From Ideation to Integration: Scaffolding Collaboration in a Large-Scale Computer Science Module

Nadine Aburumman
Nadine.Aburumman@brunel.ac.uk
Brunel University of London
London, United Kingdom

Bart de Keijzer
bart.dekeijzer@kcl.ac.uk
King's College London
London, United Kingdom

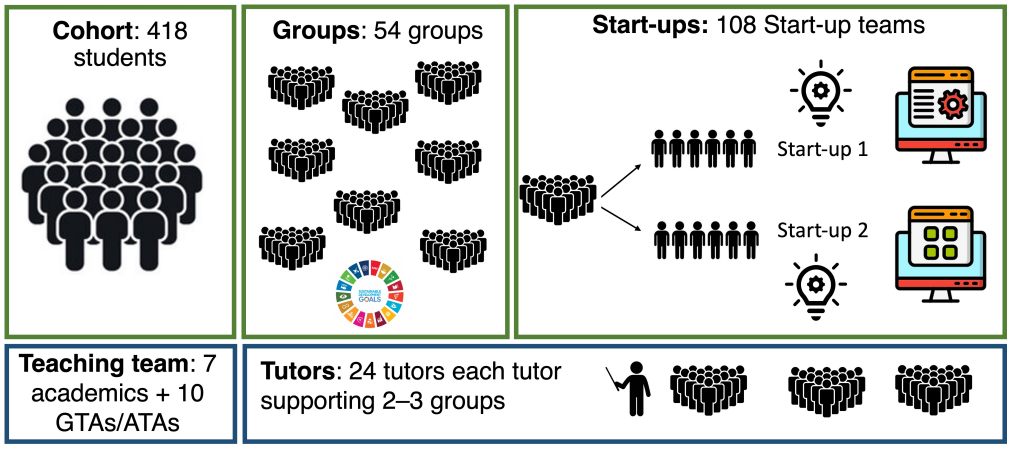

Figure 1: Two-tier structure for collaborative, project-based learning in a large cohort module. The module was supported by 7 academics, 10 GTAs/ATAs, and 24 tutors, with academics delivering lectures and laboratories and GTAs/ATAs supporting structured group activities.

Abstract

Group project modules provide valuable opportunities for students to develop collaborative, technical, and professional skills. However, they also present persistent challenges related to uneven participation, project coordination and consistent support. In this paper, we discuss the redesign and delivery of a year-long Computer Science group-project module for a large cohort of 418 students. To address these challenges, the redesigned module introduced a two-tier structure and scaffolding model that combined small start-up teams, staged formative deliverables, agile development practices and regular tutor checkpoints. We examine how this model supported collaboration, individual learning, and students' progression from project ideation to full-stack software integration. We present findings from the redesign, which indicate improved student engagement, better student outcomes, and substantially higher satisfaction than in previous years. We also reflect on the teaching strategies we used and the implementation challenges we encountered.

Koli Calling '26, Koli, Finland

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Applied computing** → **Collaborative learning**; **Interactive learning environments**.

Keywords

Collaborative Learning, Project-based Learning, Agile Methodology, Large Cohort

ACM Reference Format:
Nadine Aburumman and Bart de Keijzer. 2026. From Ideation to Integration: Scaffolding Collaboration in a Large-Scale Computer Science Module. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Koli Calling '26).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Group project modules provide an authentic setting in which undergraduate computing students can experience software development as a collaborative and professional practice. Mirroring professional software development in industry, students must negotiate requirements, make design decisions, coordinate technical tasks, and integrate their contributions into a single system. These activities support the development of communication, project management and interpersonal skills [1, 8, 17]. Group projects can also create opportunities for collaborative learning, in which students develop their understanding through discussion, peer feedback, and shared problem solving. This is consistent with related approaches such

as cooperative learning [11, 21, 22], team-based learning [14], agile software development practices [18, 25], and practicum-based learning [19]. Although these approaches differ in their organisation, they share an emphasis on interaction, shared activity, and responsibility for collective progress. However, placing students in groups does not necessarily lead to meaningful collaborative learning. Students can divide a project into isolated tasks, rely disproportionately on technically strong members, avoid difficult implementation work, or submit a shared artefact that obscures individual contributions [2, 4, 15]. Such patterns can limit opportunities for students to develop a shared understanding of the system or learn from one another's technical decisions.

These difficulties are amplified in large cohorts, where educators cannot observe interaction and where students may differ substantially in their prior experience, confidence, and engagement [7, 19]. Students can struggle to resolve conflict, coordinate interdependent work, and demonstrate what they have learned individually [15]. Therefore, educators must create conditions that support collaboration while ensuring that assessment recognises individual contribution and technical understanding [2]. Moreover, the widespread availability of generative artificial intelligence adds another dimension to the accountability challenge. A completed software artefact alone cannot demonstrate whether a student understands the work attributed to them or can explain the role of AI-assisted content within the development process. Hence, students must be able to demonstrate their participation, explain their technical contribution, and take responsibility for any AI-assisted work they submit [16]. Existing work has proposed mechanisms such as agile team processes, peer assessment, tutor feedback, and reflective reports [18]. However, these mechanisms are often considered separately. There is limited evidence showing how they can be integrated into a coherent design for a large cohort module or course. The challenge is not only to foster collaboration and progression throughout the development process, but also to identify individual contributions at the end of a project.

In this paper, we present a two-tier structure and scaffolding model for collaborative, project-based learning in a large 2nd year Computer Science (CS) group project module. In this case, the module enrolled 418 students, organised into 54 groups, with each group divided into two start-up teams, for a total of 108 teams. Each start-up team developed a full-stack web application addressing a problem related to the United Nations Sustainable Development Goals (SDGs), as illustrated in Figure 1. This model organised each large group into two small start-up teams [20], creating manageable units for project communication and accountability while retaining a shared problem context. The project development was scaffolded through staged formative deliverables, agile working practices, peer review, and regular tutor meetings. This paper addresses the following research questions:

RQ1: How can a large undergraduate group project module be structured to support collaborative learning while maintaining individual accountability?

RQ2: How do staged formative deliverables scaffold students' progression from project ideation to full-stack implementation?

2 Related Work

2.1 Collaborative Learning

Collaborative learning broadly describes situations in which two or more learners work together to develop knowledge or solve a problem [6]. Its educational value comes from the interactions through which learners explain ideas, negotiate decisions, challenge assumptions, and construct a shared understanding. Collaborative learning overlaps with cooperative learning, although the two are not identical. Cooperative learning emphasises deliberately structured interdependence and individual responsibility.

On the other hand, cooperative learning is linked to social interdependence theory. Social interdependence theory identifies the following conditions for effective cooperation: 1) positive interdependence, 2) individual accountability, 3) promotive interaction, 4) social skills, and 5) group processing [11]. These conditions have been associated with improved learning outcomes in science, technology, engineering, and mathematics education [21, 22]. They also provide a useful conceptual lens for examining whether students are learning with and from one another rather than merely dividing a shared task. Collaborative learning can be implemented through several pedagogical approaches. Team-based learning uses stable teams and structured activities to encourage preparation and collective decision making [14]. Problem-based learning organises learning around an open-ended problem, while project-based learning extends this process through the sustained development of an artefact or product [9, 10].

Software projects provide a natural context for combining these approaches. A problem can establish the purpose and context of the work, while the development of a functioning software system provides the project through which technical and professional learning is organised. Agile software practices can further structure this process by introducing shared backlogs, iterative development, regular review, and reflection [18, 25]. However, these practices support learning only when students engage with one another's reasoning and technical work. Dividing a system into isolated components can enable a group to produce an artefact without ensuring that its members develop a shared understanding of the complete system. Furthermore, in software engineering education, positive interdependence is established when students understand that their progress is connected to the team's progress [11]. Shared integration milestones and collective demonstrations can reinforce this connection by making students' contributions mutually dependent. However, these structures can also introduce challenges. In group projects, technically confident students can undertake the most complex implementation work, while other students become dependent on their contributions, making it difficult to evidence individual learning and accountability [4, 12, 15].

Software projects also require students to manage several forms of complexity at the same time. They must understand the problem, select an appropriate solution, organise team responsibilities, apply technical knowledge, and integrate multiple software components. For less experienced students, asking them to manage all of these demands without structured support can result in delayed engagement, fragmented development, and can lead to integration being postponed until the end of the project [23].

2.2 Scaffolding & Project-based Learning

Scaffolding is responsive support that enables learners to undertake activities they may not yet be able to complete independently. A review by van de Pol et al. identifies three important characteristics of scaffolding: 1) Contingency, in which support responds to learners' current needs, 2) fading, in which support is gradually reduced, and 3) transfer of responsibility, in which learners assume control over the task [24]. From this perspective, scaffolding is not simply the provision of additional teaching material. It should structure early participation while progressively enabling students to plan, monitor, and evaluate their own work.

In project-based learning, scaffolding can be provided through staged milestones, formative feedback, and opportunities for reflection [5, 13]. Formative deliverables can break down a complex project into manageable stages and make students' progress visible before the final assessment. Early activities can focus on understanding the problem, identifying skills gaps, and planning the project. Later activities can shift responsibility towards sprint review, technical implementation and integration. This staged progression can also help educators identify difficulties while there is still time to intervene.

Regular feedback is essential in large software projects because participation and coordination problems may otherwise remain hidden until final submission [26]. Effective group-project teaching therefore requires deliberate organisation and sustained support throughout the course [3, 4]. Agile practices can provide this structure when used as pedagogical tools rather than simply introduced as professional terminology. Sprint planning makes tasks and ownership visible, stand-up meetings surface blockers and dependencies, sprint reviews provide evidence of progress, and retrospectives support reflection on communication, workload, and team processes [2, 18]. Scaling this support is a complex operation. In a large cohort, students can receive varying levels or forms of guidance from different tutors, and educators cannot directly observe all team interactions. A scalable design therefore requires common milestones, shared expectations, structured tutor roles, and evidence that can be reviewed across groups.

3 Methods

This study employed an evaluative case study design to examine the redesign and delivery of a year-long group project module. The 2025–26 delivery was treated as the primary case and was compared with the three preceding academic years to examine changes in submission, student outcomes, and satisfaction. The study addressed the research questions through two complementary forms of analysis. First, the module structure, formative activities, tutor support, and assessment mechanisms were examined to identify how the design supported collaborative learning and individual accountability. Second, institutional records and project evidence were analysed to examine students' progression, participation, and outcomes following the redesign.

3.1 Module Context and Participants

This study was conducted in a 45-credit 2nd year undergraduate group project module within a Computer Science programme. The module has 7 learning outcomes (see Figure 2) and was delivered

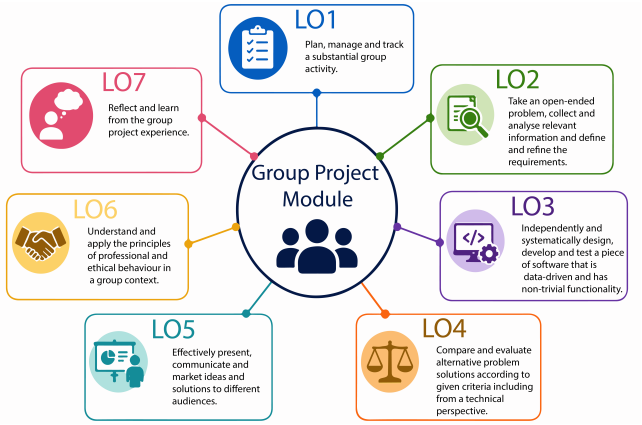

Figure 2: A diagram illustrating the 7 learning outcomes of the group project module examined in this case study.

across two teaching terms and represented 37.5% of the students' academic-year credit load. The cohort comprised 418 students organised into 54 groups of between 8 and 10 students. Each group was divided into two small start-up teams of three to five students, resulting in 108 teams. The start-up structure was informed by the principle that small autonomous teams can support communication, accountability, and agile collaboration [20]. Student groups were formed by the module team to create a mixture of backgrounds, programme pathways, and levels of prior experience, consistent with guidance on group composition and peer learning [11, 15]. Students with documented patterns of prior disengagement were allocated to designated groups so that additional support could be targeted. The module included an inclusivity statement that encouraged students to value diverse backgrounds and perspectives, collaborate respectfully, and ensure that all members had opportunities to contribute. Each group was assigned a problem statement linked to one of the United Nations Sustainable Development Goals. The two start-up teams within each group addressed the same problem but developed distinct solutions and separate software. The module was supported by 7 academics, 10 teaching assistants, and 24 group tutors. Academics led lectures and labs, while teaching assistants supported practical and group activities and acted as mentors.

3.2 Redesigned Delivery Model

The delivery model combined scheduled teaching, supervised technical practice, structured group activity, and regular formative support, as shown in Figure 3. Each week, students attended a 1-hour on-campus lecture, a 2-hour lab practical session, and a 1-hour structured group meeting. A weekly 1-hour online revision and Q&A session. This online session revisited key concepts, clarified assessment expectations, and addressed technical questions. Groups were provided with a structured agenda for their weekly meetings and were required to record decisions, assigned actions, and task ownership. Meetings incorporated agile practices, such as stand-ups, sprint planning, sprint reviews, retrospectives, and peer review. Teaching assistants supported these structured meetings, where

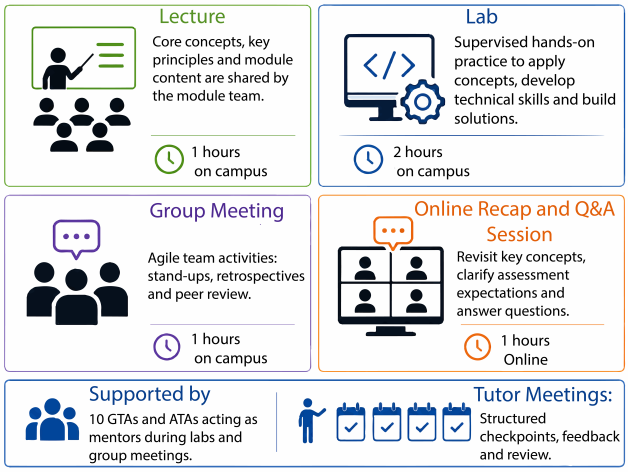

Figure 3: Weekly learning and support structure, supplemented by four structured on-campus tutor meetings for each group per term.

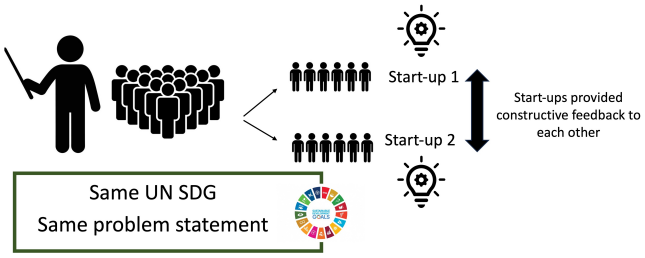

Figure 4: Tutor meetings supporting peer review and formative feedback between start-up teams.

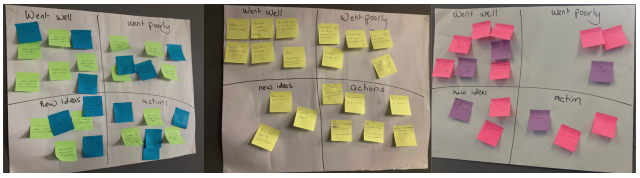

Figure 5: Examples of sprint retrospective outputs documenting team reflections and agreed actions.

activities within these meetings were used to make progress, responsibilities, and dependencies visible throughout the project. Students also had access to private group lockers and discussion areas within the Brightspace Virtual Learning Environment. These spaces were used to store meeting minutes, action records, and project discussions. Tutors had access to the spaces of their assigned groups, enabling them to monitor progress between formal checkpoints. Tutor meetings included peer review between the two start-up teams working on the same problem. During these tutor meetings, teams presented their progress to their tutor, compared alternative solutions, and provided constructive feedback on one another's work. Tutors facilitated the discussion and prompted students to explain and justify their design and implementation decisions (see Figure 4).

In this case study, we redesign the module to include 4 formative deliverables and 2 summative assessment blocks. The formative deliverables consist of 1) an Individual Professional Development Plan, 2) a Group Project Management Plan, 3) a Group Project Progress Review, and 4) an Individual Demonstration Video. These staged activities were designed to scaffold students' progression from early self-reflection and project planning to agile progress monitoring and individual technical demonstration, and are summarised in Table 1.

The module redesign had two incremental summative assessment blocks across the academic year. The first was at the end of Term 1 and focused on project proposal development, requirements analysis, design thinking, front-end development, and an individual video demo. The second was at the end of Term 2, focused on back-end development, database design, REST API implementation, system integration, a technical group demonstration, and a reflective report. Students were also required to declare and reflect on any use of their generative AI partner, including the prompts used, and to take responsibility for AI-assisted work as part of Learning Outcome 6 on responsible and ethical professional practice [16].

3.3 Assessment Design and Formative Scaffolding

Agile practices were embedded as structured pedagogical mechanisms within the deliverables. Formative Deliverable 2 operationalised agile project planning. Each start-up team produced a Lean Canvas, a backlog containing epics and user stories with identifiers and acceptance criteria, and a Sprint 1 plan. The sprint plan specified the sprint goal, agile roles, task ownership, and story point estimates. Formative Deliverable 3 extended this process through a formal Sprint 1 Review and Retrospective followed by Sprint 2 planning. start-up team evaluated completed backlog items against their acceptance criteria, reflected on their working practices, and incorporated feedback from the tutor meeting into the next sprint. Sprint 2 shifted the technical focus towards back-end implementation. This staged progression reduced the initial technical demands and supported incremental development from front-end prototyping to full-stack implementation and integration. Examples of retrospective outputs are shown in Figure 5. After Deliverables 2 and 3, the two start-up teams within a group reviewed one another's work and provided written feedback. This cross- startup team review encouraged students to compare distinct solutions, explain their technical work, and use peer feedback to improve subsequent work. At the end of each term, each start-up team demonstrated its work during a tutor meeting. Every student also answered individual interview-style questions about their contribution and technical understanding. The demonstration and questioning component accounted for between 15% and 20% of each summative assessment block.

3.4 Data Sources and Analysis

To evaluate the redesigned module, the analysis was based on five sources of evidence below.

Table 1: Formative deliverables, their purpose within the module, and their alignment with social interdependence principles.

Formative deliverable	Key requirements	Purpose within the module	Social interdependence principles
Deliverable 1: Individual Professional Development Plan	Skills-gap analysis; technical and professional development plan; teamwork expectations.	Identifies individual development needs and establishes responsibility at the start of the project.	Individual accountability; social skills; group processing.
Deliverable 2: Group Project Management Plan	Ideation output; Lean Canvas; project backlog; Sprint 1 plan; team roles and responsibilities; peer review between start-up teams.	Supports shared planning, role allocation, project ownership, and early peer feedback.	Positive interdependence; group processing; individual accountability; social skills.
Deliverable 3: Group Project Progress Review	Sprint 1 review and retrospective; Sprint 2 plan; reflection on communication, tools, workload, and teamwork; Definition of Done.	Makes progress and blockers visible and enables teams to review and adjust their working practices.	Group processing; promotive interaction; individual accountability.
Deliverable 4: Individual Demonstration Video	Back-end implementation; explanation of individual technical contribution; peer evaluation of professional and ethical behaviour.	Provides evidence of individual contribution, technical understanding, and responsibility for the submitted work.	Individual accountability; promotive interaction.

(1) **Assessment submission records**, used to calculate submission rates for the redesigned delivery and the three preceding academic years;
(2) **Grade distributions**, used to compare outcomes across grade bands before and after the redesign;
(3) **GitHub Classroom records**, used to examine evidence of individual participation and contribution across the project;
(4) **Final project artefacts and assessment records**, used to examine progression to full-stack implementation and software integration; and
(5) **Anonymous institutional student surveys**, used to compare overall student satisfaction with previous module deliveries.

All data were anonymised and collected as part of routine institutional module records. The student satisfaction survey was completed anonymously as part of the institution's standard quality assurance process. The data were analysed descriptively. Submission rates were calculated as the proportion of enrolled students submitting each assessment point on time. Grade outcomes were categorised using the institutional grade bands A, B, C, D, and E–F, and the proportion achieving the two highest bands was compared across delivery cycles. Student satisfaction was examined using the percentage of respondents providing a positive rating. GitHub Classroom records were examined as supporting evidence of individual participation rather than as a direct measure of contribution quality. Branch activity, commit frequency, pull requests, and merge history were considered alongside project artefacts, reflective reports, and group demonstrations as part of the interview simulation. This triangulation reduced reliance on any single indicator.

4 Findings

4.1 Participation and Academic Outcomes

Following the redesign, the overall assessment submission rate was 96%, compared with an average of 92% across the three preceding academic years. This suggests that a larger proportion of the cohort remained engaged through to final submission. However, as the comparison involved different student cohorts, the increase should not be interpreted as a causal effect of the redesign alone. Student outcomes also improved. The proportion of students achieving grades in the two highest grade bands increased from 57% in the previous delivery to 74% in 2025–26. Table 2 presents the grade distribution for the two summative assessment blocks. The distributions indicate that most submitted work was awarded an A or B grade in both assessment blocks. Assessment 2 had fewer A-grade outcomes than Assessment 1, but a large proportion of B grades. This reflects the technical demands of the second assessment, which required back-end development, database design, REST API implementation, and software integration.

Table 2: Grade distributions for the two summative assessment blocks. Values show the number and percentage of submitted assessments.

Grade band	Assessment 1	Assessment 2
A	190/409 (46%)	147/404 (36%)
B	118/409 (29%)	149/404 (37%)
C	63/409 (15%)	71/404 (18%)
D	10/409 (3%)	16/404 (4%)
E–F	28/409 (7%)	21/404 (5%)

Table 3 summarises submission patterns across the formative deliverables and summative assessments. Participation was high throughout the module, although submission rates varied across assessment points. Deliverables 2 and 3, the Group Project Management Plan and Group Project Progress Review, achieved 100% on-time submission across all 108 start-up teams. This indicates sustained engagement with project planning and progress monitoring. Deliverable 1 and Summative Assessment 1 also recorded high on-time submission rates of 95.2% and 94.5%, respectively. The lowest on-time submission rate was recorded for Deliverable 4, the Individual Demonstration Video, at 90.9%, with 22 students not submitting. This was the only formative deliverable requiring an

individually recorded technical demonstration. Its lower submission rate may therefore reflect the additional demands of preparing, recording, and explaining individual technical work. At every other assessment point, the non-submission rate remained below 4%.

Overall, these patterns show that participation was sustained across a large and technically demanding module. The 100% submission of the two team-based deliverables also suggests strong engagement with the staged planning and progress review activities.

4.2 Progression from Ideation to Integration

The staged formative deliverables provided visible checkpoints across the project lifecycle. Deliverables 2 and 3 achieved 100% submission across all 108 start-up teams, indicating sustained engagement with project planning, sprint review, and progress monitoring. These activities supported progression from initial ideation and backlog development to technical implementation and integration. Compared with the three preceding academic years, the 2025–26 results showed higher participation and attainment. The submission rate increased to 96%, compared with 91% in 2024–25, 93% in 2023–24, and 92% in 2022–23. The proportion of A grades increased to 41%, from 32%, 25%, and 30% in the preceding three years. B grades increased to 33%, compared with 28%, 25%, and 29%. Lower-band outcomes also decreased, with D grades falling to 3% and E–F grades to 6%. Overall, the results indicate an upward shift in attainment and a reduction in lower-band outcomes following the redesign.

Project artefacts provided further evidence of technical progression. Of the 108 start-up teams, 88 successfully integrated their full-stack web applications and pushed a complete version to the main branch. These teams progressed from separately developed front-end and back-end components to a functional, integrated software product. The remaining teams completed parts of the required system but did not demonstrate full integration by the final assessment point. In the previous academic year, less than 1/3 of teams successfully integrated their systems, indicating a substantial improvement in full-stack integration following the redesign.

4.3 Collaboration and Individual Accountability

The two-tier structure created small units of responsibility within the larger groups. Each start-up team developed a distinct solution to the shared problem. Tutor meetings brought the two start-up teams together to review progress, explain technical decisions, and discuss alternative approaches. Individual accountability was supported through multiple forms of evidence. GitHub activities and group demo with interview Q&A. GitHub provided a practical mechanism for examining contribution patterns, including branch activity, commit frequency, merge activity, and conflict resolution. GitHub Classroom records made the timing and continuity of development activity visible, while meeting records documented task allocation and project decisions. GitHub activity was not treated as a direct measure of contribution quality. A high number of commits could represent minor changes, while important design, testing, mentoring, or integration work might result in fewer commits. Repository evidence was therefore interpreted alongside the student's demonstration, reflective account, meeting records, and responses to technical questions.

The contribution data showed that 93% of students who achieved an A-band grade demonstrated sustained activity on both their individual GitHub branch and their start-up team's main branch, with commits distributed throughout the academic year. These students also demonstrated high engagement in software integration, including resolving merge conflicts and completing successful merges. The group demo with interview Q&A also reinforced accountability by requiring students to explain and justify their individual technical contributions. Students who demonstrated a clear understanding of their work during the interview achieved higher marks in this component, particularly those within the A and upper-B grade bands.

Moreover, Table 4 presents the institutional module teaching survey results for 2025–26 and 2024–25 across both semesters. The results show a clear improvement in student satisfaction during 2025–26. Overall satisfaction increased from 61% to 78% in Semester 1 and from 50% to 77% in Semester 2.

5 Discussion

Our findings show that collaboration and individual accountability were supported through the interaction of the two-tier structure, regular tutor checkpoints, agile practices, and multiple forms of contribution evidence. The small start-up teams gave students clear ownership of a software solution, while the shared problem context, peer review, and structured meetings created opportunities to explain decisions, compare approaches, and learn from one another. However, smaller teams alone did not guarantee collaborative learning. Differences in personalities, preferences, neurodiversity, wellbeing, or engagement level could affect how students participated in group work and, in some cases, limit collaborative learning. We therefore combined shared responsibility for producing a functioning system with individual demonstrations of contribution and technical understanding. This design reflects social interdependence theory by supporting positive interdependence while ensuring individual accountability [11]. Therefore, future iterations will include additional team-building activities to help students understand different personalities and working styles, build psychological safety, and communicate any support needs or barriers to participation they are comfortable sharing. This can help students develop adaptability and coping strategies that are valuable in industry. Our findings also show that no single source of evidence was sufficient to represent individual contribution. GitHub records made development activity visible across the large cohort, but they could not fully capture the quality or complexity of students' work. Moreover, triangulating multiple sources of evidence increases the workload for markers/assessors, who must review repository activity, meeting records, reflective accounts, demonstrations, and technical questioning. In future iterations, we plan to introduce an evidence-mapping system in which students link each claimed contribution to the relevant code, documentation, meeting records, and supporting context. This would make individual contributions easier to verify while reducing the assessment burden. Our redesign indicates that large-scale collaborative projects require a shared

Table 3: Submission patterns across formative deliverables and summative assessments.

Assessment point	Submission unit	On time	Late	Missed
Deliverable 1: Individual Professional Development Plan	Individual student	398 (95.2%)	7 (1.7%)	13 (3.1%)
Deliverable 2: Group Project Management Plan	Start-up team	108 (100%)	0	0
Deliverable 3: Group Project Progress Review	Start-up team	108 (100%)	0	0
Deliverable 4: Individual Demonstration Video	Individual student	380 (90.9%)	16 (3.8%)	22 (5.3%)
Summative Assessment 1: individual report, group demonstration, and individual Q&A	Individual student	395 (94.5%)	14 (3.3%)	9 (2.2%)
Summative Assessment 2: individual report, group demonstration, and individual Q&A	Individual student	385 (92.1%)	19 (4.5%)	14 (3.3%)

Table 4: Student satisfaction survey results for 2025–26 and 2024–25.

Response category	2025–26 Sem. 1	2025–26 Sem. 2	2024–25 Sem. 1	2024–25 Sem. 2
Very satisfied	24.10%	43%	15%	20%
Somewhat satisfied	54.22%	33%	45%	30%
Somewhat dissatisfied	8.43%	13%	18%	15%
Very dissatisfied	8.43%	7%	15%	30%
No response	4.82%	3%	6%	5%
Overall satisfaction	78%	77%	61%	50%

support architecture. However, scalability introduced new challenges. Tutors could interpret expectations differently, and students received varying forms of feedback. Future iterations should therefore include more tutor calibration, common checkpoint templates, and clear guidance on contribution evidence.

Acknowledgments

Bart de Keijzer was supported by EPSRC grant EP/X021696/1.

References

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* Association for Computing Machinery and IEEE Computer Society, New York, NY, USA. doi:10.1145/2534860

[2] Kevin Buffardi. 2020. Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education.* Association for Computing Machinery, New York, NY, USA, 650–656. doi:10.1145/3328778.3366948

[3] Nicholas W. Bussberg and Laura L. Taylor. 2026. Student Perceptions of Group Work and Group Formation Strategies. *Journal of Statistics and Data Science Education* 34, 2 (2026), 186–197. arXiv:https://doi.org/10.1080/26939169.2025.2507765 doi:10.1080/26939169.2025.2507765

[4] Yunjeong Chang and Peggy Brickman. 2018. When Group Work Doesn't Work: Insights from Students. *CBE—Life Sciences Education* 17, 3 (2018), ar42:1–ar42:17. doi:10.1187/cbe.17-09-0199

[5] Yu-Hui Ching and Yu-Chang Hsu. 2013. Peer Feedback to Facilitate Project-Based Learning in an Online Environment. *The International Review of Research in Open and Distance Learning* 14, 5 (2013), 258–276. doi:10.19173/irrodl.v14i5.1524

[6] Pierre Dillenbourg. 1999. What Do You Mean by Collaborative Learning? In *Collaborative Learning: Cognitive and Computational Approaches*, Pierre Dillenbourg (Ed.). Elsevier, Oxford, 1–19.

[7] Torgeir Dingsøyr. 2022. Educating Reflective Systems Developers at Scale: Towards "Productive Feedback" in a Semi-Capstone Large-Scale Software Engineering Course. In *2022 IEEE Frontiers in Education Conference.* IEEE, 1–8. doi:10.1109/FIE56618.2022.9962726

[8] Peter Freeman, Anthony I. Wasserman, and Richard E. Fairley. 1976. Essential Elements of Software Engineering Education. In *Proceedings of the 2nd International Conference on Software Engineering.* IEEE Computer Society Press, 116–122. doi:10.5555/800253.807660

[9] Pengyue Guo, Nadira Saab, Lysanne S. Post, and Wilfried Admiraal. 2020. A Review of Project-Based Learning in Higher Education: Student Outcomes and Measures. *International Journal of Educational Research* 102 (2020), 101586. doi:10.1016/j.ijer.2020.101586

[10] Laura Helle, Päivi Tynjälä, and Erkki Olkinuora. 2006. Project-Based Learning in Post-Secondary Education: Theory, Practice and Rubber Sling Shots. *Higher Education* 51, 2 (2006), 287–314. doi:10.1007/s10734-004-6386-5

[11] 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. doi:10.3102/0013189X09339057

[12] Antti Knutas, Jouni Ikonen, and Jari Porras. 2015. Computer-Supported Collaborative Learning in Software Engineering Education: A Systematic Mapping Study. *International Journal on Information Technologies & Security* 7, 4 (2015).

[13] Susan M. Land and Carla Zembal-Saul. 2003. Scaffolding Reflection and Articulation of Scientific Explanations in a Data-Rich, Project-Based Learning Environment: An Investigation of Progress Portfolio. *Educational Technology Research and Development* 51, 4 (2003), 65–84. doi:10.1007/BF02504544

[14] Larry K. Michaelsen, Neil Davidson, and Claire Howell Major. 2014. Team-Based Learning Practices and Principles in Comparison with Cooperative Learning and Problem-Based Learning. *Journal on Excellence in College Teaching* 25, 3–4 (2014), 57–84.

[15] Barbara Oakley, Richard M. Felder, Rebecca Brent, and Imad Elhajj. 2004. Turning Student Groups into Effective Teams. *Journal of Student Centered Learning* 2, 1 (2004), 9–34.

[16] Mike Perkins, Leon Furze, Jasper Roe, and Jason MacVaugh. 2024. The Artificial Intelligence Assessment Scale (AIAS): A Framework for Ethical Integration of Generative AI in Educational Assessment. *Journal of University Teaching and Learning Practice* 21, 6 (2024). doi:10.53761/q3azde36

[17] Michael Prince. 2004. Does Active Learning Work? A Review of the Research. *Journal of Engineering Education* 93, 3 (2004), 223–231. doi:10.1002/j.2168-9830.2004.tb00809.x

[18] Ken Schwaber and Jeff Sutherland. 2020. The Scrum Guide: The Definitive Guide to Scrum. Scrum Guides. Accessed 15 July 2026. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[19] Martin Shepperd. 2011. Group Project Work from the Outset: An In-Depth Teaching Experience Report. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training.* IEEE, 361–370. doi:10.1109/CSEET.2011.5876107

[20] Daniel Slater. 2023. Powering Innovation and Speed with Amazon's Two-Pizza Teams. AWS Executive Insights. Accessed 15 July 2026. https://aws.amazon.com/executive-insights/content/amazon-two-pizza-team/

[21] Robert E. Slavin. 1991. Synthesis of Research on Cooperative Learning. *Educational Leadership* 48, 5 (1991), 71–82.

[22] Leonard Springer, Mary Elizabeth Stanne, and Samuel S. Donovan. 1999. Effects of Small-Group Learning on Undergraduates in Science, Mathematics, Engineering, and Technology: A Meta-Analysis. *Review of Educational Research* 69, 1 (1999), 21–51. doi:10.3102/00346543069001021

[23] Saara Tenhunen, Tomi Männistö, Matti Luukkainen, and Petri Ihantola. 2023. A Systematic Literature Review of Capstone Courses in Software Engineering. *Information and Software Technology* 159 (2023), 107191. doi:10.1016/j.infsof.2023.107191

[24] Janneke van de Pol, Monique Volman, and Jos Beishuizen. 2010. Scaffolding in Teacher–Student Interaction: A Decade of Research. *Educational Psychology*

Review 22, 3 (2010), 271–297. doi:10.1007/s10648-010-9127-6

[25] Laurie Williams and Richard L. Upchurch. 2001. Extreme Programming for Software Engineering Education?. In *Proceedings of the 31st Annual Frontiers in Education Conference*. IEEE, T2D–12–T2D–17. doi:10.1109/FIE.2001.963882

[26] David C Wynn and Anja M Maier. 2022. Feedback systems in the design and development process. *Research in Engineering Design* 33, 3 (2022), 273–306.