Towards a Framework of Planning Collaborative Learning Scenarios in Computer Science

Anja Hawlitschek
Otto-von-Guericke-Universität
Magdeburg, Germany
anja.hawlitschek@ovgu.de

Sarah Berndt
Otto-von-Guericke-Universität
Magdeburg, Germany
sarah.berndt@ovgu.de

Sandra Schulz
Universität Hamburg
Hamburg, Germany
sandra.schulz@uni-hamburg.de

ABSTRACT

The planning and implementation of collaborative learning is perceived by teachers as demanding and time-consuming. In addition to individual learners, groups must be taken into account – with their group dynamics, demands on group coordination, and group experiences. Our aim is to develop a framework for the instructional design of collaborative learning in computer science to support a systematic evidence-based implementation. In the article, we describe a work in progress framework, encompassing didactic analyses and decision fields, which are relevant for the planning phase. We provide recommendations for actions based on the results of empirical studies as well as a short teaching vignette to illustrate application in practice.

CCS CONCEPTS

• **Social and professional topics → Model curricula**.

KEYWORDS

collaborative learning, curricula planning, programming courses, higher education, computer science education

ACM Reference Format:
Anja Hawlitschek, Sarah Berndt, and Sandra Schulz. 2021. Towards a Framework of Planning Collaborative Learning Scenarios in Computer Science. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3488042.3488044

1 INTRODUCTION

The integration of collaborative forms of learning into computer science courses offers many advantages. The ability to work well in a team is an important competence in many IT professions [33]. Developing skills in this regard should therefore play a role in undergraduate education. Collaborative learning can also help to increase the quality of learning processes. If learners explain contents to each other, ask questions, or discuss, they often invest a lot of mental effort in dealing with contents and have a greater learning outcome [42]. Team members bring additional experiences, information, ideas, perspectives, or evaluations into the learning process, from which common positions, products and knowledge bases have to be developed. Results from meta-analyses indicate better learning performance for learners in teams compared to individual learners [5]. Especially in computer science, students can often benefit from collaborative learning. Here, studies on pair programming show an advantage for learning programming in groups compared to solo programming [16]. The social component of the learner's integration into a team also has a motivational effect (especially at the beginning of the study) that should not be underestimated [29].

However, the integration of collaborative learning in courses poses challenges for teachers (and learners). Collaborative scenarios are characterized by a high degree of complexity in planning and implementation, since in addition to the individual learner, the group, with its group dynamics, with requirements for group coordination, with group dispositions, and group experiences must also be taken into account [13, 43]. Instructors perceive planning and supervision as time and cognitively demanding, especially in terms of coordinating and structuring teamwork, selecting and creating appropriate learning materials, and guiding social interaction [13, 14]. The competencies of teachers in this regard are very heterogeneous [43] and often poor in terms of structuring groups and time management [36]. In practice, structuring of teamwork is often left to students and receives little instructional guidance and support from instructors [14]. Meta-cognitive activities, such as reflection on and evaluation of teamwork, are usually given little to no space [43]. However, didactic planning and instruction is essential to the effectiveness of teamwork [24]. Teamwork is challenging also for learners, with high demands on teamwork skills as well as self-regulation. If learners perceive the costs of teamwork (time, cognitive, emotional) as too high, or the added value as too low, or their competencies insufficient, failure of such learning scenarios may result [30].

The goal of our work is to support the process of instructional design of collaborative learning scenarios in computer science by structuring the fields of analysis and decision-making. For this purpose, we develop a framework that facilitates the systematic evidence-based implementation of such scenarios. The model distinguishes between didactic analysis, which serve as a basis for decisions, and instructional design, which concern the concrete implementation.

2 PLANNING OF COLLABORATIVE LEARNING SCENARIOS

The terms cooperative and collaborative learning are used inconsistently. However, in the literature on computer-supported collaborative learning (CSCL), any form of learning in a team is referred

Koli Calling '21, November 18–21, 2021, Joensuu, Finland
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8488-9/21/11.
https://doi.org/10.1145/3488042.3488044

to as "collaborative learning" [32]. We adopt this understanding of the term in this article.

Three phases can be distinguished in the implementation of collaborative learning scenarios in courses [21]. (1) In the planning phase, teachers lay the foundations for the success or failure of teamwork by making didactic decisions about the process, structure, and organization. (2) During the implementation, they are responsible for guiding and supporting learners. (3) In the reflection phase, the aim is to evaluate the practical implementation and to identify possibilities for improvement for subsequent courses. This paper provides an orientation to the planning phase as the first step towards a substantial framework. In planning collaborative scenarios, didactic analyses have to be carried out. On the basis of those analyses, didactic decisions can be made. Based on existing didactic models [22], the following fields of analyses can be distinguished in the planning phase: The framework conditions resulting from the context of the learning scenario, the learner characteristics, and the teaching-learning objectives. Decisions on instructional design concern the structuring of the collaboration, the group formation, and the learning activities [8, 21]. In the following, the aspects of didactic analysis and instructional design based on a review of empirical studies from the field of computer science and CSCL are explained in more detail. We illustrate the application of the analysis steps and design decisions with a short teaching vignette for the following scenario: A programming course for 80 second semester computer science students. In a lecture students learn basic knowledge on the programming language. At home they have to solve programming tasks. In exercises, each for 20 students, they discuss their solutions and get help for problems. Teachers implement collaborative programming to prepare students for programming projects in IT-companies.

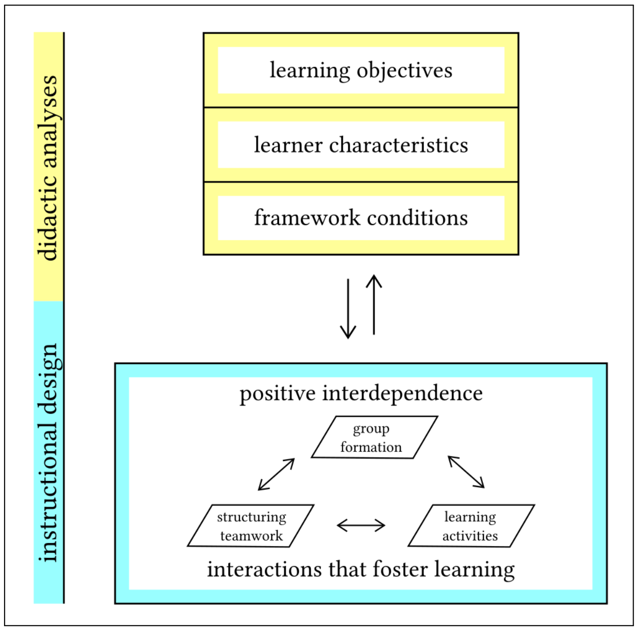

Figure 1: Framework of collaborative learning scenarios

2.1 Didactic analyses

The framework conditions of learning scenarios (see Fig. 1) can have a decisive influence on their implementation and success. Before implementing collaborative learning, it should be determined whether or to what extent the institutional framework conditions are appropriate, e.g., whether examination regulations allow for collaborative forms of examination, whether classrooms or learning platforms are conducive to teamwork, or which forms of learning appear possible based on the number of students. Socio-cultural framework conditions (e.g., specific attitudes in the academic discipline, international study programs) can also be relevant, since attitudes and approaches to collaborative learning differ [9] and common foundations in terms of academic terminology and approaches must first be laid for successful teamwork, which is a time-consuming undertaking [31].

Cognitive and meta-cognitive, motivational, and emotional learner characteristics have to be considered for instructional design, e.g., competencies in teamwork [24], the extent of intrinsic or extrinsic academic motivation [34], or perceived self-efficacy [45]. Regulatory competencies are of particular importance for teamwork, since in addition to learner's own self-regulation in the learning process, the regulation of the team must also works well [28]. It is important to identify potential obstacles to teamwork and address them in the instructional design. In particular, the problems associated with having novices and experts in the course should be highlighted. While students with a lot of previous experience in teamwork need significantly less instructional guidance and support, the opposite is true for students with little prior experience [8]. Therefore, analyzing and considering such learner characteristics is important for the success of teamwork. Similar importance can be assumed for negative attitudes towards teamwork and the effects on motivation. Here, a clarification of values and goals of the learning activities may help. Lack of regulatory competences can be compensated by didactic instructions in this regard [28].

With regard to the learning objectives, the planning process must work out, which objectives are specifically associated with teamwork. Research results indicate that teamwork in computer science is mainly applied to improve content-related learning success and to increase performance in programming [39]. However, collaborative scenarios such as pair programming are usually more time-consuming than individual work [16]. This must be taken into account when planning. However, the learning objectives can also include concrete competencies related to teamwork, e.g., because these are needed in later professional life or are an important skill in computer science studies. In that case, it is important that students can actually practice these competencies, e.g., communication in a group, analyzing and overcoming conflicts in a programming team, or organizing teamwork. The task and the instructional guidance must be aligned towards this gain in competence.

Teaching vignette: Framework condition: Course takes place online, Moodle and GitLab for online activities available, collaborative exams are possible; Learning characteristics: heterogeneous prior knowledge in programming, low skills in collaborative software engineering; Learning goals regarding collaboration: get started with project management and version control in software projects.

2.2 Instructional design

Didactic decisions for the implementation of collaborative learning are made on the basis of the analyses results. In addition, two central principles for successful collaborative learning should guide teachers here: achieving positive interdependence between the group members and promoting interactions that foster learning. Positive interdependence is promoted when every group member depends on every group member, all must contribute to the learning process, and group success can only occur when all group members succeed [19]. Interdependence can be fostered, for example, through shared group goals, distributed learning resources, roles, incentive systems for good teamwork, and tasks adapted to these goals [4]. Interactions between students in the collaboration process are key components of learning. The learning activity should be designed in a way that promotes interactions like structuring and summarizing content for each other, giving feedback to the partner, answering questions, discussing different arguments [12]. Such cognitive activities improve learning because students have to select, organize, and integrate knowledge [26]. One way to promote meaningful interactions is using collaboration scripts for structuring teamwork [8]. If positive interdependence and high quality interactions in a team occur, this also has a positive effect on individual learning outcomes in terms of content [41].

2.2.1 Structuring teamwork. The extent to which teamwork should be pre-structured by the teacher depends on the learning objectives but also on the skills of the learners. Collaborative learning can be very little structured, such as ad hoc discussions on a specific topic in small groups. However, there are also scenarios with a very fine-grained structure like pair programming. One structuring option is the use of collaboration scripts. Collaboration scripts are "a set of instructions regarding to how the group members should interact, how they should collaborate and how they should solve the problem" [8]. The degrees of freedom that learners have in designing their collaboration vary from script to script. Some scripts only provide a kind of framework for organizing the collaboration, such as the Research Structure Confront script, others are very rigidly pre-structured and leave little room for independent planning and implementation of the collaboration, such as the Group Puzzle.

Pair programming can also be considered as such a collaboration script. Results of a literature review [39] suggest that pair programming is the predominant form of structuring in collaborative learning scenarios in computer science when it comes to learning programming. Less commonly, different forms of peer reviews are used as scripts. One implementation for a review script is to have learners program individually in the first step, then give each other feedback on the resulting programs, and have them create a group solution in the final step [11]. Cross-group reviews for group solutions are also an implementation example. When the focus is on knowledge production (e.g., writing texts on specific computer science topics), various forms of structuring teamwork in computer science are used, e.g., group puzzles, distributing different roles, or, again, peer review processes. Despite such scripts being available, in current practices of teaching it is often left to the students to decide how to design the teamwork and distribute the task [39]. However, studies suggest benefits of collaboration scripts for the learning process. Compared with non-guided collaboration, scripts can increase not only content learning but also knowledge of important principles of successful collaborative learning [35]. Results of meta-analyses suggest that collaboration scripts are particularly effective when they stimulate "transactive activities" [44], i.e., when the already mentioned meaningful interaction between learners occurs, in which they use each other as a source of information, identify and discuss conflicts as well as differences in their knowledge and beliefs, ask questions, provide explanations, articulate, apply, and extend shared knowledge, and develop a common understanding of the learning content. Specific instructions and guidance can be integrated into the script for this purpose [31]. Learners with little prior knowledge and skills (both in terms of content and teamwork) especially benefit from highly guided forms of collaborative learning, such as pair programming [16, 37]. Strongly structured collaborative scripts are also recommended for group members with differing levels of knowledge and teamwork experience. Sharing ideas and reflecting on one's own and team members' program code supports the learning process.

Teaching vignette: Students are novices in collaboration, so a highly structured script is necessary. Teachers apply a review script: Student 1 solve Task 1 individually in the first step, then request feedback. Student 2 provide feedback in a second step. Both students generate a joint solution in the third step, which will be discussed in the exercise. In Task 2, Student 2 starts.

2.2.2 Group formation. Teachers should not leave the formation and size of groups to chance, as this can also have an impact on the success of the collaborative scenario. Of course, the optimal size of groups also depends on the learning objectives. If approaches to working in industrial projects are to be practiced with large groups, the instructional design must be adapted to this. However, it is important to keep in mind that learners' coordination effort increases with increasing group size [24]. Regarding the composition of groups, the results of empirical studies are not clear. With respect to programming learning, the results of literature reviews suggest that homogeneous groups consisting of learners with similar programming skills and experiences achieve best learning outcomes [37]. Given the assumption that learners with little prior knowledge in particular can benefit from more experienced team members [6], this is a surprising result. On the one hand, this could be due to the fact that students with a lot of programming experience take over too many programming activities in a heterogeneous group, either because of efficiency considerations or because of a lack of self-confidence of the inexperienced group members. Another possible explanation is that the experienced group members lack the ability to explain programming concepts and may be overwhelmed with supporting the inexperienced team members [7]. Guidance in this regard could support heterogeneous groups, e.g., the visualization of the activities of individual group members, like written lines of code, or a clear division of team work with assigned work packages.

Teaching vignette: Teachers provide a short prior knowledge test in Moodle. Based on results teachers build homogeneous groups of 2 students. Teachers provide a clear procedure on how to proceed if problems in a group occur.

2.2.3 Learning activities of the learners. The basic question in planning learning activities is: What do the learners have to do in order to be able to achieve the learning objectives? What knowledge must they acquire, what experiences must they gain, what activities must they perform? Planning learning activities involves a wide range of didactic decisions – e.g., on the design of tasks, on the distribution of learning resources, on forms of interaction and use of certain media, on the measurement of learning performance – which we address only selectively in this article. When planning teamwork, it is often advised to distribute resources to learners in order to create positive interdependence. However, this approach can lead to teamwork problems for learners with little prior content knowledge if the distributed learning content provides knowledge about core concepts or procedures of a subject or topic area [6]. The distribution of roles, on the other hand, has positive effects on interactions in groups [4]. Here learners work on a task or sub-tasks in different roles, e.g., stakeholder and developer. Often sub-tasks are integrated at the end into a common product. In software engineering in industry it is quite common that people in different roles work together in a project, so the distribution of roles in collaborative learning is also a good practice to prepare students for this [38]. When creating tasks, it should be kept in mind that they should not be too simple or too short. The higher effort for the implementation of collaborative learning, such as pair programming, must seem reasonable to the students in terms of task design [3]. Regarding the forms of interaction between learners, the following decisions should be made: Should learners collaborate synchronously or asynchronously? Should learner meet digitally (via conference tools like Teams or Zoom) or face-to-face? Which tools can be used for digital collaboration (e.g. Trello, Gather, Code Runner, Google Colaboratory)? Learners seem to prefer face-to-face collaboration in pair programming [10]. However, in terms of effectiveness, distributed pair programming is similarly effective [20]. Digital asynchronous collaboration can, in some circumstances, increase learning success compared to face-to-face processing [23]. If tasks are discussed textually instead of verbally, the processing becomes more elaborate, but of course this also increases the time required. In computer science, this form of collaboration is mainly used for discussions, but it can also be used in programming exercises [39].

The assessment of group tasks is a difficult topic. In a majority of studies, the measurement of learning outcome is done with individual tests after the end of teamwork or via performance in individual sub-tasks. Sometimes group tasks are assessed and each team member receives the same grade [40]. Here, the focus is predominantly on measuring content learning success; measurement of learner performance regarding teamwork is not usually done [15]. One exception is the approach of incorporating group members' assessments of partner's performance [38]. However, there are problems associated with these forms of assessment: Individual grades on individual sub-tasks or tests do not take group performance into account and can reduce motivation to work in teams. Especially when team skills are explicitly part of the learning objectives, they should also be part of an assessment. Group assessments, on the other hand, can quickly lead to negative attitudes toward teamwork if learners experience that uncommitted team members negatively affect the group's performance or benefit from the performance of others [40]. Learners' openness to group grades is often rather low for this reason [25]. An alternative, if the examination regulations allow for it, can be grades that consist partly of an assessment of group performance and partly of an assessment of individual performance during teamwork [25]. Transparency with regard to the evaluation criteria should be created in advance.

Teaching vignette: Since students start in the programming course with very easy tasks to understand the basics, teachers decide to start collaboration not until the third task. In the lecture, teacher explain the relevance of teamwork, key factors of successful software development, and introduce GitLab. In the script, teacher refer to GitLab features students should use. All learning activities take place online. Lecture and exercise are synchronous learning activities via virtual classroom software, teamwork takes place mainly asynchronous. At the end of semester, teacher conduct an exam where grades arise from a collaborative programming project as well as an individual oral examination.

3 DISCUSSION

The importance of careful didactic planning of collaborative learning scenarios can hardly be overestimated. For this purpose, we have described fields of analyses and instructional design decisions. Based on the results of empirical studies, we provided recommendations with a focus on learning programming and a short teaching vignette as illustration. We consider it particularly important to take into account students' prior knowledge, skills, and experience in teamwork and to adapt the instructional design accordingly. However, even the best didactic analysis and planning can lead to sub-optimal results in collaborative learning processes if the cognitive activities remain superficial. In this respect, student interaction should not only be planned but also monitored and supported [21]. Teachers should consider in advance how they want to support students in the implementation phase beyond structuring teamwork and planning team activities, and whether they want to use tools (e.g., Git, Moodle) for this purpose. One way to support teamwork through analysis of team activities is to use dashboards. Even implementing visual feedback on team participation can support successful teamwork. Data can be derived from multiple sources. Often programming and behavior activity logs from learning platforms are used, for example percentage of correct exercises per week or time spent on learning platform [1]. There are also attempts to integrate user interaction data from multiple technologies [2]. For the instructional design of feedback via dashboard different approaches can be distinguished. While some studies provide students with raw data or aggregated raw data without further interpretation aids [17], other studies provide best-practice models for comparison or give guidance on how to improve teamwork [18]. The results of Jermann and Dillenbourg (2008) suggest that such interpretive tools for visual feedback help students to adjust their activities in collaborative teamwork. In addition, reflection phases are another way to support teamwork. Here, group members should periodically consider how well they are working together and how they want to improve their performance [27]. The use of such dashboards or reflection techniques by no means eliminates the need for instructional guidance and support from instructors, but they can make their task easier by alerting about potential problems in collaboration early on.

REFERENCES

[1] David Azcona, I-Han Hsiao, and Alan F. Smeaton. 2019. Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints. *User Modeling and User-Adapted Interaction* 29, 4 (2019), 759–788.

[2] Aneesha Bakharia, Kirsty Kitto, Abelardo Pardo, Dragan Gašević, and Shane Dawson. 2016. Recipe for success: Lessons learnt from using xAPI within the connected learning analytics toolkit. In *Proceedings of the 6th International Conference on Learning Analytics & Knowledge* (Edinburgh, Scotland). ACM, 378–382.

[3] Nicholas A. Bowman, Lindsay Jarratt, KC Culver, and Alberto M. Segre. 2020. Pair programming in perspective: Effects on persistence, achievement, and equity in computer science. *Journal of Research on Educational Effectiveness* 13, 4 (2020), 731–758.

[4] Susan Brewer and James D. Klein. 2006. Type of positive interdependence and affiliation motive in an asynchronous, collaborative learning environment. *Educational Technology Research and Development* 54, 4 (2006), 331–354.

[5] Juanjuan Chen, Minhong Wang, Paul A. Kirschner, and Chin-Chung Tsai. 2018. The role of collaboration, computer use, learning environments, and supporting strategies in CSCL: A meta-analysis. *Review of Educational Research* 88, 6 (2018), 799–843.

[6] Anne Deiglmayr and Lennart Schalk. 2015. Weak versus strong knowledge interdependence: A comparison of two rationales for distributing information among learners in collaborative learning settings. *Learning and Instruction* 40 (2015), 69–78.

[7] Ömer Demir and Süleyman S. Seferoglu. 2020. The effect of determining pair programming groups according to various individual difference variables on group compatibility, flow, and coding performance. *Journal of Educational Computing Research* 59, 1 (2020), 41–70.

[8] Pierre Dillenbourg. 2002. Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In *Three worlds of CSCL. Can we support CSCL?*, Paul A. Kirschner (Ed.). Open Universiteit Nederland, Heerlen, 61–91.

[9] Anastasios A. Economides. 2008. Culture-aware collaborative learning. *Multicultural Education & Technology Journal* 2, 4 (2008), 243–267.

[10] Richard L. Edwards, Jennifer K. Stewart, and Mexhid Ferati. 2010. Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads* 1, 1 (2010), 48–54.

[11] Eustaquio S. M. Faria, Juan M. Adán-Coello, and Keiji Yamanaka. 2006. Forming groups for collaborative learning in introductory computer programming courses based on students' programming styles: An empirical study. In *Proceedings. Frontiers in Education. 36th Annual Conference* (San Diego, California). IEEE Computer Society, Los Alamitos, CA, USA, 6–11.

[12] Logan Fiorella and Richard E. Mayer. 2016. Eight ways to promote generative learning. *Educational Psychology Review* 28, 4 (2016), 717–741.

[13] Robyn M. Gillies and Michael Boyle. 2010. Teachers' reflections on cooperative learning: Issues of implementation. *Teaching and Teacher Education* 26, 4 (2010), 933–940.

[14] Robyn M. Gillies and Michael Boyle. 2011. Teachers' reflections of cooperative learning (CL): A two-year follow-up. *Teaching Education* 22, 1 (2011), 63–78.

[15] Carmen L. Z. Gress, Meghann Fior, Allyson F. Hadwin, and Philip H. Winne. 2010. Measurement and assessment in computer-supported collaborative learning. *Computers in Human Behavior* 26, 5 (2010), 806–814.

[16] Jo E. Hannay, Tore Dybå, Erik Arisholm, and Dag I. K. Sjøberg. 2009. The effectiveness of pair programming: A meta-analysis. *Information and Software Technology* 51, 7 (2009), 1110–1122.

[17] Jeroen Janssen, Gijsbert Erkens, Gellof Kanselaar, and Jos Jaspers. 2007. Visualization of participation: Does it contribute to successful computer-supported collaborative learning? *Computers & Education* 49, 4 (2007), 1037–1065.

[18] Patrick Jermann and Pierre Dillenbourg. 2008. Group mirrors to support interaction regulation in collaborative problem solving. *Computers & Education* 51, 1 (2008), 279–296.

[19] David W. Johnson and Roger T. Johnson. 2009. An educational psychology success story: Social interdependence theory and cooperative learning. *Educational Researcher* 38, 5 (2009), 365–379.

[20] Soojin Jun, Seungbum Kim, and Wongyu Lee. 2007. Online pair-programming for learning programming of novices. *WSEAS Transactions on Advances in Engineering Education* 4, 9 (2007), 187–192.

[21] Celia Kaendler, Michael Wiedmann, Nikol Rummel, and Hans Spada. 2015. Teacher competencies for the implementation of collaborative learning in the classroom: A framework and research review. *Educational Psychology Review* 27, 3 (2015), 505–536.

[22] Michael Kerres. 2013. *Mediendidaktik – Konzeption und Entwicklung mediengestützter Lernangebote.* Oldenbourg Verlag, München.

[23] Yu-Tzu Lin, Cheng-Chih Wu, and Chiung-Fang Chiu. 2018. The use of wiki in teaching programming: Effects upon achievement, attitudes, and collaborative programming behaviors. *International Journal of Distance Education Technologies* 16, 3 (2018), 18–45.

[24] Yiping Lou, Philip C. Abrami, and Sylvia d'Apollonia. 2001. Small group and individual learning with technology: A meta-analysis. *Review of Educational Research* 71, 3 (2001), 449–521.

[25] Janet Macdonald. 2003. Assessing online collaborative learning: Process and product. *Computers & Education* 40, 4 (2003), 377–391.

[26] Richard E. Mayer. 2004. Should there be a three-strikes rule against pure discovery learning? *American Psychologist* 59, 1 (2004), 14–19.

[27] Elsa Mentz, Johannes L. van der Walt, and Leila Goosen. 2008. The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education* 18, 4 (2008), 247–260.

[28] Piia Näykki, Jaana Isohätälä, Sanna Järvelä, Johanna Pöysä-Tarhonen, and Päivi Häkkinen. 2017. Facilitating socio-cognitive and socio-emotional monitoring in collaborative learning with a regulation macro script–an exploratory study. *International Journal of Computer-Supported Collaborative Learning* 12, 3 (2017), 251–279.

[29] Christopher P. Niemiec and Richard M. Ryan. 2009. Autonomy, competence, and relatedness in the classroom: Applying self-determination theory to educational practice. *Theory and Research in Education* 7, 2 (2009), 133–144.

[30] Timothy J. Nokes-Malach, J. Elizabeth Richey, and Soniya Gadgil. 2015. When is it better to learn together? Insights from research on collaborative learning. *Educational Psychology Review* 27, 4 (2015), 645–656.

[31] Omid Noroozi, Stephanie D. Teasley, Harm J. A. Biemans, Armin Weinberger, and Martin Mulder. 2013. Facilitating learning in multidisciplinary groups with transactive CSCL scripts. *International Journal of Computer-Supported Collaborative Learning* 8, 2 (2013), 189–223.

[32] Orlando J. Olivares. 2008. Collaborative vs. cooperative learning: The instructor's role in computer supported collaborative learning. In *Computer-supported collaborative learning: Best practices and principles for instructors*, Kara L. Orvis and Andrea L.R. Lassite (Eds.). IGI global, Heerlen, 20–39.

[33] Alan Peslak, Lisa Kovalchick, Paul Kovacs, Mauri Conforti, Wenli Wang, and Neelima Bhatnagar. 2018. Linking programmer analyst skills to industry needs: A current review. In *Proceedings of the EDSIG Conference* (Norfolk, Virginia). Li-Jen Lester, Huntsville, Texas, USA, 569–574.

[34] Bart Rienties, Dirk Tempelaar, Piet van den Bossche, Wim Gijselaers, and Mien Segers. 2009. The role of academic motivation in computer-supported collaborative learning. *Computers in Human Behavior* 25, 6 (2009), 1195–1206.

[35] Nikol Rummel and Hans Spada. 2005. Learning to collaborate: An instructional approach to promoting collaborative problem solving in computer-mediated settings. *The Journal of the Learning Sciences* 14, 02 (2005), 201–241.

[36] Ilse Ruys, Hilde Keer, and Antonia Aelterman. 2012. Examining pre-service teacher competence in lesson planning pertaining to collaborative learning. *Journal of Curriculum Studies* 44, 3 (2012), 349–379.

[37] Norsaremah Salleh, Emilia Mendes, and John Grundy. 2011. Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 37, 4 (2011), 509–525.

[38] Pilar Sancho-Thomas, Rubén Fuentes-Fernández, and Baltasar Fernández-Manjón. 2009. Learning teamwork skills in university programming courses. *Computers & Education* 53, 2 (2009), 517–531.

[39] Leonardo Silva, António J. Mendes, and Anabela Gomes. 2020. Computer-supported collaborative learning in programming education: A systematic literature review. In *IEEE Global Engineering Education Conference (EDUCON)* (Porto, Portugal). IEEE, 1086–1095.

[40] Jan-Willem Strijbos. 2010. Assessment of (computer-supported) collaborative learning. *IEEE Transactions on Learning Technologies* 4, 1 (2010), 59–73.

[41] Marina Supanc, Vanessa A. Völlinger, and Joachim C. Brunstein. 2017. High-structure versus low-structure cooperative learning in introductory psychology classes for student teachers: Effects on conceptual knowledge, self-perceived competence, and subjective task values. *Learning and Instruction* 50 (2017), 75–84.

[42] Carla van Boxtel, Jos van der Linden, and Gellof Kanselaar. 2000. Collaborative learning tasks and the elaboration of conceptual knowledge. *Learning and Instruction* 10, 4 (2000), 311–330.

[43] Marij A. Veldman, Mechteld F. van Kuijk, Simone Doolaard, and Roel J. Bosker. 2020. The proof of the pudding is in the eating? Implementation of cooperative learning: Differences in teachers' attitudes and beliefs. *Teachers and Teaching* 26, 1 (2020), 103–117.

[44] Freydis Vogel, Christof Wecker, Ingo Kollar, and Frank Fischer. 2017. Socio-cognitive scaffolding with computer-supported collaboration scripts: A meta-analysis. *Educational Psychology Review* 29, 3 (2017), 477–511.

[45] Shu-Ling Wang and Sunny S.J. Lin. 2007. The effects of group composition of self-efficacy and collective efficacy on computer-supported collaborative learning. *Computers in Human Behavior* 23, 5 (2007), 2256–2268.