A Systematic Literature Review of Gameful Feedback in Computer Science Education

Nico Willert

***Abstract*—For the past decade, video game- and gamification-elements get used in different fields of research. However, a contextualized usage of these elements is still underrepresented in the current research. For that reason, this research tries to identify contextualized game-elements in e-learning environments for computer science education. A systematic literature review examines the current overlap of feedback in computer science education by the use of game-elements. The relevant papers were identified by a combination of search-terms and analyzed according to a defined scope, that focuses on formative and summative feedback. In a nutshell, the majority of provided feedback in computer science education, that is not just given by an instructor, is often implemented by automated code tests. These are supported through techniques to monitor the performance of the student and their progress towards the set goal. Game- or gamification-elements do play a subordinate role, when providing feedback and are often just to enhance the monitoring process.**

***Index Terms*—Formative feedback, gamification, computer science, programming, education.**

I. Introduction

Comparable to learning different educational subjects, the handling of video games is a subject of its own, where players have to understand the different mechanics and possibilities, that are offered by a game. An important factor for both video games and education is feedback. As [1] describes feedback as one of the defining traits of games, since in this way knowledge about the objective outcome is provided to the players. Similarly, feedback provided to students can easily influence their overall learning behavior. Feedback can therefore advance or destroy the engagement, that a player has with a game or a student has with a course and its content.

For this reason, this research focuses on the feedback given to students. Hence, the general research question is: "What parallels exist between feedback in games or game-like applications and the use of feedback in education". Especially in computer science education, there are different ways of providing feedback to the students. They range from automated testing, monitoring, scaffolding to different educational and instructional designs. This study investigates the different implementations of feedback in computer science education using a systematic literature review [2]. A deeper analysis of the publications allows to gather an overview of the current use of game- or game-like elements in the educational field and to work on an approach for a collective use of the different types of feedback.

Manuscript received December 15, 2020; revised June 2, 2021.

Nico Willert is with the Information Systems Institute, University Leipzig, Leipzig, Germany (e-mail: willert@wifa.uni-leipzig.de).

II. Related Work

Prior to this work, the publication of [3] was found in the process of gathering information about the current status of gamification research and [4] was found while researching effective feedback in computer science education.

Reference [3] conducted a comprehensive literature review of the gamification research and analyzed research models and results of empirical studies on gamification. They classified the found research based on employed method, type of game-element used, psychological, and behavioral outcome. In addition, a sample of literature consisting of experimental quantitative studies was reviewed for their results and domain. The clear majority of research, reported mixed results, with only 28.7% only positive research findings.

Based on these results they conclude thematic, theoretic, and methodological agenda points that future gamification research should address. These include the context, in which the gamification is deployed or the different types of feedback that can be delivered by different kinds of game-elements and their effect on system users.

Reference [4] investigated theories for effective feedback in higher education and translate their characteristics for feedback in computer science education. Based on the various publications they worked with, they created a roadmap for effective feedback in computer science, that stretches over task, process, and self-regulation level. This roadmap also addresses a time perspective for goals, performance, and improvement feedback, as well as the right communication at each stage of the learning process.

In addition, 5 literature reviews were identified in the field of computer science education. These reviews investigate different aspects for introductory programming and software engineering courses. With the focus on feedback given to the students, there are multiple approaches and recommendations given by the publications. The most frequently used approach for assessment and feedback is automated testing to support the development process and to verify the correctness of the assignments [5], [6]. These automated tests can be extended, so that targeted feedback can be provided for specific test cases [7]-[9]. Other kinds of feedback include code visualization [5], code quality analyzation [5], [9] or the delivery of feedback via hints and the use of scaffolding to help students continue and improve their assignments [6], [8], [9] Especially noticeable is the review of [7] with their research about different types of feedback (based on the taxonomy of [10]), that is given by the examined tools. The feedback is divided into the knowledge about task constraints, concepts, mistakes, how to proceed, and metacognition. The highest amount of provided feedback is knowledge about the mistakes (96% of all reviewed tools) especially test failures and solution errors, followed by the knowledge about how to proceed (44.6%), especially bug-related hints for error correction.

As detailed above, these 7 preceding studies have examined various aspects of effective feedback and gamification-techniques as well as different approaches for feedback and testing in computer science education.

The different studies in computer science education do not specifically explore the concept of using game- or gamification-elements for providing feedback to the students. Of these 5 studies, only [8] presents a separate paragraph for games as teaching tools, where students can learn introductory programming knowledge and experience, by playing a game or using some game-like structured process. From the focus of gamification, [3] showed, that the contextual adaption and usage of game- or gamification-elements is underrepresented in the past and current gamification research. Equally underrepresented is the focus of gamification-research on different feedback types, how these can be delivered through said elements and how they can affect the system users. Therefore, it is important to look at feedback in the context of computer science education from a gameful point of view, to close the current gap of research.

III. Method

For this study a systematic literature review [2] was conducted, to obtain an overview of research issues relating to feedback in computer science courses with respect to their use of game-elements.

To define the research questions and the scope of this study, the population, intervention, and outcome were set. The population defines where the evidence is collected, therefore which group of people, programs or businesses are of interest for the review. The intervention describes the specific technology and the outcome refers to the expected measurable results.

Firstly, for conducting the review, the publications were identified by generating a search strategy and criteria to select suitable studies. A classification scheme was designed based on the defined scope and research questions, to extract data from each publication.

A. Define Scope and Research Questions

The focus of this study is on the different types of feedback and methods how this feedback is provided to students. The population, which should be affected by the intervention consists of students and learners of computer science as well as specialized scholars from the field of gamification and education. The intervention, which is under observation, refers to types of feedback or assessment given to the defined population. The frame of the intervention is thereby limited to formative and summative feedback, to tighten the overall focus. This also eliminates unwanted results, like feedback, that is gathered from students. They are compared to the use of game- or gamification-elements, that are used to provide feedback in a gameful way. The studies outcome revolves around the overall impact of the feedback implementation on the student performance, their motivation, and their engagement with the course and its learning contents.

The goal is to integrate the findings from the publications to create a general view of feedback usage in computer science education and enhance and apply the gained knowledge to our own educational work.

To analyze the use of feedback in computer science education and investigate which role gamification-elements play in its implementation, the following research questions are derived from the scope and serve to approach the general research question mentioned in the introduction,

- **RQ1:** How are the types of feedback distributed in computer science education?
- **RQ2:** How is the feedback implemented?
- **RQ3:** Which gamification-elements are used?
- **RQ4:** For what reason is the feedback/implementation used?
- **RQ5:** What is the actual outcome of the implementation?

B. Identification of Research

For the identification of research, the digital libraries of ACM, ScienceDirect, and IEEE were searched. Based on the defined scope and a preliminary study the search terms were selected. The terms programming, computer science or software development were searched in the abstract or title. In addition, the terms “gameful”, “game-based” or “gami*“ were searched and the combinations of “formative” or “summative” with “feedback”, “assessment” or “evaluation”.

This resulted in the following search string. *(("formative feedback" OR "formative evaluation" OR "formative assessment" OR "summative feedback" OR "summative evaluation" OR "summative assessment") AND ("Abstract": "programming" OR "Abstract": "computer science" OR "Abstract": "software developement" OR "Document Title": "programming" OR "Document Title": "computer science" OR "Document Title": "software developement") AND (game* OR gami*)).*

C. Study Selection Criteria

The screening process for each paper includes title, abstract and conclusion for the research questions, and further parts for the findings. The following inclusion and exclusion criteria were used to select the relevant papers. The publication is included, if the following criteria are met:

- It deals with one or more of the mentioned types of feedback or implements its feedback through game- or game-like elements.
- The feedback is in the context of computer science education.
- It is peer reviewed including full and short papers.

The publication is excluded, if one of the following criteria is met:

- The feedback given is not in computer science context.
- The feedback given is solely about the gathered feedback from students.

- It is about some form of mechanical or physical feedback of a system.
- It is another literature study.

D. Classification Scheme

The classification scheme was designed based on the established definitions from the literature to identify the different facets. The classification scheme, was extended, if a paper introduced a new category. The important facets are feedback, game elements, type of implementation, and the stated reasons of usage. The classification scheme was checked by a preliminary study. The categories and facets are summarized in Table I.

1) Feedback in education

The categories for feedback are based on [4] and have been extended by the findings. Each aspect is described to illustrate the overlapping results in RQ1 beforehand.

Formative feedback is concerned with how assessments about the quality of student responses, can shape and improve the student competence [11].

In contrast to formative feedback, summative feedback is concerned with summing up or summarizing the achievement status of a student or the end of a course unit. Therefore, it does not have an immediate impact on the learning but can influence future decisions [11].

Immediate Feedback refers to feedback given virtually at the time of test. In addition, rapid feedback focusses on feedback that is not given immediately after the submission of the results, but fast enough to have an impact on the student for the next task [12].

Self-regulation feedback serves the role of enhancing student self-regulation, by supporting the students in monitoring and self-observing their actions towards the learning goal and thus helps them to strategically adjust their goals, further actions and reactions [13].

Scaffolding is mostly used in the form of a support structure, serves to support the students in their learning process by either inserting these structures, when they are needed or providing the support from the start and gradually fading it out, when the competence of the students increases [14].

Social or peer feedback based on the practice, that feedback to tasks and assignments is given from one student to another.

2) Game elements

The differentiation of game-elements is based on the identified game-elements in [3].

3) Type of implementation

The categories for different types of implementation were mainly created from the preliminary study.

4) Reason of usage

Reference [3] and [4] identified different categories for the use of feedback or game-elements.

IV. Results

In total 247 publications were found using the initial search. Initially, 11 of them were duplicates that were included by the search. Due to the exclusion criteria or availability 186 were rejected. Finally, 50 papers were included. Their year of publication ranges from 2001 to 2020 with the majority of publications (44) published after 2013.

A. RQ1: How Are the Types of Feedback Distributed in Computer Science Education?

From the 50 accepted and analyzed papers, 31 are formative or process feedback, 19 summative or corrective feedback, 17 immediate or rapid feedback, 12 self-regulation feedback, 9 social or peer feedback, and 4 about feedback through scaffolding. As these results include contextual overlaps, Fig. 1 shows the distribution of these feedback types and how often they occurred together. The top row shows the total amount of each type, as stated above. The following rows show, how often each type occurred with another one or itself. The percentage indicator is referring to the total amount of occurrences and the numbers in each section of a bar, refers to the individual of occurrences as a pair or alone.

TABLE I: Classification Scheme for the Study

Facet	Category
Feedback in Education	Formative Feedback [11] Summative Feedback [11] Immediate Feedback [12] Self-Regulation Feedback [13] Scaffolding [14] Social/Peer Feedback
Game Elements [3]	Assistance Badges/Achievements Block-programming Puzzles Challenges Customization/Avatars Increasing/Changing Difficulty Rankings Onboarding Performance Statistics Points/Scores/XP Progress/Status Bars Rewards
Type of Implementation	Scaffolding System Monitoring Gamification Game/Game-Development Code Testing Course Redesign Development Environment Questionnaire Block Programming Instant Feedback Social/Peer Feedback
Reason of Usage [3] & [4]	Correctness Empowerment Enhancing Engagement Enjoyment/Fun Goal/Progression Help/Cues/Reinforcement Immersion Motivation Perceived Competence

B. RQ2: How Is This Feedback Implemented?

The results for the implementation type include overlaps for reporting multiple techniques in one publication. Code testing is implemented or taught by 13 papers, whereby the participants were either given feedback through automated tests or they could self-evaluate their work by using test driven development procedures.

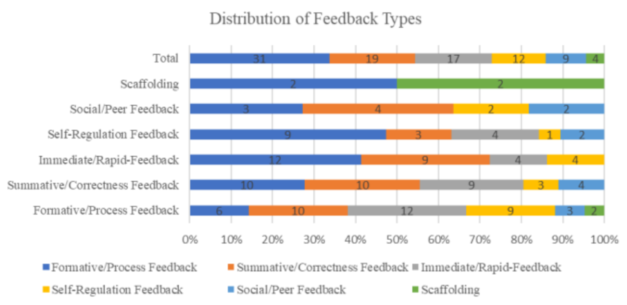

Fig. 1. Distribution of feedback.

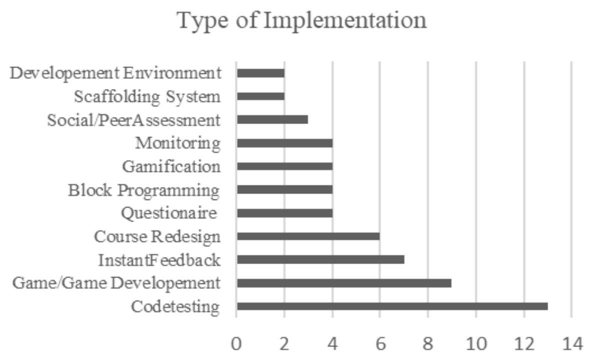

Fig. 2. Implementation type of the feedback.

In 9 papers the participants played a serious or educational game or got taught about game development, where in both parts the participants received feedback by the game they played or programmed. The implementation of immediate feedback is done by 7 papers, through mostly prewritten feedback for certain code cases or compiler states. Another 6 papers report about the redesign of the educational courses. These redesigns range from working with large groups or global learning, over monitoring and motivation to the time, when feedback should be given. Additionally, 4 papers each reported about the implementation of multiple-choice questionnaires for knowledge assessment, gamification to boost participants motivation, monitoring of progress or code performance and block programming like Scratch to program simple application or control a game. Social or peer assessment is used as feedback by 3 papers. Another 2 papers report the implementation of a development environment to help students through the development process, further 2 papers report the use of scaffolding as prewritten helping instructions either from the start of the exercise or when the participant needs them. Fig. 2 shows the different implementations for feedback in the examined publications. The individual implementations for the feedback are describes in the findings section.

C. RQ3: Which Gamification-Elements Are Used?

Only 15 papers are specifically in gamification or game-like context and 10 aren't connected to this context. The results include the overlap for multiple uses of gamification-elements. Overall, 16 papers report using feedback about the performance, 16 allow a better monitoring of lecture and exercise progress, 13 offer additional assistance either automatic or on demand, 13 use points, 10 promote challenges like social challenges against other participants, 8 use badges or achievements, 6 use an escalating difficulty to drive the participants knowledge forward, 4 allow some sort of customization, mostly in the design of the application or its content, 4 include an onboarding process or tutorial, 4 are block programming puzzles, 3 include rewards, mostly in the game, and 2 had a ranking or leaderboard, where students get ranked based on their points or performance. This distribution is shown in Fig. 3.

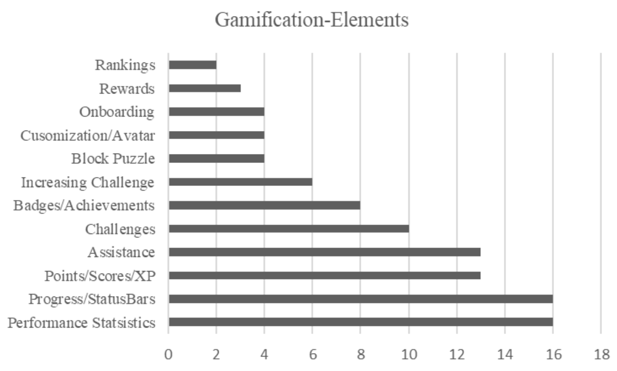

Fig. 3. Used gamification-elements.

D. RQ4: For What Reason Is the Feedback Implementation Used?

From the studied work, 10 papers did not explicitly report any reasons for the research, 28 want to enhance student engagement, 23 to give a better sense of progression and goal orientation, 17 to help the students in their work or progress, 15 for better correctness of the submitted assignments, 12 to enhance the student motivation, 11 for better perceived competence, 7 for student empowerment, 6 for enjoyment and fun, and 1 paper wants to enhance the immersion for the tasks. Fig. 4 shows the stated reasons for which the authors used their implementation, which contains overlaps for multiple stated reasons.

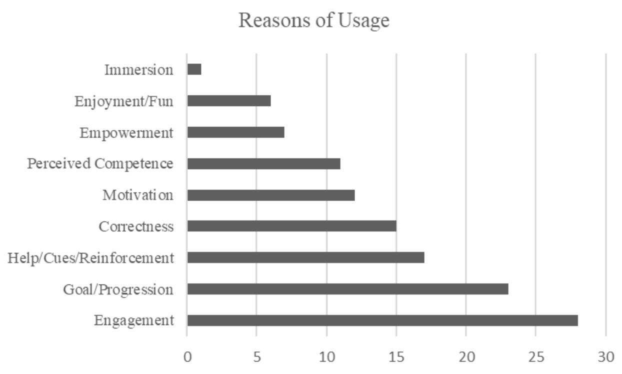

Fig. 4. Stated reasons of usage.

E. RQ5: What is the actual outcome?

Overall, 17 papers miss any indication of an outcome related to the feedback and 1 reports a neutral outcome where no significant difference between the groups was reported. Another paper reported a negative outcome, that related to the overall content and teaching staff.

Related to a positive outcome, 9 papers describe an overall satisfaction with the new course or system, but that where without any control group. Therefore, Fig. 5 shows the outcome reported by the authors, respectively the reported outcome through evaluation of their design or implementation. The majority of which were compared to traditional teaching methods. In total, 10 papers report a better engagement of the participants, 9 report an overall better performance, which resulted in a higher rate of submitted assignments, 6 report a higher student motivation, 3 report that the students were better at self-pacing their learning, 3 report a qualitative better code, 2 report a higher student satisfaction, and 1 has a better onboarding for inexperienced participants.

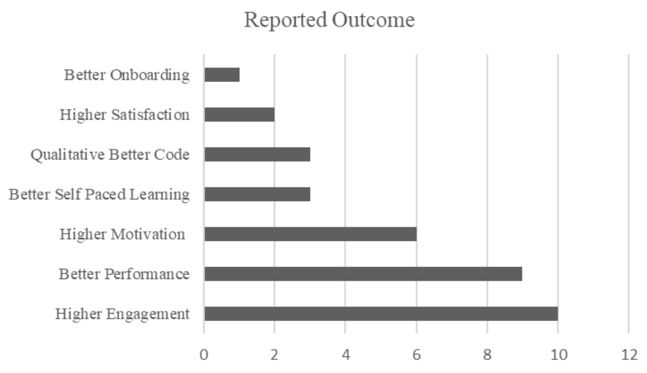

Fig. 5. Actual reported outcome.

V. Findings

Based on the classification for different feedback types in combination with their respective type of implementation and use of game-elements, there seem to be promising approaches that have a positive influence on the learning behavior of students. For this purpose, automated code assessment can be viewed as an aspect of feedback that can be used in different formative or summative ways. Similar to automated assessment, game or game-like elements can represent driving or supporting feedback by different forms of implementations. A general approach that has not yet been named, is problem learning, which emerges as regulation feedback and supports social feedback factors. The other forms of feedback that shall be mentioned here, is scaffolding as helpful structures and feedback mechanism and monitoring as self-regulation feedback, since these can help students in overcoming the obstacles and help them to not lose sight of their learning goals.

A. *Automated Assessment as Feedback*

As automated feedback is the most frequently mentioned part of feedback in computer science education, there are different varieties explored by the reviewed publications. These can be divided into automated testing of code, automated feedback that is displayed at certain cases, qualitative code assessment like metrics and the teaching about testing or test-driven development, so that students can assess themselves.

As previously shown in Fig. 2, code testing is the most prominent option to provide formative feedback, since these tests can be created to guide the students step by step towards the correct solution, by showing them in an easy to understand way, what is and isn't working [15]-[17]. However, one has to distinguish between these test as formative and as summative assessment, because many publications use automated testing in the background to assess and grade their students, but do not give them direct feedback until the end of the assignment [18].

Besides these uses of tests, [9], [19]-[21], describe the effective use of feedback that is provided to the students, in reaction to certain test results. Thereby prewritten feedback is displayed, based on certain criteria, like failed or successful tests or certain values for code metrics.

Other publications like [22], [23] teach about testing and test-driven development. They rely on the students to develop tests and ensure that their assignments work according to them. This can improve the motivation of students, but as [22] found, students that are especially confident in their programming skills, tend to skip writing tests, when the assignments difficulty does not demand their full attention.

In a similar way, code or test metrics can be used to give students a direction of how to improve their assignments. [9]. This approach requires further knowledge from students especially in the context of an introductory computer science course.

Besides the differences in the implementation, the different scholars highlight the importance of rapid or immediate feedback, so that students can assess themselves on their own terms and improve their assignments.

B. *Game-Like Feedback*

Since the use of feedback in games is an essential part, it seems more than relevant to check in which gameful way feedback is and can be provided for computer science education. As the results of this study show, there are already different game-like elements in play, that distribute over the application of gamification, serious games, block-based programming and elements of game development in the educational field.

As [24] argues, the use of game-elements should not just be an "-ification", meaning that just the use of points and badges as extrinsic rewards will not be able to foster the motivational benefits, that some scholars like to achieve with gamification. Therefore, the key concept, that connects formative assessment and good games is their ability to show a clear progress and achievable goals [24]. Depending on the design of the gamification, this can still result in different implementations. As it is shown by [24] or [25] badges and points can be applied in meaningful ways to give students a better understanding about their progress. In addition, both provide the students with multiple choice questionnaires for self-assessment and construct programming challenges either with automated tests [24] or with scaffolding and stepwise instructions [25] to guide the students through the learning process.

Since gamification applies the concepts of a game to the environment, serious games are somewhat going the opposite way, by applying the programming context to the game. Approaches like [26], [27], or [28] let students play a game, wherein certain programming related tasks should be performed, which allows to especially enhance the computational thinking of the students, without learning the specifics of a programming language. These games can provide direct visual feedback, especially when visualizing the execution of a programmed algorithm and they can inform the students about different performance stats like runtime or memory usage.

Meeting both approaches in the middle is block-based programming. Different publications take this approach, by letting their student's program with a visual block-based programming language like Scratch [29]-[31]. Similar to a game, the feedback is provided by a visual output for the students inputted code and in addition to this, additional information like scaffolding or hints can be provided.

Similar to the teaching about test-driven development, some scholars teach about game development [32]-[36]. This approach allows the students to apply their knowledge and then see the results based on the executing game. Therefore, the feedback is mostly about the visible progress, students can make towards the functional parts of the game.

C. Problem Learning

Some scholars redesign educational courses with the focus on problem-learning/-solving for programming exercises. The feedback given to students, differs in this respect from the other publications, since it is not focused on being automated feedback, rather the focus lies on feedback based on the problem context [37], [38]. Therefore, the feedback given is specifically contextualized onto the problem space and encourages to correct and improve the current solution.

In addition to the contextual feedback, [39] remarks that this feedback should be fast and regular, as well as identify and reflect on specific aspects of performance that can be improved. Reference [40] shows, that by focusing on tangible problems, students can help each other and thus promote peer evaluation as formative assessment.

D. Scaffolding

Scaffolding itself is not essential part of feedback, but it can help the learners with their assignments. References [25] and [26] take scaffolding as prewritten part of code, which can be provided as a form of feedback, when the learners don't know how to begin with a certain task. This scaffolding feedback can also be used in the form additional comments in the code to nudge learners in the right direction [26], [41]. Therefore, the scaffolding feedback can result in a change of difficulty or amount of work, that the tasks present to the learners [42].

E. Monitoring and Self-paced Learning

Monitoring as part of the feedback giving process. Many monitor the process of the course or the tasks done by the learners [16], [17], [25], [43]. These include the test results from automated tests and other performance statistics like code-metrics. This part of goal tracking can also enhance self-paced learning, when options are provided to the learners, to set their own goals.

VI. Conclusion

The goal of this paper is to give an overview of research issues relating to feedback in computer science courses with respect to their use of game-elements. In conclusion, there are multiple different techniques used to provide feedback, that are either general or otherwise specific to the programming domain.

The results show, that a high amount of feedback in computer science education is provided by tests, that evaluate automatically and can generate additional feedback. Although the use of game elements for feedback is often not the focus of research, but merely a byproduct, their application almost always brings some way of self-monitoring or goal orientation with it.

The effects of feedback and game-elements with respect to the improvement of student results and motivation are mainly positive. Especially the goal orientation, which can be created with the help of various game-elements, contributes to an increase in said motivation. However, only a fraction of the possible game-elements is used for feedback and most often than not, these game-elements are not adapted to meet the context in which they are presented. Therefore, the use of contextualized game-elements as a means of presenting feedback to students should be considered, when developing tools or environments to assist students with their learning process.

Author Contributions

This research was solely conducted by Nico Willert.

Conflict of Interest

The author declares no conflict of interest.

References

[1] J. McGonigal, "Reality is broken: Why games make us better and how they can change the world," Penguin Group, 2011.

[2] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele University, 2004.

[3] J. Koivisto and J. Hamari, "The rise of the motivational information systems: A review of gamification research," *International Journal of Information Management*, vol. 45, pp. 191-210, 2019.

[4] C. Ott, A. Robins, and K. Shephard, "Translating principles of effective feedback for students into the CS1 context," *ACM Trans. Comput. Educ.*, vol. 16, issue 1, 2016.

[5] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams *et al.*, "A survey of literature on the teaching of introductory programming," *ACM SIGCSE Bulletin*, vol. 39, issue 4, pp. 204–223, 2007.

[6] T. Clear, S. Beecham, J. Barr, M. Daniels, R. McDermott *et al.*, "Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review," in *Proc. the 2015 ITiCSE on Working Group Reports*, New York, USA, pp. 1–39, 2015.

[7] H. Keuning, J. Jeuring, and B. Heeren, "A systematic literature review of automated feedback generation for programming exercises," *ACM Trans. Comput. Educ.*, vol. 19, issue 1, article 3, 2019.

[8] A. Luxton-Reilly, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar *et al.*, Introductory programming: A systematic literature review," *ITiCSE 2018 Companion*, NY, USA, pp. 55–106, 2018.

[9] L. P. Scatalon, J. C. Carver, R. E. Garcia, and E. F. Barbosa, "Software testing in introductory programming courses: A systematic mapping study," *SIGCSE 19*, New York, NY, USA, pp. 421–427, 2019.

[10] S. Narciss, "Feedback strategies for interactive learning tasks," *Handbook of Research on Educational Communications and Technology*, 3rd. edition, pp. 125–144, 2008.

[11] D. R. Sadler, "Formative assessment: Revisiting the territory," *Assessment in Education: Principles, Policy & Practice*, vol. 5, no. 1, pp. 77-84, 1998.

[12] J. Metcalfe, N. Kornell, and B. Finn, "Delayed versus immediate feedback in children's and adults' vocabulary learning," *Memory & Cognition*, vol. 37, pp. 1077-87, 2009.

[13] Y. B. Chung and Y. Mantak, "The Role of Feedback in Enhancing Students' Self-Regulation in Inviting Schools," *Journal of Invitational Theory and Practice*, vol. 17, pp. 22-27, 2011.

[14] P. Geert and H. Steenbeek, "The dynamics of scaffolding," *New Ideas in Psychology*, vol. 23, no. 3, pp. 115-128, 2005.

[15] D. Croft and M. England, "Computing with CodeRunner at coventry university: Automated summative assessment of Python and C++ code," *CEP 2020*, New York, NY, USA, Article 1, pp. 1–4, 2020.

[16] N. Barreiro and C. Matos, "A blended learning model for practical sessions," *FedCSIS*, Gdansk, pp. 903-912, 2016.

[17] M. Rahman, R. Paudel, and M. H. Sharker, "Effects of infusing interactive and collaborative learning to teach an introductory programming course," *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, pp. 1-8, 2019.

[18] J. Park, Y. Park, J. Kim, J. Cha, S. Kim *et al*., "Elicast: Embedding interactive exercises in instructional programming screencasts," *ACM Conference on Learning at Scale*, vol. 5, pp. 1-10, 2018.

[19] H. Blau and J. B. Moss, "FrenchPress gives students automated feedback on java program flaws," *ITiCSE*, New York, NY, USA, pp. 15–20, 2015.

[20] D. Croft and M. England, "Computing with codio at coventry university: Online virtual Linux boxes and automated formative feedback," *CEP*, New York, NY, USA, Article 16, pp. 1–4, 2019.

[21] Q. Hao, J. Wilson, C. Ottaway, N. Iriumi, K. Arakawa, and D. Smith, "Investigating the essential of meaningful automated formative feedback for programming assignments," *2019 IEEE Symposium on Visual Languages and Human-Centric Computing*, Memphis, TN, USA, pp. 151-155, 2019.

[22] K. Buffardi and S. Edwards, "A formative study of influences on student testing behaviors," *SIGCSE*, New York, NY, USA, pp. 597–602, 2014.

[23] V. Ramasamy, H. Alomari, J. Kiper, and G. Potvin, "A minimally disruptive approach of integrating testing into computer programming courses," *SEEM*, Gothenburg, pp. 1-7, 2018.

[24] M. Fuchs and C. Wolff, "Improving programming education through gameful, formative feedback," *EDUCON*, Abu Dhabi, pp. 860-867, 2016.

[25] B. Morrison and B. DiSalvo, "Khan academy gamifies computer science," *SIGCSE*, NY, USA, pp. 39–44, 2014.

[26] A. Chaffin, K. Doran, D. Hicks, and T. Barnes, "Experimental evaluation of teaching recursion in a video game," *ACM SIGGRAPH Symposium on Video Games*, New York, NY, USA, pp. 79–86, 2009.

[27] T. Barnes, E. Powell, A. Chaffin, and H. Lipford, "Game2Learn: Improving the motivation of CS1 students," *GDCSE*, New York, NY, USA, pp. 1–5, 2008.

[28] J. Zhu, K. Alderfer, A. Furqan, J. Nebolsky, B. Char, B. Smith *et al*., "Programming in game space: how to represent parallel programming concepts in an educational game," *FDG '19*, New York, NY, USA, article 4, pp. 1–10, 2019.

[29] K. Katchapakirin and C. Anutariya, "An Architectural Design of ScratchThAI: A conversational agent for Computational Thinking Development using Scratch," *IAIT*, New York, NY, USA, article 7, pp. 1–7, 2018.

[30] S. Grover, M. Bienkowski, J. Niekrasz, and M. Hauswirth, "Assessing problem-solving process at scale," *L@S '16*, New York, NY, USA, pp. 245–248, 2016.

[31] W. Wang, R. Zhi, A. Milliken, N. Lytle, and T. Price, "Crescendo: Engaging Students to self-paced programming practices," *SIGCSE '20*, New York, NY, USA, pp. 859–865, 2020.

[32] K. Sung, R. Rosenberg, M. Panitz, and R. Anderson, "Assessing game-themed programming assignments for CS1/2 courses," *GDCSE '08*, New York, NY, USA, pp. 51–55, 2008.

[33] A. Basawapatna, A. Repenning, and K. Koh, "Closing the cyberlearning loop: Enabling teachers to formatively assess student programming projects," *SIGCSE '15*, New York, NY, USA, pp. 12–17, 2015.

[34] A. Dom ńguez, L. de-Marcos, and J. Mart ńez-Herr áz, "Effects of competitive and cooperative classroom response systems on quiz performance and programming skills in a video game programming course," *ITiCSE '20*, New York, NY, USA, pp. 398–403, 2020.

[35] P. Neve, G. Hunter, D. Livingston, and J. Orwell, "NoobLab: An Intelligent learning environment for teaching programming," *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Macau, pp. 357-361, 2012.

[36] S. Basu, B. Disalvo, D. Rutstein, Y. Xu, J. Roschelle, and N. Holbert, "The role of evidence centered design and participatory design in a playful assessment for computational thinking about data," *SIGCSE '20*, New York, NY, USA, pp. 985–991, 2020.

[37] P. Piwek, M. Wermelinger, R. Laney, Robin, and R. Walker, "Learning to program: From problems to code," *Third Conference in Computing Education Practice (CEP)*, Durham, UK, 2019.

[38] E. Rahimi, E. Barendsen, and I. Henze, "An instructional model to link designing and conceptual understanding in secondary computer science education," *WiPSCE '18*, New York, NY, USA, article 11, pp. 1–4, 2018.

[39] C. Izu and B. Alexander, "Using unstructured practice plus reflection to develop programming/problem-solving fluency," *ACE '18*, New York, NY, USA, pp. 25–34, 2018.

[40] Y. Guo, H. Zhao, K. Wang, and M. Li, "Formative and summative assessment in university programming course: Mediation of problem-based learning and moderation of peer evaluation," *15th ICCSE*, Delft, Netherlands, 2020, pp. 112-117, 2020.

[41] M. Dorodchi, A. Benedict, and E. Al-Hossami, "CS1 Scaffolded activities: The rise of students' engagement," *ICER '19*, New York, NY, USA, 299, 2019.

[42] A. Bart, E. Tilevich, S. Hall, T. Allevato, and C. Shaffer, "Transforming introductory computer science projects via real-time web data," *SIGCSE '14*, New York, NY, USA, pp. 289–294, 2014.

[43] C. Law, J. Grundy, A. Cain, R. Vasa, and A. Cummaudo, "User perceptions of using an open learner model visualisation tool for facilitating self-regulated learning," *ACE '17*, New York, NY, USA, pp. 55–64, 2017.

N. Willert earned his master degree in business informatics at the University of Leipzig, Germany, in 2019. His major field of study is the gameful-design of systems. Currently he is part of the Scientific Staff at the information systems institute of the University Leipzig. Previously, he worked as a quality assurance engineer.