Experiences and Lessons Learned from Real-World Projects in Software Engineering Subject

Yan Hern Ryan Sim
Computing Science Joint Degree Programme, Singapore Institute of Technology - University of Glasgow,
2717964S@student.gla.ac.uk

Zhi Zhan Lua
Computing Science Joint Degree Programme, Singapore Institute of Technology - University of Glasgow,
2717895L@student.gla.ac.uk

Kahbelan Kalisalvam Kelaver
Computing Science Joint Degree Programme, Singapore Institute of Technology - University of Glasgow,
2717909K@student.gla.ac.uk

Jia Qi Chua
Computing Science Joint Degree Programme, Singapore Institute of Technology - University of Glasgow,
2717945C@student.gla.ac.uk

Ian Zheng Jiang Lim
Computing Science Joint Degree Programme, Singapore Institute of Technology - University of Glasgow,
2717880L@student.gla.ac.uk

Qi Cao
School of Computing Science, University of Glasgow
Glasgow, Scotland, UK
qi.cao@glasgow.ac.uk
ORCID: 0000-0003-3243-5693

Sye Loong Keoh
School of Computing Science, University of Glasgow
Glasgow, Scotland, UK
SyeLoong.Keoh@glasgow.ac.uk
ORCID: 0000-0003-3640-5010

Li Hong Idris Lim
School of Engineering, University of Glasgow
Glasgow, Scotland, UK
LiHonIdris.Lim@glasgow.ac.uk
ORCID: 0000-0002-0865-3829

***Abstract*—Teamwork in software development life cycle (SDLC) and Software Engineering (SE) is a cooperative process that all Computing Science (CS) undergraduates need to undergo. It is a critical skill for the industry and is usually trained through group projects in Higher Education. Due to the nature of software development, most software projects involve collaborative efforts of a group of developers. Although teamwork has been studied in many prior works, it is still considered as a dynamic element in SDLC. As the level of complexity, type of deliverables and range of stakeholders in software projects can vary widely, prior experience cannot be applied directly to new projects. The current implementation of SE education in the Professional Software Development (PSD) and Team Project (TP) subjects contains elements to promote teamwork. Students are required to work in groups on real-world problems. This paper examines the current teamwork simulating real-world software projects through an evaluation with the existing and previous cohorts of students, who have experienced the PSD and TP subjects. Several improvements are then proposed by this study. Based on the results, the majority of the respondents agree that our proposed methods such as self-selection of groups, pair programming, and prototyping model will bring about improved teamwork in their group projects.**

Keywords—Software engineering education, teamwork, learning experience, software development, team projects.

I. Introduction

One of the main goals of the Computing Science (CS) education in Institutes of Higher Learning (IHL) is to train and prepare students for work in software industry. Undergraduate students are taught essential computing concepts, which provides a solid foundation for them to keep up with innovations in their fields. Teaching pedagogy is constantly adopted to deliver technical knowledge in CS subjects. Most IHL have also explored problem-based learning where educators engage students in deep problem-solving and critical thinking [1]. However, having a technical skill set is insufficient for career success. It is essential to train students in teamwork skills to better prepare them for the real work environment to collaborate efficiently with their colleagues. Teamwork is an important soft skill but a complex topic that is challenging to be trained. It is caused by various factors in group projects, such as project timelines, project management strategy, team composition, technical skillsets, individual motivations, levels of self-discipline, ways of individual assessment, etc.

Usually, software projects involve group efforts of software developers. The development of soft skills, such as teamwork, verbal and written communication skills is critical [2]. Prior works have been studied to improve group projects development and teamwork assessment for students to refine their teamwork skills. This is where a CS subject of Software Engineering (SE) comes in, that provides knowledge of software development life cycle (SDLC) on how to deliver high-quality software [3].

In the CS joint degree program of Singapore Institute of Technology – University of Glasgow (SIT-UofG), there are two subjects involving SE: Professional Software Development (PSD) and Team Project (TP). The PSD subject imparts SE knowledge to students with software processes, such as Agile methodology [4]. To reinforce students learning of SDLC and apply their theoretical knowledge, students simultaneously attempt the TP subject, where they work in Agile teams on real-world software projects with customers from software industry [5]. Teamwork is a fundamental factor for success in the TP subject as students are graded on their adherence to the SDLC methodology, SE practices, teamwork, and project deliverables.

This paper aims to discuss and improve the learning experience of students in teamwork for the PSD and TP courses. It analyzes the current teamwork method, followed by a proposed approach to enhance the quality of teamwork where multiple groups work jointly to develop a large scale software project. By improving the current ways of learning, we can make the learning journey more enjoyable and gain essential experience on how real-world software projects work.

II. Related Work

SE education teaches students to the intricacies of SDLC on how to work in group projects [6]. The formation of teams may receive little deliberation, where students may struggle to work efficiently [7]. Groups may consist of novice students, depriving them of advancement in group projects [8]. Some students may feel estranged from their team due to different learning paces, led to dysfunctional team dynamic [9]. Environmental factors such as onsite and remote collaboration in SE courses may impact teamwork. Onsite collaboration increases active participation of team members with facial expressions and body language, which potentially increases teamwork effectiveness [10]. While remote collaboration may result in less engaged teams [11]. Various methods are reported to enhance teamwork in SE education which are discussed next.

Extreme Programming (XP) in Agile principles is a technique to improve team collaboration and communication aiming to successful project completions [12]. High impact of XP to teamwork is reported with a more interactive learning experience [13]. The XP method can be a good fit for courses with small number of students. But workload is quickly increased as the class sizes grow larger [14].

Peer Assessment is utilized to reflect the contributions of individual members. It enables differentiation of personal efforts and fosters participation in teamwork [15]. A peer assessment conducted in [16] shows that larger teams have difficulties in coordination amongst members. Although peer assessments can derive different individual grades in group projects, it abstains from the representation of the overall team dynamics that may result in unfair assessments towards certain members [17].

Team meetings increase collaboration among members and their commitment to software projects. Regular team meetings lead to teamwork at a higher level of efficiency, where members are communicative and work in a coordinated manner [18]. Frequent stand-up meetings and retrospective meetings reduce communication barriers and increase the team synergy [19].

Agile Software Engineers Stick Together (ASEST) framework helps improve team cohesion and learning in SE subjects [20][21]. The ASEST framework increases team contributions and attains project requirements whilst avoiding conflicts [20]. It brings an increase in team performance and cohesion [21]. But cultural factors should be considered, e.g., willingness of students to adhere to rules or time constraints.

Kanban board is to visualize workflow and communicate priorities in SDLC. When a task is not progressing on the board, it flags out that one member may have difficulties in the work. This can encourage all members to work together in group projects [22]. A Kanban survey indicates a 72% reduction in dependency while maintaining a consistent development pace across the team [23]. But the Kanban board may result in the teams working on smaller user stories over the larger ones which can lead to uneven task distributions in the teams [24].

Team Formation: Grouping multiple members and forming a suitable software team is a challenging task [25]. Personality types of members are considered in the team formation in [26]. Team formations can be shaped by academic performance of students, or perceptions of lecturers, that may result in human-dependent and error-prone processes. Students prefer to forming their groups without the intervention of a lecturer [27]. But self-formed groups by students may impact some inexperienced students, who cannot learn from their experienced peers [28]. It may diminish motivation in teamwork.

III. Methodology

A. *Current Implementation*

Prior works on improving team cohesiveness in SE education are reported [29], but there is still room for improvement of teamwork. Our university has implemented several methods currently to help students work effectively as a team and apply SE practices.

Predefined Pseudo Random Groupings: Both the PSD and TP subjects are taught across two trimesters in Year 2 to CS students in parallel [5]. The lecturers allocate students into groups with five or six members each using a predefined pseudo random method. Each group consists of students mixed with different levels of academic performance, for students learning the strengths of each other. It is in the hopes of team bonding over time through the different stages of team dynamics such as Forming, Norming and Storming. Allowing students to experience teamwork in a professional environment setting where members are assigned into different groups by managers.

Scrum Framework: Scrum as an Agile framework and iterative approach in SDLC describes the incremental delivery of products [30]. In Scrum, software projects are broken down into various sprints, a period where the team completes a certain amount of work. Communication within a team is improved through the Scrum ceremonies such as sprint planning meetings, daily stand-ups meetings, sprint review meetings, and retrospective meetings, etc. Students can follow a structured framework to work efficiently in a team.

Peer Evaluations: Students conduct the peer evaluations to each member in the same teams based on their contributions to the group projects. This helps with the teamwork as it prevents freeloading by certain individual members and motivates all members to contribute with their expertise. It provides a means to voice concerns about rogue or non-participating members. Hence, it improves teamwork incentivising and acting as a deterrence to non-contributing or rogue team members.

B. *Proposed Enhancement*

The research questions are what the perceptions on teamwork of the students are in the current learning, and if there are ways to improve the teamwork in the SE education. The current learning of the TP subject involves teamwork and communications not only in the same group, but also with other groups, as students are required to work with several groups to integrate multiple projects into a single large-scale project. For example, a customer company allocated six software projects to six groups, and asked students to integrate into a final software product in 8 months duration. These six projects include front-end user interface, back-end database, software gateways, cloud computing services, mobile applications, and desktop PC software in various operations systems. Each group is required to work on the software project about 3-4 hours per week in 8 months, i.e., two trimesters.

To address the challenges involving multiple groups, the Scrum framework which is taught in PSD is adopted. Students need to perform self-exploration to practice the Scrum framework. Scrum of Scrums is as an advanced method in the Agile methodology where the representatives of each team form a Scrum team to scale Scrum beyond individual teams [31]. But there are mixed opinions and reviews on its effectiveness [32].

For multiple projects integrated by multiple groups, we propose to organize the Scrum team meeting before the start of each sprint. The Product Owner (PO) or Scrum Master (SM) of each team attend this meeting to refine their product backlogs with inputs from other Scrum teams. The representatives bring the knowledge and meeting discussions back to their own teams. Each team can keep up to date with the progress of the projects.

The teamwork for the project integrations by multiple teams becomes more complicated. To improve the teamwork, we propose several enhancements that can be implemented into future cohorts in the learning of PSD and TP subjects.

Self-Selection of Team Members: We propose the team formation of the future iterations to switch to self-selection of team members, for better internal communications and team dynamics. Self-selected team members are usually more motivated to help each other and quicker to resolve conflicts [33]. Teamwork can be enhanced with a more positive learning experience, resulting in members working effectively together.

Pair Programming: Although there is apprehension in pair programming that it might cause free rider syndrome in group projects, the efficacy of pair programming reveals that participants can accomplish tasks quicker than working individually [34]. This is a result of teams covering each other in projects when one developer is unavailable, the other in the pair can continue programming [35]. Source codes can be reviewed and discussed by the pair of developers leading to efficiency in delivering software products on time [34]. As such, pair programming can boost the teamwork in the SDLC.

Prototyping Model: The prototyping process model is typically used for newly formed inexperienced software teams, due to its characteristics of "start small" and "fail fast", when project requirements are unclear. Software prototypes can be developed while feedback is gathered from the customers. It enables the delivery of more elegant solutions. The Scrum approach and spike development of rapid prototyping model could provide a structured procedure to better assist the students [36]. The prototyping model allows good collaboration among teammates. It helps that all team members are harmonious in the project. Students could help each other in the group projects.

IV. Findings and Analysis

The research instrument, survey was selected to collect the opinions of the CS students in our university. It was to analyse the benefits and effectiveness of our proposed enhancements on teamwork. The anonymous survey was conducted by inviting students from the existing and previous cohorts of the CS joint degree programme at SIT-UofG, who have experienced the PSD and TP subjects. The students were randomly selected with different factors such as academic performance and ages to avoid skewed results due to confounding variables. There is a sample size of 30 anonymous participants. For the central limit theorem, the sample size over 30 avoids inaccurate results [37].

A. *Questionnaires and Analysis*

Participants were asked to indicate the software process models used in the TP projects. As shown in Fig. 1, about 83.3% participants indicate that they are using the Scrum framework. This feedback is within our expectations as the TP subject highlights the Scrum methodology. The lecturers also encourage students to practice the Scrum methodology in TP projects.

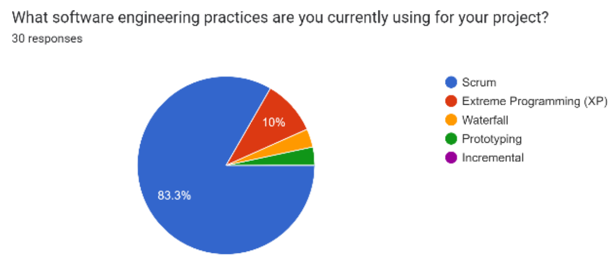

Fig. 1. Survey response on software process models used.

Next, the participants were asked if the self-selection of group members can improve the teamwork, with the Likert scale of "1 – Strongly Disagree" and "5 – Strongly Agree". The survey response is shown in Fig. 2. Most participants agree that being able to choose their own team members will improve teamwork. It supports our first proposed enhancement of self-selected group formations to improve the teamwork in SDLC.

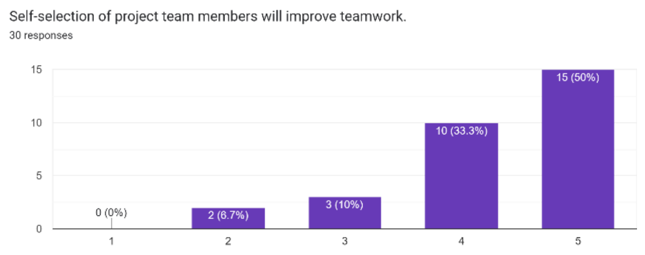

Fig. 2. Responses on teamwork by self-selection of project team members

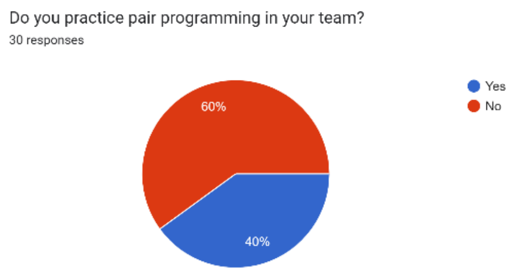

Fig. 3. Survey responses on practice of pair programming

The participants were asked if they adopted the pair programming practice in their projects. From Fig. 3, about 40% of responses show that they use pair programming. This result comes as a surprise, as pair programming was only taught in the PSD subject near the end of the first trimester. We did not expect so many participants to practice this in their TP projects.

As the participants have prior experience in working on team projects, their opinions were sought if the pair programming can help improve the teamwork in their own software projects. As shown in Fig. 4, about 80% of participants agree or strongly agree that their teamwork is enhanced by pair programming. The observations support our second proposed enhancement on teamwork in SDLC.

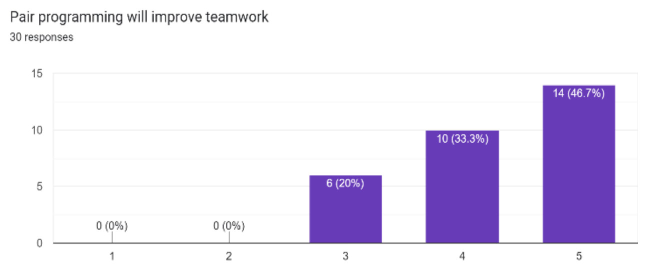

Fig. 4. Responses on teamwork improvement by the pair programming

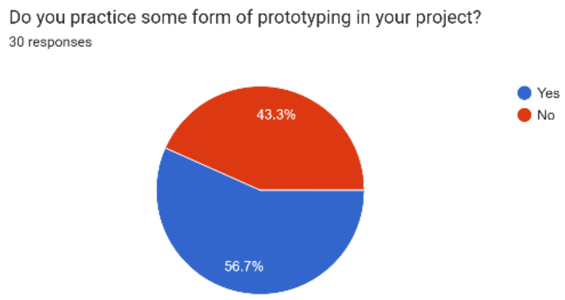

Fig. 5. Survey responses on adoption of prototyping model

Although most teams adopt the Scrum framework in their SDLC with various sprints, some parts of the TP projects need un-taught programming skillsets in Year 2 curriculum, such as database and cloud computing knowledge. The next survey question asks participants if their teams practice the prototyping process model in those parts of the TP projects. The responses are shown in Fig. 5. It is seen that more than half of the participants were practising prototyping model in their projects, even though the prototyping model was not emphasized in the TP module. It means that the prototyping model can be helpful to students in the iteration of the software development phases.

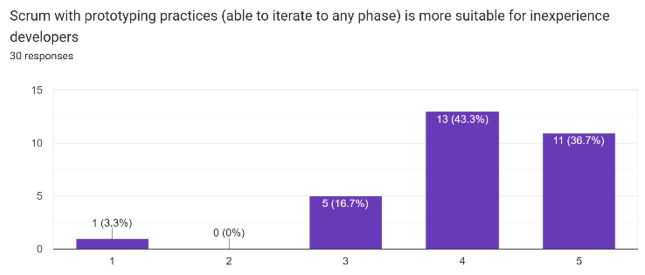

Fig. 6. Scrum with prototyping model more suitable to inexperience developers

The survey was also conducted to get their opinions if the combination of Scrum framework and the prototyping process model for certain software modules is helpful to them in their learning stage. Most participants agree or strongly agree that it is suitable for inexperienced developers like them, as shown in Fig. 6. It supports our third proposed enhancement on teamwork by practicing the prototyping model in the projects.

B. *Discussions and Limitations*

The study on the embracement amongst students in SE of our proposed solutions of self-selection of teams, pair programming, and the use of the prototyping model has produced positive results, in line with the survey results for students working in group projects of SE subjects. It shows that most students would prefer to choose their team members due to the unspoken understanding that students already have members in mind and are familiar with. It seems to prevent the additional step of getting to know new teammates and building the bonding. Educators conducting SE education can consider allowing students to select their own team members, if possible. It is seen that about 40% of students use pair programming practice in their software projects, so that closer team collaboration and fewer errors are brought into the projects. As students come with an array of experience, pair programming can serve as a mentoring session for inexperienced students to improve their technical skills. Pair Programming should be taught early in the PSD subject, as it is beneficial to teamwork and code quality in their real-world software projects. The survey results also indicate that the prototyping model is used by teams, so that all members can be involved and better understand the overall projects. Furthermore, it supports the basis that the prototyping model prevents teams from developing the final software products without a clear direction, resulting in an excessive workload for the team. Educators should encourage students to use the prototyping model besides the Agile framework, as they are new to SE practices.

There are a few limitations in the current study on teamwork enhancement for SE education. The findings may not translate to actual improvements in teamwork among students. The generalizability of the survey results is limited by participants from the CS students at SIT-UofG who are in the process or have completed the PSD and TP courses. Further research should consider how the proposed enhancements impact the teamwork of other programme besides CS, or other universities. The current results collected are based on teams which have already been allocated by the lecturers based on predefined pseudo random groupings. A more balanced study will be needed to survey participants with other types of group formations. Further studies should consider self-formed groups and groups allocated based on skillsets or personality types to analyze the impact on how it affects the team's performance.

V. CONCLUSION

The complexity of large-scale software development projects makes it challenging for many teams to reach their full potential. There are myriad studies made to research the ways to improve teamwork. The current course structures of PSD and TP encourage teamwork. To explore further enhancement on the teamwork of students in their corresponding groups, we examine prior works in literature to improve team cohesiveness. We propose three enhancements to teamwork in the TP projects. Shown from the survey results, about 83.3% of respondents agree or strongly agree that self-selected group formations can improve their teamwork. About 80% of respondents agree that teamwork can be improved if pair programming is adopted. About 80% of respondents agree that the prototyping model can help with improving their teamwork. It shows that the proposed enhancements could improve the teamwork, as students prefer to choosing whom they work with, and a pair programming practice is more forgiving for less experienced members.

References

[1] A. Chis, A. N. Moldovan, L. Murphy, *et al*., "Investigating Flipped Classroom and Problem-based Learning in a programming module for computing conversion course," *Educational Technology and Society*, vol. 21, pp. 232–247, 2018.

[2] M. Stevens and R. Norman, "Industry expectations of soft skills in IT graduates: A regional survey," *Australasian Computer Science Week Multiconference*, 2016, pp. 1–9.

[3] Mohammed and H. Abushama, "Popular agile approaches in software development: Review and analysis," *International Conference on Computing, Electrical and Electronic Engineering*, 2013, pp. 160–166.

[4] Y. C. Chan, C. M. Gan, C. Y. Lim, *et al*., "Learning CS subjects of professional software development and team projects," *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, 2022.

[5] Y. X. Chia, K. H. Loh, Z. Y. B. Ong, *et al*., "Sentiments analysis and feedback among three cohorts in learning software engineering modules," *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, 2022.

[6] B. Bruegge, S. Krusche, and L. Alperowitz, "Software engineering project courses with industrial clients," *ACM Transactions on Computing Education*, vol. 15, no. 4, pp. 1–31, 2015.

[7] M. Rizwan, J. Qureshi, S. Alshamat, and F. Sabir, "Significance of the teamwork in Agile software engineering," *Science International-Lahore*, vol. 26, no. 1, pp. 117–120, 2014.

[8] D. Oguz and K. Oguz, "Perspectives on the gap between the software industry and the software engineering education," *IEEE Access*, vol. 7, 2019.

[9] M. L. Pertegal-Felices, A. Fuster-Guillo, M. L. Rico-Soliveres, *et al*., "Practical method of improving the teamwork of engineering students using team contracts to minimize conflict situations," *IEEE Access*, vol. 7, 2019.

[10] J. Porras, A. Happonen, and J. Khakurel, "Experiences and lessons learned from onsite and remote teamwork based courses in software engineering," *Int'l Conference on Data and Software Engineering*, 2021.

[11] R. Vivian, K. Falkner, N. Falkner, and H. Tarmazdi, "A method to analyze computer science students' teamwork in online collaborative learning environments," *ACM Transactions on Computing Education*, vol. 16, no. 2, pp. 1–28, 2016.

[12] M. Almseidin, K. Alrfou, N. Alnidami, and A. Tarawneh, "A comparative study of Agile methods: XP versus Scrum," *Int'l Journal of Computer Science and Software Engineering*, vol. 4, no. 5, 2015.

[13] S. Al-Ratrout, "Practical implementation of Agile approaches in the teaching process," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 4, 2019.

[14] L. Pan, "Designing an extreme-based teaching model for the first programming course," *International Conference on Computational Science and Computational Intelligence*, 2021.

[15] T. Robal, "Fair and individualized project teamwork evaluation for an engineering course," *28th EAEEIE Annual Conference*, 2018.

[16] R. Lingard and E. Berry, "Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance," *32nd Annual Frontiers in Education*, 2002.

[17] M. Marques, S. F. Ochoa, M. C. Bastarrica, *et al*., "Enhancing the student learning experience in software engineering project courses," *IEEE Transactions on Education*, vol. 61, no. 1, pp. 63–73, 2018.

[18] C. Y. Chen, Y. C. Hong, and P. C. Chen, "Effects of the meetings-flow approach on quality teamwork in the training of software capstone projects," *IEEE Transactions on Education*, vol. 57, pp. 201–208, 2014.

[19] M. Paasivaara, C. Lassenius, D. Damian, *et al*., "Teaching students global software engineering skills using distributed Scrum," *35th International Conference on Software Engineering*, 2013.

[20] D. Tamayo Avila, W. Van Petegem, and A. Libottonc, "ASEST framework: a proposal for improving teamwork by making cohesive software engineering student teams," *European Journal of Engineering Education*, vol. 46, no. 5, pp. 750–764, 2020.

[21] D. T. Avila, W. Van Petegem, and M. Snoeck, "Improving teamwork in Agile software engineering education: the ASEST+ framework," *IEEE Transactions on Education*, pp. 1–12, 2021.

[22] M. Ahmad, J. Markkula, and M. Oivo, "Kanban for software engineering teaching in a software factory learning environment," *World Transactions on Engineering and Technology Education*, vol. 12, 2014.

[23] M. K. Yacoub, M. A. A. Mostafa, and A. B. Farid, "A new approach for distributed software engineering teams based on Kanban method for reducing dependency," *Journal of Software*, vol. 11, no. 12, pp. 1231–1241, 2016.

[24] C. Matthies, "Scrum2kanban: integrating Kanban and Scrum in a university software engineering capstone course," *Int'l Workshop on Software Engineering Education for Millennials*, 2018, pp. 48–55.

[25] A. Costa, F. Ramos, M. Perkusich, *et al*., "Team formation in software engineering: a systematic mapping study," *IEEE Access*, vol. 8, 2020.

[26] A. Mujkanovic and A. Bollin, "Improving Learning Outcomes through Systematic Group Reformation," *IEEE/ACM Cooperative and Human Aspects of Software Engineering*, 2016, pp. 97-103.

[27] H. H. Løvold, Y. Lindsjørn, and V. Stray, "Forming and assessing student teams in software engineering courses," In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*, 2020.

[28] D. Dzvonyar, L. Alperowitz, D. Henze, and B. Bruegge, "Team composition in software engineering project courses," *IEEE/ACM International Workshop on Software Engineering Education for Millennials*, 2018, pp. 16-23.

[29] C. Iacob and S. Faily, "Exploring the gap between the student expectations and the reality of teamwork in undergraduate software engineering group projects," *Journal of Systems and Software*, vol. 157, 2019.

[30] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for Agile methodology," *International Conference on Computing, Communication and Automation*, 2017.

[31] A. Mundra, S. Misra, and C. A. Dhawale, "Practical Scrum-Scrum team: Way to produce successful and quality software," *13th Int'l Conference on Computational Science and Its Applications*, 2013, pp. 119-123.

[32] M. Paasivaara, C. Lassenius, and V. T. Heikkilä, "Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work?," *ACM-IEEE Int'l Symposium on Empirical Software Engineering and Measurement*, 2012, pp. 235-238.

[33] K. J. Chapman, M. Meuter, D. Toy, and L. Wright, "Can't we pick our own groups? The influence of group selection method on group dynamics and outcomes," *Journal of Management Education*, vol. 30, no. 4, pp. 557–569, 2006.

[34] S. Faja, "Evaluating effectiveness of pair programming as a teaching tool in programming courses," *Information Systems Education Journal*, vol. 12, no. 6, 2014.

[35] I. D. Coman, P. N. Robillard, A. Sillitti, and G. Succi, "Cooperation, collaboration and pair-programming: Field studies on backup behavior," *Journal of Systems and Software*, vol. 91, pp. 124–134, 2014.

[36] B. Peterson and B. Vogel, "Prototyping the Internet of Things with web technologies: is it easy?," *IEEE Int'l Conference on Pervasive Computing and Communications Workshops*, 2018, pp. 518-522.

[37] S. G. Kwak and J. H. Kim, "Central limit theorem: the cornerstone of modern statistics," *Korean Journal of Anesthesiology*, vol. 70, no. 2, pp. 144–156, 2017.