters with our bodies	1) It reduces fatigue due to prolonged computer use by requiring to move around and release energy before focusing again; 2) it introduces an element of fun when observing collected objects
∗∗	Game: Boosting attention games	3-4	10	Teamwork, simple design		It increases engagement and fun, fosters networking, releases energy before focusing again

of this activity is about the importance of working together toward a solution by identifying small steps.

Thirty items (∗∗). This activity substitutes the *letters with our bodies* [15], which is clearly not feasible in an online setting. During session 4, teams compete for 15 minutes in finding 30 items with given characteristics (e.g., shining, useless, broken) at the locations of all team members. The takeaway message of this activity is about the importance of understanding ambiguous requirements (e.g., is it possible to consider an object valid for more than one category?), and team self-organization with little or no guidance from facilitators.

Boosting attention games (∗∗). At the beginning of sessions 3 and 4, the first 10 minutes are dedicated to two games to start focusing: 1) *who likes what?* (Session 3) consists of sharing a bingo-like screen including a series of 9 boxes with items written within. These items refer to hobbies, activities, sports, and entertainment items to choose from. The facilitator reads them out loud (for instance, "Who likes football?") so that participants can add themselves a mark on the box if they feel like they appreciate the mentioned item. This activity has been introduced in the online setting, as it is valuable for networking, for participants to connect with other members with similar preferences. 2) *Gimme five* (Session 4). In this activity, we mention the order in which students' screens are

Figure 1: Color wheel.

sorted in the facilitator screen. For example, "In the first row, we have left to right, John, Maria, and Joe; in the second row we have Bob, Stella, and Tanya". After mentioning this arrangement, we encourage students to "high-five" the persons right next to them according to their screen arrangement. It is amusing and requires a big deal of coordination to have all students high-five each other. In the online setting, these games also aim at increasing engagement and fostering teamwork.

3.2 Participants

Similar to the face-to-face version, the online coding camp targets high school students (aged 15-19) attending different schools (from non-vocational to computer science), i.e., having diversified disciplinary background. Moreover, the participants have little or no previous software development knowledge.

3.3 Assessment framework

The goal of the assessment framework is understanding if the online coding camp successfully emulates the development process and achieves the same quality of the developed products with respect to the face-to-face setting, while keeping fun alive. Therefore, we extended the framework proposed in [15], which included product and process assessment, by including fun assessment. Under consideration of the underlying principles of Project-Based Learning [52], during our coding camps we limit the handing out of test/questionnaires and we prefer critique and revision, supported by observation and code inspections [15].

Fun assessment. Research has demonstrated that fun increases engagement with learning activities [64] and has positive effects on learning outcomes [7]. Despite this, the concept behind the term is not always clearly defined; moreover, there is a lack of reliable measurement tools, especially for adolescents [60]. To understand better the reach and impact that fun activities (i.e., games) had on the participants, the facilitators organized two types of reflection sessions to collect information from participants about perceived fun:

(1) upon completion of each game, there was always time dedicated to reflecting about what could be important lessons that such activity can bring to the professional software development process;

(2) at the end of the coding camp, focus group interviews [10] of around 30 minutes, involving participants and student tutors separately, served to elicit views and opinions to complement the observations collected during the activities and reflection moments.

After asking questions, the facilitators only took notes while encouraging the students to express their opinions. Reflection sessions served to explore how participants connected games to their learning process and how they considered games helpful to reduce fatigue, increase engagement, and *keep fun alive* in the online setting. Student tutors did participate in the previous year's face-to-face coding camp. Therefore, besides reporting how much fun they perceived while observing the participants, they could compare the two editions in terms of fun.

Thematic analysis [10] was conducted on the collected notes based on the factors proposed by Tisza and Markopoulosto to measure adolescents' fun [60], i.e., answers were coded as *fun* when they mentioned specific concepts (such as, curiosity, flying time, new friends, doing something new) and as *not fun* with opposite concepts (such as, feeling bad, angry, sad, or forced to participate).

Product assessment. As we use Thunkable instead of App Inventor, we adapted the framework in [15] to extract five groups of metrics to analyze Thunkable projects from a Software Engineering perspective, namely: component metrics, computational concepts blocks, code smells, complexity metrics, and size (Table 3).

Table 3: Metrics for product assessment (adapted from [15]).

Group	Metric
Component metrics	Number of components by functionality based on the categories in the Thunkable palette: authentication, data, image, layout, screens, sensors, user interface
	Total Number of Components (TNC): the sum of all the components by functionality
	Total Number Of Unique Blocks (NOUB)[67]: length of the distinct list of blocks
Computational concepts	Count of six types of blocks: conditional, function, list, logic, loop, and variable blocks
Complexity	Cyclomatic Complexity (CC): number of decision points in the code plus one
Size	Number of Logical Lines Of Code (LLOC)
Code smells	Component names [65]; Superfluous stuff [65]; duplication [25, 65]; long method [25, 65]; Meaningful variable names [21]

Process assessment. Our framework capitalizes on the one proposed in [15], as it focuses on observations of process-relevant traits focusing on XP practices. However, in the online setting, the use of breakout rooms places the challenge for facilitators of directly observing participants' behavior. To overcome it, we developed a protocol to collect observations (Table 4) so that we could train student tutors and delegate them to observe the assigned teams (on the second and last day of the event) by using the protocol (in a Google Form).

Table 4: Process assessment protocol.

Observed behavior	Corresponding XP practice
1. The team uses paper/digital sketches to drive the development of the app [never / sometimes / often]	User stories and metaphor
2. The team gets ready to present a prototype at the end of each iteration [never / sometimes / often]	Small releases and iterations
3. The team tests frequently [never / sometimes / often]	Refactoring/testing
4. The entire team knows and can modify the code [never / sometimes / often]	Collective ownership
5. Two/three team members work together on same piece of code [never / sometimes / often]	Pair programming
6. The team takes advantage of meetings with customers to get feedback [never / sometimes / often]	On-site customer

Based on the questions in the observation protocol, the student tutors observed the teams continuously and reported the results of their observations to the facilitators. Using the collected data, the facilitators could identify any critical issues and support the student tutors to manage them. In doing so, we turned the challenge into the opportunity that allowed us to understand better participants' behavior through analyzing the information that could not be gathered easily in the face-to-face version.

4 FACE-TO-FACE VS. ONLINE CODING CAMP

This section first assesses whether the online camp kept alive the fun that participants typically obtained in the face-to-face edition, then compares the achieved results in terms of quality of product and process in the two editions. The comparison allows us to understand if our effort of replicating fun and engagement in the online version produced desired results, which are reflected in the quality of product and process.

The coding camp hosted 80 participants (14 F, 66 M) from ten different high school types (computer science, scientific, vocational, and non-vocational), i.e., participants had diversified disciplinary backgrounds. The participants were aged 15-19 and had little or no previous software development knowledge. In total, 27 teams that were tutored by a group of 15 second/third-year students (7 F, 8 M) who attended the previous year's face-to-face coding camp. Each tutor was assigned one or two teams.

The activities started on Monday afternoon and concluded on Friday afternoon, four hours per day. Two authors of this paper facilitated all the sessions and also the face-to-face version described in [15], which hosted 28 participants (6 F, 22M) having similar characteristics and organized in 10 teams.

Fun assessment. During reflection sessions, the participants never reported not-fun factors (e.g., boredom or sadness [60]). Instead, they frequently mentioned fun and explained how *fun activities* changed the context and dynamics of the course (*"I wondered how to survive four hours of sitting in front of the screen. Instead, time passed more quickly than I thought"*; *"Games helped us socialize with people we did not know at first, which is definitely more difficult in an online context"*), assuring moments of fun (*"At the end of the thirty items game, my jaws were aching from laughing"*) and relaxation (*"Thanks to these activities I had to run all over my apartment, which helped stretching my legs and resting my eyes"*) . On top of that, the students commented that even though fun activities can be perceived upfront as unrelated eventually, they were able to find a connection and eventual learning that does relate to the software development process (*"At first, I wondered why I had to do parkour to find objects [...] Then, I realized that deciding who had to run around to look for what was a great team game!"; "When we were developing our app, we often thought about the paper tower metaphor"*).

According to the observations collected by the facilitators and tutors, the participants' engagement was a clear indicator of fun. Indeed, they all did their best to achieve good results; to mention one example, some participants went all the way to the kitchen to find carrots to increase the number of orange objects in the color wheel. Moreover, the *camaraderie* effect was pronounced and led to the different teams playfully competing against each other. The student tutors confirmed that they considered fun to be an essential ingredient of the face-to-face coding camp; for this reason, they were initially concerned that the online experience would be very different for new participants. However, they confirmed that they could observe fun during the coding camp and that the games played a crucial role in maintaining the participants' fun and engagement.

Product assessment. Figure 2 compares size and complexity of the projects developed at the online coding camp (27 projects in total, one per team) with those of the face-to-face edition (10 projects [15]). The only noticeable difference is represented by project size (LLOC) being slightly lower in the online edition, but still comparable to the face-to-face edition. This leads to the consideration that the capacity of the teams in terms of size and complexity of the product does not decay because of the variation in the course delivery channel.

Figure 3 shows the presence of *components* (based on the categories in the Thunkable palette) in the projects of the online coding camp. User Interface (UI) components are the most present, while other components (such locations/sensors and authentication) are used less frequently. This result is similar to what happened in the products developed face-to-face [15].

Figure 4 compares the number of components in the two editions of the coding camp (i.e., face-to-face and online). We do not compare *data*, which is almost absent in both editions, and *authentication*, which was only present in the Thunkable palette). Moreover, although *image* components are quite common in the online edition, their distribution is not compared to the face-to-face edition because the *image* category was not present in App Inventor with the same features as the category in Thunkable. Figure 4 shows no major variations, which were indeed not expected as we did not change the portfolio of topics offered during the online coding camp with respect to the face-to-face syllabus.

Figure 5 shows that, in the online coding camp, loops and lists are the least used *computational concept blocks*, while variables, logic, and function blocks are frequently used.
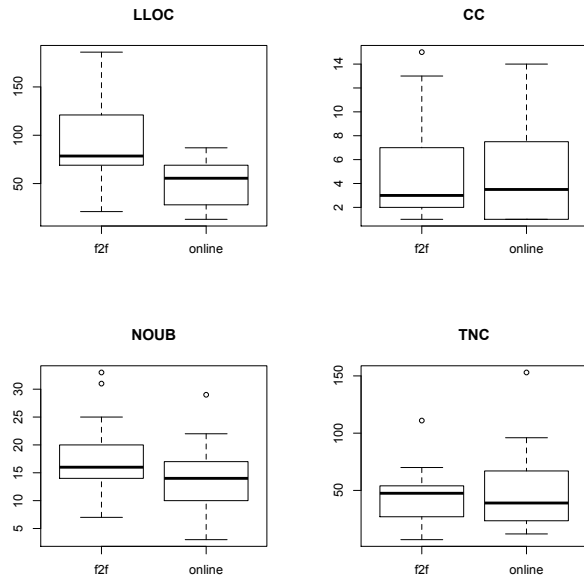
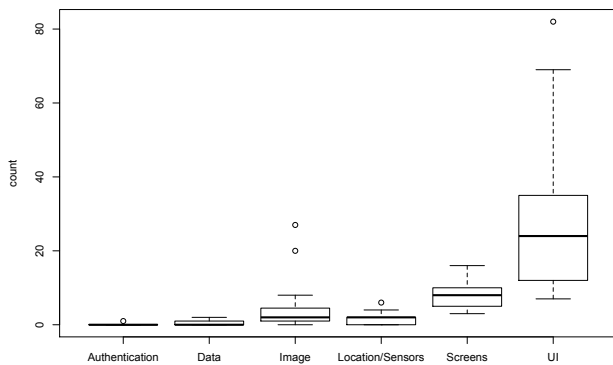**Figure 2: Size and complexity: face-to-face vs. online coding camp.**
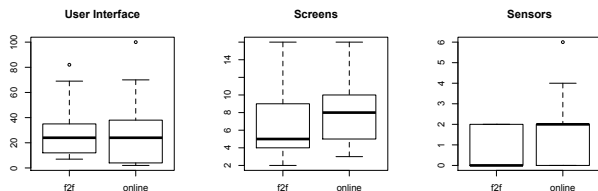

Figure 3: Online coding camp: components per type.

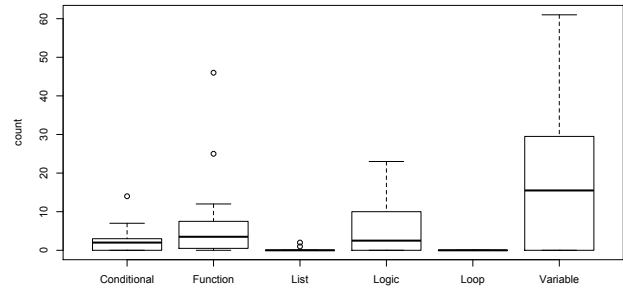**Figure 4: Components: face-to-face vs. online coding camp.**

Figure 5: Online coding camp: computational concept blocks.

Also in this aspect, the trend offered by products developed remotely is similar to the trend observed in the face-to-face delivery method (Figure 6).
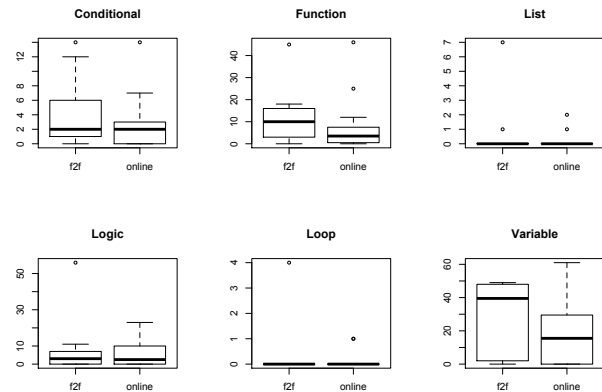

Figure 6: Computational concept blocks: face-to-face vs. online coding camp.

The code inspection of the 27 projects developed during the online coding camp shows that half of the projects (i.e., twice the value of the face-to-face edition) suffer from the superfluous stuff code smell *code smell* (Figure 7). Instead, duplication is a rather limited problem, less present with respect to the face-to-face edition. Compared to the face-to-face coding camp, an increased number of pair programming sessions have been observed, which might have contributed to reducing duplication (and to lowering project size down). Superfluous stuff in the code (i.e., discarded blocks left around) might be explained by how blocks are deleted in Thunkable: instead of dragging blocks back into the palette as in other BBPLs that participants might have already used (e.g., Scratch), it is, in fact, necessary to drag blocks to the bin in the corner; otherwise, blocks remain abandoned in the project.

Process assessment. Student tutors collected data on the second and last day of the coding camp using the observation protocol
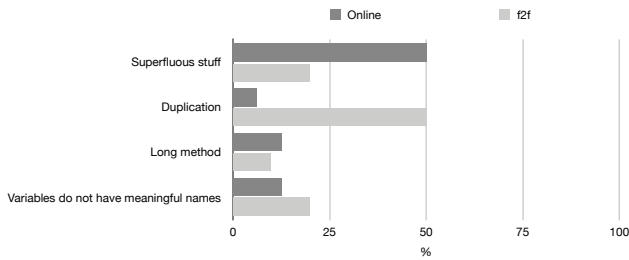
Figure 7: Code smells: face-to-face vs. online coding camp.

(Table 4), where each observation was an indicator of an XP practice. Figure 8 compares the usage of each XP practice at the two observation days.
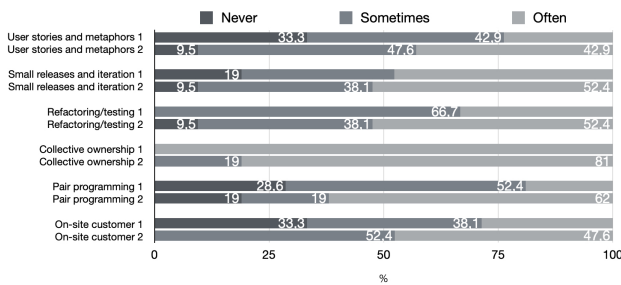

Figure 8: Process assessment (beginning (1) and end (2) of the coding camp).

Most of the teams used *user stories and metaphors* (especially during the second part of the coding camp), i.e., they frequently used paper/digital sketches to drive app development. This could be explained by geographic distance affecting distributed design sessions [28]. *Small releases and iteration* and *refactoring/testing* were widely used, while the possibility to get feedback from the *on-site customer* was leveraged more towards the end of the coding camp. Despite the distributed setting, the teams managed to maintain *collective ownership* of the code and to work in *pair programming* sessions.

Similar to the face-to-face delivery method (in which a similar observation protocol was not used), agile practices are highly promoted by facilitators during the coding camp, so observing their implementation is not surprising. However, it is important to note that students had to be more resourceful and creative in creating a proper setting to carry out some agile practices remotely (for example, pair programming sessions).

5 DISCUSSION

The results evidence that it is possible to emulate the development process and achieve the same quality of the developed product in both settings, while keeping fun alive in online coding camps, when the instructional goal of the camp is accompanied by support activities. Fun and relaxing items in the camp's agenda ensure not only the existence of mere "knowledge transfer" tasks, but also elements that activate participants promoting an overall engagement towards the technical content of the camp. The results detailed

in Section 4 show that the effort adapting face-to-face activities enabling them to be executed fully remote supported the attainment of the desired technical results, which are observed through the quality of products developed, and the stages of the process executed, extensively discussed in Section 4 as well.

5.1 Lessons Learned on the Methodological Approach

- One great opportunity enabled by the online coding camp, which we were able to grasp and utilise, is that we provided student tutors with an observation protocol so that they could help us monitoring the teams' activities in their Zoom breakout rooms. It helped us understanding better participants' behavior through analyzing the information which we could not be gathered easily in the face-to-face version, but now with online edition we could collect more easily. Before starting the coding camp, it is important to unify among student tutors the observation method and the expectations as to what results such observations should convey.

- In the online version of the coding camp, we aimed at keeping the same type of fun generated by the activities and games in the face-to-face camp, therefore the adaptations of the games emulated the face-to-face versions and intended to keep as much as possible the physical nature of the replaced games. We demonstrated this can be done and can help to achieve the same intended results as in the face-to-face versions. An opportunity is to explore more on what "fun" means in online settings, and what are the online "fun factors", and blend online activities with physical activities to enhance the fun that the participants could obtain, and in turn achieve better learning outcomes. This blended approach could also better attend the differences in the participants in terms of personality, background and other personal traits. Different participants can use the blended activities to achieve the same level of fun.

- Due to the unexpected COVID situations, we had to manage to run online coding camps as quickly as possible. Therefore, it is a natural choice of adapting our existing face-to-face version and making the minimal changes possible. A missed opportunity here is to design online coding camps with a clean slate, making the best use the full potential of online tools and platforms, and design new instruction, interactivity as well as time management strategies. This is a direction worth exploring further, with fresh perspectives.

- In this experience report, we presented the assessment results of the online coding camp without considering the participants characteristics (e.g., diversity of previous experiences, previous courses taken, previous learn/work experience online, extroverts or introverts) which might have influenced the results, since we only knew limited background information of the participants. This reminds us that the need of finding a valid way of collecting the background information of participants so that we could inspect if participants encounter barriers [59] and if these barriers depend on background characteristics.

5.2 Lessons Learned on the Incorporation of Games and Fun activities

Some of the insights from the introduction of fun and activating activities in an online coding camp include:

- Games are essential as a strategic activity to reinforce learning: fun, engagement, move around, change context, and release energy, especially in sessions of several hours.
- Games must be designed in such a way as to have a take-away message that relates to the subject matter at hand (in this case, to Software Engineering).
- To obtain such take-away, instructors should reserve a reflection moment to share thoughts and reflections after the game. Otherwise, participants can be distracted about the real point of playing that game, or may be disappointed because they cannot show what they have done.
- Participating in the game and succeeding in its goal should be enabled by simple material available wherever the participant is, or found easily if notified with reasonable notice (for example the day before).
- When conducted virtually, instructors and student tutors should dedicate time to manage the games in breakout rooms to allow for exposure and interaction. Even if there are few rooms, participants are "alone" in the rooms with little or no guidance, whilst in the face-to-face version, games are played in a co-located classroom where facilitators and participants work together. Moreover, a reserved staff resource (technical facilitator) must also take care of the management of the breakout rooms so instructors can focus on mentoring.
- Solution sharing and reflection after the games should be run in the main Zoom room because they help keeping a connection between the teams. During the experience presented in this paper, when they later worked in breakout rooms, the teams occasionally visited other breakout rooms to help solving issues or to cross-check apps and provide comments, as what happened during the face-to-face coding camp [15].
- As any context-changing activity, there is always a risk that playing a game disrupts the didactic pace of a class. By completing a totally different activity, the group should be ready to switch gears, resume the technical content and proceed. This makes the take-away speeches and the reflection discussions mentioned above especially relevant.

5.3 Lessons Learned on Remote Teamwork and Collaboration

With regard to teamwork, collaboration and communication activities, our major insights are:

- Based on the tutors' observations, the facilitators may as well intervene to encourage the group to attain a solution, or to mentor the group to explore alternative ways to find a solution.
- The teaching staff should give as much support and facilitation as possible to enable collaboration even in the offline aspects of the course (for example, if an exercise requires the class to draft a user interface with pencil and paper).

Facilitators must inspire in this sense, using tools that the participants can replicate (such as whiteboards or paper sheets) instead of sketching or prototyping licensed tools that others may not access.

- In a face-to-face setting, it is always easy to turn the head and see what students in the next table are doing. In the online edition, partial and final presentations acquire particular relevance, because the teams may lose track as to what other teams are doing, or what could be their final outcome.

5.4 Lessons Learned on Enabling Technology

The success of a technical camp relies much on the technology available to carry out the activities required by the course. In this regard, we observed the following learning:

- The enabling technology should be selected to be executed in a common platform that can be easily supplied in any setting. For instance, working with Thunkable permits to execute a fully web-based development environment without the need of installing any software. The developed software can be tested via web, or if participants opt in, the two major mobile operating systems (iOS, Android) are fully supported by the tool. Still, minor problems commonly arise and the teaching staff should be available to mitigate eventual technical problems.
- A face-to-face course depends on the infrastructure provided by the course manager. A distributed, remote participation depends a lot on the technology, connectivity and infrastructure provided directly by participants. To this end, instructors should plan ahead what to do, how to deal with, and advise others what to do if technology (internet connection, software tools) were to fail, both on the instructor's and the student's end. This plan is to be discussed by the beginning of the course, to mitigate anxiety, promote resourcefulness, and creating a technology-resilient environment for all participants.

6 CONCLUSIONS

In this experience report, we described the implementation of a fully-remote coding camp directed to high school students, and the evaluation of the results of the experience from the point of view of assessing the ability to develop high-quality software following an agile-based Software Engineering process while keeping the fun alive. This work yields relevant results to configure future strategies that enable students to embrace a work environment to collaborate remotely with peers elsewhere in the world, simulating and stimulating a working environment that is common in a professional setting. The results show that the participants were successful attaining the goals of the camp, by constructing a working software product utilizing block-based programming tools, yet displaying important deficiencies related to the good command of structured programming. Informally, students reported overall satisfaction on the implementation of Software Engineering practices.

A solid technical course should consider a number of didactic goals that are accomplished via traditional knowledge transfer activities (expositions, supervised exercises, repetition, and independent

work). However, we believe that context-changing, gamified activities deliver important value in the process of obtaining new knowledge, as these activities fulfill a two-fold strategy: on the one hand, they reinforce messages related to the technical track of the course, and such messages are effectively discussed in the take-away sessions. On the other hand, games allow for necessary relief when spending long sessions online, and permit physical activation that is necessary after long periods of sitting in front of a screen.

Even though not targeted by this experience report, replicability and reproducibility efforts are highly recommended to generalize the conclusions yielded by this work. The observations discussed in this work follow the technical and didactic strategy of a very specific course, so it is recommended to extend the scope of this strategy to other contexts. Moreover, we advise that instances of the reproduction of this work occur in a non-distributed setting, to determine quantitatively the impact that the co-location or the remote setting can have on groups, with measurable accounts on which specific points create a major difference.

7 ACKNOWLEDGMENTS

REFERENCES

[1] Wahab Ali. 2020. Online and remote learning in higher education institutes: A necessity in light of COVID-19 pandemic. *Higher education studies* 10, 3 (2020), 16–25. https://doi.org/10.5539/hes.v10n3p16

[2] Sarah Beecham, Tony Clear, Daniela Damian, John Barr, John Noll, and Walt Scacchi. 2017. How best to teach global software engineering? Educators are divided. *IEEE Software* 34, 01 (2017), 16–19. https://doi.org/10.1109/MS.2017.12

[3] Lee Millar Bidwell, Scott T Grether, and JoEllen Pederson. 2020. Disruption and difficulty: Student and faculty perceptions of the transition to online instruction in the COVID-19 pandemic. In *COVID-19*. Routledge, 31–46. https://doi.org/10.4324/9781003142065

[4] William S Bolton, Shu Ng, Angela Lam, James Kinch, Victor Parchment, William P Foster, Manuela R Zimmermann, Jye Quan Teh, Abigail Simpson, Karisma Sharma, et al. 2021. Virtual hackathon to tackle COVID-19 unmet needs. *BMJ Innovations* 7, 2 (2021). https://doi.org/10.1136/bmjinnov-2020-000456

[5] Frank R. Castelli and Mark A. Sarvary. 2021. Why students do not turn on their video cameras during online classes and an equitable and inclusive plan to encourage them to do so. *Ecology and Evolution* 11, 8 (2021), 3565–3576. https://doi.org/10.1002/ece3.7123

[6] Champagne, J. 2016. Are coding bootcamps worth it? . https://blog.capterra.com/are-coding-bootcamps-worth-it/.

[7] Simon CH Chan, CL Johnny Wan, and Stephen Ko. 2019. Interactivity, active collaborative learning, and learning performance: The moderating role of perceived fun by using personal response systems. *The International Journal of Management Education* 17, 1 (2019), 94–102. https://doi.org/10.1016/j.ijme.2018.12.004

[8] Alistair Cockburn. 2002. *Agile Software Development*. Addison-Wesley Longman Publishing Co., Inc., USA.

[9] A. Cockburn and J. Highsmith. 2001. Agile software development, the people factor. *Computer* 34, 11 (2001), 131–133. https://doi.org/10.1109/2.963450

[10] John W Creswell and J David Creswell. 2017. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

[11] Adrienne Decker, Kurt Eiselt, and Kimberly Voll. 2015. Understanding and improving the culture of hackathons: Think global hack local. In *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–8. https://doi.org/10.1109/FIE.2015.7344211

[12] Anita DeWitt, Julia Fay, Madeleine Goldman, Eleanor Nicolson, Linda Oyolu, Lukas Resch, Jovan Martinez Saldaña, Soulideth Sounalath, Tyler Williams, Kathryn Yetter, et al. 2017. What we say vs. what they do: A comparison of middle-school coding camps in the cs education literature and mainstream coding camps. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 707–707. https://doi.org/10.1145/3017680.3022434

[13] Per Engzell, Arun Frey, and Mark D Verhagen. 2021. Learning loss due to school closures during the COVID-19 pandemic. *Proceedings of the National Academy of Sciences* 118, 17 (2021). https://doi.org/10.1073/pnas.2022376118

[14] Ilenia Fronza, Luis Corral, Gennaro Iaccarino, and Claus Pahl. 2021. Enabling Peer-Led Coding Camps by Creating a Seed Effect in Young Students. *SIGITE 2021 - Proceedings of the 22nd Annual Conference on Information Technology Education*, 117–122. https://doi.org/10.1145/3450329.3476860

[15] Ilenia Fronza, Luis Corral, and Claus Pahl. 2020. End-user software development: Effectiveness of a software engineering-centric instructional strategy. *Journal of Information Technology Education: Research* 19 (2020), 367–393. https://doi.org/10.28945/4580

[16] Ilenia Fronza, Luis Corral, Claus Pahl, and Gennaro Iaccarino. 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*. 248–253. https://doi.org/10.1145/3368308.3415391

[17] Ilenia Fronza, Nabil El Ioini, Claus Pahl, and Luis Corral. 2019. Bringing the benefits of Agile techniques inside the classroom: a practical guide. In *Agile and Lean Concepts for Teaching and Learning. Bringing Methodologies from Industry to the Classroom*. Springer, Singapore. https://doi.org/10.1007/978-981-13-2751-3_7

[18] Kiev Gama. 2019. Developing course projects in a hack day: an experience report. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 388–394. https://doi.org/10.1145/3304221.3319777

[19] Kiev Gama, Carlos Zimmerle, and Pedro Rossi. 2021. Online Hackathons as an Engaging Tool to Promote Group Work in Emergency Remote Learning. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Virtual Event, Germany) *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 345–351. https://doi.org/10.1145/3430665.3456312

[20] Elizabeth Gammie and Morag Matson. 2007. Group assessment at final degree level: An evaluation. *Accounting Education: an international journal* 16, 2 (2007), 185–206. https://doi.org/10.1080/09639280701234609

[21] S Grover. 2017. Tackling novice learners' naive conceptions in introductory programming. *Hello World* 2 (2017).

[22] Ari Happonen, Daria Minashkina, Alexander Nolte, and Maria Angelica Medina Angarita. 2020. Hackathons as a company–University collaboration tool to boost circularity innovations and digitalization enhanced sustainability. In *AIP Conference Proceedings*, Vol. 2233. AIP Publishing LLC, 050009. https://doi.org/10.1063/5.0001883

[23] Robert Heininger, Loina Prifti, Victor Seifert, Matthias Utesch, and Helmut Krcmar. 2017. Teaching how to program with a playful approach: A review of success factors. In *2017 IEEE global engineering education conference (EDUCON)*. IEEE, 189–198. https://doi.org/10.1109/EDUCON.2017.7942846

[24] James D Herbsleb and Deependra Moitra. 2001. Global software development. *IEEE software* 18, 2 (2001), 16–20. https://doi.org/10.1109/52.914732

[25] Felienne Hermans and Efthimia Aivaloglou. 2016. Do code smells hamper novice programming? A controlled experiment on Scratch programs. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 1–10. https://doi.org/10.1109/ICPC.2016.7503706

[26] Faisal Hossain, Nicholas Elmer, Margaret Srinivasan, and Alice Andral. 2020. Accelerating applications for planned NASA satellite missions: a new paradigm of virtual hackathons during a pandemic and in the Post-Pandemic era. *Bulletin of the American Meteorological Society* 101, 9 (2020), E1544–E1554. https://doi.org/10.1175/BAMS-D-20-0167.1

[27] Daqing Hou, Patricia Jablonski, and Ferosh Jacob. 2009. CnP: Towards an environment for the proactive management of copy-and-paste programming. In *2009 IEEE 17th International Conference on Program Comprehension*. IEEE, 238–242. https://doi.org/10.1109/ICPC.2009.5090049

[28] Rodi Jolak, Andreas Wortmann, Grischa Liebel, Eric Umuhoza, and Michel RV Chaudron. 2020. The design thinking of co-located vs. distributed software developers: distance strikes again!. In *Proceedings of the 15th International Conference on Global Software Engineering*. 106–116. https://doi.org/10.1145/3372787.3390438

[29] Petra Kastl, Ulrich Kiesmüller, and Ralf Romeike. 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*. 60–65. https://doi.org/10.1145/2978249.2978257

[30] Nenagh Kemp and Rachel Grieve. 2014. Face-to-face or face-to-screen? Undergraduates' opinions and test performance in classroom vs. online learning. *Frontiers in psychology* 5 (2014), 1278. https://doi.org/10.3389/fpsyg.2014.01278

[31] David A Kolb. 2014. *Experiential learning: Experience as the source of learning and development*. FT press.

[32] Stan Kurkovsky, Stephanie Ludi, and Linda Clark. 2019. Active Learning with LEGO for Software Requirements. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 218–224. https://doi.org/10.1145/3287324.3287444

[33] Julia Köhlke, Sebastian Hanna, and Johann Schütz. 2021. Cross-Domain Stakeholder-Alignment in Collaborative SoS – Lego©Serious Play©as a Boundary

Object. In *2021 16th International Conference of System of Systems Engineering (SoSE)*. 108–113. https://doi.org/10.1109/SOSE52739.2021.9497469

[34] Miguel Lara and Kate Lockwood. 2016. Hackathons as community-based learning: a case study. *TechTrends* 60, 5 (2016), 486–495. https://doi.org/10.1007/s11528-016-0101-0

[35] Miguel Lara, Kate Lockwood, and Eric Tao. 2015. Peer-led hackathon: An intense learning experience. *thannual* (2015), 255.

[36] Colleen M Lewis, Ruth E Anderson, and Ken Yasuhara. 2016. "I Don't Code All Day" Fitting in Computer Science When the Stereotypes Don't Fit. In *Proceedings of the 2016 ACM conference on international computing education research*. 23–32. https://doi.org/10.1145/2960310.2960332

[37] Janet Liebenberg, Magda Huisman, and Elsa Mentz. 2015. The relevance of software development education for students. *IEEE Transactions on Education* 58, 4 (2015), 242–248. https://doi.org/10.1109/TE.2014.2381599

[38] Beáta Lőrincz, Bogdan Iudean, and Andreea Vescan. 2021. Experience report on teaching testing through gamification. In *Proceedings of the 3rd International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*. 15–22. https://doi.org/10.1145/3472673.3473960

[39] Thomas D Lynch, Michael Herold, Joe Bolinger, Shweta Deshpande, Thomas Bihari, Jayashree Ramanathan, and Rajiv Ramnath. 2011. An agile boot camp: Using a LEGO®-based active game to ground agile development principles. In *2011 Frontiers in Education Conference (FIE)*. IEEE, F1H–1. https://doi.org/10.1109/FIE.2011.6142849

[40] Mosima Anna Masethe, Hlaudi Daniel Masethe, and Solomon Adeyemi Odunaike. 2017. Scoping Review of Learning Theories in the 21 st Century. In *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 1. 25–27.

[41] Orni Meerbaum-Salant and Orit Hazzan. 2010. An agile constructionist mentoring methodology for software projects in the high school. *ACM Transactions on Computing Education (TOCE)* 9, 4 (2010), 1–29. https://doi.org/10.1145/1656255.1656259

[42] Catherine Mooney and Brett A Becker. 2021. Investigating the impact of the COVID-19 pandemic on computing students' sense of belonging. *ACM Inroads* 12, 2 (2021), 38–45. https://doi.org/10.1145/3408877.3432407

[43] Miguel Ehécatl Morales-Trujillo. 2021. Learning Software Quality Assurance with Bricks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 11–19. https://doi.org/10.1109/ICSE-SEET52601.2021.00010

[44] R. Morelli, T. De Lanerolle, P. Lake, N. Limardo, E. Tamotsu, and C. Uche. 2011. Can android app inventor bring computational thinking to k-12. In *Proc. 42nd ACM technical symp. on Computer science education (SIGCSE'11)*. 1–6.

[45] Arnab Nandi and Meris Mandernach. 2016. Hackathons as an informal learning platform. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 346–351. https://doi.org/10.1145/2839509.2844590

[46] Barbara Oakley, Richard M Felder, Rebecca Brent, and Imad Elhajj. 2004. Turning student groups into effective teams. *Journal of student centered learning* 2, 1 (2004), 9–34.

[47] Jasmine Paul and Felicia Jefferson. 2019. A comparative analysis of student performance in an online vs. face-to-face environmental science course from 2009 to 2016. *Frontiers in Computer Science* 1 (2019), 7. https://doi.org/10.3389/fcomp.2019.00007

[48] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. 2018. Designing corporate hackathons with a purpose: the future of software development. *IEEE Software* 36, 1 (2018), 15–22. https://doi.org/10.1109/MS.2018.290110547

[49] Jari Porras, Jayden Khakurel, Jouni Ikonen, Ari Happonen, Antti Knutas, Antti Herala, and Olaf Drögehorn. 2018. Hackathons in software engineering education: lessons learned from a decade of events. In *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*. 40–47. https://doi.org/10.1145/3194779.3194783

[50] Jari Porras, Antti Knutas, Jouni Ikonen, Ari Happonen, Jayden Khakurel, and Antti Herala. 2019. Code camps and hackathons in education-literature review and lessons learned. (2019). https://doi.org/10.24251/hicss.2019.933

[51] Jeaime Powell, Linda Bailey Hayden, Amy Cannon, Boyd Wilson, and Alexander Nolte. 2021. Organizing online hackathons for newcomers to a scientific community–Lessons learned from two events. In *Sixth Annual International Conference on Game Jams, Hackathons, and Game Creation Events*. 78–82. https://doi.org/10.1145/3472688.3472700

[52] Ralf Romeike and Timo Göttel. 2012. Agile projects in high school computing education: emphasizing a learners' perspective. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*. 48–57. https://doi.org/10.1145/2481449.2481461

[53] Aivars Šāblis, Javier Gonzalez-Huerta, Ehsan Zabardast, and Darja Šmite. 2019. Building LEGO towers: an exercise for teaching the challenges of global work. *ACM Transactions on Computing Education (TOCE)* 19, 2 (2019), 1–32. https://doi.org/10.1145/3218249

[54] Bruce A. Scharlau. 2013. Games for Teaching Software Development. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) *(ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 303–308. https://doi.org/10.1145/2462476.2462494

[55] Darja Šmite, Claes Wohlin, Tony Gorschek, and Robert Feldt. 2010. Empirical evidence in global software engineering: a systematic review. *Empirical software engineering* 15, 1 (2010), 91–118. https://doi.org/10.1007/s10664-009-9123-y

[56] Keyur Sorathia and Rocco Servidio. 2012. Learning and experience: teaching tangible interaction & edutainment. *Procedia-Social and Behavioral Sciences* 64 (2012), 265–274. https://doi.org/10.1016/j.sbspro.2012.11.031

[57] Steven Stack. 2015. Learning Outcomes in an online vs traditional course. *International Journal for the Scholarship of Teaching and Learning* 9, 1 (2015), n1. https://doi.org/10.20429/ijsotl.2015.090105

[58] Sachiko Takeda and Fabian Homberg. 2014. The effects of gender on group work process and achievement: an analysis through self-and peer-assessment. *British Educational Research Journal* 40, 2 (2014), 373–396. https://doi.org/10.1002/berj.3088

[59] Kyle Thayer and Andrew J Ko. 2017. Barriers faced by coding bootcamp students. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 245–253. https://doi.org/10.1145/3105726.3106176

[60] Gabriella Tisza and Panos Markopoulos. 2021. FunQ: Measuring the fun experience of a learning activity with adolescents. *Current Psychology* (2021), 1–21. https://doi.org/10.1007/s12144-021-01484-2

[61] Erik H Trainer, Arun Kalyanasundaram, Chalalai Chaihirunkarn, and James D Herbsleb. 2016. How to hackathon: Socio-technical tradeoffs in brief, intensive collocation. In *proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. 1118–1130. https://doi.org/10.1145/2818048.2819946

[62] Peace UNESCO. Division for Inclusion and Education Sector Sustainable Development. 2017. Education for sustainable development goals. (2017).

[63] Silvia Vermicelli, Livio Cricelli, and Michele Grimaldi. 2021. How can crowdsourcing help tackle the COVID-19 pandemic? An explorative overview of innovative collaborative practices. *R&D Management* 51, 2 (2021), 183–194. https://doi.org/10.1111/radm.12443

[64] Luiz Carlos Vieira and Flávio Soares Corrêa da Silva. 2017. Assessment of fun in interactive systems: A survey. *Cognitive Systems Research* 41 (2017), 130–143. https://doi.org/10.1016/j.cogsys.2016.09.007

[65] J Waite. 2017. Smelly code. Do we pass on best practice when we teach block-based programming to primary school pupils. *Hello World* 3 (2017).

[66] Wujec, T. 2010. Build a tower, build a team [Video]. TED Conferences. https://www.ted.com/talks/tom_wu-jec_build_a_tower_build_a_team/.

[67] Benjamin Xie, Isra Shabir, and Hal Abelson. 2015. Measuring the usability and capability of app inventor to create mobile applications. In *Proceedings of the 3rd International Workshop on Programming for Mobile and Touch*. 1–8. https://doi.org/10.1145/2824823.2824824

[68] Shimaa Mohammad Yousof, Rasha Eid Alsawat, Jumana Ali Almajed, Ameera Abdulaziz Alkhamesi, Renad Mane Alsuhaimi, Shrooq Abdulrhman Alssed, and Iman Mohmad Wahby Salem. 2021. The possible negative effects of prolonged technology-based online learning during the COVID-19 pandemic on body functions and wellbeing: a review article. *Journal of Medical Science* 90, 3 (2021), e522–e522. https://doi.org/10.20883/medical.e522