International Journal of Educational Technology in Higher Education

RESEARCH ARTICLE Open Access

Teaching technology with technology: approaches to bridging learning and teaching gaps in simulation-based programming education

Md Golam Jamil* and Sakirulai Olufemi Isiaq

* Correspondence: md.jamil@solent.ac.uk
Research Fellow, Solent Learning & Teaching Institute, Solent University, East Park Terrace, Southampton SO14 0YN, UK

Abstract

The learning of programming using simulation involves unique educational environments and human factors. However, research in this field has been mainly centred on the efficacy of the simulation tool whereas there is a lack of comparative studies between the associated teaching and learning procedures. To address the gap, this study facilitates an evidence-driven discussion on learning and teaching, as well as their relationship, in simulation-based programming education. Investigation areas include virtual and physical environments of simulation sessions, relevant learning enablers and impediments, and roles of students and faculty members in the process. The study followed qualitative methodology using focus groups and semi-structured interviews. Thirty-seven students and four lecturers on a computing course at a British university shared experiences and perceptions on simulation-based programming sessions. The data were analysed thematically and through cross-evaluation. The findings have provided fresh insights on several enabling and challenging aspects of simulation-based programming education. On the one hand, visualisation, consistency of learning procedures, and student engagement emerged as empowering factors. On the other, the negative implications of collaborative tasks, students' attention diversion while shifting between virtual and physical environments of learning, and lecturers' over-emphasis on technology in teaching preparation, appeared as challenges. The paper contributes to understanding the advantages and challenges of using simulation in programming education. It suggests essential teaching principles and their application procedures, which add value to the overall computing education at tertiary level. The learning is transferrable among other engineering programmes and academic disciplines that use simulation for educational purposes.

Keywords: Simulation, Programming, Learning, Teaching, Higher education

Introduction

The learning and teaching of programming is a common element in tertiary computing syllabuses. Programming requires higher level cognitive competences, such as creativity and critical thinking, which are also important in humanities disciplines (Bergin, Reilly, & Traynor, 2005). Programmers need to 'select, reflect, evaluate, justify, communicate and be innovative in their problem solving', making programming demanding and enthralling (QAA, 2016, p. 12). However, historically, there are high dropout rates in this academic subject as many students experience boredom and alienation while learning (Bennedsen & Caspersen, 2007; Giannakos, Pappas, Jaccheri, & Sampson, 2017; Mann & Robinson, 2009). Many academic practitioners also perceive programming as a very difficult subject to teach, thus prompting a rethink about how this subject should be taught (Guzdial, 2015; Jenkins, 2002).

Research findings indicate various educational benefits of using simulation in engineering education including computing (Magana & Silva Coutinho, 2017; Xie, Schimpf, Chao, Nourian, & Massicotte, 2018). Simulation can link real actions of future academic and professional work with similar learning environments and procedures (Kelly, Forber, Conlon, Roche, & Stasa, 2014). In some academic programmes, simulation is used as an alternative for industry placement (Rochester et al., 2012). Because of these advantages, researchers and education practitioners have expanded their knowledge of simulation from the purely technological characteristics to thinking about using technology to teach technology (Harder, 2009; Oliveira et al., 2019). As a result, the need for understanding simulation-based educational activities and best practice has been established (Rystedt & Sjoblom, 2012). However, individual aspects of the learning and teaching of simulation, and their interweaving relationships, in programming education are still under-researched.

Can simulation improve programming skills?

The use of simulation in formal education has been in existence for more than 200 years and the approach has been applied widely in medical, aviation and maritime courses (Woolley, 2009; Wyatt, Archer, & Fallows, 2007). Presently, there is a greater use of simulation in business and education disciplines (Chini, Straub, & Thomas, 2016). Ironically, engineering, more specifically computing education, has not been at the forefront of simulation-based teaching.

There are disciplinary divides in traditional higher education practices because of unique academic objectives and learnings cultures of different academic programmes (Knotts, Henderson, Davidson, & Swain, 2009). As a result, the meaning and applications of simulation are diverse and, in many cases, dissimilar in different disciplines. Despite differences in understanding the construct and its applications, simulation has shown success in creating engaged and meaningful learning environments in various academic fields. Examples include, Patient Simulators in nursing education (Wyatt et al., 2007), Mini Simulators in supply chain courses (Yahaya, Mustapha, Jaffar, Talip, & Hassan, 2017), and TeachLivE simulators in teacher development (Chini et al., 2016; Dieker, Straub, Hughes, Hynes, & Hardin, 2014). Evidence of enhanced student learning experiences in these disciplines indicates potential advantages for simulation-based programming education in computing.

In engineering, simulation broadly refers to a technological device or model which can facilitate elements of reality for supplying practical experiences and learning enhancement (McGaghie, Issenberg, Petrusa, & Scalese, 2010). With many creative and interactive features, simulation-based programming education promises a great deal of benefits, for example self-directed and specialised learning content (Bryan, Kreuter, &

Brownson, 2009); 'learning by doing', the approach that follows Kolb's experiential learning model (Kolb, 1984); and conscious and repetitive practice which can help gain mastery of certain technical skills (Sawyer et al., 2011).

Traditional vs simulation-based programming education: an overview

Creating a computer programme generally follows the process of code writing to enable the computer to carry out specific tasks. This process involves code writers' understanding of language, procedural steps and the computer, hence, the importance of high-level languages. Commonly, problems of code writing arise from the lack of a shared understanding of language syntaxes and language semantics, whether this is high- or low-level language, using the code editor called the Integrated Development Environment (IDE). Therefore, the code writing process is prone to semantic and syntax errors, making it a complex task for programmers, particularly the novices. In recent years, the traditional ways of completing programme codes with a code editor has become easier due to several supporting features, such as debugger, colour-coding, code-lining and syntax error highlights. However, a simulation environment for code writing enables further steps, such as error notification, suggestive corrections, visual display of steps and data processing. For instance, in this research, the simulation work was based on the use of Web programming that entails client and server-side technologies i.e. codes are designed to perform tasks on both client and server sides. A high-level programming language, PHP (acronym for Hypertext Preprocessor) was adopted for writing codes and various concepts including database communication, server-side scripting, data format, SQL (Structured Query Language) injections and web services.

A significant difference between the traditional and simulation-based environments is the visual representation of activities from the front end to the back-end i.e., both on the client and the server sides. Traditionally, IDEs for code writing using PHP do not provide a clear understanding regarding how and what transpires i.e., what action a specific line of code is carrying out and how data move from one end to the other. This process is more transparent in simulation-based teaching.

By acknowledging the unique features of programming simulation, this study explored the perceptions of the lecturers and students about their learning experiences and technical know-how related to simulation-based programming. The students were tasked with undertaking similar concepts to those commonly taught in any traditional higher education programming modules.

So, what is the problem?

Researchers have discussed various educational aspects of programming, such as the influence of learning culture (Sharma & Shen, 2018), roles of student engagement and teaching (White, 2017), and the creativity and applied features of pedagogies (Kujansuu & Tapio, 2004). Additionally, simulation techniques, and associated algorithm or complete programme animation have been explored from a design point of view, as well as on the use of simulation as a demonstration tool in education (Korhonen, 2003; Makransky, Thisgaard, & Gadegaard, 2016). In this connection, several support systems have been proposed to improve programming education, but still various learning-related difficulties are reported by users (Gomes & Mendes, 2007). However, a detailed discussion on programming pedagogy in simulation environments has not developed in the literature although programming simulation involves distinct subject matter and unique educational approaches. Therefore, a comprehensive understanding of the learning environment as well as the role of lecturers and students is important to address the pedagogic difficulties and suggest solutions accordingly.

In simulation-based teaching, curricular content is generally integrated for providing standardised practical learning experiences (Gonczi, 2013; Park, 2016). This approach offers collaborative and supportive learning opportunities for imitating risky actions in a safe and corrective learning environment (Jeffries, 2012). However, simulation itself cannot lead to effective learning if the design and facilitation are not properly conducted (Dieckmann, 2009; Kelly, Hopwood, Rooney, & Boud, 2016). Pedagogical barriers, for example lack of study resources, inadequate teaching preparation and professional development, and teachers' lack of simulation experience may hamper the success of the learning and teaching (Hayden, 2010). There is also a gap of theoretical understanding about how simulation contributes to learning (Bland, Topping, & Wood, 2011). Moreover, it is not very clear how a technological device or model assists transparent and systematic simulation procedures (Tun, Alinier, Tang, & Kneebone, 2015). Therefore, it is essential to discuss the concepts of simulation linked to educational principles. It is also important to balance the teaching-focused and learning-focused theoretical explanations (Kaakinen & Arwood, 2009). Hence, we need a rigorous discussion of simulation-based pedagogy, particularly its relevance, challenges and the solutions.

Scope of exploring simulation-based programming pedagogy

Programming students require higher levels of creativity and imagination as well as concrete understanding of functional procedures in computing (Ma, Ferguson, Roper, & Wood, 2011; Tuomi, Multisilta, Saarikoski, & Suominen, 2018). In this regard, a substantial level of attention, engagement and understanding is vital (Craft, Chappell, & Twining, 2008; Kujansuu & Tapio, 2004). According to research findings, these cognitive and behavioural features are influenced and developed by teaching approaches, the relationship between teachers and students, and the physical environments and resources (Davies et al., 2013). Therefore, to design and implement effective programming education, it is important to explore what pedagogical approaches impact positively on students' programming related competences, and how they influence the change process within the learning conditions.

Emerging educational concepts

Generally, inquiry and multiple perspectives in formal education are expected to facilitate authentic learning (Tong, Standen, & Sotiriou, M. (Eds.)., 2018; Lombardi, 2007). The learning process also needs to be resource rich in terms of students' time and effort; and real-world relevant, for example connected with their future profession and work (Blackmore, 2009). Additionally, there is a strong role of students' prior knowledge and skills, such as known concepts, learning expectations and learning strategies on their academic achievements (Nasir & Hand, 2006; Schmidt, Rothgangel, & Grube, 2017; Schoenfeld, 1999). In science education, maintaining connections and consistency of tasks is an important requirement (Sikorski & Hammer, 2017). The current literature on computing, programming and simulation related pedagogy highlights strengths of collaborative, flipped, and cognitive apprenticeship-based educational models.

Collaboration

Programming and coding are considered as digital humanities and software-as-art, which demand co-creation of ideas and shared learning (Blackwell, McLean, Noble, & Rohrhuber, 2014). Generally, in engineering, collaboration is treated as an essential strategy and enabler for enriching students' learning capabilities as well as generating diverse engineering outputs (Martin-Gutierrez, Fabiani, Benesova, Meneses, & Mora, 2015). Therefore, it is important to involve collective notions in programming designs by allowing students to share critical minds and creative ideas.

It has been established by research that participatory simulation can engage students through collaboration and problem solving amongst other characteristics of the learning (Colella, 2000). However, the goals of programming involve independent accomplishments as well as the ability to work in team. This educational objective is unique because some academic disciplines emphasise more on frequent collaboration by students for achieving meaningful learning.

Collaboration is essential in programming education for various reasons. For example, programming is a flexible and creative academic subject which encourages students to follow different design procedures and application purposes while undertaking the same task. However, simulation sessions create a unique learning and teaching environment where students may not find traditional collaboration and group activities supportive to this type of learning, mainly because of the over dominance of the virtual environment.

Flipped approach

Flipped Classroom is comparatively a modern learning and teaching approach that expands students' individual learning space through the inclusion of pre- and post-group learning practices (Flipped Learning Network, 2014; Bergmann & Sams, 2012). This educational model is built on the core idea of utilising students' prior knowledge which they can apply in face to face academic sessions through active learning activities with ongoing support by faculty members (Latulipe, Rorrer, & Long, 2018). There is an emerging trend in using partial or full flipped approach in computing education (Sanders, Boustedt, Eckerdal, McCartney, & Zander, 2017).

In flipped classes, the teaching and learning of regular academic sessions are determined by some pre-session learning activities (Karabulut-Ilgu, Jaramillo Cherrez, & Jahren, 2018). Students gain prior knowledge through accessible educational materials, such as readings, videos, and online lectures. Flipped classes can save time for deeper interactions between teachers and students in class; can provide flexible learning opportunities to students, and more opportunities for teachers to provide feedback on students' performances (Karaca & Ocak, 2017; Tucker, 2012). However, the approach may contain limitations, such as lack of student input and weak classroom teaching, if the instructional plans are not properly designed and implemented (Chen & Chen, 2014). Research findings suggest that flipped approach is more effective in computing education when students learn collaboratively and use social interactions in the process (Latulipe, Long, & Seminario, 2015).

Cognitive apprenticeship model

Programming involves dynamic actions, more specifically practical and often independent problem-solving skills (Gomes & Mendes, 2007). Apprenticeship model is one approach which can help students solve problems in programming sessions by using expanded 'vision of expertise' and particular skills suitable in changed situations (Morley, 2018).

Traditionally, in an apprenticeship model, a teacher or an expert demonstrates to learners or trainee practitioners how a task can be accomplished, and then helps them do similar tasks by themselves. Contemporary apprenticeships offer wider educational opportunities as they suggest expanding the interaction and collaboration among peers and communities (Eraut, 2004; Morley, 2016). The process is generally observable, and it prepares students with specialised learning skills (Donaldson, 2015).

Cognitive apprenticeship, a modern and extended apprenticeship approach, involves critical analysis and systematic reflections in the teaching and learning where students gradually transform from observers to skilled performers (Collins, 2006; Collins, Brown, & Newman, 1989). This approach encourages students to gain diverse learning experiences through a variety of complex assignments (Dennen & Burner, 2008). Previous work, for example Olsson, Mozelius & Collin (2015), evidenced that programming concepts are easily comprehended by learners through various channels of visualisation. As a result, learners can demonstrate substantive control and understanding of improved participation with the use of visual tools.

The focus of the apprenticeship approach is generally to support the individual development of learners through improving occupational skills (Fuller & Unwin, 2011). An apprenticeship approach can help lecturers communicate these complex tasks more efficiently through demonstrations (MacLellan, 2017). It can also expand their capacity to enhance students' confidence and programming expertise by ensuring informed and immediate support during the task progression. While following a cognitive apprenticeship approach, lecturers may support students through providing practical exercises and real-time feedback (Crick, Davenport, & Hayes, 2015).

The starting point

Simulation-based programming sessions contain unique educational objectives as well as experiential, repetitive and self-directed learning procedures. Therefore, the teaching principles and approaches for such learning environment need to be considered carefully. However, discussions on simulation-based programming education are very limited, thus effective teaching guidelines for this type of academic programme are not available in the literature. The studies in this field are mainly centred on the design of simulators and students' comprehension of the taught concepts, particularly in areas of computer architecture and assembly language (Nova, Ferreira, & Araujo, 2013; Topaloglu & Gurdal, 2010). In addition to this, there are a few studies on student learning dynamics, for example, students' learning behaviours by Suzuki, Hirokawa, Mukoyama, Uehara, and Ogata (2016). Additionally, for the last three decades, there has been a development of discussion on the use of visual algorithm simulation (TRAKLA), a graphical manipulation tool, for providing learning improvement through digital assessment over traditional pen and paper approach (Hyvonen & Malmi, 1993; Korhonen & Malmi, 2000). TRAKLA2, on the other hand, is a framework that was developed for the automatic assessment with a structured schema for new programming exercises (Malmi et al., 2004). Generally, these visual algorithm simulation systems address assessment activities based on electronic feedback techniques such as integrated chats, email and discussion groups. Eventually, the human factors (human elements in teaching and learning) as well as the elements of educational environment and associated learning processes have not been extensively explored. Similarly, the psychology of the visualisation of live code in programming is also under-investigated, thus learning processes in the simulation environment and the role of lecturers as facilitators are not well-explained (Aleksic & Ivanovic, 2016; McLean, Griffiths, Collins, & Wiggins, 2010. Therefore, exploring effective teaching principles and their application guidelines in programming simulation may add value to this particular area of computing education.

The findings of our previous research (Isiaq & Jamil, 2018) indicated a strong relationship between behavioural, cognitive and emotional dimensions of student engagement in simulation-based programming sessions. We found simulation capable of facilitating personalised learning, engagement and links between learning content and students' future work and profession. The findings also revealed stronger collaboration and focus on learning goals in simulation sessions compared to traditional sessions. However, the study showed the need for more cognitively challenging tasks for students to accomplish meaningful learning. The key lesson we learned is that the use of simulation for delivering programming sessions becomes more effective when the pedagogical activities involve a balanced intervention of behavioural, emotional and cognitive exercises. This has encouraged us to study feasible approaches to designing and implementing suitable pedagogical activities which can ensure a balance in the learning and teaching. Our current research in this article builds on these insights. We wanted to explore the perceptions of students and lecturers for understanding the educational situations as well as the associated challenges and advantages of simulation-based programming sessions. Our expectation was that the findings would generate a rich pedagogical discussion by amalgamating diverse experiences and perceptions of the key stakeholders of programming education.

Research questions

This study had a specific focus on pedagogical aspects including teaching preparation, content delivery approaches, and learning-related challenges in simulation-based programming education. We captured different experiences and perceptions of lecturers and students through investigating the following four research questions:

(i) How do the students perceive the learning environment and the teaching in simulation-based programming sessions?
(ii) What are the pedagogical benefits and challenges for lecturers in teaching programming using simulation?
(iii) How do the lecturers prepare for simulation-based programming teaching?

(iv) To what extent are the perceptions of the students and lecturers directed towards best teaching practices for simulation-based programming education?

The study

In this research, we followed the qualitative methodology of investigation. The approach was appropriate because it has the capacity to provide in-depth explanations of any unexplored areas (Creswell, 2006). For data collection, we used focus group and semi-structured interviews. The methods are suitable for drawing detailed opinions and experiences of research participants through reinforcing and challenging their responses (Stewart & Shamdasani, 2014). Moreover, the interviews were semi-structured, allowing the interviewers options to improvise and extend questions to clarify the answers.

Thirty-seven students and four lecturers from a second-year computing course in a British university participated in the study. The course contained a series of traditional and simulation-based programming sessions for around 6 months. The participation of the students and lecturers was entirely voluntary. The number of students who attended the focus groups was more than half of the total students participating in the course. Four lecturers had experience of delivering both traditional and simulation-based programming courses, and we interviewed all of them. We obtained ethical approval from the university before collecting data, and all participants gave informed consent in writing prior to attending a focus group or interview session. The students and lecturers shared their observations and opinions based on personal experiences, therefore the data were valid and related to their experience and our research questions.

Data collection techniques

The students attended five focus group sessions, each containing four to nine participants and lasting for about 30 minutes. The discussion topics were mainly on simulation-based learning activities that had accelerated or hindered their academic performances in the programming sessions. Additionally, the students shared opinions on the roles of their peers and lecturers in such learning environments. Examples of the focus group questions include, 'Did the lecturers play any role for you in cross scripting?', 'You have said how teachers can engage you, but how can you be engaged?', 'What makes you inattentive in programming classes and less engaged?', and 'What was the role of your classmates in that session?' The questions created opportunities for the students to share personal observations and reflections as mature scholars in a co-operative and non-threating environment. As a result, the information of the focus groups was generally shared, complementary and negotiated.

The interview participants were engaged through purposive sampling (Patton, 2005). This technique was suitable because we needed to explore views of those faculty members who had taught programming in both traditional and simulation environments. The lecturers attended individual and face-to-face interview sessions (except one, which was done via Skype) lasting between 20 to 30 minutes each. The interviews were semi-structured, and the questions mainly covered pedagogical issues, such as lesson preparation and teaching techniques. Examples of the interview questions include, 'How do you prepare the delivery of a simulation-based programming lesson?' and 'What challenges do your students face while participating in your programming classes?' In the third interview, a trend of speech saturation emerged as the lecturer started repeating several points which the previous lecturers had discussed in their answers. It is plausible that the key reasons for such recurrence of data at that stage were the limited number of interview questions, and their specific focus on learning and teaching aspects only. The fourth lecturer contained a high proportion of speech saturation confirming the adequacy of the interview data for this study.

Data processing and analysis procedures

Both the focus group and interview sessions were audio-recorded and transcribed verbatim by a professional transcriber. We used qualitative data processing software NVivo to categorise student and teacher responses in several thematic areas, such as learning environments, teaching preparation, and student engagement (see Section 4). We used thematic analysis to discuss these categories in two broad areas: student experiences and perceptions (see Section 4.1), and lecturers' experiences and perceptions (see Section 4.2). Then, we cross-evaluated the two circles of narration, and linked them with educational theories for achieving a critical understanding (see Section 5). This data triangulation provided richer perspectives of simulation-based teaching of programming and the impacts of the approach on students' participation and learning (Johnson, Onwuegbuzie, & Turner, 2007; Teddlie & Tashakkori, 2009). While processing the focus group data, we did not try to locate individual respondents because the sessions together contained a comparatively high number of students. Besides, it was not possible to identify exact students based on their voice in the audio recordings. Moreover, we only needed to analyse similarities and differences between the students' responses for gaining a general description of their experiences and perceptions, thus code naming for these data was not necessary. However, while processing and presenting the interview data, we used code names, namely Paul, John, Richard and Joseph for four interviewee lecturers. This helped us understand the common ground and divergence of perceptions among faculty members of the same academic programme which is common in the education sector at all levels. The code naming also ensured confidentiality and anonymity of the interviewees.

Findings

The two sets of data individually and together constructed a useful interpretation of the learning and teaching of programming using simulation. The findings identified several enabling and challenging factors ('Learning perspectives' and 'Teaching perspectives' sections), which helped achieve useful pedagogical principles and their application guidelines for this academic subject ('Analysis of findings' section).

Learning perspectives: students' experiences and perceptions

The students shared both enabling and challenging features of simulation-based programming learning. On the one hand, they described engagement, visualisation and consistency of learning procedures as empowering factors. On the other, they questioned the feasibility of collaborative tasks, and considered the shifting between virtual and physical educational environments a challenge. The following themes summarise their experiences and reflect their perceptions.

Engagement and consistency are major learning catalysts

The students indicated the need for active engagement and participation for accomplishing meaningful learning of programming. One important pre-condition they considered is the provision of cognitively challenging tasks.

> … they (lecturers) start from the beginning. I think, some students already did some units about database before, so maybe you can get some uses and go to the next level.

> It was like there are different levels (of students) studying database and it was from the beginning, and I was a little bit bored. I think it was so simple, I wasn't engaged at all…

The students also placed an emphasis on ensuring consistent pedagogical progression. Some students found the beginning phase problematic as they felt unprepared, particularly with essential programming concepts.

> …they (students) haven't studied the concept before and they don't have solid knowledge, they get lost so easy… sometimes they do not get everything …

This finding evidences the importance of prior knowledge marrying the new knowledge intended. Therefore, a progressive act of teaching appears to be essential in simulation sessions.

Linking between the 'environments' enhances students' confidence

The learning environments of simulation and traditional sessions are apparently similar as students in both settings sit in front of computers and write programming codes. In simulation sessions, the students concurrently dealt with two learning environments, physical and virtual. On one hand, in the classroom or physical environment, they were connected with the peers and lecturers through task explanations, questioning and problem-solving activities. The space and facilities of the classroom also influenced the level of their engagement or dis-engagement, particularly given the fact that engagement is not restricted to the rational and analytical but has emotional and other dimensions. As student engagement is multi-faceted, the following example shows how some flexibility in the sessions influenced the learning of students:

> … in the first class you could bring drinks in there, basically to feel comfortable, so you can learn in the most comfortable way.

On the other hand, simulation sessions also attracted the students in a virtual environment leading to working more independently using an intuitive and interactive simulation tool. Learning within the two environments required reciprocal communication and understanding by students and lecturers. Students mentioned that there was a hindrance to learning when they encountered problems and faced difficulties in translating those problems to the lecturers who were outside of the virtual environment. Lecturers were remote or removed from the virtual process in simulation-based learning causing a shift between environments (virtual and physical).

> [Learning hampered] … as tutor didn't interact too much with us … he was like- we're doing the work, and he would come to see whether we are doing alright with the task, or just we're sitting there.

In connection to this, students suggested the simulator should have a synchronous monitoring system through which the lecturers could access the virtual learning environments of students and the works in progress. These can minimise frequent cross-migration between environments and thus improve the quality of lecturer-student interaction.

> It would be much better if he (the lecturer) could explain from his computer and show how it's working, and then it would be like probably better to do it with us and then ask us to do it alone.

Collaboration is not always feasible

Collaborative and shared learning approaches, for example group discussions and team projects, are important for ensuring pedagogical excellence in computing (Sentance & Csizmadia, 2017). However, the students generally found it unsuitable for simulation sessions and thus preferred working alone.

> … all I'm gonna do is work on the problem sheet, so I might as well do it at the comfort of my own…

> … there isn't much group work, it's just individuals doing those tasks and things like that; so, yeah you feel like I'm gonna put my headphones on and start to listen to music and don't worry about others … it's just you, yourself and the computer.

Although the students preferred to learn alone in simulation sessions, they did not undermine the advantages of collaboration. They mentioned that the occasional support from their peers and lecturers 'promoted a lot of camaraderie between them'. Some students also expressed the need of a communication platform, an alternative approach to collaboration, for sharing and consulting with their lecturers and classmates while completing tasks.

Demonstration and visual presentations are effective preparation tools

The students understood programming actions and processes better when the lecturers explained the relevant application areas and possible outcomes. In this connection, the students found demonstration a useful task for preparatory learning and engagement in the simulation environment.

> With xx (the lecturer) it was like visual examples- how to do it yourself and then we were asked, to do- to perform the task; and for me this was better because watching how he's doing like the task I think made it, built in the logic in there, and how I'm going to do it, and how the method he used to finish the task really helped me to

> visualise. I feel more confident with going into a task rather than like reading something and trying to figure out the stuff myself.
>
> Demos, questions, going through a working example before we have to do the work, so we know what the goal is, which is a great thing to keep engaged.
>
> It was a first unit about web development, you don't know anything, so you need some like guide, after that you don't need it because you can learn it by yourself, at the beginning you absolutely need it.

The students also mentioned the demands of clear goal setting, provision of questioning, and quick problem-solving support from the peers and lecturers. However, they did not expect 'spoon feeding' in the learning process rather wanted a degree of autonomy with the facility of occasional and quick support from the peers and lecturers.

> There's couple of things with the programming every now and then where you think the task you're about to take is going to be incredibly daunting, and then when explained properly and well, it goes actually that's really easy, you know and then you get it really fast, and then applying it to other situations then again it also becomes easy, but when it's not explained there is just no way.

Teaching perspectives: lecturers' experiences and perceptions

The lecturers reflected on a number of teaching aspects regarding the simulation sessions. Their opinions not only covered the features of teaching, they also provided critical observations on students' learning experiences. In presenting the findings below, we have used pseudonyms to refer to the lecturers.

Teaching is less-challenging in simulation-based sessions

The lecturers expressed an overall satisfaction about using simulation for teaching programming. They did not report any resistance from the students, rather mentioned they were more engaged and motivated in simulation sessions compared to traditional programming sessions.

> I have not noticed or seen any student complaining that they don't like simulation classes, but I've noticed, I've had instances, where students will say they don't like traditional classes, maybe they don't understand, but with simulator they find it a lot easier to understand ... (Richard).

Richard also stated that the students 'want to see what is going to happen and how things actually work' (Richard). In this regard, John referred to various visual features of simulation sessions and claimed the element as the key reason for the students' higher engagement and participation.

> I feel it's easier in terms of pictorial aspect and seeing things, actually like working the way you are explaining them... (John).

Although the lecturers generally showed a positive attitude towards simulation-based teaching, they did not consider the approach entirely free from challenges. Paul and Richard mentioned the following two difficulties mainly associated with technological applications.

> … when there's problem with the software itself or the network, or the simulator is not properly explaining what they should know, then you will see quite a few challenges (Paul).

> ... obviously there will be some bugs as well and you only notice most of the bugs when you use them. ... the challenge is if there's any problem, they (students) get infuriated and evidently that can obstruct their learning (Richard).

The lecturers also found a lack of standardisation in the learning tasks and simulation designs in online and offline environments. In this regard, they emphasised addressing the needs and preferences of students.

> I think when you can test it in multiple environments, you're not reliant on a particular piece of hardware that can go wrong. I'm certainly a fan of web-based simulators, you know other simulators may involve installation and not particularly friendly. This wouldn't really work in the way that students need to work these days which is online…(Joseph).

There is a potential risk of students' attention diversion

The lecturers' descriptions of simulation-aided programming activities depicted a process-oriented and visual learning journey. Simulation places an emphasis on the understanding of programming steps which can help students execute similar tasks in a non-simulation environment, as Joseph ascertained,

> .... simulation shows you everything, it does all the way through there, you can rely on that a little bit too much rather than actually trying to think about getting your code right to start with or thinking how you would fix the problem if you don't have the simulator (Joseph).

Although the programming stages and processes can be observed through using simulation, the lecturers anticipated that these rigorous process descriptions may make the students over reliant on the simulator. According to John, this could result in inefficiency when doing programming in non-simulation environment.

> … the students are often more attracted to the simulator itself so that caught the attention and that will then lead it into what they actually needed to know… they (students) were more reliant on the simulator than the class procedures and lesson instructions… (John).

The lecturers observed that some students were extremely engaged in the virtual environment and thus could not even link with classroom activities. Joseph shared an instance of a student's distraction: 'look at the problem worksheets, you know you're allowed but the student was so detached to instruction and only wanted to get it all virtually' (Joseph).

Technological challenges override teaching preparation

Simulation-based programming sessions were in a face-to-face class environment with appropriate computer facilities. The sessions included students' individual and hands-on programming practices supported by lecturers' verbal instructions, brief demonstrations and feedback on student work. While the sessions combined varied learning and teaching activities, the lecturers mainly mentioned technological aspects when describing challenges in his teaching. The following statement of Richard evidences this aspect.

> In terms of delivery, majorly technology related problems are the only thing we've noticed so far. I have not noticed, or I have not seen any student complain that they don't like the simulator (Richard).

Similarly, John's preparation for simulation sessions included some minor modifications of simulation-based worksheets and checking if 'a specific platform topic [for the session] is actually in place from the student end as well as the lecturer end' (John). The interview data showed the lecturers' indifference about non-technological issues of teaching, for example instructional procedures and preparation activities. Therefore, the need for pedagogical preparation was not generally acknowledged and the lecturers, such as Richard and John, only considered the technological side of teaching.

> ... preparation involves you having to have a look at the simulator to make sure things are actually working well so one of the main things you want to look at in it would be the connection to the internet for instance... So, make sure the simulator works both from the back end, from the lecturer end as well as the student end... (Richard).

> We design the worksheet for simulation-based activity, which is different from the traditional style. Initially, the worksheets were written in a certain way, I've had to re-write them to take advantage of using simulator (John).

Feedback needs to be personalised and immediate

Programming is a step-by-step process where a single error at any stage leads to failure of the entire programming design. Therefore, assisting students while they are doing programming or coding requires close monitoring of the individual programming tasks and the progression pathways. This also helps the lecturers recognise students' difficulties and inquiries while executing the programming tasks, as Joseph described,

> …you can put all these tools in front of the students, until they really understand what they're looking at, they are just pressing a few buttons. They've seen this button that makes it run, so I'm often standing over students so with the simulator, and it's going line by line... (Joseph).

As a general rule, students decide and use coding actions from multiple options which make their programming designs and outputs different from each other. In simulation sessions, when students raise problems linked to their individualised programming processes, the variances of programming design and procedure sometimes confuse the lecturers in perceiving the meaning of the students' inquiries.

> … sometimes people can ask questions where you think that they have mastered everything or mastered what they'd done in the class, and yet then again later on you see them and I mean we got one guy at the moment who comes in and asks the most insanely complicated questions... (John).

Therefore, to overcome the complexities of understanding students' difficulties properly, the lecturers mentioned the need for reviewing students' programming actions and processes. They also emphasised the importance of giving immediate feedback on the inquiries so that the students do not progress in an incorrect direction. However, because of the mixed programming backgrounds of the students, sometimes the lecturers were unable to assume the nature and depth of the students' difficulties.

> .. it's difficult to totally assess something because when you think you're being asked, you know clever intelligent questions, but still people may not know the basics on the back end... (Joseph).

To understand students' difficulties in simulation-based programming, Paul followed an explanatory approach where he asked students to explain their programming processes, for example, 'what you see going on in this loop, you now try and explain it on your own words' (Paul). This approach not only helped the students recall and review the process, but also helped him have an idea about the steps the students followed in their programming design or coding.

Analysis of findings and lessons learned

The experiences and perceptions of the research participants elucidate important educational features of simulation-based programming learning and teaching. The findings show various supportive and challenging aspects which, in the light of learning theories and educational models, supply the following three pedagogical directions.

Address pedagogical issues in teaching preparation

Overall, the lecturers were satisfied with simulation-based teaching (see 'Teaching perspectives' section). Conversely, the students experienced some challenges, such as lack of consistency in teaching and the need to get immediate feedback while conducting programming tasks (see 'Learning perspectives' section). These slightly contrasting findings suggest revisiting existing teaching plans and their implementation procedures, even though the learning environment seems to function adequately.

In terms of teaching preparation, the lecturers were mainly concerned about the difficulties related to technology, more specifically the simulator. However, several non-technological issues, for example task selection, instructional procedures, and assessment techniques emerged in the focus group and interview data. The findings suggest that lecturers need to address pedagogical issues in simulation sessions in equal measure as technological aspects.

The study shows the need for maintaining connections and consistency of tasks in teaching, which is generally an important requirement in science education (see Sikorski & Hammer, 2017; in 'Emerging educational concepts' section). Consistency in simulation-based programming sessions is hampered by unbalanced learning activities, for example when there are too difficult or too easy tasks in different periods of a course. This problem with the learning content can be minimised by incorporating appropriately challenging learning activities across the course. To engage the students effectively, it is also important to keep a proper balance of cognitive, behavioural and emotional tasks; for example complex coding, time-demanding assignment, and practical programming design for future professional work. However, for designing suitable tasks, lecturers need to take account of students' level of competency and prior knowledge (see Schmidt et al., 2017; Nasir & Hand, 2006; and Schoenfeld, 1999 in 'Emerging educational concepts' section). In this regard, a pre-course assessment of programming competencies may help lecturers understand the baseline knowledge and skills of their student groups, and also decide appropriate programming tasks for them.

The study sheds light on three areas significant to helping students participate and learn effectively.

First, it shows how best to construct an effective educational environment to maximise student learning and attainment (UKPSF, 2018). The findings indicate the feasibility of flipped approach in simulation-based programming education which can accommodate students' prior knowledge and allow teachers to use teaching time more effectively (see 'Flipped approach' section). Second, as discussed in the findings about students' experiences and perceptions (see 'Learning perspectives' section), students often require immediate feedback and suggestions on the difficulties they face while conducting simulation tasks. One of the barriers here is lecturers' reliance on simulation when they are in the virtual environment, and it seems like lecturers are almost absent as teachers. It is essential to support students promptly to avoid inaccurate first responses as well as incorrect programming procedures (Epstein et al., 2002). For efficient monitoring and speedy responses to students' difficulties, co- or team teaching and peer observation by students may be effective (Shaffer & Thomas-Brown, 2015; Sweigart & Landrum, 2015). Third, in terms of assigning tasks, the students felt the need for having clear learning goals and procedures to accomplish them. This requirement indicates the need for standardising simulation tasks, preferably in line with possible applications in the students' future professions.

Rethink the scope and procedures of collaboration

Collaboration is a key enabler in engineering education (see 'Collaboration' section). However, on the surface, collaboration got mixed reviews (see 'Learning perspectives' section). On the one hand, the students found it inconvenient and ineffective during some simulation activities, particularly when they were engaged in live programming tasks. In such situations, collaboration diverted their attention from the virtual to physical environment and impacted negatively on the progression of their tasks. On the other hand, the students found the support from lecturers and peers very helpful when they faced any difficulties. These suggest that collaboration and support can aid students' programming if they do not divert their attention.

The findings of this study suggest that the implications of any collaborative activities in simulation sessions should be evaluated carefully before their implementation. In this regard, the lecturers can consider developing a built-in collaboration mechanism within the simulation environment. A networked or virtual platform for peer observation of programming tasks can provide opportunities to follow the task completion processes which can also assist students to share ideas and solve problems more efficiently. Additionally, for taking educational advantage of collaboration, lecturers may consider the provision of pre- and post- collaborative activities using flipped approach (see 'Flipped approach' section), such as programming related group planning and reflective practices with peers on completed programming tasks.

Explore the feasibility of cognitive apprenticeships

The students and lecturers described positive experiences with cognitively challenging tasks and visual approaches to learning (see 'Learning perspectives' and 'Teaching perspectives' sections). Students expected effective demonstration by lecturers whereas the lecturers highlighted the need for personalised and immediate feedback for students' effective learning. Students also expressed the need for strong cognitive engagement, enhanced confidence in dealing with multiple learning environments, and capabilities to reflect on their learning tasks and progression. These features of learning and teaching suggest that cognitive apprenticeship pedagogy is an appropriate method for simulation-based programming education.

The cognitive apprenticeship approach (see 'Cognitive apprenticeship model' section) has two key strengths which can facilitate effective simulation-based programming sessions. First, programming generally involves live tasks for accomplishing defined skills and actions transferable to future professions of the students. The core objectives of the cognitive apprenticeship approach are the same with this goal. Second, programming competence involves specialised computing skills and knowledge which are often tacit and difficult to explain verbally. The lecturers mentioned this difficulty and voiced the importance of providing task related detailed clarifications in simulation sessions (see 'Teaching perspectives' section).

Despite many educational advantages of cognitive apprenticeships (see 'Cognitive apprenticeship model' section), the lecturers of simulation-based programming may find the approach pedagogically demanding as it entails extended teaching preparation and more contribution in teaching students from them. They may also face confrontations with their colleagues and administrative systems while applying this apparently new pedagogical approach as the delivery of university academic programmes is often homogeneous and controlled by conventions. Moreover, the same educational goals for all programmers and single pedagogical approach in all programming sessions may not work (Guzdial, 2015), so the apprenticeship-based teaching will need flexibility in terms of teaching styles and resources. Overall, a carefully designed pedagogical plan will be required for implementing cognitive apprenticeship teaching and learning in simulation-based programming sessions.

Conclusion

The study raises three distinct issues. First, it recognises that pedagogy is the blind-spot in the programming education, particularly when simulation is used. Therefore, it is a wakeup call for the computing educators to take pedagogical aspects seriously while designing and delivering simulation-based programming sessions. Second, the study shows the importance of understanding learning environments within and outside simulation. The findings have expanded the knowledge on student engagement to the effective styles and processes of teaching and collaborative learning in simulation environment. Third, the art of effective learning and teaching in computer simulation sessions suggests considering cognitive apprenticeships as a suitable educational approach where students can learn through visual demonstrations, hands-on practice and reflections. The approach also offers opportunities to use more cognitively challenging tasks and collaboration in the learning process. Overall, the study shows the need for systematic pedagogical plans as well as their careful implementation in a skills-oriented and performance-driven academic learning culture.

Although the implications of the explored pedagogical principles and procedures are centred on programming or computing subjects, the lessons learned from this research may have resonance for simulation practices in a wider spectrum of education. Therefore, the findings may explain the learning and teaching issues or help identify any pedagogy-related questions attached to simulation in the higher education context. However, this study draws evidence from one British university only, thus it may not fully represent the diverse features of academic environment and pedagogical practices of programming education in other higher educational institutions. It is also possible that students from different cultural and educational backgrounds respond to technology-enhanced approaches differently. Therefore, further research in dissimilar educational institutions is needed to corroborate the findings of this study. Particularly, more specific and in-depth inquiry can explain various complex pedagogical areas of simulation-based programming education, for example teacher preparation strategies and approaches to collaborative learning by students.

Acknowledgements
The authors are grateful to Professor Tansy Jessop and Dr. Nick Whitelegg of Solent University, UK, for their continuous support and encouragement in conducting this study. The authors also like to thank Solent University's Learning and Teaching Institute (SLTI) for funding this research.

Declarations
We would like to confirm that the research presented in this manuscript is original and has not been published elsewhere. To our knowledge, there is no conflict of interest to disclose and we have followed standard ethical procedures in our study.

Authors' contributions
MGJ planned this study, conducted the literature review, designed the data collection tools, assisted in data collection, and led the data analysis and reporting work. SOI led the data collection work and assisted in analysing the findings. Both authors read and approved the final manuscript.

Authors' information
Dr. Md Golam Jamil is a Research Fellow at the Learning and Teaching Institute (SLTI) of Solent University, UK. Dr. Sakirulai Olufemi Isiaq is a Senior Lecturer - Computing at Solent University's School of Media Art and Technology.

Funding
Solent Learning and Teaching Institute (SLTI), Solent University, UK.

Availability of data and materials
The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests in the manuscript.

Received: 7 March 2019 Accepted: 25 June 2019
Published online: 01 August 2019

References

Aleksic, V., & Ivanovic, M. (2016). Introductory programming subject in European higher education. *Informatics in Education*, *15*(2), 163–182.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGcSE Bulletin*, *39*(2), 32–36.

Bergmann, J., & Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. Virginia: International Society for Technology in Education (ISTE).

Blackmore, J. (2009). Academic pedagogies, quality logics and performative universities: Evaluating teaching and what students want. *Studies in Higher Education*, *34*(8), 857–872.

Blackwell, A., McLean, A., Noble, J., & Rohrhuber, J. (2014). Collaboration and learning through live coding (Dagstuhl seminar 13382). In *Dagstuhl Reports, 3*(9):130–168. Wadern: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Bland, A. J., Topping, A., & Wood, B. (2011). A concept analysis of simulation as a learning strategy in the education of undergraduate nursing students. *Nurse Education Today*, *31*(7), 664–670.

Bryan, R. L., Kreuter, M. W., & Brownson, R. C. (2009). Integrating adult learning principles into training for public health practice. *Health Promotion Practice*, *10*(4), 557–563.

Chen, H. Y. L., & Chen, N. S. (2014). Design and evaluation of a flipped course adopting the holistic flipped classroom approach. In *2014 IEEE 14th International Conference on Advanced Learning Technologies*, (pp. 627–631). Athens: IEEE.

Chini, J. J., Straub, C. L., & Thomas, K. H. (2016). Learning from avatars: Learning assistants practice physics pedagogy in a classroom simulator. *Physical Review Physics Education Research*, *12*(1), 010117 -1-15.

Colella, V. (2000). Participatory simulations: Building collaborative understanding through immersive dynamic modeling. *The Journal of the Learning Sciences*, *9*(4), 471–500.

Collins, A. (2006). Cognitive apprenticeship. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning science*, (pp. 47–60). New York: Cambridge University Press.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, (pp. 453–494). Hillsdale: Lawrence Erlbaum Associates.

Craft, A., Chappell, K., & Twining, P. (2008). Learners reconceptualising education: Widening participation through creative engagement? *Innovations in Education and Teaching International*, *45*(3), 235–245.

Creswell, J. W. (2006). Five qualitative approaches to inquiry. In J. W. Creswell & C. N. Poth (Eds.), *Qualitative Inquiry and Research Design: Choosing Among Five Approaches* (pp. 53-80). Thousand Oaks: SAGE.

Crick, T., Davenport, J. H., & Hayes, A. (2015). Innovative pedagogical practices in the craft of computing. In *Innovative Pedagogies in the Disciplines*. York: Higher Education Academy.

Davies, D., Jindal-Snape, D., Collier, C., Digby, R., Hay, P., & Howe, A. (2013). Creative learning environments in education - a systematic literature review. *Thinking Skills and Creativity*, *8*, 80–91.

Dennen, V. P., & Burner, K. J. (2008). The cognitive apprenticeship model in educational practice. In J. M. Spector, M. D. Merrill, J. Merrienboer, & M. P. Driscoll (Eds.), *Handbook of research on educational communications and technology*, (pp. 425–439). New York: Taylor & Francis.

Dieckmann, P. (2009). Simulation settings for learning in acute medical care. In P. Dieckmann (Ed.), *Using simulations for education, training and research*, (pp. 40–138). Lengerich: Pabst.

Dieker, L. A., Straub, C. L., Hughes, C. E., Hynes, M. C., & Hardin, S. (2014). Learning from virtual students. *Educational Leadership*, *71*(8), 54–58.

Donaldson, A. L. (2015). Pre-professional training for serving children with ASD: An apprenticeship model of supervision. *Teacher Education and Special Education*, *38*(1), 58–70.

Epstein, M. L., Lazarus, A. D., Calvano, T. B., Matthews, K. A., Hendel, R. A., Epstein, B. B., & Brosvic, G. M. (2002). Immediate feedback assessment technique promotes learning and corrects inaccurate first responses. *The Psychological Record*, *52*(2), 187–201.

Eraut, M. (2004). Informal learning in the workplace. *Studies in Continuing Education*, *26*(2), 247–273.

Flipped Learning Network. (2014). The four pillars of FLIP. Retrieved April June 5, 2019, from https://flippedlearning.org/wp-content/uploads/2016/07/FLIP_handout_FNL_Web.pdf

Fuller, A., & Unwin, L. (2011). Apprenticeship as an evolving model of learning. *Journal of Vocational Education & Training*, *63*(3), 261–266.

Giannakos, M. N., Pappas, I. O., Jaccheri, L., & Sampson, D. G. (2017). Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. *Education and Information Technologies*, *22*(5), 2365–2382.

Gomes, A., & Mendes, A. J. (2007). An environment to improve programming education. In *Proceedings of the 2007 international conference on Computer systems and technologies* (Article 88). Bulgaria: ACM.

Gonczi, A. (2013). Competency-based approaches: Linking theory and practice in professional education with particular reference to health education. *Educational Philosophy and Theory*, *45*(12), 1290–1306.

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, *8*(6), 1–165.

Harder, B. N. (2009). Evolution of simulation use in health care education. *Clinical Simulation in Nursing*, *5*(5), 169–172.

Hayden, J. (2010). Use of simulation in nursing education: National survey results. *Journal of Nursing Regulation*, *1*(3), 52–57.

Hyvonen, J., & Malmi, L. (1993). *TRAKLA-a system for teaching algorithms using email and a graphical editor*.

Isiaq, S. O., & Jamil, M. G. (2018). Enhancing student engagement through simulation in programming sessions. The International Journal of Information and Learning Technology, *35*(2), 105–117.

Jeffries, P. R. (2012). *Simulation in nursing: From conceptualization to evaluation*. New York: National League for Nurses.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, (vol. 4, No. 2002, pp, pp. 53–58).

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research*, *1*(2), 112–133.

Kaakinen, J., & Arwood, E. (2009). Systematic review of nursing simulation literature for use of learning theory. *International Journal of Nursing Education Scholarship*, *6*(1), 1-20.

Karabulut-Ilgu, A., Jaramillo Cherrez, N., & Jahren, C. T. (2018). A systematic review of research on the flipped learning method in engineering education. *British Journal of Educational Technology*, *49*(3), 398–411.

Karaca, C., & Ocak, M. (2017). Effect of flipped learning on cognitive load: A higher education research. *Journal of Learning and Teaching in Digital Age*, *2*(1), 20–27.

Kelly, M. A., Forber, J., Conlon, L., Roche, M., & Stasa, H. (2014). Empowering the registered nurses of tomorrow: Students' perspectives of a simulation experience for recognising and managing a deteriorating patient. *Nurse Education Today*, *34*(5), 724–729.

Kelly, M. A., Hopwood, N., Rooney, D., & Boud, D. (2016). Enhancing students' learning through simulation: Dealing with diverse, large cohorts. *Clinical Simulation in Nursing*, *12*(5), 171–176.

Knotts, G., Henderson, L., Davidson, R. A., & Swain, J. D. (2009). The search for authentic practice across the disciplinary divide. *College Teaching*, *57*(4), 188–196.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs: Prentice-Hall.

Korhonen, A. (2003). *Visual Algorithm Simulation (Doctoral thesis)*. Helsinki: Helsinki University of Technology.

Korhonen, A., & Malmi, L. (2000). Algorithm simulation with automatic assessment. *ACM SIGCSE Bulletin*, *32*(3), 160–163.

Kujansuu, E., & Tapio, T. (2004). Codewitz–An international project for better programming skills. In L. Cantoni & C. McLoughlin (Eds.), In *Proceedings of EdMedia: World Conference on Educational Media and Technology 2004*, (pp. 2237-2239). Waynesville: Association for the Advancement of Computing in Education (AACE).

Latulipe, C., Long, N. B., & Seminario, C. E. (2015). Structuring flipped classes with lightweight teams and gamification. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, (pp. 392-397). Missouri: ACM.

Latulipe, C., Rorrer, A., & Long, B. (2018). Longitudinal data on flipped class effects on performance in cs1 and retention after cs1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, (pp. 411-416). Baltimore: ACM.

Lombardi, M. M. (2007). Authentic learning for the 21st century: An overview. *Educause learning initiative*, *1*(2007), 1–12.

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education*, *21*(1), 57–80.

MacLellan, C. J. (2017). *Computational models of human learning: Applications for tutor development, behavior prediction, and theory testing (Doctoral thesis)*. Pittsburgh: Carnegie Mellon University.

Magana, A. J., & Silva Coutinho, G. (2017). Modeling and simulation practices for a computational thinking-enabled engineering workforce. *Computer Applications in Engineering Education*, *25*(1), 62–78.

Makransky, G., Thisgaard, M. W., & Gadegaard, H. (2016). Virtual simulations as preparation for lab exercises: Assessing learning of key laboratory skills in microbiology and improvement of essential non-cognitive skills. *PLoS One*, *11*(6), e0155895.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in education*, *3*(2), 267.

Mann, S., & Robinson, A. (2009). Boredom in the lecture theatre: An investigation into the contributors, moderators and outcomes of boredom amongst university students. *British Educational Research Journal*, *35*(2), 243–258.

Martin-Gutierrez, J., Fabiani, P., Benesova, W., Meneses, M. D., & Mora, C. E. (2015). Augmented reality to promote collaborative and autonomous learning in higher education. *Computers in Human Behavior*, *51*, 752–761.

McGaghie, W. C., Issenberg, S. B., Petrusa, E. R., & Scalese, R. J. (2010). A critical review of simulation-based medical education research: 2003–2009. *Medical Education*, *44*(1), 50–63.

McLean, A., Griffiths, D., Collins, N., & Wiggins, G. A. (2010). *Visualisation of live code. In Proceedings of Electronic Visualisation and the Arts*, (pp 26-30). London: Computer Arts Society.

Morley, D. A. (2016). *A Grounded theory study exploring first year student nurses' learning in practice (Doctoral thesis)*. Bournemouth: Bournemouth University.

Morley, D. A. (2018). The 'ebb and Flow'of student learning on placement. In *Enhancing Employability in Higher Education through Work Based Learning*, (pp. 173–190). Cham: Palgrave Macmillan.

Nasir, N. I. S., & Hand, V. M. (2006). Exploring sociocultural perspectives on race, culture, and learning. *Review of Educational Research*, *76*(4), 449–475.

Nova, B., Ferreira, J. C., & Araujo, A. (2013). Tool to support computer architecture teaching and learning. In *2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*, (pp. 1-8). Porto: IEEE.

Oliveira, A., Behnagh, R. F., Ni, L., Mohsinah, A. A., Burgess, K. J., & Guo, L. (2019). Emerging technologies as pedagogical tools for teaching and learning science: A literature review. *Human Behavior and Emerging Technologies*, *1*(2), 149–160.

Olsson, M., Mozelius, P., & Collin, J. (2015). Visualisation and Gamification of e-Learning and Programming Education. Electronic journal of e-learning, *13*(6), 441–454.

Park, Y. S. (2016). A study on the standardization of education modules for ARPA/radar simulation. *Journal of the Korean Society of Marine Environment & Safety*, *22*(6), 631–638.

Patton, M. Q. (2005). Qualitative research. In B. S. Evritt & D. Howell (Eds.), *Encyclopedia of statistics in behavioral science*, (pp. 1631-1636). Chichester: Wiley.

QAA (2016). Subject benchmark statement UK quality code for higher education Part A: setting and maintaining academic standards computing. Retrieved on 15 September 2018 from https://www.qaa.ac.uk/docs/qaa/subject-benchmark-statements/sbs-computing-16.pdf?sfvrsn=26e1f781_12.

Rochester, S., Kelly, M., Disler, R., White, H., Forber, J., & Matiuk, S. (2012). Providing simulation experiences for large cohorts of 1st year nursing students: Evaluating quality and impact. *Collegian*, *19*(3), 117–124.

Rystedt, H., & Sjoblom, B. (2012). Realism, authenticity, and learning in healthcare simulations: Rules of relevance and irrelevance as interactive achievements. *Instructional Science*, *40*(5), 785–798.

Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., & Zander, C. (2017). Folk pedagogy: Nobody doesn't like active learning. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, (pp. 145-154). Washington: ACM.

Sawyer, T., Sierocka-Castaneda, A., Chan, D., Berg, B., Lustik, M., & Thompson, M. (2011). Deliberate practice using simulation improves neonatal resuscitation performance. *Simulation in Healthcare*, *6*(6), 327–336.

Schmidt, H. K., Rothgangel, M., & Grube, D. (2017). Does prior domain-specific content knowledge influence students' recall of arguments surrounding interdisciplinary topics? *Journal of Adolescence*, *61*, 96–106.

Schoenfeld, A. H. (1999). Models of the teaching process. *The Journal of Mathematical Behavior*, *18*(3), 243–261.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469–495.

Shaffer, L., & Thomas-Brown, K. (2015). Enhancing teacher competency through co-teaching and embedded professional development. *Journal of Education and Training Studies*, *3*(3), 117–125.

Sharma, R., & Shen, H. (2018). Does education culture influence factors in learning programming: A comparative study between two universities across continents. *International Journal of Learning, Teaching and Educational Research*, *17*(2), 1-24.

Sikorski, T. R., & Hammer, D. (2017). Looking for coherence in science curriculum. *Science Education*, *101*(6), 929–943.

Stewart, D. W., & Shamdasani, P. N. (2014). *Focus groups: Theory and practice* (Vol. 20). London: SAGE.

Suzuki, S. V., Hirokawa, S., Mukoyama, S., Uehara, R., & Ogata, H. (2016). Student behavior in computer simulation practices by pair programming and flip teaching. In *24th International Conference on Computers in Education, ICCE 2016*, (pp. 212-221). Mumbai: Asia-Pacific Society for Computers in Education.

Sweigart, C. A., & Landrum, T. J. (2015). The impact of number of adults on instruction: Implications for co-teaching. *Preventing School Failure: Alternative Education for Children and Youth*, *59*(1), 22–29.

Teddlie, C., & Tashakkori, A. (2009). *Foundations of mixed methods research: Integrating quantitative and qualitative approaches in the social and behavioral sciences*. Thousand Oaks: SAGE.

Tong, V. C., Standen, A., & Sotiriou, M. (Eds.). (2018). *Shaping higher education with students: Ways to connect research and teaching*. London: UCL Press.

Topaloglu, T., & Gurdal, O. (2010). A highly interactive PC based simulator tool for teaching microprocessor architecture and assembly language programming. *Elektronika ir Elektrotechnika*, *98*(2), 53–58.

Tucker, B. (2012). The flipped classroom. *Education next*, 12(1), 82–83.

Tun, J. K., Alinier, G., Tang, J., & Kneebone, R. L. (2015). Redefining simulation fidelity for healthcare education. *Simulation & Gaming*, *46*(2), 159–174.

Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. *Education and Information Technologies*, *23*(1), 419–434.

UKPSF (2018). Dimensions of the framework. Retrieved on 20 September 2018 from https://www.heacademy.ac.uk/ukpsf

White, M. (2017). Keep calm and simulate on: Faculty experiences and insights into implementing best practices in simulation. *Teaching and Learning in Nursing*, *12*(1), 43–49.

Woolley, M. (2009). Time for the navy to get into the game! In *US Naval Institute Proceedings*, (pp. 34–39).

Wyatt, A., Archer, F., & Fallows, B. (2007). Use of simulators in teaching and learning: paramedics' evaluation of a patient simulator? *Journal of Emergency Primary Health Care (JEPHC)*, *5*(2), 1-16.

Xie, C., Schimpf, C., Chao, J., Nourian, S., & Massicotte, J. (2018). Learning and teaching engineering design through modeling and simulation on a CAD platform. *Computer Applications in Engineering Education*, *26*(4), 824–840.

Yahaya, C. K. H. C. K., Mustapha, J. C., Jaffar, J., Talip, B. A., & Hassan, M. M. (2017). Operations and supply chain mini simulator development as a teaching aid to Enhance student's learning experience. In *7th Annual Conference on Industrial Engineering and Operations Management, IEOM 2017*, (pp. 358-367). Rabat: IEOM Society.

Publisher's Note