

Excited

Centre for Excellent IT Education





Lessons learned from ten years of innovation in computing education

Edited by Birgit Krogstie and Robin Isfold Munkvold

Copyright

Excited - Lessons learned from ten years of innovation in computing education © 2026 by Birgit Krogstie and Robin Munkvold (Eds.) is licensed under CC BY-NC-ND 4.0



Preface

The idea to share insights from Excited Centre of excellent IT education in a format useful for educators came up in the planning of our 2026 Anniversary Conference. The Excited team decided to write a set of book chapters. Rather than aiming for a full coverage of the Centre's activities, we agreed to focus on a representative selection of topics, somewhat biased toward recent activity in the Centre and the available workforce represented by the current Centre team.

It became clear early on that most, if not all, of the contributions had some connection to project based learning, which we decided to reflect in the title of the book. Given the limited time, it was decided that the book was to be self-published on the Centre website, with a premiere in printed format at the Anniversary Conference.

The editors wish to thank the authors as well as all those involved in the work on which the chapters are based, including researchers, educators, students, and administrative staff at our home institutions. Our heartfelt thanks also go to our funding bodies (Norwegian Directorate of Higher Education and Skills, Nord University, NTNU).

We are happy to share results from the Centre's work in this book, hoping that it will trigger new endeavors in innovative IT education!

Introduction

The Centre for Excellence in IT Education (Excited) was established as a collaboration between the *Norwegian University of Science and Technology (NTNU)* and *Nord University*. As a Centre for Excellence in Education (SFU), Excited has operated over a ten-year period from 2016 to 2026, bringing together educators, researchers, students, and industry partners with the shared ambition of strengthening higher IT education in Norway.

The Centre's vision is to bring Norway to the forefront of innovative IT education and provide IT education for a digital and sustainable future. To this end, Excited has developed arenas for experience exchange, knowledge building and culture building among educators locally and nationally, promoting a culture for innovative IT education grounded in research. Through active collaboration with IT students the Centre has made a positive and lasting contribution to the quality of student life. Several international research contributions have resulted from the Centre's work, mainly in the fields of curriculum development, pedagogy, learning technology, and sustainability in computing education. A large part of this research builds on interventions and improvements in courses and study programs at Nord University and NTNU.

The book shares experiences, research, and insights generated during a ten-year period of educational development and research. While the chapters reflect a diversity of topics and perspectives, they are united by a shared interest in project-based learning and related educational approaches within higher education. These are the chapters:

Chapter 1, Learning Theories and their Use in Computing Education Research, introduces learning theories and demonstrates their relevance for educators and researchers in computing education. The chapter is organized around three broad orientations to learning: behaviorist, cognitive, and social. Key theories within each of these orientations are presented, with examples of how they can inform the design and evaluation of teaching practices in computing education. The chapter concludes with a set of recommendations to educators in terms of selecting a learning theory to support their teaching and research.

Chapter 2, Designing Projects for Learning, addresses Project-based learning (PjBL), which is widely used in higher computing education because it enables students to work on authentic problems, collaborate with others, and develop competencies relevant for professional practice. The chapter examines how teachers can design projects that support both project outcomes and student learning. Four different frameworks for PjBL are presented, along with experience-based insights from higher computing education related to blended project-based learning and the use of external mentors.

Maintaining the focus on student projects, **Chapter 3, Dynamics of Groupwork**, turns to the topic of teamwork competence, which is critical in PjBL. The chapter shows how game-based learning can be used to promote students' understanding of conflict management by using the Cards of Conflict (CoC) game. Two instructional designs for implementing the game in higher education are presented. The chapter offers practical guidance for educators seeking to strengthen students' teamwork competence and conflict-management skills.

The major sustainability challenges in society typically involve IT. Computing education thus needs to provide the competencies needed for IT professionals to work with sustainability. **Chapter 4, Sustainability in Computing Education**, discusses how this can be done in practice. After introducing relevant sustainability models and principles along with sustainability competencies, the chapter elaborates on pedagogy for teaching about sustainability. Examples from computing education teaching

and research are provided, illustrating for instance the use of a framework for sustainability analysis which can easily be integrated into project-based design and software development courses.

In **Chapter 5, The Educational Production Gap**, we turn to student projects in game development. A challenge in these projects is the distance between what the teams are asked to deliver, a finished and playable game, and the production structure they are given to deliver it. The chapter introduces the Agile Learning Team Framework (ALTF) to address this challenge. Grounded in scaffolding and self-determination theory, the framework supports a higher degree of supervision at the start of the projects, fading as the teams mature. The chapter also discusses agentic AI in student teams, with practical guides for educators.

To succeed in their jobs, IT professionals need non-technical (generic) competencies. How to provide such competencies is the topic of **Chapter 6, Educating for Generic Competences**. Constructive alignment is an important principle in curriculum design, but research suggests that generic competences are not well defined in intended learning outcomes on the study programme level. The chapter provides a foundation for good alignment between programme and course level regarding generic competences, presenting a core set of generic competences and discussing how generic competences can be progressively developed.

The potential of having students give feedback on each other's work is addressed in **Chapter 7, Peer Assessment in Project-Based IT Education**. Peer assessment trains critical thinking and evaluative judgement, builds students' capacity to give and receive critique, and can make it possible to provide formative feedback on a large scale. The chapter gives a practical overview of peer assessment in project-based IT education. Three use cases from courses at NTNU illustrate how the purpose, design and experience of peer assessment vary and how students' use of generative AI is beginning to change the conditions for the activity.

Chapter 8, Self-Paced Mastery Learning presents Do It Yourself (DIY) advice for readers who want to know more about self-paced mastery learning, possibly to try it out in their own courses. After presenting experiences with mastery learning in an introductory programming course at NTNU, the chapter explains the background and principles of self-paced mastery learning and summarizes some key results in terms of student performance and satisfaction. The chapter also outlines key design decisions and trade-offs to be made when developing similar courses and gives some advice on the course design process, highlighting the benefits of student involvement.

Chapter 9, Gamification in Higher Education - From Research to Educational Practice, asserts that gamification should be understood as a spectrum of design choices, not as a question of points, badges and leaderboards. Drawing on self-determination theory and empirical work connected to Excited, the chapter compares three computing education cases. The cases show that the same motivational function in a game can take very different forms. Also, it matters more what a mechanic *does* than whether it looks like a game. The chapter offers eight design principles for gamification in higher education.

In **Chapter 10, The Learning Environment,** we turn to the important topic of students' learning environment. The chapter presents interventions and research efforts conducted in Excited to understand and improve the students' learning environment, covering social aspects, physical aspects (e.g., learning areas), and the academic learning environment. A special focus in the research has been student identity and belonging, often in the context of teamwork and project-based learning. Lessons learned from the interventions are provided, along with advice for others who want to try out something similar.

Collectively, the chapters explore how teaching in computing education can be designed, facilitated, assessed, and studied. They examine the roles of teamwork, reflection, industry engagement, feedback, learning environments, and digital technologies. A unifying theme is the focus on the development of generic competences that are increasingly recognized as essential for IT professionals, including communication, collaboration, critical thinking, conflict management, professional responsibility, and sustainability competencies. The chapters together have a strong focus on the role of authentic, development-oriented projects as arenas for learning, whether students are developing software applications, digital solutions, games, or other technological artefacts. The work presented in this book thus illustrates how such projects provide great opportunities to integrate disciplinary knowledge with professional competence.

Use of generative AI in the writing process

Where nothing else has been stated in a chapter, the use of generative AI tools has been limited to literature search, simple editing and proofreading.

Keywords:

Project-Based Learning (PjBL); IT Education; Generic Competences; Teamwork Competence; Programme-Course Alignment; Peer Assessment; Conflict Management; Game-Based Learning; Professional Skills; Collaboration; Reflection; Industry Engagement; Student Feedback; Educational Design; Higher Education; Experiential Learning; Authentic Learning, Educational Innovation; Professional Development; Learning Outcomes Alignment.

About the authors



Bashira Jaradat is a PhD candidate at the Norwegian University of Science and Technology (NTNU) in Norway, where she is a member of the Applied Informatics group the Department of Computer Science. Her research area is the integration of sustainability in computing education, with a particular focus on project-based user-centered design courses and students' acquisition of sustainability competencies. Through her research and teaching at NTNU she has successfully applied the SusAF sustainability analysis framework and adapted its use to various settings.



Rune Hjelsvold is a Professor in the Department of Computer Science at NTNU. He has a background in Computer Science (PhD). In recent years, his research has been on higher computing education. Hjelsvold is an Excellent Teaching Practitioner at NTNU. In Excited, Rune was the project leader of the P4, Sharing and Diversity, project in phase 1 and the C1, Learning Outcomes and Structure, cluster in phase 2.



Guttorm Sindre holds MSc and PhD degrees in Computer Science (1987, 1990). He has been a full professor at NTNU since 2003. He was Head and Deputy Head of the Department of Computer Science (2+2) in the period 2009-2013 and lead the failed (though positively evaluated) Centre application 2IT in 2013. Sindre was the leader of the Excited Centre for Excellent IT Education from 2016-21, thereafter the vice leader 2021-26. Sindre has teaching experience across a wide range of IT topics, from CS1 to PhD level research courses, and he obtained the status of Excellent teaching practitioner at the NTNU in 2025. His recent research has focused on computing education, in particular didactics of programming, self-paced mastery learning, project-based learning, digital exams, automated question types, and assessment integrity. He has been involved in several conferences in computing education research and engineering education, such as ITiCSE 2020 (general co-chair), CDIO 2023 (org. committee member), FSE-SEET 2025 (program co-chair), and ICSE-SEET 2027 (program co-chair).

Line Kolås is an Associate Professor at Nord University, Norway - also awarded the title "Excellent Teaching Practitioner". She holds a PhD in informatics and is part of the "Technology and Learning" research group. Her research interests are e-learning systems, ICT in education and IT didactics. She is currently a researcher and cluster leader in Excited - Centre for Excellent IT education, focusing on pedagogical practices.



Helga Dís Ísfold Sigurðardóttir is an Associate Professor at Nord University, Norway. She holds a PhD in science and technology studies, and her research interests include gaming, learning, artificial intelligence, the game industry and social impact games. She is currently leading the FutureSkill Alliance Erasmus+ project, focusing on artificial intelligence and future skills in education, and serves as a Work Package Leader in NEON Young Norway.



Aamna Rais Ahmed is a PhD candidate at Nord University, Norway. She holds an MPhil in Education, and her research interests are collaborative learning and digital pedagogy in higher education. She is currently a PhD candidate in Excited - Centre for Excellent IT education, focusing on blended learning in project-based computing courses, and previously led the implementation of the UNICEF Learning Passport in Pakistan.



Birgit Rognebakke Krogstie is an Associate Professor in the Department of Computer Science at NTNU. She has a background in Computer science (PhD) and Education (Cand.Phil.). Her research areas include reflective learning, technology supported work and learning, and higher computing education. In recent years, her main research focus has been the meaningful integration of sustainability in computing education. In Excited, Krogstie is Centre director. Formerly she was Centre vice director and project leader in the area of Employability and industry relevance.



Ole Goethe is an Associate Professor at Kristiania University of Applied Sciences and Nord University in Norway, where he leads the Gamification Lab and the Accessibility Lab. He is pursuing doctoral research in Human–AI Coevolution, focusing on how technology can strengthen human capabilities rather than replace them. His work spans gamification, Human–Computer Interaction (HCI), User eXperience (UX), accessibility, and AI in education and creative practice. Grounded in Human-Engaged Computing (HEC), his research



explores how design can support autonomy, competence, and meaningful collaboration.

Robin Isfold Munkvold is an Associate Professor at Nord University, Norway. His work focuses on designing engaging, authentic learning experiences that connect educational theory and practice. His research interests include project-based learning, game-based learning, gamification, and educational technology, with particular emphasis on enhancing student motivation, participation, and competence development.



His work examines the integration of game-based pedagogies into diverse educational settings, emphasizing learning experiences that encourage students to take an active role in addressing authentic challenges and real-world projects.

Trond Aalberg is a Professor at the Norwegian University of Science and Technology (NTNU), Norway, where he is a member of the Data and Artificial Intelligence Group at the Department of Computer Science. His research interests are in computing education, particularly peer assessment and feedback in project-based courses, where students learn by collaborative construction of software artefacts. In recent years he has given talks and held seminars and workshops on the use of generative AI in software engineering and its implications for how software development is taught and assessed. He is the project leader of the Excited cluster on infrastructure and learning technologies.



Somayeh Bayat Esfandani is a PhD candidate at the Norwegian University of Science and Technology (NTNU) in Norway, where she is a member of the Data and Artificial Intelligence Group at the Department of Computer Science. She is a member of the Excited Cluster of Infrastructure and learning technologies. Her PhD research explores the integration of generative AI into peer assessment to enhance the learning experience of Computing Education students, with a particular focus on peer code assessment.



Mariusz Nowostawski is a Professor at Norwegian University of Science and Technology (NTNU), leader of the Bachelor in Programming study programme and director of the Decentralized Systems Engineering Lab. Researcher and computer scientist, focusing on self-organizing systems, adaptive and autonomous computation, peer-to-peer systems and decentralization technology. He is a member of the Excited SFU cluster on innovative teaching methods and student-centered active learning. Current interests include peer assessment and the use of AI in IT education.



Table of contents

1. LEARNING THEORIES AND THEIR USE IN COMPUTING EDUCATION RESEARCH .. 17

1.1. Introduction 17

1.2. What is a learning theory? 17

1.3. The behaviourist orientation 19

1.4. The cognitive orientation 21

2. DESIGNING PROJECTS FOR LEARNING 38

2.1. Project-based learning 38

2.2. Frameworks/models for PjBL 38

2.3. Comparing the frameworks/models 39

2.4. Experiences and reflections on designing PjBL 45

2.5. Limitations 52

2.6. Designing projects for learning: Concluding remarks 53

3. THE DYNAMICS OF GROUP WORK 56

3.1. Introduction 56

3.2. Teamwork 57

3.3. Conflict Management 58

3.4. The tool 60

3.5. Example practice (suggested approaches for implementation) 61

3.6. End Comments 64

4. SUSTAINABILITY IN COMPUTING EDUCATION 67

4.1. Introduction 67

4.2. Sustainability models and principles 68

4.3. Sustainability competencies 71

4.4. Pedagogy for teaching about sustainability 72

4.5. Computing and sustainability 73
4.6. Sustainability as a topic in computing education 76
4.7. Examples from NTNU 77
4.8. Insights on sustainability integration in CS education 79
5. THE EDUCATIONAL PRODUCTION GAP 87
5.1. Introduction 87
5.2. Agile Practice and Supervision in Student Projects 88
5.3. What Makes Student Game Projects Different 90
5.4. The Agile Learning Team Framework (ALTF) 92
5.5. Discussion 98
5.6. Conclusion 99
6. EDUCATING FOR GENERIC COMPETENCIES 103
6.1. Introduction 103
6.2. Exploring the Term Generic Competence 104
6.3. Commonly Recognised Generic Competences 106
6.4. Designing Generic Competence Progression in Study Programmes 108
6.5. Integrating Generic Competences into Programmes and Courses 110
6.6. Conclusion 115
7. PEER ASSESSMENT IN PROJECT-BASED IT EDUCATION 118
7.1. Introduction 118
7.2. Why Peer Assessment Suits IT Education 119
7.3. Pedagogical Foundations 119
7.4. The Process and System Support 122
7.5. Peer Assessment Across the IT Programme 126
7.6. Designing Peer Assessment: Recommended Practice 128
7.7. Artificial Intelligence in Peer Assessment 130

7.8. Use Cases from Three NTNU Courses 131

7.9. Concluding Remarks 135

8. SELF-PACED MASTERY LEARNING 141

8.1. Introduction: Who is this chapter for? 141

8.2. What is mastery learning? 141

8.3. Experiences from the IT1001 Course 142

8.4. Make your own course: Design trade-offs 145

8.5. Learning resources and activities – and mitigating procrastination 153

8.6. Detailed design considerations 156

8.7. Make your own course - Process suggestions 157

9. GAMIFICATION IN HIGHER EDUCATION FROM RESEARCH TO EDUCATIONAL PRACTICE 166

9.1. Context and Motivation 166

9.2. Beyond Points, Badges and Leaderboards 166

9.3. Three Cases, One Spectrum 166

9.4. Understanding Gamification in Education 167

9.5. The Excited Context 169

9.6. Three Gamification Designs from Excited 170

9.7. Lessons Learned: What Works? 173

9.8. When Gamification Goes Wrong 177

9.9. Practical Framework for Educators 178

9.10. Discussion 179

9.11. Conclusion 179

10. THE LEARNING ENVIRONMENT 183

10.1. Introduction 183

10.2. Interventions for the learning environment 184

10.3. Excited interventions and research 186

10.4. CS1 as a signature course for the STEM Teacher Students, NTNU 187

10.5. Workdays at NTNU and Nord..... 188

10.6. Conflict management in project work (game-based learning) 189

10.7. Students as Teachers at NTNU and Nord 190

10.8. Hackathons and Creative Media Jams (Nord and NTNU) 191

10.9. Catch IDI Department Symposium, NTNU 192

10.10. Hybrid and online learning activities 192

10.11. Student identity and belonging 193

10.12. Concluding advice 194

APPENDIX: BOOK CHAPTER ABSTRACTS 198

1. Learning Theories and their Use in Computing Education Research

Birgit R. Krogstie

Norwegian University of Science and Technology (NTNU)

1.1. Introduction

The aim of this chapter is to introduce learning theories and show how they may be relevant to computing educators who wish to develop and improve their teaching.

We provide a brief overview of key theories and give some examples of how they are applied in computing education and Computing Education Research (CER), drawing on experience from Excited as well as the research literature.

Learning theories is a vast field, and this chapter can only provide a taster and point to a selection of relevant theories. After reading the chapter we hope you will have some new ideas about

- Main orientations towards learning (what learning is about)
- Why it is useful to be conscious about your orientation towards learning in the development of teaching in computing education
- Which (type of) theories could be relevant/useful for you to draw on, maybe with inspiration from studies presented in the research literature

While we believe this chapter could be relevant for different groups of readers interested in computing education, the chapter is written having in mind an educator with the goal of developing their teaching.

1.2. What is a learning theory?

Szabo and Sheard (2022) provide the following definition of an educational or learning theory as «*a generalisation, abstraction, explanation, or prediction of a learning phenomenon*» (Szabo and Sheard 2022; Szabo et al. 2019).

It is possible to view a learning theory as a theory describing what learning "really" is. Most practitioners and researchers would however consider a specific theory as *one* perspective, useful to shed light on important aspects of a phenomenon while focusing less on others.

Broadly, theories about individual learning have evolved from being primarily about *behaviour* (stimulus-response, reinforcement) through focusing on *mental processes* (e.g. processing capacity, mental models), to considering human activity and learning as fundamentally *social* (e.g. ingrained in practice/culture). These can be considered as *fundamental orientations* to learning (Szabo and Sheard 2022).

In addition to a fundamental orientation, the choice of theory to illuminate a teaching-related phenomenon typically implies selecting a *unit of analysis* in your work. Are you studying individual learners, groups of learners, or organizations? Furthermore, what is the role of *technology/artifacts* –

are they a key element of learning? Whose perspective are you considering – that of the educator developing a curriculum, or a student striving to learn?

To understand and act upon a situation or phenomenon it is often viable to use a combination of theories, e.g., to shed light on different levels or perspectives.

1.2.1. Why does learning theory matter for teaching?

It is always useful to reflect on your fundamental assumptions about important aspects of your work, and students' learning is at the heart of the work of any practitioner in computing education.

In a teaching context, your perspective on learning determines what you see, how you understand things, what to you is foreground and what is background, what are causal connections (i.e. by changing A, what might lead to change in B), what is within and beyond control, and what is important. For instance, if you believe individual goal setting to be particularly important for learning, you might focus on encouraging students to set individual learning goals and adjust their effort in accordance with their progress towards these goals. If you believe learning is basically a social activity, you might focus on students' teamworking skills or individual learning aided by a helper who is more knowledgeable or skilled. If you believe reflection to be essential to learning, you might want to make sure students engage in some form of reflection during or after a learning activity.

A theory of learning, or a combination of such theories, provides a lens for understanding and can also serve as a tool or framework for action, depending on the theory. For instance, theory of self-regulated learning (Zimmermann, 2002), sociocultural theory including the zone of proximal development (Vygotsky 1978) or reflective learning (Boud et al. 1985) can inform decisions on how to design learning activities along with associated learning objectives and assessment.

Importantly, the conscious consideration of your take on learning does not exclude the use of your practical experience and intuition about what is good practice and promising actions. On the contrary, drawing on theory that fits your basic views could help you build a stronger fundament around your approach and help you argue about it and share it, whether in local pedagogical communities or in the computing education research community. Awareness of your position in the landscape of learning theory could also help you *expand* your view by utilizing insights from related or complementing theories, which might help you identify promising ideas and avenues for further work.

1.2.2. Theories currently used in Computing Education research (CER)

A significant part of Computing Education research is based on practical teaching interventions. Looking into the CER literature, it is thus possible to get an impression of the use of theory in the development of teaching in our field.

(Szabo and Sheard 2022) examined the CE research literature to identify which learning theories are applied and how they form clusters of related theories. The overall network of learning theories in computing education is shown in Figure 1. Bigger nodes are more important in the network, and thicker lines show strong connections. The colouring indicates sub-networks of related theories.

Szabo and Sheard observed that computing education research is currently applying a rather narrow spectrum of learning theories compared those applied in computer science research at large. As examples of potentially relevant and applicable, but currently under-used theories, they mention self-determination theory, Maslow's hierarchy of needs, mindsets, and self-efficacy.

1.2.3. Main orientations of learning theories

In what follows, we present theories under the umbrellas of behaviourist, cognitive and social orientation. In practice, this differentiation is not clear-cut. Many theories do not fall strictly within one orientation, but combine elements e.g. from two of them. Also, there are aspects of theories that are not well distinguished by the main orientations, e.g. the role of artifacts. We briefly address some such aspects in connection with specific theories.

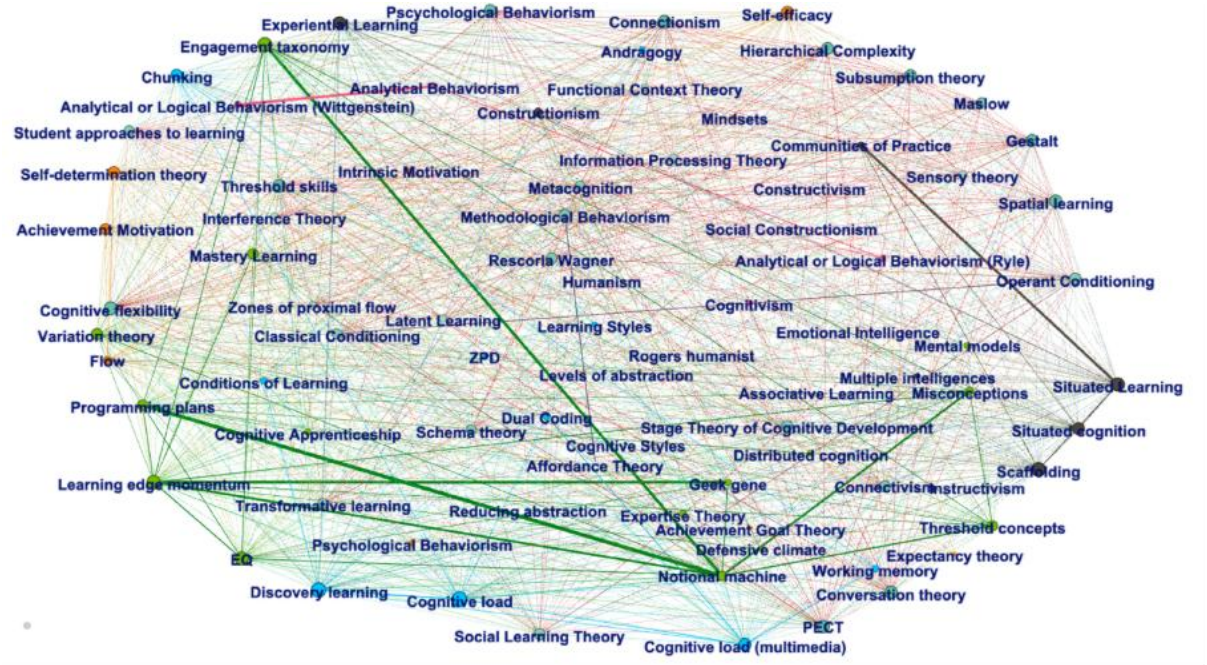


Figure 1: Learning theories in computing education. From (Szabo and Sheard 2022)

1.3. The behaviourist orientation

Originating late in the 19th century, the theory of *connectionism* (Thorndike [1911] 2026) launches *instrumental learning:* Learning occurs from *doing* something. Key elements of the theory include the *Law of effect*: Responses that produce a desired effect are more likely to occur again, whereas responses that produce an unpleasant effect are less likely to do so; the *Law of readiness*: Learning is most effective when a learner is physically, mentally, and emotionally prepared to act; the *Law of exercise*: Connections are strengthened through practice and weakened when practice stops.

The theory might be old, but we recognize fundamental assumptions about learning that most educators would still share: learning tends to benefit from students being motivated and ready to engage in learning tasks, from repeated practice of a type of task, and from the experience of pleasant effects of an effort, including mastery.

The term *behaviourism* was coined by John Watson, who saw its ultimate goal as the "prediction and control of behaviour"[1]. At the most fundamental level, behaviourism is about stimulus and response, reward and punishment, conditioning and reinforcement. Famously, Ivan Pavlov[2] investigated the natural/innate response of a dog presented with tempting food (i.e., meat). The dog will salivate, the

[1] https://learn.akkadium.com/behaviourism-roots-of-learning-technology/
[2] https://en.wikipedia.org/wiki/Classical_conditioning

food serving as an *unconditioned stimulus*. If, over time, food is systematically paired with a neutral stimulus, e.g. the sound of a bell, the same salivation response will be triggered by the sound of the bell, which serves as a *conditioned stimulus*. For the dog, the bell has become associated with food. This is called *classical/respondent conditioning*.

In *operant conditioning* (Skinner 1963, 1938) an organism learns to associate a behaviour with its consequences. Action followed by reward is more likely to be repeated, whereas action resulting in punishment, will be avoided. Reward and punishment can thus be applied to encourage certain behaviours and discourage others. In an educational context, Skinner was a proponent of *programmed instruction* whereby instant positive feedback to the student leads to desired adjustment, conscious or unconscious, of behaviour.

Thinking about our day-to-day teaching and curriculum development in higher education, common ways of motivating student behaviour could probably be considered as forms of reward and punishment.

Rewards and instant positive feedback are key elements in gamification, as elaborated in the chapter *Gamification in Higher Education*. A related concept is that of nudging, which means a gentle intervention steering behaviour in a predictable direction without forbidding options or significantly changing economic incentives (Thaler and Sunstein 2008). Learning technology focusing on the detection of, and adaptation to, learners' behaviour, can also be associated with behaviourism. This includes sentiment analysis (Wankhade et al. 2022), which aims to detect learners' emotions, for instance anxiety, happiness or anger. Based on the diagnosed emotion, the system might adapt, e.g. by aiming to make the ongoing task less stressful or by suggesting a break, thus influencing learner behaviour. Overall, however, adaptive learning systems, while focusing on learner behaviours, typically have characteristics associating them more with a cognitive orientation to learning.

1.3.1. Example: Encouraging student attendance

Computing education typically includes events and activities intended to foster not only students' learning, but also their feeling of belonging and thriving as part of a community. An example from NTNU is IDI Workday, organized by Excited over a nine year period. The Workday is a weekly half-day activity for first year students in the Computer Science department. The attending students are served a simple breakfast (DIY sandwiches + coffee) and get help from teaching assistants with assignments, mainly in the demanding programming courses, The purpose of the Workday is to offer an arena which is welcoming to all first year students in the computing study programs, and which combines the social with the curricular.

Evaluation of the Workday program indicates that the breakfast is an important part and that students appreciate the encouragement to start their day relatively early in the morning. Attendance at the Workday can be seen as based on a mix of intrinsic and extrinsic motivation: The breakfast helps encourage student behaviours we believe to be beneficial to students' belonging in their study program and to their intention to stay.

1.3.2. Example: Parsons problems

An approach applying elements of gamification is the use of Parsons problems to teach programming (Parsons and Haden 2006). Parsons problems are drag-and-drop style puzzles where the student is given a selection of code fragments and the task to choose the relevant ones and put them together in the right order to solve a given problem. The student may then get immediate feedback on the accuracy of their

solution. The approach is based on 1) using gamification to increase engagement, while 2) helping students focus on the structure/logic of a solution rather than syntax, i.e. minimizing the cognitive load. Work on Parsons problems require a way of thinking which is essential in computing (Ericson et al. 2022), i.e. ordering a set of steps that together will implement an algorithm.

The Parsons problems example illustrates how a pedagogical approach can draw on more than one main orientation to learning. Gamification and engagement, arguably, tend toward a behaviourist orientation through the focus on reward and reinforcement. The concept of cognitive load, the learner's understanding of logical structure and the correct sequencing of steps in an algorithm, as well as the provision of scaffolding of the learner's learning process can be more associated with a cognitive orientation, to which we will now turn.

1.4. The cognitive orientation

The mental or cognitive orientation to learning draws on an idea of human learning as involving information processing and mental structures.

1.4.1. The information processing model of human learning

A way of viewing the workings of the human mind is by comparison to a computer. The brain can be seen as having a sensory memory, a working/short-term memory and a long-term memory. Sensory memory is for capturing and very briefly holding large amounts of data taken in through sight, sound and touch stimuli. Attention brings this information into working memory, where it can be processed/rehearsed. Without rehearsal, the information will disappear after 5-15 seconds. The lasting, long-term memory is for storing the result of this processing, which can happen when the data is encoded in a meaningful way, making it available for later retrieval. Throughout these steps, information will be lost, not reaching the learner's attention or not being processed for lasting storage. Learning, in this model, means having contents from the short-term memory, through rehearsal, become part of long-term memory.

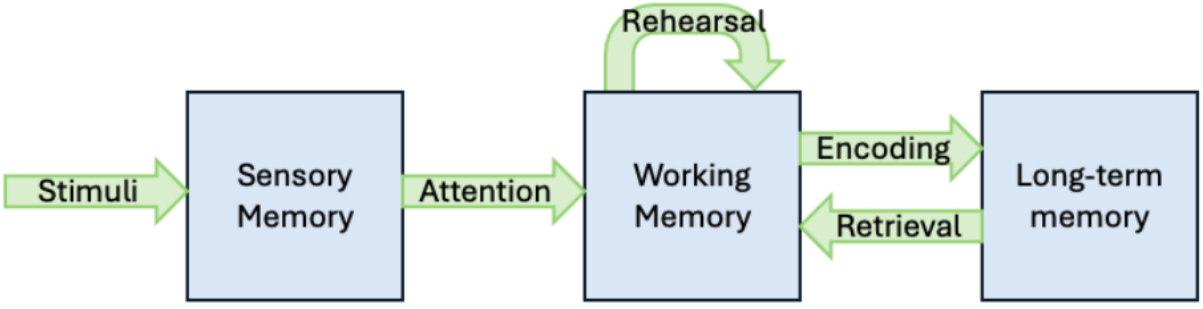


Figure 2: The information processing mode of human learningl[3]

Independently of how accurate an information processing model of human learning might be in terms of describing how the human brain works on a fine-grained level, it has proven accurate enough to serve as a framework for key studies in psychology and education and to lead to valuable insights for everyday teaching.

[3] For an elaboration of this model, see https://dataworks-ed.com/blog/2014/07/the-information-processing-model/

Practical implications for educators include:

- To get what starts as stimuli into the working memory through attention and further into the long-term memory through rehearsal, some types of learning activities are more effective. Student-active forms of learning are known to be more effective, e.g. for attention, and the processing of information such as through application, contextualization and reflection, should involve higher-order forms of learning (cf Bloom's taxonomy, addressed later in this chapter).
- The capacity and limits of human working memory should be taken into account. The total number of "items" a person can hold in working memory varies among individuals, in the range of 7 +/- 2. Techniques to remember more information includes bundling information into larger chunks.
- To get the most out of sensory memory, different modes of sensing could be utilized, e.g. in teaching or learning activities including both visual and hands-on elements.

A current topic where a human information processing perspective may help us reach insights as educators, is the use of AI tools in work and learning tasks. The AI tools can be considered as part of the overall information processing. It is largely acknowledged among educators as well as students is that it is impossible to have an AI tool do the learning *for* the student. The question is, then, what does it take for AI to help *strengthen* the learner's information processing, e.g. helping them learn more or more efficiently, as opposed to short-circuiting steps necessary for human learning. This is an area of much ongoing research.

Cognitive load

A concept that has gained popularity in education research, is that of *cognitive load*. Cognitive load theory is based on insights about the relations between working memory and long-term memory (Sweller et al. 2011). The cognitive load is the load that performing a specific task imposes on a learner's cognitive system. Cognitive load refers to the load that performing a particular task imposes on a learner's cognitive system. On a practical level, for an educator, theory of cognitive load points to the importance of designing learning activities and tools so that the student's cognitive load does not become too heavy.

Designing for a reasonable cognitive load might involve diagnosing and if possible adapting to the students' cognitive load at any time in the learning process. It may also involve enabling learners themselves to regulate their activity to achieve a manageable cognitive load, i.e. through having some form of flexibility.

Importantly, cognitive load is not only a question of processing a set of items related to a specific learning task. There is connection between cognitive load and emotions, e.g. when emotions represent extra cognitive load and compete for limited working memory resources, or when emotions strengthen cognitive processing by helping the learner utilize their cognitive resources more effectively (Plass and Kalyuga 2019). Also, the learner might be cognitively engaged with other tasks than those linked to the learning activity at hand, in the educational context or on other arenas. All in all, learners' cognitive load might be influenced by a number of factors, many of which can be influenced by the educator (e.g. the learning task itself, or the presence of distracting elements in the learning situation, such as games or social media conversations). Other elements are largely beyond the educator's control (e.g. students' life challenges). A functional and welcoming learning environment (socially and physically) is likely to play a role and is within the influence of the higher education institution. For more on the importance of learning environments for students' learning, see chapter X.

Cognitive development

Jean Piaget (Piaget 1980) famously developed a theory of cognitive development. Fundamental to this theory is that human knowledge and skills are based on cognitive structures/schemas. Learning takes place as new information meets existing cognitive structures/schemas. *Adaptation* then happens through either of two main mechanisms: *Assimilation*, whereby new information is fit into existing cognitive structures, and *accommodation*, by which the cognitive structures are changed to make the new information fit in. These mechanisms can be considered fundamental to human learning independent of age.

Going back to the information processing model, we may think of the existing cognitive structures/schemas as the long-term memory, where encoded information from working memory is adapted through assimilation or accommodation.

Another way of thinking about the difference between assimilation and accommodation is to conceive your existing knowledge as a kind of «theory» serving as a frame for interpreting your experiences «in practice». Assimilation, then, is like fitting practice to theory, and accommodation means fitting theory to practice!

Cognitive levels of learning

A concept frequently encountered in education literature is *metacognition*: "thinking about thinking". Metacognition was defined by John Flavell[4] in 1979 as the ability to monitor and control your own thinking and learning processes[5]. Metacognition implies being *aware* of what is happening in your mind (e.g.: "I seem to be losing focus", "This is confusing", "My strategy to learn this is not working") and take *control* of it (e.g. select a learning approach or adjusting the one you are currently using).

Metacognition can thus be associated with more advanced forms of learning, e.g. as a necessity for reflection and self-regulated learning.

A well-known framework distinguishing between different cognitive levels of learning was published by Benjamin Bloom in 1956. Bloom describes six levels of learning objectives in the cognitive domain, ranging from the rudimentary (remembering facts and basic concepts) to the advanced (creating something original). The taxonomy was later revised and expanded by Anderson and Kratwohl (Kratwohl 2002), who turned it into a two-dimensional framework with levels of learning on one dimension (Remember, Understand, Apply, Analyze, Evaluate, and Create) and types of knowledge (Factual, Conceptual, Procedural, and Metacognitive) on the other.

Bloom's taxonomy as well as its revised version can be useful for educators defining learning objectives in a curriculum, for instance to help make sure that higher levels of learning are being addressed in a course or lecture. To make more concrete what type of student activity represents learning on each level, the framework provides sample verbs. For instance, learning on the level of understanding might involve classifying, describing, discussing, explaining, identifying, locating, and reporting. This operationalization can be helpful when designing for constructive alignment between learning outcomes, learning activities and forms of assessment. A useful resource on Bloom's taxonomy and its revised version has been published by The Centre for the Advancement of Teaching Excellence at the University of Illinois Chicago[6]. For more on learning outcomes, see chapter in this book.

[4] https://en.wikipedia.org/wiki/John_H._Flavell
[5] https://www.psychologynoteshq.com/metacognition/
[6] https://teaching.uic.edu/cate-teaching-guides/syllabus-course-design/blooms-taxonomy-of-educational-objectives/

Computational thinking

For learners in computing, it is important to be skilled in *computational thinking.* This concept can be understood as related to human thinking - solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science (Wing 2006). It can also be seen as information processing: «a thinking process where "… solutions are represented in a form that can be effectively carried out by an information-processing agent» (Cuny, Snyder, and Wing 2010).

In higher computing education, courses on programming and algorithms will depend on, as well as further improve, students' computational thinking skills.

Threshold concepts

As part of the learning process in an area such as computing, students will encounter *threshold concepts*. Threshold concepts represent a way of understanding, or interpreting, or viewing something, without which the learner cannot progress. Insight about threshold concepts can help structure learning/teaching and predict what the students will struggle with. Threshold concepts are characterized by being transformative, integrative, irreversible, potentially troublesome for students, and bounded (Sanders and McCartney 2016). Threshold concepts are thus key concepts that require the student to work hard and think in new ways. After a period of effort and frustration, the student gets an "Aha"-experience, and once they get the point, they really get it.

Threshold concepts in programming are partially specific to programming language/language types, and partially more general. Example threshold concepts include recursion, pointers, complexity, state, and object-orientation-related concepts such as classes, objects and inheritance. Experienced educators are likely to be aware of what are the threshold concepts in their courses, targeting their teaching to helping the students overcome the known barriers. Identified threshold concepts may also serve as pointers to relevant education research literature on how to address teaching related to the concepts in question.

Notional machines

The computing disciplines are characterized by abstraction in many layers. In professional computing work there is a plethora of conceptual models which are designed, reasoned about, improved, and turned into working software and information systems. To support human work on conceptual models they are given visual representations.

The abstraction involved in computing can be challenging to students. At the same time, a model at a higher conceptual level is easier to comprehend than the real machine. In teaching, models and their visualizations are an ingrained part, both as a reflection of professional, disciplinary work and for pedagogical purposes. Visualizations include for instance simulations showing what happens in a process, e.g. when a program executes. In case of an object-oriented program, for instance, it may be visualized how objects are instantiated based on classes, and how they interact.

A *notional machine* is an abstraction designed to provide a model to aid students in understanding a particular programming language construct or program execution. This abstraction does not need to accurately reflect the exact properties of the real machine. There may be different visualizations of one notional machine. Figure 3 shows a visualization of a notional machine (Berry and Kölling 2014).

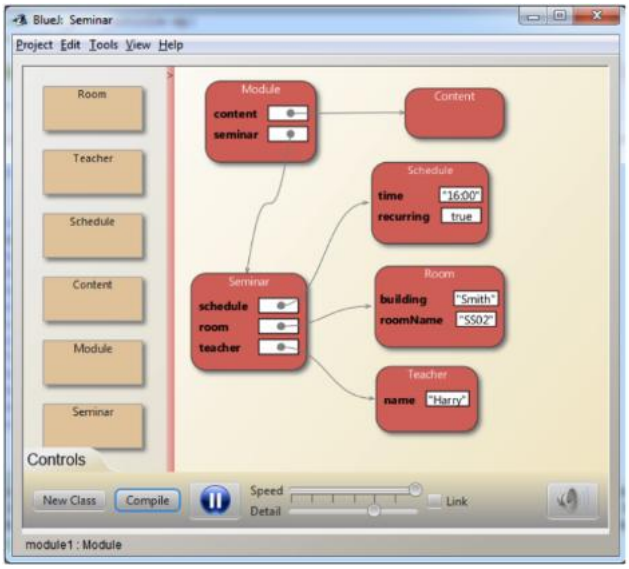


Figure 3: Novis Notional machine visualizer (Berry and Kolling 2014)

Today's computing students, when working on course syllabi and threshold concepts, are likely to consult the Internet and find a rich selection of educational resources with relevant visualizations and simulations. The same type of resources can also be applied by the educators in their teaching.

When to learn computational thinking

To reach the learning objectives of higher education we depend on students to be reasonably skilled in computational thinking when they *start* their studies – and probably as a prerequisite for choosing to enter a study programme in computing in the first place. This points to higher education institutions being a stakeholder in what happens in the primary and secondary school levels. For instance, Excited has been involved in research on courses teaching in-service K-12 to teach programming (Rouhani et al. 2022).

For students in primary and secondary education, apart from receiving good teaching, becoming skilled in computational thinking may also depend on more general interest in computing and technology as such. The computing education field has long strived to recruit a diversity of students, acknowledging the necessity to particularly encourage girls and pupils from different social backgrounds. To create interest in computing, technologies intended to be fun and interesting have been introduced, focusing e.g. on co-design and creative use of technology in familiar, real-life settings (Saxegaard and Divitini 2019). While engaging tools may inspire potential students, however, students' intention to continue learning computer science may depend on more profound perceptions and beliefs, including a more general affinity for technology (Jin and Divitini 2020).

In sum, while it is important for us in computing education to focus on strengthening students' computational thinking skills, this partially has to be addressed at the primary and secondary school level, while also recognizing that it is not alone sufficient to inspire a diverse recruitment of students for computing education.

Self-Directed Learning (SDL) and Self-Regulated Learning (SRL)

Two practical theories that can at least partially be associated with a cognitive orientation, are SDL and SRL. Both of focus on the learner's active role in monitoring and managing their learning based on learning goals and progress.

Self-Directed Learning was seen by the American educator Malcolm Knowles (1975) as an approach to learning which is common to all adult learners: A tendency to want to direct their own learning process. SDL is based on the following steps for the learner:

1. Assess readiness to learn
2. Set learning goals
3. Engage in the learning process
4. Evaluate learning.

While being a kind of learning theory, this is just as much a model of a learning process that an educator can design for when planning learning activities and curricula. Each of the four steps can be addressed with appropriate support for the learners. Among the resources available for educators in this regard, we recommend a website from The University of Waterloo[7].

Self-Regulated Learning has been explained in the following way by B. Zimmermann (2002): *"Self-regulated learning and performance refers to the process whereby learners personally activate and sustain cognitions, affects, and behaviours that are systematically oriented toward the attainment of personal goals. By setting personal goals, learners create self-oriented feedback loops through which they can monitor their effectiveness and adapt their functioning"*. (Zimmermann, 2002) As part of this, motivational beliefs are important. So is the involvement of others in social learning, e.g. getting help from peers, coaches, and teachers.

Practical implications for educators from a SRL perspective on learning include the importance of helping learners

- being aware of their learning strategies (with their strengths and weaknesses)
- having a repertoire of strategies to choose from
- having a growth mindset (believing that they can develop and learn), and – relatedly:
- attributing their learning outcomes not only to intrinsic factors ("I cannot do this because I am just not a computer guy") to extrinsic ones ("I cannot do this yet because I have never had programming as a hobby").

Educators can focus their teaching on providing students with such skills and attitudes. The question of attribution can be considered linked to self-efficacy and identity as a (future) computing professional.

Mastery learning

Mastery learning is based on the idea of adapting the instructional method and time to match individual learning needs. The student needs to master a topic before progressing to the next. Thus, with the right individual support, learning is mainly a matter of time and instruction, not ability, and differences in achievement among students can be substantially reduced. The most well-known early contributor to

[7] https://uwaterloo.ca/centre-for-teaching-excellence/catalogs/tip-sheets/self-directed-learning-four-step-process

this theory is Benjamin Bloom . A more recent and available research paper on mastery learning and Bloom's work is (Guskey 2007). Guskey points out that mastery learning has been shown to be particularly effective when the instruction is focusing on higher level learning goals, e.g. problem solving.

In computing education research, mastery learning has been applied to introductory programming courses. Research on *learning edge momentum* (Robins 2010) focuses on mastery learning as a way of establishing a positive learning momentum. The common sense-based notion that there are two "natural" groups of learners: programmers and non-programmers has been refuted in many research studies. Theory on learning edge momentum helps address how learning can be supported for any learner of programming. In this book, chapter goes into more detail on mastery learning in computing education.

1.4.2. The social orientation

The third and final orientation to learning to be addressed in this chapter, is the social one. Focus thus turns to human activity and learning as fundamentally *social* and ingrained in practice and culture.

Social learning theory

Learning happens in a social context. Learners compare themselves to others and learn by observing others. This is reflected in the social learning theory of Albert Bandura (Bandura 1977).

One concept that we will particularly highlight from this theory is *self-efficacy*. Self-efficacy is the individual's judgment of his or her capabilities to organize and execute a course of action, i.e. a kind of self-confidence to act in a domain. A student's self-efficacy can impact on their learning (Batra and Atiq 2025), e.g. when a more confident student dares to participate, explore and experiment and thereby learns from the experience, whereas a less confident student holds back and miss the same opportunity. In computing education research, studies have explored the connection between gender and self-efficacy, social background and self-efficacy, and the impact of self-efficacy on computing students' teamwork (Lundberg, Gunhild 2016). These studies typically derive implications for how to help strengthen the self-efficacy of the students in in question, e.g. through positive experiences of mastery.

A useful aspect of self-efficacy as a scientific construct is that there exist validated questionnaire instruments to measure it. Instruments for self-efficacy in particular domains can be found by searching the research literature, and there are also resources for tailoring your own instrument based on a generic template (Bandura, Albert 2006).

Social constructivism

Social constructivism sees human development as socially situated. Knowledge and meaning are actively created through communication and shared experiences taking place in social settings highly influenced by culture and history.

A prominent contributor to this strand of theory is Lev Vygotsky. Vygotsky was a proponent of sociocultural theory considering learning to be a social process where individual mental growth happens through interaction with other people, language and people. Educators are typically familiar with Vygotsky's concept *zone of proximal development (ZPD)*. Originating in Vygotsky's work on children's development (Vygotsky 1978), this concept is now widely applied in pedagogical settings including higher education. The ZPD denotes the gap between what a learner can do independently and what they can achieve with guidance (see Figure 4). Receiving guidance, also denoted *scaffolding*, the learner will gradually become able to perform the tasks in question more independently. In the end, the

scaffolding can be removed, and the ZPD will have expanded so that the learner can achieve even more with assistance.

A pedagogical implication from this theory is that the educator should focus on the student's ZPD as a learning potential, and on what scaffolding might help the student forward.

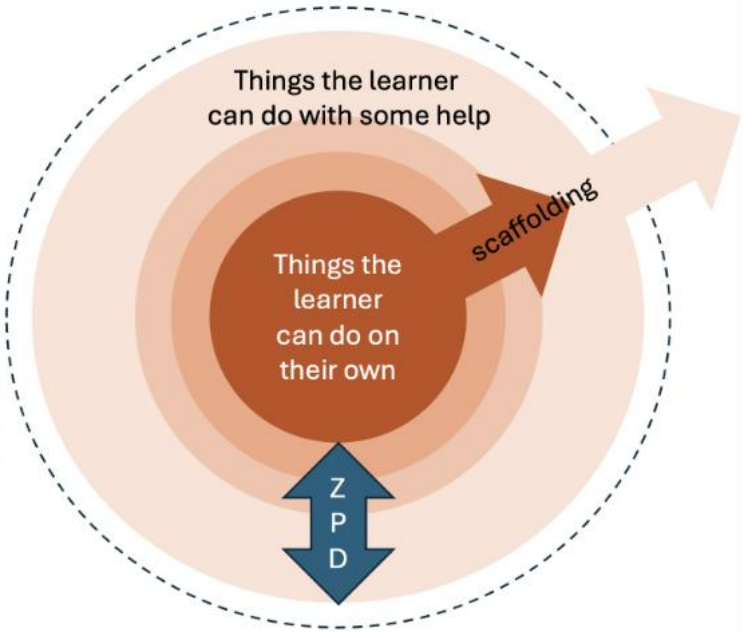


Figure 4: Scaffolding in the Zone of Proximal Development (ZPD)

Students at different stages of progression within a learning domain will typically benefit from different scaffolding. In project-based learning in software development, for instance, it is common to provide more structure in early courses, allowing the students to focus on fundamental skills in areas such as teamwork and the systematic application of particular technologies, languages and frameworks. In later courses it can be assumed that the students have the basic skills in these areas and that their learning will benefit from more challenges and less structure in other areas e.g. in handling the relationship with an external project customer (Bygstad et al. 2009). For more on project-based learning, see chapter.

One thing that should be noted about scaffolding as a term in education, is that its current meaning has as shifted from its meaning in Vygotsky's work. While the latter refers to scaffolding as guidance of a temporary nature, to be removed when the scaffolding is no more needed by the learner, scaffolding is now often used as a general term for support for a learning activity, whether provided by a human or by technology, and whether it is intended to be removed or not.

Experiential learning

The idea of experience as the key element in learning, "learning by doing", is associated with John Dewey (Dewey, J. 1986). Dewey argued about the importance of students relating information to prior experiences. For learning to be effective, the learner should be autonomous and engage in hands-on experimentation, thus deepening the connection to the new knowledge.

The reflective learning cycle (Figure 4) outlined by Kolb and colleagues (Kolb 1984) builds on the same idea. Taking place in a social context heavily influencing the learning process, experiential learning consists in a series of steps forming a cycle: 1) having a concrete experience, 2) reflecting on this experience, 3) learning from the experience by conceptualizing and abstracting, and 4) actively planning for new experience, e.g. a next version of the activity leading to experience.

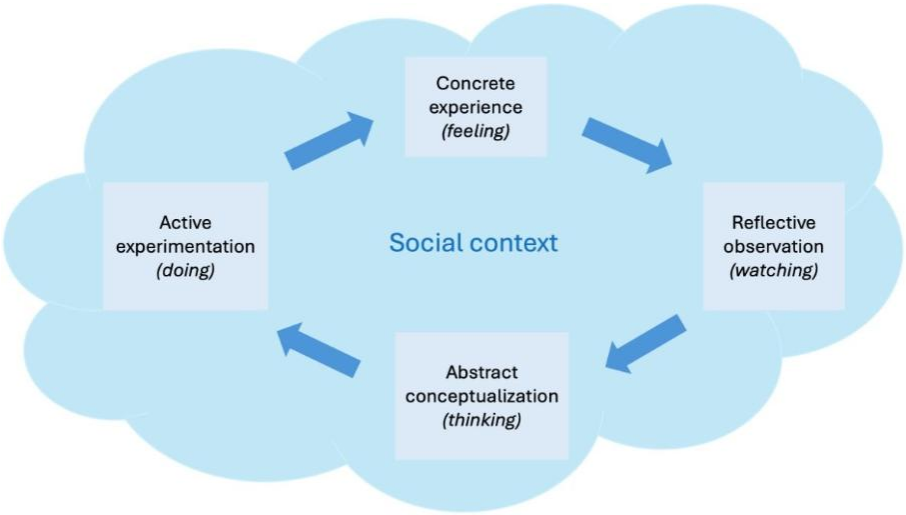


Figure 5: The experiential learning cycle (Kolb 1984)

In designing learning activities, educators can aim for explicitly supporting these steps and the transitions between them.

The reflective learning process

David Boud and colleagues (Boud et al. 1985) focus on the reflective process as a key element of learning. The reflective learning process model (Figure 5) distinguishes between experience(s), the reflective process, and its outcomes. Experience(s) include not only behaviour/action, but also ideas and feelings. Emotional aspects are often key part of an experience. For instance, in the context of computing education, there could be great frustration from trying to solve a difficult task, or corresponding joy upon of finally finding a solution. The reflective process, which returns to the experiences, includes re-evaluating the experience and attending to the emotional aspects. The possible outcomes of reflection include actual change in behaviour, but also less visible outcomes such as new perspectives and a readiness and/or commitment to action.



Figure 6: The reflective learning process (Boud et al. 1985)

As with Kolb's learning cycle, the reflective learning model can serve as the backbone for an educator's design of support for a learning process. The elements of the model help identify important aspects to

consider. For instance, when asking students in a project course to reflect on the project process, it may be important to encourage reflection also on the emotional aspects of the project experiences. Tools for doing so exist, see e.g. (B. R. Krogstie and Divitini 2009). Regarding reflection outcomes, it may be beneficial to be clear about what type of outcomes are expected (a modified project plan, for instance).

Considering support for reflective learning, ii should be noted that this tends to assume that reflection is something that can be deliberately triggered through specific types of *activity* (e.g. a reflection workshop, the writing of reflection notes). From a psychological point of view, however, reflection can be seen as a more fundamental and inevitable part of a learning process, happening spontaneously following a perceived a mismatch between experience and expectation. This might involve feelings such as surprise, confusion and frustration. Providing support for reflection, then, can be seen as facilitating activity leading to this type of feeling through recall and reconsideration of experience.

Collaborative learning

Learning often takes place through some form of collaboration in a group. At the time of graduation, students in computing education have taken part in group- or team-based learning activity in numerous courses. The term "collaborative learning" probably fits much of this. It is however inconsistently applied in the education literature, serving as a kind of umbrella term for many different approaches. We will therefore be a bit more specific on what many would agree to.

Collaborative learning not the same as groupwork based on any form of participation. Rather, participation is based on principles of democracy and respect. Elements considered necessary for collaborative learning include

- respect for group members' individual abilities and contributions
- group members sharing authority and the responsibility for the group's actions
- cooperation based consensus building – not competition

This involves positive interdependence among team members, a considerable amount of interaction, and social skills (Laal and Laal 2012).

Communities of practice (CoP)

Theory of community of practice considers learning as an element of practice, e.g. in a workplace or in a non-profit organization. A community of practice is a group of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise by interacting on an ongoing basis[8]. A new member of a community of practice starts out as a beginner who, through engagement, interaction and collaboration within the community, gradually moves towards expertise. Learning thus takes place within communities. Learning may also happen across communities, aided by *boundary objects* and *brokering*. Boundary objects are abstract or concrete objects that have an existence and (different) meaning in different communities. They can help create coherence and support collaboration across communities. Brokers are individuals who are members in more than one community and can help bridging the two, translating and integrating practices (Lave and Wenger 1991; Wenger 1998).

[8] https://wenger-trayner.com/introduction-to-communities-of-practice/



Figure 7: Learning through participation in a Community of Practice

In computing education, a CoP perspective can shed light on learning as something happening in a community within the educational setting (e.g. in a course or a student project team, or among colleagues in the teaching staff) and through interaction with other communities. Topic areas that can benefit from being addressed from a CoP perspective include student competencies vs. real-life practices in the profession (including industry, public sector, developer communities, and professional associations), professional and pre-professional identity, employability, industry/university collaboration, and project-based learning involving external stakeholders.

Example of applying CoP in computing education research

In Software Engineering, the dynamics and work within the team can be influenced by members' role and participation in other communities. For instance, a study found that a team member's ability to work as a broker between the team and a technology-related community can be decisive for the group's progress and also strengthen the position of the broker within the project team (B. Krogstie 2008).

Socio-cultural theory

Socio-cultural theory, i.e. the home turf of Vygotsky, is particulary concerned with the influence of history and culture on human activity and learning. An important name in this tradition is Michael Cole. In his sociocultural perspective, 1) culture is the "medium" humans live in (but are often unaware of), 2) human activity is mediated by cultural tools and artifacts, which have a dual ideal/material nature, and 3) psychological development happens through joint/collective activity and is transferred among generations.

Another influential scholar in this tradition is Yrjö Engeström. In his *activity theory* (Engeström 1987), the *activity system* captures important elements of human activity, represented as nodes in an extended triangle (see Figure 8): A *subject* acts on an *object* by use of (mediated by) an *instrument*, in order to create an *outcome*. This is happens in the context of a *community* and is subject to *rules* and a *division of work*. Different activity systems are connected through the nodes. For instance, the output of one activity (say, a learning tool) might be the instrument in another (e.g. a teacher applying that tool in a course). The activity system serves as a useful analytical building block for making sense of how various activities are structured and connected. What makes the theory even more attractive, e.g. for an educator seeking to understand a particular learning context, is that is also addresses the dynamics of change. There may be tensions within and between nodes inside and across activity systems, and also between an activity system and a perceived future or alternative version of the system. These tensions drive change, which can be considered as learning.

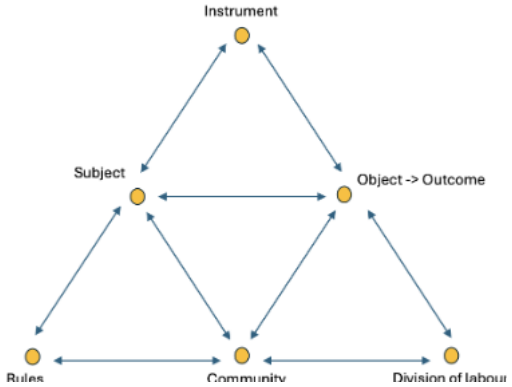


Figure 8: The activity system (Engeström 1987)

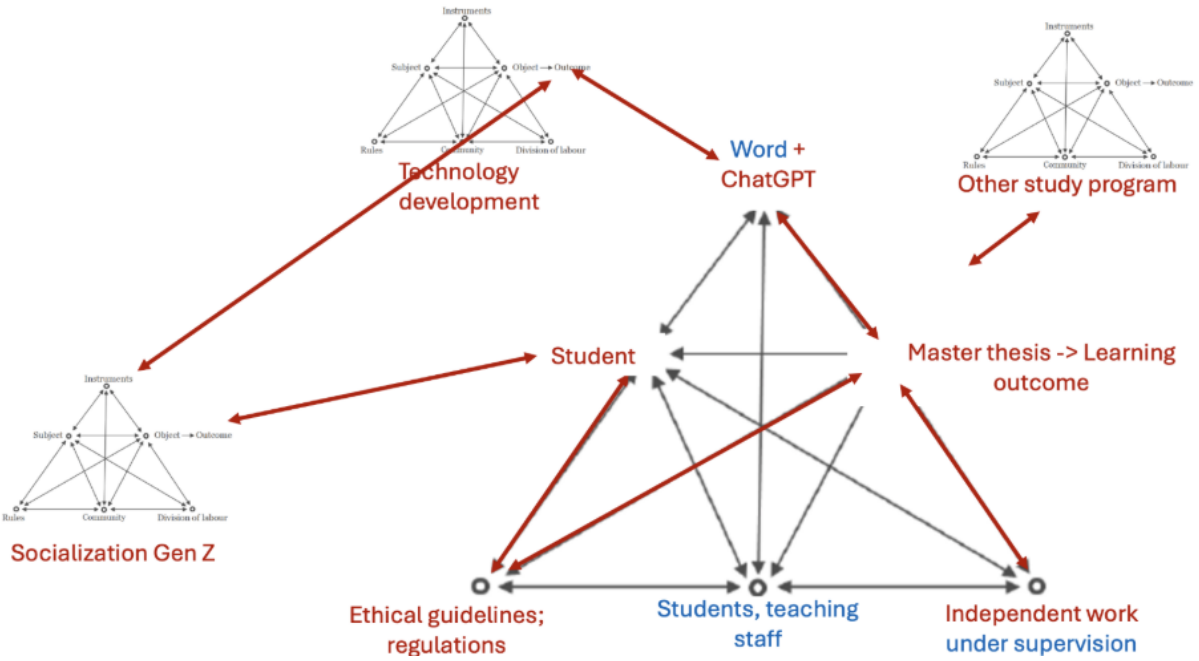


Figure 9: An activity system perspective on introducing ChatGPT in master thesis work

Example of applying activity theory in CER

The following example presents a rudimentary modelling of a challenging situation in computing education (Figure 9). In the activity system shown with a large triangle in the diagram, the subject is a student working on his master thesis (the object). To simplify quite a bit, we assume that the old way of using tools in this setting is to use Word for writing the text, whereas now Chat GPT has been introduced to aid some of this work. The use of colours is a choice by the author: nodes and relationships now subject to some tension, are shown in red.

Represented as two separate activity systems, the model depicts the socialization of GenZ along with the general technology development as sources of influence, affecting the choice of tools and how the students use them. Today's students are obviously familiar with GenAI and skilled in using GenAI tools e.g. to find and structure information. Furthermore, also represented by an activity system, the use of GenAI in similar contexts in other study programs, especially at the same university, is likely to be a

source of comparison, potentially influencing the master thesis activity. Turning to the nodes of the main activity in the model, the use of GenAI will affect the thesis and the learning outcome – positively or negatively, and not necessarily in the same direction (e.g., a thesis that looks nice after the use of GenAI is not necessarily evidence that the student's learning has improved similarly). The thesis and the student's work may or may not now be in line with ethical regulations and guidelines regarding the use of GenAI. Also, the division of labour involved in a master thesis, e.g. the student doing independent work under supervision, may now have been changed: is the work still independent? Is GenAI part of supervision?

This example illustrates that a teaching context and situation can be analyzed by identifying activity systems with their relationships and tensions. More potential tensions could probably have been identified in this case. It should be noted that in this type of analysis it is not always evident what the activity systems are – one activity system might for instance alternatively be considered as a combination of systems. From the point of view of gaining insight about and describing a situation, the level of abstraction and choice of activities to include is up to researcher creating the model.

Connectivism

A final theory to be touched upon in this chapter is Connectivism. It is a theory which - analytically - places technology on the level of human actors as part of a network through which activity and learning takes place and over which knowledge is distributed. Other theories resembling connectivism in this regard include Distributed cognition (Salomon 1993; Hutchins 1995) and Actor-network theory (Latour 2005).

Siemens (2004) describes connectionism in the following way: "Connectivism is the integration of principles explored by chaos, network, and complexity and self-organization theories. Learning is a process that occurs within nebulous environments of shifting core elements – not entirely under the control of the individual. Learning (defined as actionable knowledge) can reside outside of ourselves (within an organization or a database), is focused on connecting specialized information sets, and the connections that enable us to learn more are more important than our current state of knowing." (Siemens 2004). Connectivism includes *connection making* as a form of learning activity. Goldie (2016) explains: "Connectivism asserts that knowledge and learning knowledge are distributive, i.e. they are not located in any given place, but instead consists of networks of connections formed from experience and interactions between individuals, societies, organizations and the technologies that link them. Knowledge is viewed as a process, fluid and dynamic flowing through networks of humans and their artifacts» Learning, then «occurs through the construction and traversing of networks. Participation in network activities results in the creation, removal or adjustment in strength of connections. (Goldie 2016).

Connectivism has informed pedagogy for MOOCs, massive open online courses. In our age of extensive use of AI and corresponding need for insights about its role and appropriate usage in learning, connectivism might be useful for shedding light on the role of GenAI as parts of a network connecting human learners and technology.

Computing Education Research and related areas

For an educator interested in computing education research and learning theory relevant for such research, it is worth noting that there are multiple fields to consider beyond CER.

First, communities aimed at STEM education, engineering education and higher education in general all address pedagogical topics and challenges of relevance to computing education.

Computer supported collaborative learning (CSCL) as a research area started in the 1990s as a reaction to software that forced students to learn as isolated individuals (Stahl et al. 2006). CSCL was conceived as a pedagogical approach where learning takes place via digital social interaction. It is characterized by the sharing and construction of knowledge among participants using technology as their main means of communication or as a common resource. Small groups are the unit of analysis. Over the last years, CSCL has ended up spanning a variety of theories (Stahl and Hakkarainen 2021).

Technology Enhanced Learning (TEL) is another research area focusing on technology support for learning, covering both individual and collaborative learning. A useful introduction to TEL is provided in the DE-TEL book[9]. According to this source, a broadly used definition of TEL is "a field of research aiming at improving learning by integrating current technologies and designing innovative ones" (Bourdeau and Balacheff 2014).

Computing education research has some overlap with CSCL and TEL in terms of studies involving technology support for learning in computing.

The *Learning Sciences* (LS) is another field with some overlap with those mentioned above. According to a handbook of LS (Fischer et al. 2018), the field represents "the interdisciplinary, scientific study of human learning and its processes in the real world. It explores how people learn, identifies what might be important for human learning, engages in the design and implementation of learning environments, and improves instructional methodologies." Studies in Learning Sciences often apply a design research methodology. LS comprises research involving Learning analytics (LA).

Finally, the major fields of social science such as psychology, education, organization management and sociology, provide a potential (and huge!) treasure chest for those interested in useful and enriching perspectives on learning.

Choosing a learning theory to inform your teaching

As concluding remarks in this chapter, we offer a small set of general recommendations for those looking for learning theory to inform or inspire their work as educators. These recommendations implicitly assume that the educator looks into resources beyond what could be presented in in this chapter.

We suggest that you look for learning theory and associated frameworks and methodologies that…

- ..shed light on the aspects you believe to be most important, based on your experience in education
- ..shed light on aspects you are required to address and evaluate, if any
- ..is well aligned with research method(s) you consider applying, if any
- ..have been used in types of interventions similar to ones you have in mind, possibly in exemplary studies
- ..have been applied in domains close to the one you are working on, possibly in exemplary studies
- ..somehow inspire you, trigger new thoughts and help you connect the dots of your experience and knowledge in interesting ways

[9] https://ea-tel.eu/detel-book/chapter/tel-core/tel-overview/

- ..are applied by colleagues that you may consult and discuss with

Furthermore:

- Have in mind what aspects of learning a chosen theory or approach will *not* shed much light on
- Consider whether a combination of theories/frameworks could be viable and useful

Copyright

Learning Theories and their Use in Computing Education Research © 2026 by Birgit R. Krogstie is licensed under CC BY-NC-ND 4.0



References

Bandura, Albert. 1977. *Social Learning Theory*. Prentice Hall.

Bandura, Albert. 2006. "Guide for Constructing Self-Efficacy Scales." In *Self-Efficacy Beliefs of Adolescents*, edited by Pajares, Frank and Urdan, Timothy C. Information Age Publishing.

Batra, Rakhi, and Zahra Atiq. 2025. "Emotions and Self-Efficacy of Undergraduate Computing and Engineering Students: A Systematic Literature Review." Paper presented at SIGCSE TS.

Berry, Michael, and Michael Kölling. 2014. "The State of Play: A Notional Machine for Learning Programming." Paper presented at ITiCSE'14.

Boud, David, Rosemary Keogh, and David Walker. 1985. *Reflection: Turning Experience into Learning*. RoutledgeFalmer.

Bourdeau, Jacqueline, and Nicolas Balacheff. 2014. "Technology-Enhanced Learning : From Thesaurus and Dictionary to Ontology." In *Technological and Social Environments for Interactive Learning*, edited by Jelena Jovanovic and Raymond Chiong. Informing Science Press.

Bygstad, Bendik, Birgit R. Krogstie, and Tor Morten Grønli. 2009. "Learning from Achievement: Scaffolding Student Projects in Software Engineering." *International Journal of Networked and Virtual Organizations* 6 (2).

Dewey, J. 1986. "Experience and Education." *The Educational Forum* 50 (3): 241–52.

Engeström, Yrjö. 1987. *Learning by Expanding. An Activity-Theoretical Approach to Developmental Research*. Orienta-Konsultit Oy, Helsinki.

Ericson, Barbara, Paul Denny, James Prather, et al. 2022. "Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs." Paper presented at ITICSE-WGR, Dublin, Ireland.

Fischer, F., C. E. Hmelo-Silver, S. R. Goldman, and P. Reimann. 2018. *International Handbook of the Learning Sciences*. Routledge.

Goldie, John Gerard Scott. 2016. “Connectivism: A Knowledge Learning Theory for the Digital Age?” *Medical Teacher* 38 (10): 1064–69.

Guskey, Thomas R. 2007. “Closing Achievement Gaps: Revisiting Bejnamin S. Bloom’s ‘Learning for Mastery.’” *Journal for Advanced Academics* 19 (1).

Hutchins, Edwin. 1995. *Cognition in the Wild*. The MIT Press.

Jin, Fufen, and Monica Divitini. 2020. “Affinity for Technology and Teenagers’ Learning Intentions.” Paper presented at ICER ’20.

Kolb, D. A. 1984. *Experiential Learning: Experience as a Source of Learning and Development*. Prentice Hall.

Kratwohl, David R. 2002. “A Revision of Bloom’s Taxonomy: An Overview.” *Theory Into Practice* 41 (4): 212–18.

Krogstie, B. 2008. “Power through Brokering. OSS Participation in SE Projects.” Paper presented at International Conference on Software Engineering (ICSE) 2008. May 10.

Krogstie, Birgit R., and Monica Divitini. 2009. “Shared Timeline and Individual Experience: Supporting Retrospective Reflection in Student Software Engineering Teams.” Paper presented at Software Engineering Education and Training, 2009. CSEET ’09. 22nd Conference on.

Laal, Marjan, and Mozhgan Laal. 2012. “Collaborative Learning: What Is It?” *Procedia - Social and Behavioural Sciences* 31: 491–95.

Latour, Bruno. 2005. *Reassembling the Social. An Introduction to Actor-Network-Theory*. Oxford University Press.

Lave, Jean, and Etienne Wenger. 1991. *Situated Learning. Legitimate Peripheral Participation*. University of Cambridge Press.

Lundberg, Gunhild. 2016. “Nettbasert teamarbeid og betydningen av mestringstro.” Master of ICT-based cooperation, Norwegian University of Science and Technology.

Parsons, Dale, and Patricia Haden. 2006. “Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses.” 52.

Piaget, J. 1980. “The Psychogenesis of Knowledge and Its Epistemological Significance.” In *Language and Learning*, edited by M. Piattelli-Palmarini. Harvard University Press.

Plass, Jan L., and Slava Kalyuga. 2019. “Four Ways of Considering Emotion in Cognitive Load Theory.” *Educational Psychology Review* 31.

Robins, Anthony V. 2010. “Learning Edge Momentum: A New Account of Outcomes in CS1.” *Computer Science Education* 20 (1).

Rouhani, Majid, Miriam Lillebo, Veronica Farshchian, and Monica Divitini. 2022. “Learning to Program: An In-Service Teachers’ Perspective.” Paper presented at IEEE Global Engineering Education Conference (Educon). March 28.

Salomon, Gavriel. 1993. *Distributed Cognitions*. Cambridge University Press.

Sanders, Kate, and Robert McCartney. 2016. “Threshold Concepts in Computing: Past, Present, and Future.” Paper presented at Koli Calling ’16.

Saxegaard, Evelyn, and Monica Divitini. 2019. “CITY: A Game to Raise Girls’ Awareness About Information Technology.” Paper presented at ISSEP. *Informatics in Schools. New Ieas in School Informatics*.

Siemens, George. 2004. “Connectivism: A Learning Theory for the Digital Age.” *International Journal of Instructional Technology & Distance Learning* 2 (1).

Skinner, B. F. 1938. *The Behavior of Organisms: An Experimental Analysis*. B. F. Skinner Foundation.

Skinner, B. F. 1963. “Operant Behaviour.” *American Pshychologist* 18 (8): 503–15. https://doi.org/https://doi.org/10.1037/h0045185.

Stahl, Gerry, and Kai Hakkarainen. 2021. “Theories of CSCL.” In *International Handbook of Computer-Supported Collaborative Learning*. Springer Nature.

Stahl, Gerry, Timothy Koschmann, and D. Suthers. 2006. “Computer-Supported Collaborative Learning: An Historical Perspective.” In *Cambridge Handbook of the Learning Sciences*, edited by R. Keith Sawyer. Cambridge University Press.

Sweller, John, Paul Ayres, and Slava Kalyuga. 2011. *Cognitive Load Theory*. Springer. https://doi.org/10.1007/978-1-4419-8126-4.

Szabo, Claudia, Nickolas Falkner, Andrew Petersen, et al. 2019. “Review and Use of Learning Theories within Computer Science Education Research: Primer for Researchers and Practitioners.” Paper presented at ITiCSE-WGR ’19.

Szabo, Claudia, and Judy Sheard. 2022. “Learning Theories Use and Relationships in Computing Education Research.” *ACM Transactions on Computing Education* 23 (1). https://doi.org/10.1145/3487056.

Thaler, Richard H., and Cass R. Sunstein. 2008. *Nudge: Improving Decisions about Health, Wealth, and Happiness*. Yale University Press.

Thorndike, Edward. (1911) 2026. *Animal Intelligence: Experimental Studies*. Antiquum Novum Verlag UG.

Vygotsky, Lev S. 1978. *Mind in Society. The Development of Higher Psychological Processes.* Harvard University Press.

Wankhade, Mayur, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. 2022. “A Survey on Sentiment Analysis Methods, Applications, and Challenges.” *Artificial Intelligence Review*.

Wenger, Etienne. 1998. *Communities of Practice. Learning, Meaning, and Identity*. Learning in Doing: Social, Cognitive, and Computational Perspectives. Cambridge University Press.

Wing, Jeanette M. 2006. “Computational Thinking.” *Communications of the ACM* 49 (3).

Zimmermann, Barry J. 2002. “Becoming a Self-Regulated Learning: An Overview.” *Theory Into Practice* 41 (2): 64–70.

2. Designing Projects for Learning

Line Kolås, Helga Dís Ísfold Sigurðardóttir, and Aamna Rais Ahmed

Nord University

2.1. Project-based learning

Project-based learning (PjBL) is a well-known teaching method in IT/computing education. It helps prepare students for a future career by giving them the opportunity to work on authentic problems, collaborate with others, and work in meaningful contexts. However, research has also shown that high-quality project work requires more than authentic projects alone. Effective project-based learning depends on careful attention to cognitive development and socio-emotional learning (Sánchez-García & Angosto, 2026), stakeholder involvement (Naseer et al, 2025), and the overall structure through which projects are organized (Berggren et al, 2003). In addition, blended project-based learning is proposed as an anticipated trend for engineering education (Graham, 2018).

Edström & Kolmos (2014) use the acronym PBL for problem/project-based learning, but in this chapter, we use the acronym PjBL to emphasize the project work. Edström & Kolmos described how PBL is interpreted and applied in diverse ways, and that a range of definitions and perspectives have emerged. Also in Sindre et al (2018) the characteristics of project-based learning are described to include a large variety of project-based learning. Edström & Kolmos (2014) emphasize that historically seen, PBL should be recognized as a milestone for student-centred education while also preparing the ground for (more specific) models like CDIO. In our chapter we therefore compare four models and frameworks for project-based learning, not treating PBL as one specific approach like Edström & Kolmos (2014) did in their comparison of problem/project-based learning (PBL) and CDIO.

Based on experience, research and development, a number of models and frameworks have been developed to help teachers plan and implement project-based learning. Examples include the CDIO framework (CDIO office – Chalmers university of technology, 2026), the I-PBL framework (Naseer et al, 2025), the CPBL model (Sánchez-García & Angosto, 2026) and The Aalborg model of problem- and project-based learning (Aalborg University). Each framework contributes to a distinct but interconnected perspective on the design and facilitation of project-based learning. In this chapter, these four frameworks are presented and compared to help teachers design projects for learning. Then, based on own experiences and educational development work, we will present our experiences and reflections on blended project-based learning and use of external mentors in project-based learning. All subchapters conclude with a set of planning questions for teachers, intended to support teachers in the design of project courses in higher computing education.

2.2. Frameworks/models for PjBL

2.2.1. CDIO

The CDIO framework (Conceive-Design-Implement-Operate) provides an overall project architecture, both for student projects and for teachers' planning. Originally developed to connect academic learning with professional engineering practice, CDIO offers a logical process structure that guides students from the identification of a challenge through the development, implementation, and evaluation of solutions. Through its emphasis on authentic professional processes, CDIO helps students experience the complete lifecycle of projects and develop competencies associated with real-world practice (CDIO

office – Chalmers university of technology, 2026). For teachers, a central premise of CDIO is that curriculum design should begin with the competencies needed in professional practice. The adoption of project-based learning therefore emerges as a logical consequence of these educational goals, rather than as a pedagogical choice made independently of them. The CDIO framework consists of 12 standards, which are a set of principles/best practices underlying the implementation of CDIO in an engineering programme. CDIO's standard 5 prescribes "A curriculum that includes two or more design–implement experiences, including one at a basic level and one at an advanced level", indicating project-based learning activities. Standard 7, focusing on integrated learning experiences, means that students simultaneously develop disciplinary knowledge and professional engineering skills (Edström & Kolmos, 2014), which is relevant when planning PjBL.

2.2.2. Industry-Enriched PBL

The Industry-Enriched Project-Based Learning (I-PBL) framework strengthens the authenticity of project work by systematically incorporating external stakeholders throughout the project lifecycle. Through industry partnerships, mentoring, workplace interactions, client feedback, and real-world challenges, I-PBL creates opportunities for students to connect academic learning with societal and professional needs while developing competencies associated with employability and future career readiness (Naseer et al, 2025).

2.2.3. The Aalborg model of problem- and project-based learning

Since its founding in 1974, Aalborg University in Denmark has organized its educational programs around project work. The Aalborg model of problem- and project-based learning (Aalborg University) is a learning approach where students learn by collaboratively investigating authentic problems, thereby developing both disciplinary knowledge and professional identity. The six foundational principles for problem- and project-based learning at Aalborg University are: problem orientation; project organization; student-directed; collaboration; process orientation; and exemplarity. The model is guided by these six pedagogical principles, with particular emphasis on problem orientation and process orientation. The model focuses on how students engage in collaborative inquiry and knowledge construction throughout a project.

2.2.4. The Comprehensive PBL design model

The newly published Comprehensive PBL design model (CPBL) contributes a holistic understanding of learning by emphasizing the integration of three dimensions: procedural processes (what students do), cognitive activities (how students think), and socio-emotional activities (how students collaborate, communicate, and regulate their learning). Building on both the Gold Standard PBL model (Larmer et al, 2015) and a systematic analysis of contemporary PBL research, the CPBL model highlights the importance of intentionally designing learning experiences that foster critical thinking, creativity, metacognition, collaboration, self-awareness, emotional regulation, and responsible decision-making alongside project completion. By doing so, CPBL addresses common shortcomings in project work where procedural activities can overshadow deeper learning processes (Sánchez-García & Angosto, 2026).

2.3. Comparing the frameworks/models

While CDIO provides a structured project lifecycle through the phases Conceive, Design, Implement, and Operate, the other models are not organized around predefined project stages. Taken together, these

four frameworks create a comprehensive foundation for designing project-based learning in higher education. In this chapter, CDIO provides the project architecture, I-PBL provides authentic stakeholder engagement, CPBL ensures balanced procedural, cognitive and socio-emotional development, and the Aalborg model provides a pedagogical approach to learning within an architecture like CDIO. The combination of these perspectives enables teachers to move beyond viewing projects simply as instructional activities and instead design learning experiences that simultaneously support disciplinary knowledge, professional competence, reflective practice, socio-emotional growth, and future career readiness. This aligns with the broader educational aspirations of both PBL and CDIO, which emphasize not only knowledge acquisition but also students' skill development and personal growth as part of becoming professionals (Edström & Kolmos, 2014).

Table 1. Comparison of four frameworks for project-based learning, using the CDIO phases as an organizing lens.

CDIO phase	***CDIO***	***I-PBL***	***Aalborg PBL model***	***CPBL***
Conceive	Identify and understand problems, needs, context, and requirements	Authentic industry challenges, stakeholder needs, and problem definition	Problem orientation; student involvement in defining and investigating authentic problems; exemplarity	Cognitive engagement in problem understanding; motivation, self-awareness, and collaborative goal-setting
Design	Explore alternatives; develop and evaluate solution concepts	Industry expertise, professional constraints, co-design, and formative feedback	Student-directed inquiry, collaborative knowledge construction, and project organization	Critical thinking, creativity, problem-solving, collaboration, and decision-making
Implement	Develop, build, test, and integrate solutions	Professional practices, industry mentoring, iterative feedback, and application in authentic contexts	Collaborative project work, student responsibility, ongoing inquiry, and reflection on the learning process	Application of knowledge, adaptability, teamwork, self-regulation, and iterative problem-solving
Operate	Apply, demonstrate, and evaluate solutions in relevant contexts	Industry evaluation, stakeholder feedback, professional relevance, and career readiness	Evaluation of project outcomes, reflection on processes and learning, and transfer of knowledge through exemplarity	Metacognition, reflection, socio-emotional development, and evaluation of learning

Note. The CDIO phases are used as an organizing lens, not as phases prescribed by the other three frameworks. The Aalborg principles, CPBL learning dimensions, and I-PBL stakeholder engagement can be relevant throughout the entire project lifecycle. The entries illustrate possible pedagogical emphasis rather than exclusive or sequential activities.

A useful way to compare these four frameworks is to treat CDIO's four phases (Conceive, Design, Implement, Operate) as an organizing lens. We apply this lens in the following sections, not assuming that the other three models literally use the four phases, but based on the assumption that a typical project will go through somewhat similar stages.

2.3.1. Conceive: Identifying meaningful problems and opportunities

The Conceive phase focuses on identifying authentic challenges, understanding stakeholder needs, and establishing the purpose of a project. Decisions made during this phase influence student motivation, engagement, and the quality of subsequent learning.

From an I-PBL perspective, teachers should seek to anchor projects in authentic problems connected to industry needs and real-world professional contexts. Industry partners can help identify relevant problems and expose students to organizational priorities and workplace expectations. Early stakeholder involvement helps students understand the relevance of the challenge and provides insight into the context in which their solutions may eventually be applied.

From the perspective of the Aalborg model of problem- and project-based learning, the Conceive phase should provide opportunities for students to explore and develop an understanding of the problem rather than simply receive a predefined project assignment. Problem orientation is a central principle, emphasizing that learning should be driven by investigating meaningful problems that provide a context for acquiring disciplinary knowledge. Student-directed learning implies that students should have opportunities to influence how the problem is understood, formulated, and investigated, while teachers provide appropriate guidance and scaffolding. Through collaborative discussions, preliminary investigations, and dialogue with relevant stakeholders, students can develop a shared understanding of the problem and identify questions that will guide their subsequent project work. The principle of exemplarity also encourages teachers to consider how investigating a specific problem can help students develop knowledge and competencies that are transferable to broader disciplinary and professional contexts. Thus, when planning the Conceive phase, teachers should consider not only whether a problem is authentic and engaging, but also whether it provides sufficient opportunities for student inquiry, ownership, collaboration, and meaningful learning (Aalborg University, 2025).

The CPBL model contributes an explicit focus on both cognitive and socio-emotional processes. Teachers should support students in developing empathy for stakeholders, identifying assumptions, asking critical questions, and exploring multiple perspectives on the problem. Activities such as stakeholder interviews, field observations, and problem-mapping exercises can help students connect emotionally and intellectually with the challenge.

Planning questions for teachers

- What disciplinary knowledge and competencies should students develop?
- What forms of thinking, such as problem solving, critical thinking, creativity, or decision-making should the project require?
- What challenging problems or questions can drive this learning?
- Could industry or other external stakeholders contribute problems, expertise, feedback, or constraints?
- Why is the problem authentic and meaningful, and to whom?

- How will student voice and choice be incorporated?
- How can students participate in defining and investigating the problem?
- How can the specific problem support learning that is transferable to other contexts?
- How will students investigate stakeholder needs?
- What scaffolding will students need in the startup of a project?
- Does the project provide equitable opportunities for all students to take on technically challenging tasks and responsibilities? (Naseer et al, 2025).
- By the end of the Conceive phase, what should students understand about the problem, stakeholders, requirements, and criteria for success before moving into the Design phase?

2.3.2. Design: Developing ideas and project plans

During the Design phase, students move from understanding the problem toward developing possible solutions and planning their work.

The CDIO framework emphasizes systematic design processes, while I-PBL recommends involving industry partners and stakeholders as mentors during this phase. External experts can provide feedback on project concepts, help identify constraints, and contribute professional perspectives that improve the feasibility and relevance of student proposals.

The Aalborg model emphasizes student-directed learning, collaboration, and process orientation throughout project work. During the Design phase, these principles suggest that students should take an active role in exploring possible solutions, making and justifying design decisions, and planning how their project will be carried out. Rather than prescribing a specific solution or process, teachers should provide guidance and scaffolding that enable students to investigate alternatives, apply disciplinary knowledge, and make informed choices. Collaborative discussions and negotiation of ideas can help students develop shared understanding and responsibility for the chosen direction. Process orientation also highlights the importance of reflecting on how the group organizes its work, makes decisions, and responds to challenges. Teachers can support this through supervision, project planning activities, and opportunities for students to reflect on both their design choices and their collaborative processes (Aalborg University, 2025).

CPBL suggests that teachers explicitly scaffold cognitive development through activities that promote creativity, problem-solving, critical analysis, and metacognitive thinking. At the same time, socio-emotional competencies such as communication, collaboration, negotiation, and leadership should be intentionally supported through group processes and structured teamwork.

Planning questions for teachers

- What disciplinary knowledge and competencies should students apply during the Design phase?
- How will students investigate, develop, and compare possible solutions?
- Which design decisions should students make themselves, and what guidance will they need?
- How will students justify their choices using evidence and disciplinary knowledge?

- What tools, methods, examples, or scaffolding will support creative exploration and critical thinking?
- How will students organize their project work, collaborate, and negotiate design decisions?
- When and how will students receive feedback from teachers, peers, industry partners, or other stakeholders?
- How will students document and reflect on their design decisions and collaborative processes?
- What should students have demonstrated before moving into the Implement phase?

2.3.3. Implement: Creating, testing, and improving solutions

The Implement phase focuses on transforming plans into tangible outcomes through prototyping, production, testing, and revision.

Within I-PBL, stakeholders become active participants who provide coaching, mentoring, and feedback throughout development. Workplace visits and exposure to professional tools, standards, and ways of working can be valuable for students. Authentic feedback loops expose students to professional expectations and encourage continuous improvement. I-PBL emphasizes sustained mentor relationships, for example through regular project milestones involving industry feedback rather than isolated guest lectures. Naseer et al. (2025) reported particularly notable improvements in technical self-efficacy among female students participating in I-PBL, suggesting that authentic project experiences and industry interaction may help strengthen students' confidence in their technical abilities.

The Aalborg model emphasizes student responsibility for organizing and carrying out collaborative project work. During the Implement phase, this means providing students with opportunities to manage their project activities, distribute responsibilities, address emerging challenges, and make decisions about how to proceed. The principle of process orientation highlights that learning takes place not only through developing a solution but also through reflecting on how the group works, responds to difficulties, and adapts its approach. Teachers and supervisors play an important role in supporting these processes through academic guidance, feedback, and facilitation of group reflection, while allowing students to retain ownership of their project. In this way, implementation becomes an opportunity to develop disciplinary knowledge alongside competencies in collaboration, project management, and independent problem-solving (Aalborg University, 2025).

CPBL emphasizes that implementation should not focus exclusively on producing project outcomes. Teachers must also support reflection on learning processes. Metacognitive prompts, progress reviews, and collaborative reflection sessions can help students understand why particular decisions are made and how challenges are addressed. Socio-emotional dimensions such as resilience, emotional regulation, teamwork, and conflict management become especially important when projects encounter difficulties (Sánchez-García & Angosto, 2026).

Planning questions for teachers

- What will students create, develop, test, or put into practice, and which disciplinary and practical competencies should they develop?
- In what authentic or realistic contexts can students test and evaluate their solutions?

- How will students organize their project work, distribute responsibilities, and take ownership of implementation?
- How much iteration will the project allow, and how will testing, feedback, and revision cycles be organized?
- When and how will students receive formative feedback from teachers, peers, industry mentors, or other stakeholders?
- How will students document their decisions, reflect on their progress, and learn from unsuccessful attempts?
- How will teachers support collaboration, individual contributions, conflict management, and challenges encountered during project work?
- How to use industry-enriched PBL activities to deliberately support female students' confidence in their technical abilities? (Naseer et al., 2025)
- How will both project progress and student learning be assessed, rather than focusing only on the final product?
- What should students have demonstrated before moving into the Operate phase?

2.3.4. Operate: Presenting, evaluating, and reflecting on impact

The Operate phase extends beyond project completion by focusing on the use, evaluation, and broader implications of project outcomes.

I-PBL emphasizes authentic dissemination opportunities and stakeholder evaluation. Presenting solutions to clients, organizations, industry representatives, community members, or public audiences help students understand the real-world value of their work and provides opportunities to develop professional communication skills and professional identity (Naseer et al., 2025).

The Aalborg model of problem- and project-based learning emphasizes the principle of exemplarity, meaning that students should develop knowledge and competencies through investigating a specific problem that can also be applied in broader contexts. During the Operate phase, teachers can encourage students to consider how the insights gained from their project relate to other problems, disciplinary concepts, and professional situations. Process orientation further highlights the importance of reflecting on how the project was carried out, including the decisions made, challenges encountered, and ways in which the group collaborated. Students should take an active role in evaluating their project outcomes and identifying what they would approach differently in future projects, supported by guidance and feedback from teachers and supervisors. In this way, the Operate phase provides opportunities to connect project outcomes with broader learning and professional development (Aalborg University, 2025).

CPBL contributes a focus on transfer of learning, responsible decision-making, self-awareness, and socio-emotional growth. Reflection activities should encourage students to consider not only what they produced but also what they learned about themselves, their team, the problem, and the professional or societal context (Sánchez-García & Angosto, 2026).

Planning questions for teachers

- Who are the intended users or audiences for the project outcomes, and how will students communicate and demonstrate their solutions?
- How will students evaluate whether their solutions address the original problem and stakeholder needs?
- What criteria will be used to assess the quality, performance, and broader implications of the project outcomes?
- How can industry partners, users, or other external stakeholders contribute to authentic evaluation and feedback?
- How will students present, explain, and justify their work to relevant audiences?
- How will students reflect upon their disciplinary knowledge, problem-solving processes, collaboration, and professional development?
- How can students identify knowledge and competencies gained from the specific project that are applicable to other problems or contexts?
- How will students evaluate their project processes and identify what they would improve if the project continued?
- How can learning from this project be connected to subsequent courses, future projects, or professional practice?

2.4. Experiences and reflections on designing PjBL

In the following sub-chapters, we draw on our experiences with project-based learning in higher computing education, with particular attention to blended project-based learning and the involvement of external mentors. We will discuss our experiences in relation to the frameworks / models for PjBL presented above, and conclude with planning questions for teachers, which we hope can be valuable teaching resources for anyone planning project courses in higher computing education.

2.4.1. Blended project-based learning

Blended learning has become an increasingly relevant approach in computing education, offering opportunities to combine the flexibility of online learning with the collaborative and experiential qualities of face-to-face (F2F) activities. Picciano et al. (2018) defines blended learning as the strategic combination of online and in-person learning, emphasizing that the value of blended learning lies not merely in combing the learning modes, but using them purposefully to support learning goals. Within computing education, blended learning has gained increasing attention as institutions seek to provide flexible learning opportunities while maintaining authentic and practice-oriented learning experiences (Rais Ahmed & Kolås, 2026). O'Connor et al. (2023), in their systematic review of CDIO research, highlighted blended and hybrid learning as important areas for further development within the CDIO framework. They argue that while CDIO has traditionally emphasized active, hands-on, project-based learning, it now needs to adapt to the growing role of digital and hybrid education.

Blended project-based learning (BPBL) represents one response to this challenge. BPBL refers to the use of online and F2F learning environments within a project-based pedagogical approach (Barbosa, 2022). In project-based courses, blended learning can be organized in many ways, and Robles Dimaano (2021) describes twelve models that differ in how online and in-person activities are combined. One widely used model is the flipped classroom, in which students engage with content, such as lecture videos or readings, online before class, and F2F time is used for discussion, problem solving, and

applied work (Robles Dimaano, 2021; Veras et al., 2020). Our bibliometric study examining 34 studies of BPBL in undergraduate computing education, found that the flipped classroom approach was the approach most often implemented with project-based learning, while the integration of online and F2F activities was often insufficiently described. The literature primarily emphasizes students' experiences and quantitatively measured learning outcomes, with comparatively limited attention given to the perspectives of teachers and industry partners involved in project-based activities (Rais Ahmed & Kolås, 2026). This highlights the need to move beyond viewing blended project-based learning primarily as a simple combination of delivery modes and instead consider how learning environments can be deliberately designed to support project-based learning.

2.4.2. Different BPBL project phases have different needs

Drawing on our experiences with project-based learning in computing education, including student feedback, a round table discussion at Læringsfestivalen, and educational development within the Excited Centre for Excellent IT Education, we view blended learning an additional dimension of pedagogical design. The central question is not simply which activities belong online, and which belong F2F, but how different learning environments can best support students' learning, collaboration, and project progression. In this perspective, decisions about learning environments should be guided by the pedagogical purpose of activities, the needs of participants, and the intended learning outcomes. Recent research provides empirical support for this position by demonstrating that different phases of inquiry/project work require different forms of interaction and instructional support. In a study of 232 students across ten blended project-based computing courses in Norway, Sweden, and Finland, activities associated with exploration were particularly dependent on interaction among students, whereas activities related to framing project challenges, integrating ideas, and validating solutions were more strongly associated with instructional guidance and feedback (Rais Ahmed et al., in press). These findings can be understood through the Community of Inquiry framework, which conceptualizes learning as progression through four phases: triggering event, exploration, integration, and resolution (Swan et al., 2008). The results suggest that exploration activities benefit especially from opportunities for F2F peer interaction and collaborative meaning-making, whereas guidance and feedback can be facilitated online. This implies that decisions regarding online and F2F learning environments should be informed not only by logistical considerations but also by the pedagogical requirements of different inquiry activities (Rais Ahmed et al., in press).

In our own teaching practice, we have found it useful to distinguish between activities that benefit from direct interpersonal interaction and those where digital environments provide greater flexibility. F2F meetings can be particularly valuable when students establish project teams, develop a shared understanding of the project, explore ideas, and negotiate decisions. Such activities often involve dialogue, spontaneous interaction, and collaborative problem-solving. Online environments, on the other hand, can support ongoing coordination, documentation, communication, and access to learning resources throughout the project. Digital collaboration tools also enable students to continue working together between scheduled teaching activities.

2.4.3. Preparing students for blended work environments

An additional consideration in BPBL is how students are prepared for participation in professional work environments. Students in computing disciplines are expected to work in distributed teams, collaborate across digital platforms, and engage with colleagues, clients, and stakeholders through a combination of F2F and online interactions (Pohan & Maulina, 2022). Project-based learning is known to support

both subject-specific learning and skills needed to work in professional environments such as collaboration, communication and problem solving (Naseer et al, 2025). In addition to these skills, as per our experience, project-based courses can also help students develop skills needed to function effectively across physical and digital work environments. The literature describes this as *blended readiness*, and it is the ability to navigate learning and work across multiple environments, selecting communication channels, collaboration practices and digital tools to achieve shared goals (Barbosa, 2022).

However, blended readiness is more than just basic digital literacy and selecting the right tools for communication and collaboration. It also involves a combination of communication skills, self-regulation, ability to collaborate and technological adaptability (Pohan & Maulina, 2022). Blended project work requires students to coordinate and communicate effectively. These interactions take place across different communication channels and may involve peers, teachers, and external professionals. Teams commonly use a combination of F2F meetings, online collaboration platforms, messaging tools, shared documentation systems, and project repositories. Our teaching experience suggests that students must learn not only how to use digital tools, but also how to establish communication practices, negotiate responsibilities, document decisions, and maintain productive collaboration over time. Teachers can support the development of these competencies in complementary ways. For example, using F2F activities to provide opportunities for relationship building, trust development, collaborative exploration, and negotiation of ideas; and scaffolding the use of online environments to support ongoing communication, documentation, project coordination, resource sharing, and asynchronous collaboration. Such support is consistent with the emphasis on collaboration, reflection, and the integration of cognitive and socio-emotional dimension in CPBL (Sánchez-García & Angosto, 2026) Moving between these environments encourages students to develop flexible communication and collaboration strategies while experiencing different forms of participation and engagement.

2.4.4. Online involvement of external stakeholders in BPBL

The development of blended readiness becomes particularly important when projects involve external stakeholders. Blended learning environments enable industry representatives to participate in project activities. They may contribute to authentic problems, professional expertise, mentoring, and feedback, but their availability for campus-based activities can be limited. Discussions with computing educators at Læringsfestivalen similarly emphasized the practical challenges of sustaining industry involvement throughout project courses. Digital collaboration platforms provide opportunities to reduce these barriers by enabling external partners to contribute through online meetings in students' projects without requiring their physical presence throughout the project. In our experience, combining an initial F2F encounter, when feasible, with subsequent online meetings can be a practical way of establishing relationships while maintaining regular contact. This approach aligns with the emphasis on industry engagement in I-PBL (Naseer et al., 2025) and the importance of external support in project-based computing education (Suleiman et al., 2025).

In our courses, which we will discuss in the following subchapter, we observed that students experience interacting with professionals beyond the classroom while learning to adapt their communication and collaboration practices to diverse stakeholder groups. Such interactions supported valuable preparation for professional work settings where collaboration frequently extends across organizational, geographical, and disciplinary boundaries. BPBL has the potential to contribute not only to disciplinary knowledge and project outcomes but also to the development of transferable competencies valued in contemporary workplaces. Therefore, teachers must also consider how different learning environments

support inquiry, collaboration, communication, stakeholder engagement, and the development of blended-ready graduates. Making these decisions requires a systematic approach to pedagogical design.

2.4.5. Pedagogical design dimensions

The CDIO framework provides a useful structure for considering how blended learning environments can support different activities throughout the project lifecycle. While CDIO organizes project work around the phases of Conceive, Design, Implement, and Operate, BPBL adds a complementary design consideration: how these learning activities and interactions can be organized across F2F and online environments. Drawing on our teaching experiences, we suggest that teachers consider three interrelated dimensions when designing blended project activities: the project phase, the pedagogical purpose of the activity, and the learning environment. Rather than assigning entire CDIO phases to either online or F2F settings, teachers can use these dimensions to make deliberate choices about how different activities should be organized.

Table 2 illustrates how this perspective can be applied across the CDIO lifecycle. The examples are intended as practical design possibilities rather than fixed recommendations. They demonstrate how F2F and online environments can complement one another in supporting collaboration, guidance, stakeholder involvement, and student learning. Since project-based learning is iterative, the same types of activities may occur across several phases, and the most appropriate learning environment will depend on the project context and intended learning outcomes.

2.4.6. BPBL in relation to other frameworks

From a pedagogical design perspective, BPBL complements the frameworks discussed earlier in this chapter. CDIO provides a structure for organizing project activities throughout the project lifecycle, I-PBL emphasizes the involvement of industry and external stakeholders, CPBL highlights the procedural, cognitive, and socio-emotional dimensions of meaningful project-based learning, and the Aalborg PBL model emphasizes student-directed learning, problem orientation, collaboration, process orientation, and the development of transferable knowledge and competencies through exemplarity. BPBL complements these frameworks by focusing on how digital and physical learning environments, resources, and interactions can be intentionally combined to support student learning throughout the project. BPBL adds the consideration of how learning activities and interactions can be distributed and connected across physical and digital learning environments. The objective is not to maximize the use of technology, but to create a coherent learning experience in which different environments contribute to the overall pedagogical purpose of the project, but also to prepare students for blended projects in working life.

In summary, blended project-based learning requires deliberate decisions about the relationship between project activities, learning environments, participants, and learning outcomes. Based on our experiences, we recommend approaching blended learning as an integrated part of project design rather than as an additional technological component. By considering when and how F2F and online activities can complement one another, teachers can support flexibility, collaboration, authentic stakeholder engagement, and continuous learning throughout the project lifecycle.

Table 2. Examples of aligning CDIO phases, project activities, pedagogical purposes, and learning environments in blended project-based learning**.**

CDIO phase	**Project activities**	**Pedagogical rationale**	**Possible learning environments**
Conceive	Team formation, problem exploration, stakeholder dialogue, project goals	Establish shared understanding, support relationship building, clarify expectations, and initiate inquiry	F2F kick-off meetings and workshops, complemented by online resources and stakeholder communication
Design	Brainstorming, exploring alternatives, prototyping, design reviews, feedback	Support creative exploration, negotiation of ideas, collaborative decision-making, and iterative feedback	F2F collaborative workshops combined with online design tools, shared documents, and feedback meetings
Implement	Development, testing, project management, documentation, iterative improvement	Support continuous collaboration, coordination, practical problem-solving, and timely guidance	Online collaboration platforms, repositories, and project management tools, complemented by F2F supervision and teamwork
Operate	Demonstrations, stakeholder evaluation presentation of project outcomes	Support authentic evaluation, communication of results, stakeholder engagement, and reflection on learning	F2F or online demonstrations, stakeholder presentations, feedback sessions, and reflection activities

Note. F2F=face-to-face. The examples reflect an experience-informed pedagogical design perspective.

Planning questions for teachers: Blended Project-Based Learning

- How can online and F2F learning activities be combined to support students' learning and project progress across the CDIO phases?
- What competencies do students need for communication, collaboration, and feedback across online and F2F environments, and how can teachers support their development?
- Who needs to participate in each activity, including students, teachers, and external stakeholders?
- Which activities benefit from F2F interaction, which can be conducted online, and how can the two environments complement one another?
- Are there opportunities for F2F peer interaction and collaborative meaning-making in the exploration activities?
- How can online participation make industry involvement more feasible?
- Where is F2F interaction important for building relationships and trust?
- What blended communication and feedback competencies will students need?
- How to support students in developing these competencies?

2.4.7. External mentors as a design resource in PjBL

Game Lab has been a learning concept that has run throughout all three years of the Bachelor's degree programme in Games and Entertainment Technology at Nord University. Game Lab has functioned as a company simulator, and students have completed one Game Lab project each semester. This organization closely resembles the logic of the CDIO framework, where students repeatedly move through cycles of conceiving, designing, implementing, and evaluating solutions in authentic project settings. Across six consecutive Game Lab experiences, students have gradually encountered increasingly complex design challenges and professional responsibilities.

Students have worked in project teams to develop their products, either complete games or prototypes. Through group collaboration, guidance from faculty members, and mentoring from external industry experts, each team has developed a game. Each of the six Game Labs has had a different focus, e.g. retro games, serious games. Game Lab gives students insight into professional working life from the very beginning of their studies. The concept is mostly free of traditional classroom teaching. At the start of each project, students pitch three different ideas. Faculty members, together with one or more external mentors, select the idea they believe has the greatest potential. The teams then assign roles, begin development, and manage the project themselves. Team leaders meet weekly with their supervisors (teachers) and an external mentor, and the semester concludes with students presenting their products to one another. Game Lab is closely integrated with the rest of the study programme. Alongside their project work, students take courses in areas such as programming, drawing, 3D modelling, and game design.

The case discussed here is Game Lab 3 (10 ECTS), a project-based course that has focused on the design and development of serious games and gamified solutions. Students have worked in teams to address authentic challenges in areas such as mental healthcare, sustainability and educational topics, combining theoretical perspectives on serious games with the design, development, and evaluation of game-based artefacts. In addition to the external mentor from the game industry, the students of Game Lab 3 have, when relevant, also had a mentor with expertise within the field their game is based around, such as recovery from mental health challenges. While the industry mentor has been the same person across all the student teams, each team has gotten their own mental health expert mentor. Throughout the semester, students have engaged in iterative development processes supported by faculty supervisors and external mentors.

2.4.8. The professional legitimacy of external mentors

The involvement of external mentors in the game lab 3 course reflects a core principle of the I-PBL framework, which emphasizes the integration of external stakeholders as mentors and active contributors to students' learning processes rather than merely end-users or evaluators of project outcomes. A connection to the Aalborg model is the principle of exemplarity, which emphasizes that learning from a specific problem or project should enable students to develop knowledge and competencies that can be applied in broader contexts (Aalborg University, 2025). External mentors can support this process by helping students connect the specific challenges encountered in their projects to more general principles, professional practices, and experiences from industry. For example, discussions about a particular game design decision, technical implementation, or collaborative challenge can be used to explore broader questions concerning software development, user needs, project management, or professional responsibility. In this way, mentoring can help students move beyond solving the immediate project problem towards developing transferable knowledge and a deeper understanding of professional practice.

We also experience that due to their industry background and experience, external mentors are often perceived by students as highly credible and trustworthy sources of professional knowledge, particularly in comparison with university teachers whose expertise may be viewed as more academic than practice oriented. External mentors bring professional legitimacy to the learning environment.

2.4.9. External mentors as facilitators of cognitive learning

While PjBL is often justified through authenticity and professional relevance, our experiences from the Game Lab 3 course suggest that external mentors contribute to student learning in ways that extend beyond industry engagement alone. Viewed through the lens of the CPBL design model, external mentors can serve as important design facilitators ("elements"), supporting both cognitive and socio-emotional development throughout the project lifecycle.

The CPBL model argues that PjBL should be intentionally designed to foster not only procedural skills and project completion, but also critical thinking, creativity, metacognition, collaboration, self-awareness, emotional regulation, and responsible decision-making (Sánchez-García & Angosto, 2026). In practice, however, project work can easily become dominated by production tasks and deadlines. Students focus on completing a video game, delivering a product, or satisfying assignment requirements, while deeper learning processes receive less attention. One approach we have found valuable is the systematic involvement of external mentors. In our game lab courses, the mentors typically fall into two categories. Some are game developers with experience from industry, while others contribute specialist knowledge related to the project domain. In projects focused on mental health serious games, for example, each student team has worked with two different mentors, possessing professional experience relevant to the themes being explored.

Although these mentors provide valuable expertise, their most important contribution may be the questions they ask rather than the answers they provide. Encounters with external mentors often require students to explain and defend design decisions, respond to critique, reconsider assumptions, and reflect on the consequences of different solutions. Such interactions stimulate critical thinking, metacognition, and responsible decision-making, all central components of the CPBL framework. From an I-PBL perspective, these interactions help create the reciprocal dialogue between students and stakeholders that supports authentic problem exploration and solution development. The principle of process orientation in the Aalborg PBL model (Aalborg University, 2025) emphasizes that learning involves not only producing a final artefact but also reflecting on the processes through which the work is carried out, including communication, shared decision-making, knowledge sharing, and constructive feedback. Mentoring sessions can therefore be designed as opportunities for students to reflect on both their technical decisions and their collaborative processes. This perspective reinforces the importance of designing mentoring interactions that support learning throughout the project rather than focusing exclusively on the quality of the final product. From a CDIO perspective, mentor feedback mirrors the iterative review processes that characterize professional design and implementation work.

2.4.10. External mentors as facilitators of socio-emotional learning

Mentor interactions also support socio-emotional learning. Students must learn to present unfinished work, receive criticism without becoming defensive, communicate with professionals, and navigate situations where different stakeholders hold different opinions. These experiences often require emotional regulation, perspective-taking, and collaboration. The mental health expert mentors have challenged students to engage sensitively with complex and sometimes emotionally demanding topics, encouraging empathy, ethical reflection, and a deeper understanding of the people for whom they are

designing. Furthermore, repeated encounters with external professionals help students develop confidence and begin to see themselves as future practitioners rather than solely as students.

Interestingly, many students describe communication, feedback, teamwork, and self-development as some of their most important learning outcomes, often placing greater emphasis on these experiences than on the technical artefacts they produced. This suggests that the most significant contribution of external mentors may not be best measured through project outcomes, but rather through their ability to create opportunities for the deeper cognitive and socio-emotional learning processes emphasized by CPBL, the stakeholder engagement advocated by I-PBL, and the authentic professional learning experiences that underpin CDIO. From the perspective of the Aalborg model, external mentors can further support student-directed learning, collaboration, process reflection, and the development of knowledge and competencies that extend beyond the immediate project. Together, these perspectives highlight the importance of treating external mentoring as an intentional pedagogical design resource rather than simply a means of providing technical expertise or professional feedback.

Planning Questions for Teachers: External mentors

- What learning purpose should the external mentor serve beyond providing expertise?
- What type of mentor: industry professional, domain expert, user, or other stakeholder, best supports this purpose?
- At which stages of the project will mentor involvement have the greatest value?
- How will mentors be prepared for their pedagogical role and the intended learning outcomes?
- How can mentors be encouraged to ask questions to stimulate critical thinking, reflection, and reconsideration of assumptions?
- How can mentors provide feedback without taking ownership of students' decisions?
- How will students be prepared to present unfinished work, receive critique, and respond professionally to feedback?
- How can teachers scaffold the students' learning process about handling conflicting advice from teachers, mentors, and other stakeholders?
- What role, if any, should external mentors play in the final evaluation of students' work?
- How will students reflect on and make use of what they learn through mentor interactions?
- Have you scheduled industry touchpoints at ca.5 distinct interactions per academic session? (Naseer et al, 2025).
- Did you plan for *sustained* mentorship relationships with mentors from industry? (Naseer et al, 2025).
- Do you have formal partnership agreements that clarify expectations for both educational institutions and industry partners? (Naseer et al, 2025).

2.5. Limitations

This chapter does not aim to provide a comprehensive review of all project-based learning approaches used in higher education. Instead, it focuses on four frameworks, that align with the authors' experiences from course design and teaching practice, and where especially CDIO and the Aalborg model have been influential in higher computing education. Other frameworks and educational contexts may offer additional perspectives on project-based learning. Furthermore, the examples of blended learning and external mentoring presented in the chapter are intended as illustrative design considerations rather than

evidence of universally effective practices. Consequently, the planning questions should be adapted to the specific context, discipline, and learning objectives of individual courses.

It is important to acknowledge that CDIO was primarily developed as a framework for curriculum development and outcomes-based programme design, rather than as a pedagogical model for individual courses (Edström & Kolmos, 2014). Consequently, its applicability to the design of a single project-based course may be limited, although its principles can still provide useful guidance for structuring project activities and aligning them with intended learning outcomes.

2.6. Designing projects for learning: Concluding remarks

Project-based learning is widely recognized as a valuable approach for preparing students for professional practice. However, designing projects for learning involves more than assigning students a project and expecting learning to emerge naturally. Across the frameworks discussed in this chapter, a common message is that meaningful project-based learning requires deliberate pedagogical design. Authentic problems, stakeholder engagement, collaboration, reflection, and opportunities for procedural, cognitive and socio-emotional development need to be planned and supported throughout the project lifecycle.

The four frameworks presented in this chapter highlight complementary aspects of project-based learning. CDIO provides a structure for organizing project activities across the lifecycle of a project. The Aalborg model emphasizes student-directed inquiry, collaboration, and learning through engagement with authentic problems. I-PBL highlights the value of involving external stakeholders, while CPBL draws attention to the cognitive, socio-emotional, and procedural dimensions of learning that can otherwise remain implicit. Together, these perspectives illustrate that project-based learning is not simply about completing a project, but about designing learning experiences that support the development of knowledge, skills, professional identity, and personal growth.

Our reflections on blended project-based learning and the use of external mentors suggest that these should be viewed as pedagogical design resources rather than separate educational initiatives. Blended learning provides opportunities to purposefully combine online and F2F environments, while external mentors can contribute authenticity, challenge students' assumptions, and support reflection and professional development. Both approaches are most effective when aligned with the intended learning outcomes and integrated into the overall project design.

Ultimately, the difference between project-based learning and merely "doing a project" lies in intentional design. By systematically considering project structures, learning processes, stakeholder involvement, learning environments, and opportunities for reflection, teachers can create project experiences that contribute not only to project outcomes, but also to student learning. The planning questions presented throughout this chapter are intended as practical resources to support such design processes.

Copyright

Designing Projects for Learning © 2026 by Line Kolås, Helga Dís Ísfold Sigurðardóttir and Aamna Rais Ahmed is licensed under CC BY-NC-ND 4.0



References

Aalborg University. Aalborg University Principles for Problem- and Project-Based Learning (PBL). https://www.aaudxp-cms.aau.dk/media/oecfn4iu/pbl_folder_uk_digital.pdf

Rais-Ahmed, A., Kolås, L, & Holand, A.M. (in press). Using Inquiry Phases to Inform the Design of Blended Project-Based Courses. Proceedings of the UDIT conference 2026.

Barbosa, M. W. (2022). Using blended project-based learning to teach project management to software engineering students. *International Journal of Mobile and Blended Learning (IJMBL)*, *14*(1), 1–17. https://doi.org/10.4018/IJMBL.291978

Berggren, K.-F., Brodeur, D. R., Crawley, E. F., Ingemarsson, I., Litant, W. T. G., Malmqvist, J., Östlund, S. (2003). *CDIO: An International Initiative for Reforming Engineering Education.* https://cdio.org/sites/default/files/docs/file/cdio_international.pdf

CDIO office – Chalmers university of technology (2026, Oct 15) *What is a CDIO-based education? https://cdio.org/cdio-vision & https://cdio.org/content/cdio-standards-30*

Dimaano, M. R. (2022). Students' experience on blended learning approaches. In *Proceedings of the 5th International Conference on Digital Technology in Education* (pp. 67–75). Association for Computing Machinery. https://doi.org/10.1145/3488466.3488472

Edström, K., & Kolmos, A. (2014). PBL and CDIO: complementary models for engineering education development. European Journal of Engineering Education, 39(5), 539–555. https://doi.org/10.1080/03043797.2014.895703

Graham, R. 2018. *The Global State of the Art in Engineering Education*. In Massachusetts Institute of Technology (MIT). ISBN 13: 9780692089200. https://www.rhgraham.org/resources/Global-state-of-the-art-in-engineering-education---March-2018.pdf

Larmer, J., Mergendoller, J. R., & Boss, S. (2015). *Setting the Standard for Project Based Learning*. ASCD. ISBN: 978-1-4166-2033-4

Naseer, F., Tariq, R., Alshahrani, H. M., Alruwais, N., & Al-Wesabi, F. N. (2025). Project based learning framework integrating industry collaboration to enhance student future readiness in higher education. *Scientific Reports*, *15*(1), 24985–24985. https://doi.org/10.1038/s41598-025-10385-4

O'Connor, S., Power, J., & Blom, N. (2023). A systematic review of CDIO knowledge library publications (2010 – 2020): An Overview of trends and recommendations for future research. *Australasian Journal of Engineering Education*, 28(2), 166–180. https://doi.org/10.1080/22054952.2023.2220265

Pohan, L., & Maulina, J. (2022). Blended Learning Integrated with Project-Based Learning: Its Effect on Learning Outcomes, Perception, and Self-Regulated Learning. *Jurnal Pendidikan Kimia Indonesia*, *6*(2), 97-106.

Rais Ahmed, A., & Kolås, L. (2026). *Research gaps concerning blended project-based learning in computing education.* 22nd International Conference on Information Technology Based Higher Education and Training (ITHET 2026), Norway.

Picciano, A. G., Dziuban, C. D., Graham, C. R., & Moskal, P. D. (Eds.). (2022). *Blended learning: Research perspectives, volume 3* (1st ed.).

Sánchez-García, R. & Angosto, S. (2026). Mapping the path to effective Project-Based Learning: a systematic review and analysis of models and practices. *Humanit Soc Sci Commun* (2026). https://doi.org/ 10.1057/s41599-026-07870-0

Sindre, G., Giannakos, M., Krogstie, B. R., Munkvold, R. & Aalberg, T. (2018). Project-Based Learning in IT Education: Definitions and Qualities. *Uniped*, 41(2), 147–163. https://doi.org/10.18261/ISSN1893-8981-2018-02-06

Suleiman, A. D., Hou, D., Liu, Y., DeWaters, J., Shepherd, D., & Souza, J. (2025). Systematic Literature Review on Project-Based Learning in Computing Education. *ACM Transactions on Computing Education*, 25(4), 1–53. https://doi.org/10.1145/3743684

Swan, K., Richardson, J., Ice, P., Garrison, D. R., Cleveland-Innes, M., & Arbaugh, J. B. (2008). Validating a measurement tool of presence in online communities of inquiry. *E-mentor* 2(24).

Veras, N. L., Rocha, L. S., & Viana, W. (2020). Flipped classroom in software engineering: A systematic mapping study. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (SBES '20, pp. 720–729). Association for Computing Machinery. https://doi.org/10.1145/3422392.3422490

3. The Dynamics of Group Work

Robin Isfold Munkvold

Nord University

This chapter is about the challenges of collaborative learning. Through a game-based educational approach we explore conflict and team dynamics in project-based learning.

3.1. Introduction

During my years of teaching project-based courses I have experienced many challenges when students work in teams over a longer period (typically one semester or more). The challenges are several, i.e. struggling with technology, finding good solutions to given problems, being able to scope their projects, taking on too many tasks, having a hard time planning and assigning tasks to peer students, group members having different goals and expectations from both the work and the group members, and more. Other typical issues are related to differences between the students in terms of work habits, expectations, ambition, etc. Many of these issues have at times led to frustration and have hindered proper progress with the project during the semester. Some of these frustrations have led to students disagreeing and not being able to solve or have a hard time solving conflicts that arise during the project. Not only does this result in dissatisfied students but also a delay in progression that affects the learning outcomes. As a teacher, I have had to spend some time mediating issues that arose within the team to make sure they do not evolve further. I have also experienced a lack of interest among the students when addressing the topic of handling conflicts as part of the teaching. The reason for this seemingly is that they are more focused on technical challenges than teamwork challenges. I therefore wished for my students to be more aware of how to address possible conflicts, the process and the approach to solving them. Among several pedagogical strategies that I have tried, the development of a serious game stands out as it makes students more engaged during classes on conflict resolution and management. This does not mean that the students haven't been able to solve all types of conflicts themselves, but they have reported being more engaged and having learned more from the teaching sessions and learning activities related to the topic (Munkvold, et al., 2023).

The goal of this chapter is to give the teacher a guide on how to plan and arrange learning activities related to the topic of conflict management in group work. This will hopefully make students better prepared for handling possible conflicts in general and issues related to projects that they currently are working on.

The chapter introduces the topic of teamwork and addresses previous research and experiences in conflict management. It gives insight into different theoretical models regarding conflict management. A serious game, based upon the Thomas-Kilmann Conflict Mode Instrument (TKI) (Kilmann & Thomas, 1977), is presented, along with different educational approaches to using the tool. The game, titled Cards of Conflict (CoC), was developed as part of a summer job project for the Center for Excellent IT-education (Excited). A team of students were given the opportunity of developing it based on requirements given by the Excited team. The result was a game designed to be used in a classroom with the aim of working with the topic of conflict management.

3.2. Teamwork

Teamwork competence is vital for students' learning outcomes in a student development project. There are many factors that affect work and group dynamics. Being able to work in teams is prioritized by future employers (O'Neill et al., 2024).

Studies show an increasing need for teamwork competence (Nadal, C. T. et al., 2015; Eglīte, Ž., 2023), hence pointing to the need for training solutions on the topic. This is supported by van Woerden et al. (2025), as they address the importance of including teamwork competence as part of the learning goals and activities.

Typical themes when addressing teamwork competences and skills assessment in project-based learning include learning orientation; group goals; planning and coordination; communication; active participation; completing tasks on time; displaying appropriate assertiveness; and feeling being part of the team (Shofiyah et al., 2022; Hebles et al. 2022).

Some of these themes are further elaborated to give better insight into possible conflicts that can arise during teamwork.

Achieving a **shared understanding** of the project's purpose and objectives is often challenging. Even when team members believe they are aligned, differences in interpretation may remain. This raises an important question: how can a team determine whether everyone truly shares the same understanding, and is such assurance ever fully attainable?

Another challenge concerns determining **which individuals should be assigned specific tasks** based on predefined criteria. While these tasks must contribute to the project's objectives, they should also support the professional development outcomes embedded in the educational goals. Different team members might have different priorities, depending on what they see as important.

Should the team try making the best possible project/game/ app, or should they make sure that each team member develops their own skills / competencies in line with the course's learning objectives? This is discussed more in-depth in the chapter "Educating for generic competencies".

On the topic of **communication,** several aspects should be addressed. Some are related to setting and having a common understanding of project goals, how to divide responsibilities and follow up on work that needs to be done (within set time limits) and how to deal with disagreements during a project.

There is, in general, also an expectation that all team members **actively participate** in the group work, discussions, decisions taken, and more. Lack of active participation by team members will affect group dynamics, hence how the group acts as a team and how they contribute to the project.

Challenges often arise when fellow team members do not fulfil their assigned responsibilities. This may involve tasks such as facilitating meetings, booking rooms, developing software, creating digital assets, or completing other activities required to achieve project objectives. While there may be legitimate reasons for a student's inability to complete assigned tasks, unmet responsibilities can negatively affect team performance and place additional pressure on other members of the group.

The ability to **contribute constructively** to team processes is important, as it is likely to influence communication among team members. Individuals who struggle to express their views clearly and respectfully may negatively affect motivation, attitudes, group dynamics, and overall team performance.

Further consideration is the need to ensure that all team members experience a **sense of inclusion and belonging** within the group. Exclusion from discussions, social interactions, or other team activities can undermine group dynamics and contribute to miscommunication, frustration, and other obstacles that may hinder progress during a student development project.

All in all, the above-mentioned aspects might lead to disagreements and tension within the team, and they will need to make sure they have a way of handling disagreements and possible conflicts.

The continuation of this chapter addresses theories for how to work with disagreements during the project phase, what actions can be taken to keep the team perform well and examples on how to use the serious game, Cards of Conflict, as an educational approach, making students more aware of possible issues and how to manage them.

3.3. Conflict Management

Minor conflicts or disagreements between members of a team are often seen as necessary for solving issues and challenges that arise during the development phase of a student project, as these can be both constructive and help students evolve (Jehn, 1995; Johnson & Johnson, 2005). Though these conflicts or differences in perspective can also lead to bigger disagreements between the students, and hence there will be a need to find strategies for solving these.

Social Interdependence Theory (Johnson & Johnson, 2005) provides a framework for understanding cooperation and conflict within groups. From this perspective, constructive controversy is viewed as an educational process in which learners engage with opposing viewpoints, take alternative perspectives, and work toward unifying solutions and deeper understanding.

In project-based learning environments, students also face challenges (personal, practical, topic-based, individual responsibilities, etc.) that lead to frustration and conflicts (Kolås & Munkvold, 2017). As they spend most of the time working on the project, the teacher is normally less present, hence not able to support the learner (student) as much as in a traditional classroom-based activity.

Based on my experience as a teacher and supported by previous research (Shofiyah et al., 2022; Hebles et al. 2022), some reoccurring topics are addressed when students are working with group projects: learning orientation; group goals; planning and coordination; communication; active participation; completing tasks on time; displaying appropriate assertiveness; and feeling being part of the team.

O'Neill et al., (2024) and Pruitt & Kim (2004) address several factors on how conflicts emerge and what students perceive as valuable and important to understanding themselves, based on theories on conflict management and possible pitfalls during project-based learning in teams. Some factors are related to personal development goals, group dynamics, and teamwork.

Pruitt & Kim (2004) give deeper insight into conflict resolution strategies and suggests that moderating task-related disagreement can be constructive, but unmanaged escalation can damage trust, communication, and collaboration. Early intervention, perspective-taking, and problem-solving approaches are therefore critical for maintaining productive teamwork.

According to Orchard et al. (2023), team conflict is typically conceptualized in three forms: relationship conflict, task conflict, and process conflict. Conflict management training emphasizing cooperation has been shown to enhance both project team productivity and managerial preparedness (Zelenko et al., 2024).

With intensive group-based project work, however, things may take directions that do not support a constructive process.

A commonly used theory of Conflict Management is the Thomas-Kilmann Conflict Mode Instrument (TKI) (Kilmann & Thomas, 1977). The instrument assesses an individual's behavior when faced with a conflict. The instrument builds on the assessment of assertiveness and cooperativeness. Assertiveness captures the extent to which a person attempts to satisfy their own concerns whilst cooperativeness refers to the degree to which the individual tries to satisfy others' concerns. The two dimensions are further combined with five methods of dealing with a conflict: Competing, Collaborating, Compromising, Avoiding, and Accommodating. Depending on the case, given a certain conflict involving many personalities and other relevant factors, different approaches can be taken to resolve a conflict.

Figure 1 illustrates these methods and suggests possible approaches for handling different types of conflicts, depending on where one can place the conflicting parties on the dimensions of assertiveness and cooperativeness.

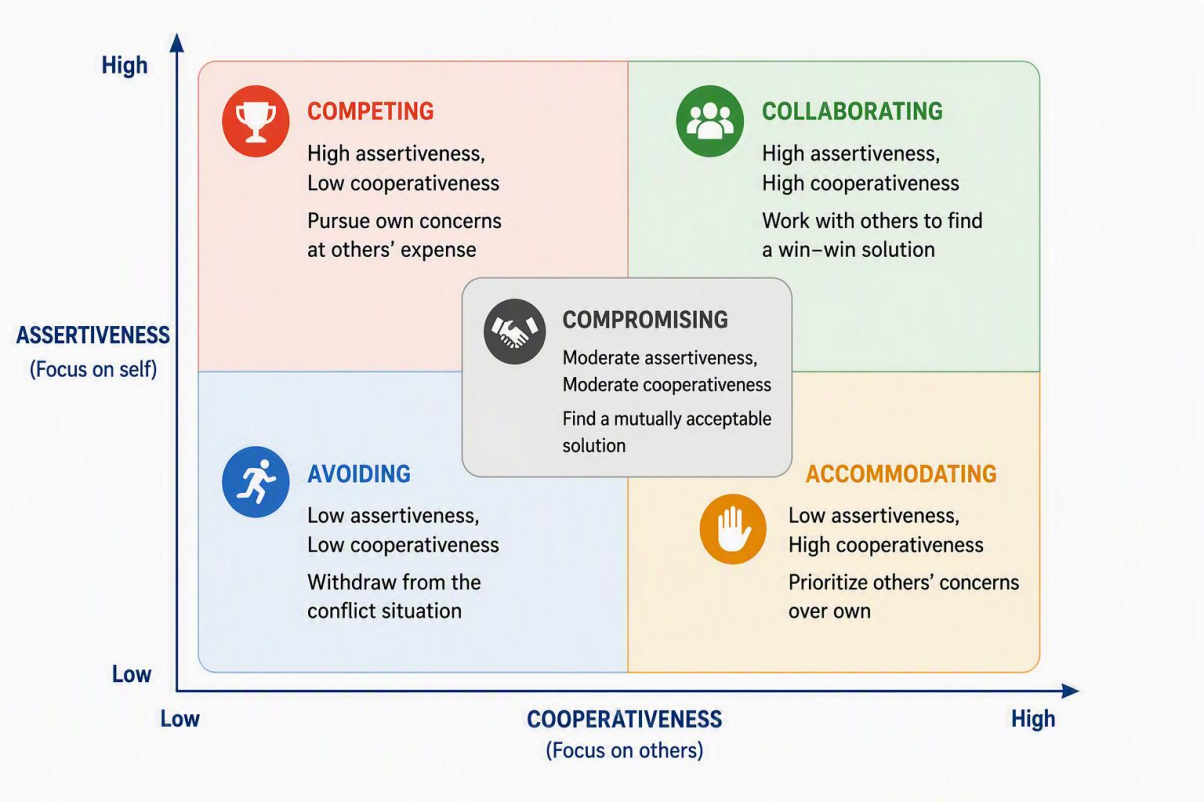


Figure 1 - Two-dimensional model illustrating Conflict Management styles adapted from the Thomas-Kilmann conflict mode framework.

The selection of conflict-management methods should be guided by the specific type of conflict involved. In some situations, a competing conflict-management style may be effective for maintaining project momentum and facilitating timely decision-making. However, excessive reliance on this approach may increase the likelihood of interpersonal tensions and contribute to the emergence of additional conflicts within the team. It is therefore important that the teacher, sometimes together with the students, defines a set of authentic and relevant cases and makes use of these during the learning activity. The goal is to help students develop conflict management skills and understand how project-related conflicts can be addressed. Depending on the situation, resolutions may be made by the team

leader, by members directly involved in the conflict, by the team as a whole or with the help of external staff (e.g. the teacher).

3.4. The tool

As mentioned earlier in this chapter, a group of students took on the challenge of developing a serious game that could be used as part of teaching activities. The result of this process was the digital game Cards of Conflict. The game is based on the Thomas-Kilmann conflict model's five modes of conflict management, consisting of 12 different conflict scenarios.

In this game-based learning activity, student teams are presented with a series of conflict scenarios. For each scenario, the instructor provides a set of potential conflict-management strategies from which teams must select the solution they consider most appropriate. Team members are required to discuss the scenario and reach a consensus before submitting their choice. Once all teams have responded, the instructor reveals the recommended strategy, and the game allocates points accordingly. Throughout the activity, the instructor serves as a game master, guiding the process while students engage in collaborative decision-making and reflection on conflict resolution.

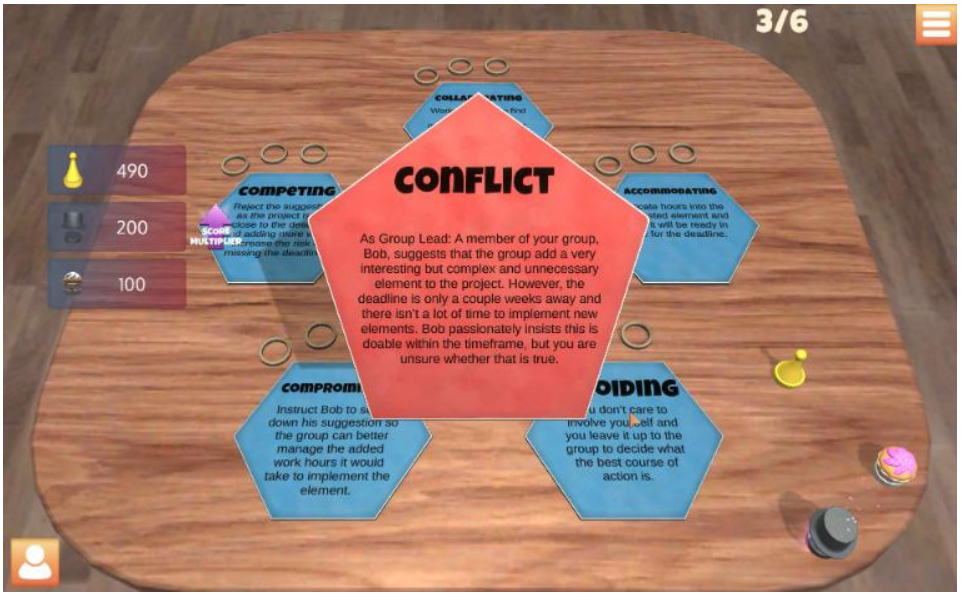


Figure 2: The CoC interface showing one conflict scenario and five conflict resolution strategies

Munkvold et al. (2023) confirm that the game-based learning approach using Cards of Conflict influences student social interaction during gameplay. This approach also raises students' comprehension and self-awareness of conflict management.

To make sure the students find the learning activity interesting, it is important that they discuss authentic scenarios. Munkvold et al. (2023) and Chang et al. (2024) confirm that the use of authentic scenarios enhances students' engagement when they are presented with learning resources and cases that align with scenarios that both feel relevant and comply with industry standards. The outcomes from the research (Munkvold et al, 2023 and Chang et al., 2024) also show how the use of authentic scenarios positively affects group dynamics and facilitates a deeper understanding of the topic at hand.

The following chapter demonstrates how Cards of Conflict can be implemented as part of a game-based learning activity, with the aim of students learning about conflict management.

3.5. Example practice (suggested approaches for implementation)

This chapter presents two distinct approaches to incorporating Cards of Conflict into learning activities and learning processes.

The pedagogical design is primarily informed by Kolb's (1984, 2015) experiential learning model. This model conceptualizes learning as a cyclical process consisting of concrete experience, reflective observation, abstract conceptualization, and active experimentation. Within the learning activity, concrete experience is provided through gameplay, where participants engage with authentic (and realistic) scenarios and decision-making processes. This is followed by structured reflection on actions and outcomes, which serves as the basis for linking practical experiences to relevant theoretical perspectives. Participants are then encouraged to apply these insights by revising existing scenarios or developing new strategies, thereby completing the cycle through active experimentation.

While Kolb's experiential learning cycle provides the overall didactic framework, the model is further enhanced through the integration of game-based elements designed to promote engagement, motivation, and active participation. Kolb's theory builds on earlier contributions from Dewey (1938), who emphasized learning through experience and reflection; Lewin (1951), whose work highlighted iterative action-reflection cycles and the social dimensions of learning; and Piaget (1970), who focused on cognitive development through interaction with the environment. Drawing on these theoretical foundations, the learning activities presented in this chapter combine experiential learning principles with strategies to support both participant engagement and the development of conflict-resolution competencies.

The section below describes two different ways of using the tool when addressing the topic of conflict management. The didactical approach applied across the different scenarios follows a Tell, Show, Do + Practice and Review structure (Kuhlmann, 2022) which is consistent with the demonstration, application, and integration principles described in Merrill's First Principles of Instruction (Merrill, 2002).

3.5.1. Scenario 1 - Making a playful experience with the learning activity (Cards of conflict with fixed scenarios)

This scenario is better suited for a *shorter session* with an introduction to the topic and a playful in-classroom activity to enhance student engagement.

- **Goal:** Teaching students the theory of conflict management and practicing this in a classroom setting.
- **Constraints:** One class with a maximum of 30 participants. 2 hours are available for the learning activity.

Didactical approach

1: Activation and demonstration (Tell and Show):

The teacher starts by presenting the students with the theory on the Thomas-Killman instrument for conflict management and shows them a possible conflict scenario with possible solutions.

2: Application (Do + Practice):

The students are divided into groups, and the teachers initiate the game, which contains pre-defined cases, and where the teacher takes the role as a game master. Figure 3illustrates the steps taken. During the process, the student groups get to practice how to discuss the different cases and then reach consensus on what they consider to be the right answer. The groups are further given points depending on their choice of solution, and a scoreboard is shown after each round. This process is iterated 10-15 times (with 10-15 different cases).

3: Integration and reflection (Review) during and at the end of the session:

During the process, the suggested right solutions to the case are discussed in plenary with the participants reflecting on how the different resolution alternatives could give different results.

At the end of the session, one group will be announced as the winning team. Depending on the time available for the session, the students and the teacher can engage in a new round of reflections.

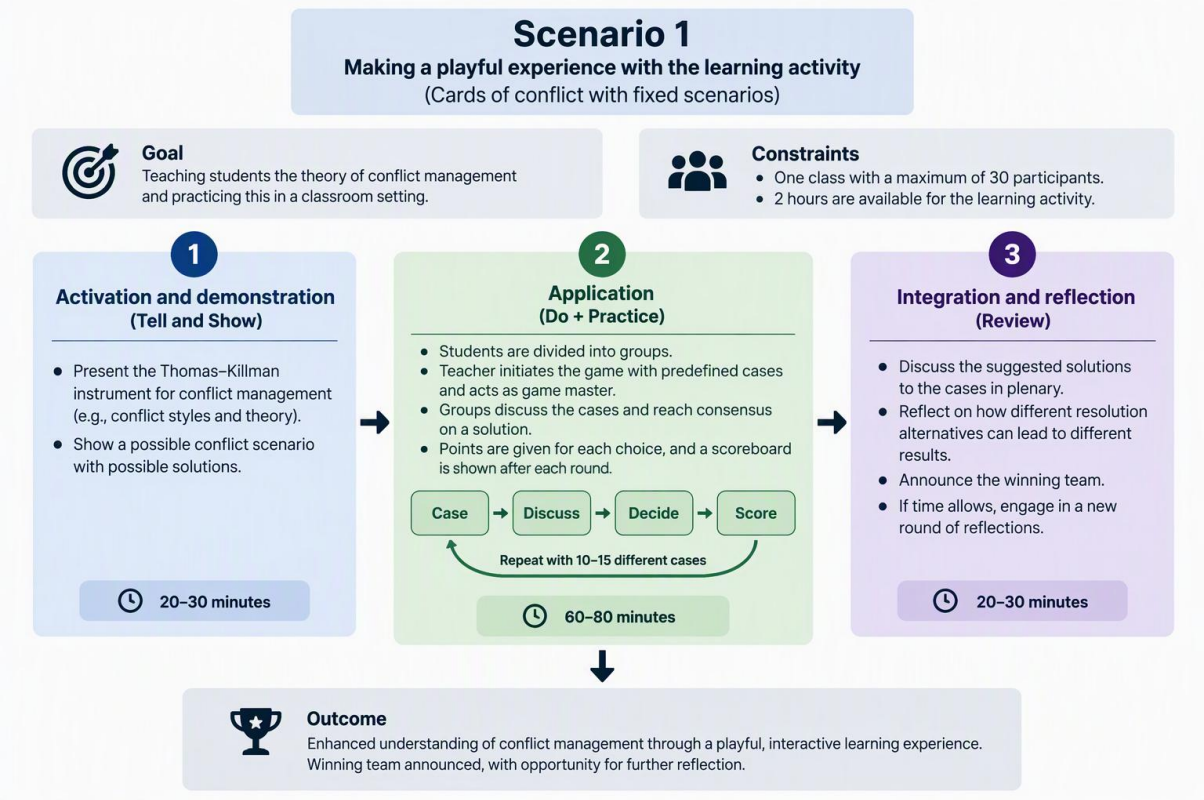


Figure 3 – Scenario 1: Learning activity flow for conflict management training, using predefined scenarios.

3.5.2. Scenario 2 - Making a playful experience with the learning activity, where the participants design their own scenarios

This scenario is suited for a *longer session* with an introduction to the topic, students designing scenarios and a playful in-classroom activity to enhance student engagement.

- **Goal:** Teaching students the theory of conflict management and practicing this in a classroom setting.

- **Constraints:** One class with a maximum of 30 participants. 6-10 hours are available for the learning activity.

Didactical approach

1: Activation and demonstration (Tell and Show):

- The teacher starts by presenting the students with the theory on the Thomas-Killman instrument for conflict management and shows them a possible conflict scenario with possible solutions.

2: Application (Do + Practice):

- Each student is given the task of defining up to 5 possible cases from where a conflict can arise.
- The next step is to divide the students into groups of +/- 5 students and have them discuss five cases they would like to take further and to define 5 different ways of solving the conflict based on the Thomas-Killman model.
- The students then continue the process, rating the possible solutions (from 1-5). They add this information to the Cards of Conflict game where rating 1 is the solution that they recommend as most important and add information on why they find this one solution to be the best.
- When the different cases, along with the rated conflict resolutions, have been added to the tool, the teacher initiates the game, following the same gameplay structure as in scenario 1.

3: Integration and reflection (Review) during and at the end of the session:

- During the process, the suggested “right solution” to the case is discussed in plenary, and the participants discuss and reflect on how the different resolution alternatives could give different results.
- At the end of the session, one group will be announced as the winning team, and depending on the time available for the session, the students and the teacher will engage in a new round of reflection together.

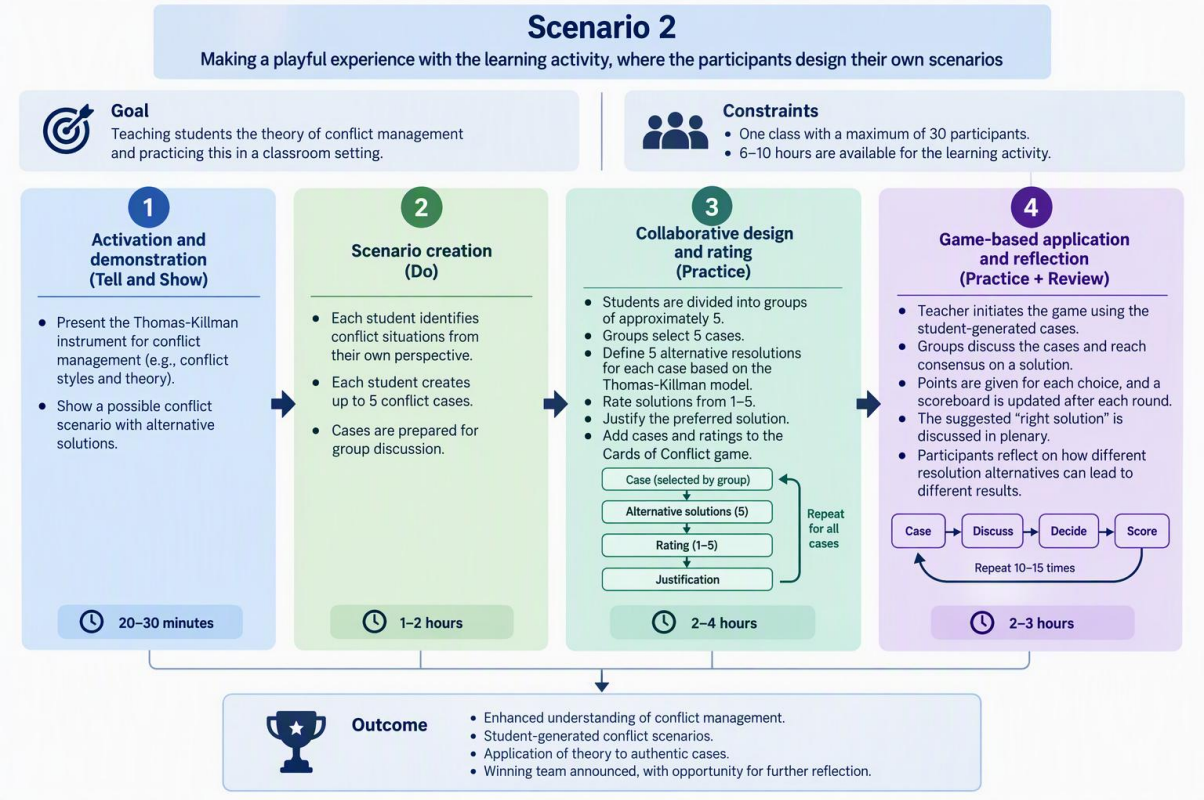


Figure 4 – Scenario 2: Learning activity flow for conflict management training using student-generated scenarios.

3.6. End Comments

Conflict management is an essential component of teamwork competence in project-based learning. While conflicts can create challenges for student teams, they also represent valuable opportunities for learning, reflection, and personal development. This chapter has presented a game-based learning approach where the game Cards of Conflict is used as a tool to facilitate the learning activity, with the goal of helping students explore authentic conflict situations, discuss alternative response strategies, and reflect on the consequences of different actions.

The chapter highlights factors that are particularly important for the success of the approach. First, the conflict scenarios need to be *authentic* and recognizable, enabling students to connect the situations to their own project experiences. Second, the activity should encourage *collaboration* within teams while introducing a degree of *competition* between teams, creating engagement and motivation for active participation. Finally, the educational value of the activity is strengthened when teachers facilitate structured *reflection* both during and after gameplay. These reflections will help students relate their experiences to relevant theories, to critically evaluate different conflict-management strategies, and develop a deeper understanding of how conflicts influence teamwork and project outcomes.

By combining authentic scenarios, collaborative gameplay, competition, and guided reflection, Cards of Conflict provides an engaging framework for learning about conflict management. The approach not only helps students develop greater awareness of conflict dynamics but also supports the development of communication, teamwork, and problem-solving competencies that are valuable throughout higher education and in future professional practice.

The game can be downloaded from https://gamejolt.com/games/CardsofConflict/689375.

Copyright

The Dynamics of Group Work © 2026 by Robin Isfold Munkvold is licensed under CC BY-NC-ND 4.0



References

Chang, Y., Choi, J., & Akbulut, M. Ş. (2024). Undergraduate Students' Engagement in Project-Based Learning with an Authentic Context. Education Sciences, 14(2), 168–168. https://doi.org/10.3390/educsci14020168

Dewey, J. (1938). Experience and education. Macmillan.

Eglīte, Ž. (2023). Collaboration in Creative Industries: From Creative Individuals and Intermediaries to Networks. *Culture Crossroads, 23.*

Hebles, M., Alvarez-de-Eulate, C. Y., & Santos, M. A. D. (2022). Teamwork competency scale (TCS) from the individual perspective in university students. Journal of Technology and Science Education, 12(2), 510–510. https://doi.org/10.3926/jotse.1478.

Jehn, K. A. (1995). A multimethod examination of the benefits and detriments of intragroup conflict. Administrative Science Quarterly, 40(2), 256-282. https://doi.org/10.2307/2393638

Johnson, D. W., & Johnson, R. T. (2005). Creative Controversy: Intellectual Challenge in the Classroom. Interaction Book Company.

Kilmann, R. H., & Thomas, K. W. (1977). Developing a forced-choice measure of conflict-handling behavior: The "MODE" instrument. Educational and Psychological Measurement, 37(2), 309–325.

Kolb, D. A. (1984). Experiential Learning: Experience as the Source of Learning and Development. Englewood Cliffs, NJ: Prentice Hall. (http://academic.regis.edu/ed205/Kolb.pdf)

Kolb, D.A. (2015). Experiential Learning: Experience as the Source of Learning and Development. 2nd Edition, Pearson Education, Inc.

Kolås, L., & Munkvold, R. I. (2017). Learning through construction: A roller coaster ride of academic emotions? In Proceedings of the 6th Computer Science Education Research Conference (CSERC '17) (pp. 29-34). Association for Computing Machinery. https://doi.org/10.1145/3162087.3162090

Kuhlmann, T. (2022, June 14). Tell, show, do plus practice & review to build effective online training. Articulate E-Learning Heroes. https://www.articulate.com/blog/tell-show-do-plus-practice-review-to-build-effective-online-training/

Lewin, K. (1951). Field theory in social science: Selected theoretical papers (D. Cartwright, Ed.). New York: Harper.

Merrill, M. D. (2002). First principles of instruction. Educational Technology Research and Development, 50(3), 43-59. https://doi.org/10.1007/BF02505024

Munkvold, R. I., Kolås, L., & Palmquist, A. (2023). A Conflict Management Game in Project-Based Learning. Proceedings of the 17th European Conference on Games Based Learning (ECGBL). DOI: 10.34190/ECGBL.17.1.1375.

Nadal, C. T., Mañas, G. P., Sabrià-Bernadó, B., & Mora, C. A. (2015). Assessing teamwork competence. Psicothema, 4(27), 354–361. https://doi.org/10.7334/psicothema2014.284

Pruitt, D. G., & Kim, S. H. (2004). Social Conflict: Escalation, Stalemate, and Settlement (3rd ed.). McGraw-Hill.

Shofiyah, N., Wulandari, F. E., Mauliana, M. I., & Pambayun, P. P. (2022). Teamwork skills assessment for STEM Project-Based Learning. Jurnal Penelitian Pendidikan IPA, 8(3), 1425–1432. https://doi.org/10.29303/jppipa.v8i3.1678

O'Neill, T. A., Aldana, A. J., Espinoza, J. A., Sperry, M. P., Brykman, K. M., & Donia, M. B. L. (2024). Conflict management styles assessment and feedback for student self-awareness and team development. Assessment & Evaluation in Higher Education, 49(8), 1091-1104. https://doi.org/10.1080/02602938.2024.2350682

Orchard, C., King, G., Tryphonopoulos, P., Gorman, E., Ugirase, S., Lising, D., & Fung, K. (2023). Interprofessional Team Conflict Resolution: A Critical Literature Review. *Journal of Continuing Education in the Health Professions*, *44*(3), 203–210. https://doi.org/10.1097/ceh.0000000000000524

Piaget, J. (1970). Science of education and the psychology of the child (D. Coltman, Trans.). Orion Press APA PsycNet.

van Woerden, R., van Goch, M. M., van der Tuin, I., & Schruijer, S. G. L. (2025). Students' Needs for Multidisciplinary Student Teamwork Teaching and Learning in Interdisciplinary Higher Education: Relational Competences and Educational Design Needs. Active Learning in Higher Education.

Zelenko, O., Zhdanova-Nedilko, O., Stepanenko, V., Chernyahviska, V & Nonik, V. (2024). Managing conflicts within a team: Conflict resolution strategies and effective cooperation. *Ad ALTA Journal of Interdisciplinary Research*, *14*(1), 69–72. https://doi.org/10.33543/1401396972

4. Sustainability in Computing Education

Birgit Krogstie and Bashira Jaradat

Norwegian University of Science and Technology (NTNU)

4.1. Introduction

Sustainability can broadly be understood as the idea that we should all act with future generations in mind, giving them the chance to lead a good and meaningful life.

Negative consequences of human activity can currently be seen on a global scale. Global warming, largely caused by the burning of fossil fuels, has severe consequences for the climate. Also, the use of AI has unpredictable effects on future society; exactly what they might be is currently disputed, but negative impacts are potentially devastating.

Consequences of human activity and technology span different areas, e.g. social, economic, and environmental ones. Effects of technology-related activity can be positive or negative, are intertwined in complex ways, and can be considered on different time scales. For instance, decision-making politicians or tech managers may prioritize short term benefits of a certain activity, whereas long-term effects may be downplayed or ignored. Apart from costs and profit-related concerns, arguments for such decisions may refer to uncertainty and the possibility of a later "technological fix".

The use of technology increasingly permeates our ways of living, acting and reasoning as individuals and citizens or as professionals. In this, computing experts have an important role. Crucially, such influence applies beyond the more obvious decision-making power following management positions. Experts focusing their work on technology development or the implementation of digital solutions in an organization, frequently make professional choices with sustainability implications. Professional ethics dictates that an expert have a responsibility to use their expertise for the good. To a company whose business includes technology development, employee awareness and competence in this area may very well also align with business-driven concerns: Apart from managing their reputation as responsible and ethical, a company is legally required to adhere to sustainability requirements and to perform corporate sustainability reporting. This also levels the ground for competition among companies in terms of addressing sustainability.

Information Technology (IT) has a huge impact on sustainability, whether through resource use (e.g. area and energy needs of server parks) or their influence on human behavior (e.g. encouraged consumption of material goods; production or logistics efficiency gains based on AI). The footprint of computing in use, which is mainly related to energy efficiency, can be denoted "green IT" (Murugesan 2008). The use of computing to improve conservation of resources in all areas of society (including energy) has been denoted "green by IT" (Hilty 2012).

Overall, there are three general perspectives on IT for sustainability (Peters et al. 2024):

- *Incremental:* Focus is to adjust computing to reduce its footprint
- *Enabling*: IT is the means for addressing society-wide sustainability issues
- *Disruptive:* Acknowledging the severity of the sustainability issues and the need for major, structural societal changes where IT norms, practices and processes also must change.

On this backdrop, we recognize that computing experts need adequate competence to understand the sustainability impacts, direct or indirect, of their work. Importantly, this involves not only individual knowledge and sense making of technological and other consequences of a solution, but also, to a significant extent, interaction and collaboration with other stakeholders. As part of professional teams, organizations and society in general, computing professionals needs be able to engage in discussion, negotiation and reflection on sustainability aspects of the technology solutions and their use.

Given the complexity of sustainability effects from technology, thus, and given the strong link to professional work ethics and collaborative work, it is clear that competence to address sustainability must be provided through higher computing education. It is furthermore evident that this cannot happen by adding sustainability considerations to educational offerings as an afterthought or as an optional perspective.

In this chapter we ask: How can sustainability be integrated in computing education in a way that is seen as meaningful by students as well as teaching staff, and that leads to the relevant competencies? Referring to examples of integrating sustainability in computing education through work in Excited, we propose an approach based on the following principles:

The integration of sustainability in computing education needs to

- ..be meaningful by being part of the **professional practice** in question. This includes the scientific discipline as well as the profession.
- ..involve a degree of **transformation** – the need for transformation in society necessitating a need for change in the profession, which again depends on education to prepare the students to make a change.
- .. be directed at relevant **competencies** and considered in a study program perspective (think progression and red thread)
- ..use **relevant tools (methodology, technology)**. In our case, we have focused on use and adaptation of the SusAF framework

We start by visiting some basic sustainability models and principles, having a look at sustainability competencies and pedagogical approaches for teaching sustainability. Next we turn to the relevance of sustainability for computing and the integration of sustainability in computing education.

4.2. Sustainability models and principles

When addressing sustainability in education, it is useful to start with the concept itself. Many will associate it with environmental issues. However, a broader and widely established understanding consider sustainability as a set of interconnected aspects in different areas. The connections and complexity on a systemic level are a major reason that sustainability challenges tend to be hard to solve.

One well-known, broad understanding of sustainability is as development that "meets the needs of the present without compromising the ability of future generations to meet their own needs» (Brundtland 1987). The foundation of sustainability has three pillars: environment, economy and society.

The UN describes 17 Sustainable Development Goals (SDGs)[1], each of which having a set of sub-goals/targets. The link between the SDGs and the three pillars is established in the so called "wedding cake model"[2].

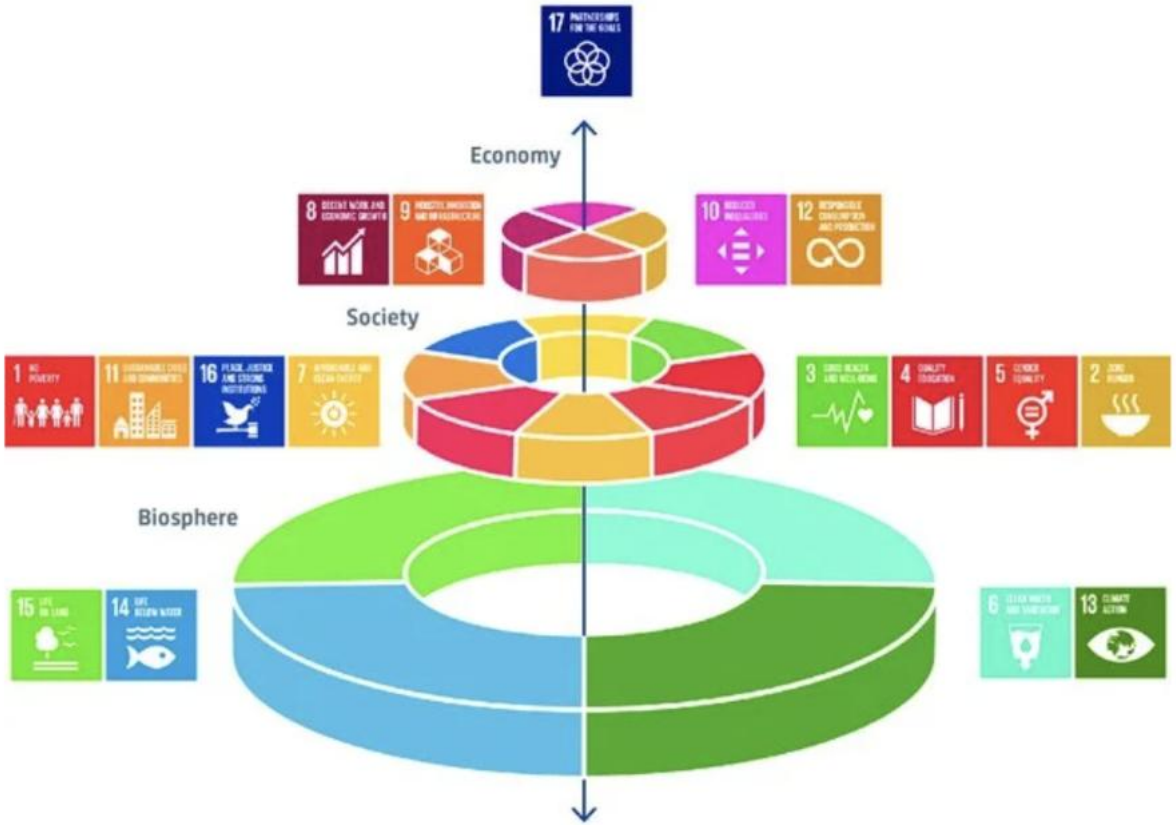


Figure 1: The wedding cake SDG model. Credit: Azote for Stockholm Resilience Centre, Stockholm University CC BY-ND 3.

Another holistic take on sustainability is provided in the "doughnut model" (Raworth 2022). This model sees the safe and just space for humanity as delimited by a social foundation and an ecological ceiling, where sustainability implies that the social foundation is sufficiently supported, while the activity does not overshoot the ecological ceiling. The current global situation is unfortunately characterized by a severe shortfall in many social areas and an overshoot in many ecological ones.

A necessary part of addressing sustainability challenges in society is to work for responsible use of resources. This will ultimately require a transformation of the economy towards a much higher degree of resource re-use and a reduction of waste. This can be described as a *circular economy*, illustrated in Figure 3.

In the context of higher education, fundamental concepts and models such as the UN SDGs, the doughnut/planetary boundaries, and circular economy can be considered basic knowledge for students and educators. This provides a shared platform and language for addressing more advanced and discipline-specific issues.



Figure 2: The Doughnut model. By DoughnutEconomics - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=75695171

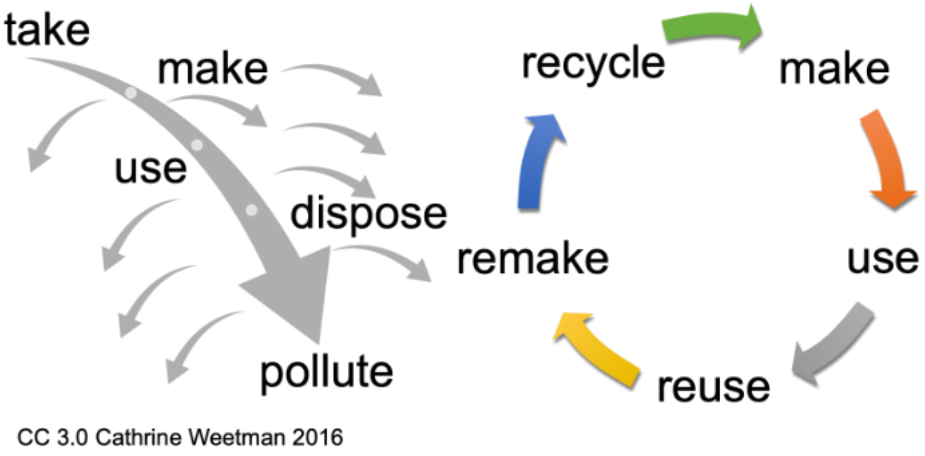


Figure 3: Linear vs. circular economy. Original: PennineFoxhound Vector: Clonewayx Catherine Weetman, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>

4.3. Sustainability competencies

Today's students, when entering higher education, have acquired some basic awareness and knowledge about sustainability through learning in primary, secondary, and/or high school [3]. They are likely to have encountered the above-presented models or similar ones and are familiar with some key sustainability-related challenges and principles, e.g. of complexity and circularity. Turning to higher education, learning outcomes in the area of sustainability should accordingly build on students' existing knowledge and be more advanced and specialized.

To successfully address sustainability in a professional context, there is a need to define the relevant competencies. Competencies recognized as *key sustainability competencies* (Wiek et al. 2011) include systems thinking, future thinking, strategic thinking, normative thinking, disciplinary competence, inter-disciplinary competence, and inter-personal competence. Such competencies are often included in the intended learning outcomes of courses focused on analysis, design and problem-solving skills. These courses often include student projects, project-based learning (PjBL), being a well-established approach e.g. in higher computing education. There is, accordingly, a significant overlap between key sustainability competencies and competencies already aimed for in many courses and study programs.

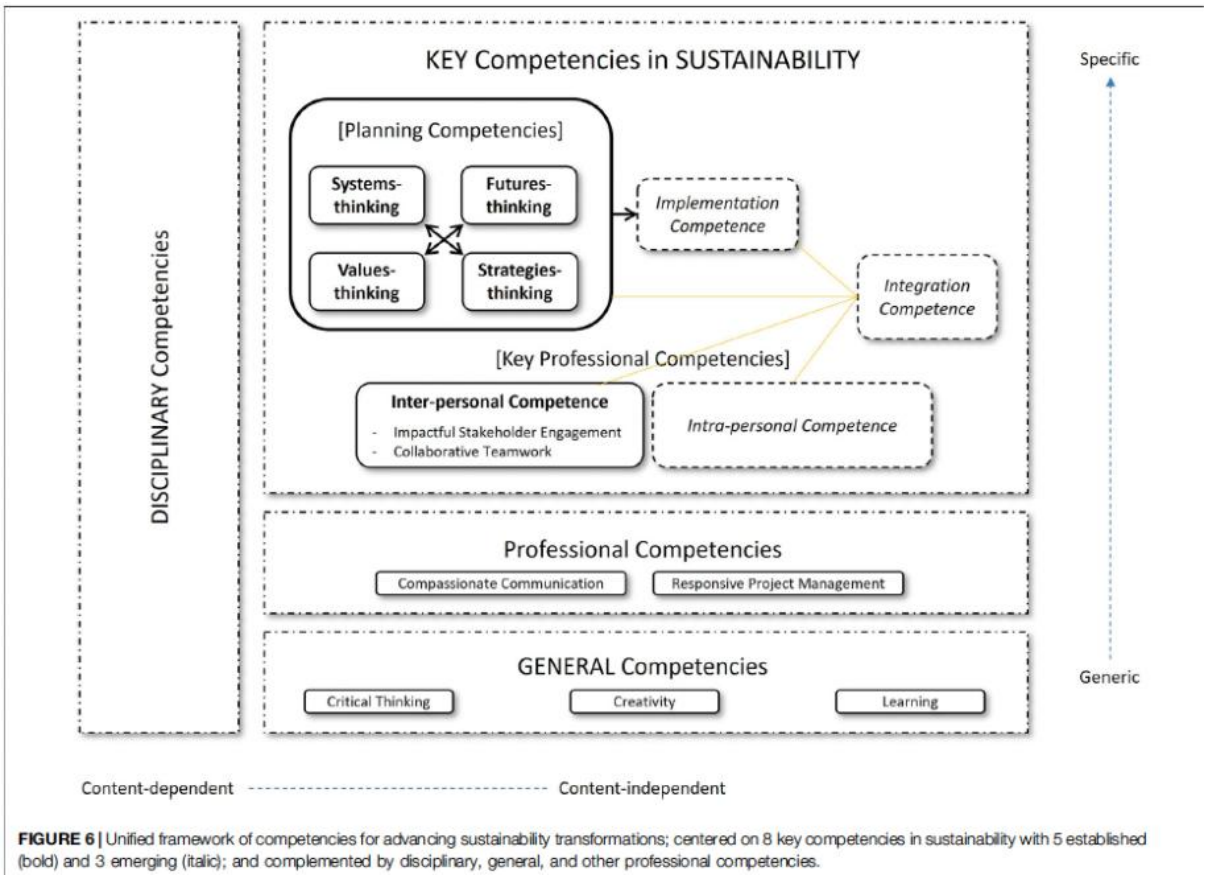


FIGURE 6 | Unified framework of competencies for advancing sustainability transformations; centered on 8 key competencies in sustainability with 5 established (bold) and 3 emerging (italic); and complemented by disciplinary, general, and other professional competencies.

Figure 4: The unified sustainability competencies framework by Redman and Wiek (2021)

Two noteworthy frameworks for sustainability competencies are the EU sustainability competencies framework, GreenComp (Bianchi et al. 2022), and the unified framework by Redman and Wiek (Redman and Wiek 2021) shown in Figure 4. These frameworks are both broad and holistic and thereby covering largely the same areas of competence. A noteworthy aspect of GreenComp is that it includes

political agency, collective action and individual initiative, thus pointing to the need for competence in *acting* for sustainability. As for Redman and Wiek, the framework incorporates disciplinary and professional competencies and lays out a map that seems well adapted to the language of curricular work in higher education.

For educators developing courses and study programs, competency frameworks can be helpful in several ways. The frameworks can be applied as a reference suggesting a set of competencies that *should* be covered in a study program, providing a starting point for further work on where, how (and possibly whether) the related learning should happen through the various courses in the program. To achieve constructive alignment (Biggs and Tang 2011) in addressing sustainability competencies, some of the framework-related literature may be useful by presenting insights on how sustainability competencies can be operationalized within specific domains, acquired by students through appropriate learning activities, and evaluated (Redman et al. 2021; Wiek et al. 2015; Bianchi et al. 2022) . The operationalization of sustainability in computing education in terms of learning outcomes, activities and assessment is however an area in need of more research, for instance empirical studies on the integration of sustainability in real-life educational settings.

4.4. Pedagogy for teaching about sustainability

To support students' learning about sustainability, some pedagogical approaches are recognized as effective. We will touch upon some general characteristics before elaborating on one approach: project-based learning (PBL).

4.4.1. General characteristics

UNESCO points out that education for sustainable development should be transformative and competence-based (UNESCO 2018). The complexity of sustainability should be addressed through inter- and transdisciplinarity. Teaching should be learner-centered, action-oriented and transformative. A *learner-centered* approach sees students as autonomous learners. Knowledge is actively developed, not merely transferred. Reflection on one's own learning process is essential. *Action-oriented* learning entails students engaging in action and reflecting on experiences in light of the intended learning process and the student's individual development. This corresponds to the elements in Kolb's cycle of experiential learning (Kolb 1984).

Transformative learning for sustainability (d'Abreu et al. 2025) seeks to empower students to questions their perspective and change their way of seeing the world in order to achieve better understanding. This again influences action. Considering the state of affairs and degree of current progress on handling sustainability on a societal level, many argue that the world needs more rapid and radical change, requiring a transformation of the educational system so that it can prepare students to transform society. There are different approaches to transformative learning, aiming for varying degrees of transformation (Yacek 2020). Regarding computing education for sustainability, current practices tend to focus on increments and improvement within existing structures rather than more radical transformation (Peters et al. 2024).

An aspect of learning about sustainability that has increasingly been acknowledged as important, is the emotional side. Sustainability issues can be severe and difficult to fully take in, both for students and for academic staff, and this should be deliberately addressed in teaching (Eriksson et al. 2022; Holmén and Lönngren 2025). To support reflective learning (Boud et al. 1985), it is important to acknowledge emotions as part of the reflection process and the experiences reflected upon.

4.4.2. Project-based learning

A teaching method with the potential to meet the requirements to be learner-centered, action-oriented, transformative and interdisciplinary, is project-based learning (PBL). Blumenfeld and colleagues (Blumenfeld et al. 1991) described the approach in the following way:

"Project-based learning is a comprehensive perspective focused on teaching by engaging students in investigation. Within this framework, students pursue solutions to nontrivial problems by asking and refining questions, debating ideas, making predictions, designing plans and/or experiments, collecting and analyzing data, drawing conclusions, communicating their ideas and findings to others, asking new questions, and creating artifacts."

In higher computing education there is a strong tradition for PBL-based courses. These courses are typically designed for learning cycles mapping the iterations in the software development process, e.g. including reflection at the end of each iteration (B. R. Krogstie and Divitini 2009). There are numerous aspects on which PBL courses can be varied to adapt to the specific educational context (Sindre et al. 2018). For instance, the project tasks may be common to all projects in a course or different among the projects. There may be a project customer, e.g. a member of scientific staff or an industry partner. Degree of imposed structure on the projects may vary, student projects in the first and second year typically having more scaffolding (e.g. in terms of mandatory templates and deliverables) than projects toward the end of a study program (Bygstad et al. 2009). Motivating factors of PBL for the students include perceived authenticity and ownership of the project tasks as well as independence in the work process. Some projects serve as capstone projects. These are usually taken during the final year and meant to provide "a culminating and integrative educational experience" (Clear and et al. 2001). A capstone project is often seen by students as a demonstration of competence; a kind of informal professional qualification. Given the sustainability effects of the work in a profession such as computing, and given the sustainability related learning objectives of the study programs, capstone projects can be considered a particularly relevant occasion for developing and demonstrating sustainability competencies.

4.5. Computing and sustainability

As previously argued, a computing professional needs to be aware of the potential sustainability consequences of digital solutions, understand the mechanisms leading to these consequences, and know how digital solutions can be designed to promote positive effects and limit negative ones.

4.5.1. Ethics

How we address sustainability in computing clearly has a normative aspect. Our conduct in this regard is guided by professional ethics, e.g. as elaborated in the ACM Code of Ethics and Professional Conduct (https://www.acm.org/code-of-ethics). As a professional you should be doing good while avoiding harm. How to achieve this in practice might of course involve conflicting concerns and difficult trade-offs.

From a societal perspective, especially in a country offering free higher education, one may also argue that the individual having the privilege of higher education should use their resulting competency and position for the greater good. This overarching aim is also reflected in the vision of the universities providing the education, e.g. NTNU with its vision "Technology for a better world".

We cannot, thus, approach the topic of sustainability in a "neutral" way in our teaching; there is an agenda with which aiming for sustainability is not an optional concern for those particularly interested.

Rather, it is an expectation and an inherent aspect of professional computing work, albeit with different degrees of focus depending on the domain and task at hand.

4.5.2. Sustainability effects and the SusAF framework

The need to address sustainability in the development of IT solutions has been acknowledged in the Software Engineering and Requirements fields: *All* projects involving IT will – if implemented in real life – have sustainability effects (Duboc et al. 2019), e.g. in terms of resource consumption or changes to individual or social conditions.

Several frameworks have been developed to help developers identify potential sustainability effects of technological solutions. One of them is SAF (Lago et al. 2024), another is SusAF (Sustainability Awareness Framework) (Becker et al. 2015, 2016; Duboc et al. 2019). As this chapter's examples of integration of sustainability in computing education involves the use of SusAF, we will briefly describe this framework.

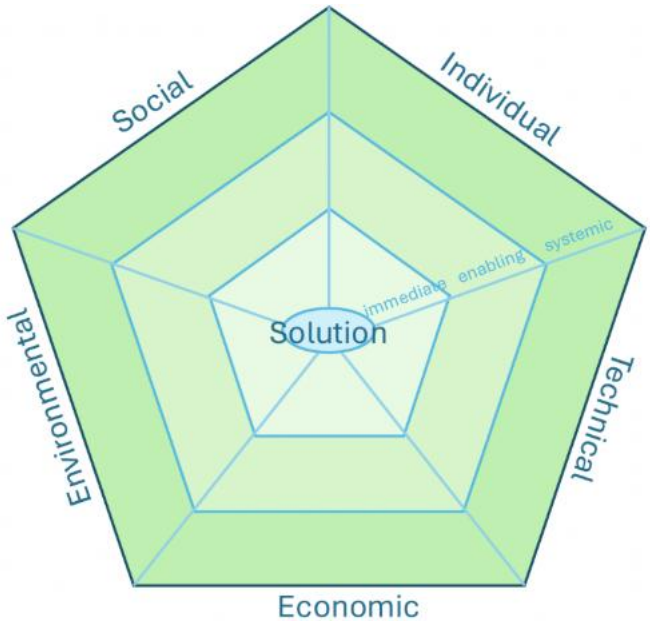


Figure 5: Template for the SusAF diagram - SusAD

SusAF is a framework for conducting a high-level sustainability analysis of possible effects of a solution (a digital solution / technology) on sustainability. A development team may use SusAF to help identify system requirements, possibly in collaboration with stakeholders such as a project customer.

Sustainability effects are considered on five *dimensions*: *Individual* effects pertain to the individual´s ability to thrive and develop. *Societal* effects address relationships between individuals and groups. *Economic* effects relate to the financial side and business value. *Environmental* effects cover the use and management of natural resources. Finally, *technical* effects relate to the ability to maintain and improve systems over time. Effects can be positive and/or negative, and intended or unintended.

Effects have different *orders*, which partly corresponds to time frames: *Immediate* effects follow directly from the production, use, or disposal of the solution. *Enabling* effects emerge from applying the solution over time. *Systemic (Structural)* effects capture the persistent changes that are bserved over time. Effects are interlinked by forming *chains of effects* when one effect (possibly) leads to another. Chains of effects, conflicting concerns (e.g. co-occurrence of positive and negative effects) and

uncertainty of possible scenarios all contribute to the complexity involved in considering the sustainability of a solution.

The SusAF model is visually represented in a diagram, the SusAD (Sustainability Awareness Diagram). The diagram template is shown in Figure 5. Identified sustainability effects (represented as rectangles, e.g. post-its if a physical format is used) are placed in the relevant section, in accordance with their assumed dimension and time frame. Effects being positive or negative can be explicitly represented in use of colour in the SusAD.

in the diagram, e.g. by use of arrows linking the effects, are key to understanding complexity and the systemic nature of sustainability. A filled-in diagram from a SusAF workshop is shown in Figure 6. In this case, there are no arrows showing links between effects, one reason possibly being that the post-its representing sustainability effects take up much of the space in the diagram, leading the developers to skip the arrows.



Figure 6: A filled-in SusAD (Sustainability Awareness Diagram). Photo: Excited

The SusAF framework includes a set of guiding questions to help identify effects on the various dimensions, making more concrete what types of effects might be involved.

Guiding material for applying SusAF in a workshop can be found online (SUSO Sustainable Software 2026).

4.5.3. Sustainability reporting

Professional computing practice entails reporting. Regarding sustainability reporting, there are EU regulations: the Corporate Sustainability Reporting Directive, CSRD (Operato et al. 2025). Companies in Norway are required to follow a sustainability reporting directive from the Norwegian government [4]. It follows that knowledge about sustainability reporting is relevant to students in computing as part of preparation for work life.

4.6. Sustainability as a topic in computing education

A common problem when integrating sustainability in a course or study program is that it may end up as an add-on competing over resources in an already crowded curriculum. The inclusion of sustainability may have been imposed "from above" by the institution. This could lead to resistance among teaching staff as well as students. It is not uncommon that sustainability is deferred until the last lecture of the semester (perhaps as part of a session on "sustainability, ethics and society"), often half-heartedly taught and – partly as a consequence - reluctantly received.

A way to avoid students seeing sustainability as something not really belonging in a computing course, is to deliberately present sustainability as an essential *part of the scientific discipline and/or professional practice*. If sustainability is considered as an integral aspect of computing, it makes sense that courses, e.g. project courses, include experiential learning addressing sustainability, e.g. through sustainability analysis as part of software development.

Reflecting the various perspectives on IT for sustainability (incremental, enabling, and disruptive), sustainability in IT education can be addressed at different "levels of interruption", from the incremental to the transformative. An active stance towards curriculum transformation might be required on part of the teacher (Murugesan 2008), the study program management and possibly higher-level units in the organization: How much do we want to change?

Starting with courses and study programs in their current form, gradual changes can be made to incorporate a focus on sustainability. Limited modification e.g. by substituting a lecture, providing an updated project case, or introducing a new framework to be applied in assignments, can be considered a "low hanging fruit". Such changes may even be possible within the constraints of the current course description and its intended learning outcomes, making it possible to develop and implement a course change on short notice (e.g. based on insights gained the semester before the course starts). Incremental changes also tend to assume that sustainability can be incorporated into largely existing professional and disciplinary practices. The examples presented later in the final part of this chapter are representative of an incremental approach.

A greater potential for change comes from developing a new course or module. The course/module may be more or less general/discipline-dependent and more or less advanced, depending on its intended role in students' learning progression. Elective courses can be easier to integrate into existing study programs, while mandatory courses, if you find room for it, is likely to have a greater impact.

Making computing education transformative might require more profound changes. (Peters et al. 2024) found that few papers on sustainability in computing education engage with the severity and complexity of sustainability challenges and with the dysfunction of the current system. The same authors argue that computing education should be not only about acquiring competence but also about *contributing to democratic change*, with a caring orientation to the world and the future and by taking inspiration from other fields. Current norms and values in computing education tend to convey a picture of computing as purely technical, abstract, and rational. While this perspective needs to be understood by computer scientists, the norms and values also need to be challenged and changed (Peters et al. 2024).

It is possible to make radical changes to a single course or module. A new course might for instance include real democratic participation and community impact (e.g. through a project with an external project owner/client), encourage exploration of emotions, acknowledge complexity, uncertainty and the lack of correct answers, and so on. To get beyond providing the students with an interesting and

rewarding experience, however, changes might need to happen on the level of the study program - or through the development of a new study program.

The development of new or profoundly changed study programs requires the support of the study program owner (university/faculty/department) who needs to consider the purpose and role of the new or radically changed study program considering the institution's strategy and study program portfolio. Successful implementation of the study programs furthermore depends on potential applicants to be convinced that they will get the relevant qualification by choosing something different and missing at least some of the "safe and sound" education based on a long tradition.

Considering the alternatives outlined above, for higher computing education (as for education in other fields) it is important to address whether or how we can bring about the desired sustainability competencies and societal changes by working incrementally on our existing offerings, or if we need to be more radical. Here, opinions will differ.

(Lago et al. 2025) developed a roadmap outlining a set of areas that should be addressed to integrate sustainability in Software Engineering education. This can be considered a set of leverage points for computing education when aiming to integrate sustainability, both from an incremental and a transformative perspective:

- raising awareness
- establishing harmonized and clear sustainability concepts
- integrating ethical thinking
- creating a holistic view
- establishing sustainability metrics and indicators
- integrating SE competencies for sustainable software
- integrating skills for inter- and intrapersonal teamwork
- building the business case for sustainability
- changing cultures through advocacy and lobbying
- reorienting AI to drive sustainability
- activating academic orientations
- facilitating industrial adoption
- identifying greenwashing.

4.7. Examples from NTNU

In what follows we briefly present three examples from the NTNU context, reflecting the author's affiliation to NTNU Department of Computer Science (IDI).

4.7.1. Case 1: Integrating sustainability on a study program level

In the period 2019-2021 NTNU conducted the FTS project - Technology Education of the Future[5]. The project addressed what NTNUs technology education should look like in the future and resulted in a set of 10 overarching principles as well as a set of competence profiles for various types of study programs. The principles include holistic competence; contextual learning; student active learning, relevant assessment and good learning culture; educators' competence development; holistic thinking in study

programs and study program portfolio; continuous improvement and culture for quality; international collaboration on quality in education; systematic collaboration with work life; infrastructure for learning, health and wellbeing.

Considering these principles in light of the competence framework by Redman and Wiek (Redman and Wiek 2021) (Figure 4) we recognize several principles as addressing sustainability competencies. The principle of holistic competency indicates that our education should aim for sustainability as an integral part of students' professional identity and competency. The principle of holistic thinking in study programs points to the need for sustainability to be integrated throughout a study program, the progression in development of sustainability competencies being closely linked to the development of other competencies (e.g. disciplinary competencies) and contextualized in the discipline. Also, the principles of international collaboration and collaboration with work life implies addressing sustainability in industry, at the research front and in society at large. It is tempting to add here that NTNU should be actively leading in this regard, not depend on being dragged along by other stakeholders.

NTNU has started applying the insights from FTS in their development of their technology education, and will continue this effort. The SEED Center for Science and Engineering Education Development[6] has a role in this. For all of us involved in the development of study programs in technology, the FTS principles can be regarded as a guide grounded in NTNU strategy, providing an anchoring in the institution of sustainability as essential and integral to education.

An example of a study program that has actively integrated sustainability over the last years, is the Bachelor's Degree Programme in Computer Science[7], which is a three year engineering program. The study program leaders have actively worked on sustainability competencies as a common thread in the program, making sure that each semester contains one course addressing sustainability and taking into account that the students' sustainability competence should develop through the courses in question.

4.7.2. Case 2: Integrating sustainability in HCI courses

HCI traditionally addresses the interaction of one human with a computer system. The growing field of sustainable HCI (Hansson et al. 2021) broadens the scope of HCI to address the broader impacts of technology.

In three different introductory HCI courses at IDI we have integrated sustainability analysis by use of SusAF, actively involving course staff in collaboration with computing education researchers. The interventions all represented an incremental approach to change where sustainability analysis and associated activity was incorporated in the curriculum without change of intended learning outcomes for the course. Rather, the interventions were meant to strengthen students learning towards existing outcomes linked to experiencing project work in user centered design.

The interventions and associated research activities have resulted in new insight about how sustainability can be included in introductory courses in HCI/user centered design. This includes lessons learned on

- aligning sustainability analysis with the development of personas and scenarios
- encouraging reflective learning on sustainability within a design context, e.g. as part of HCI practice
- broadening designers' perspective from the traditional HCI focus on the user and their interaction with the technology to the wider context, e.g. possible effects of the technology and

its use on society and the environment, in line with the more recent tradition of Sustainable HCI.

- integrating sustainability in large scale, introductory project courses with an incremental approach, e.g. through targeted lectures and modified assignments

For more details, see (B. Krogstie and Krogstie 2020; Jaradat et al. 2026; Jaradat and Krogstie 2025).

4.7.3. Case 3: Sustainability in bachelor and master theses

A challenge that has not yet been fully resolved at NTNU is how to ensure sustainability is addressed in bachelor and master theses. Based on the university strategy and the FTS project along with its sister project for study programs in social science and the humanities, it was clear that sustainability should somehow be addressed in the thesis. The Faculty of Information Technology and Electrical Engineering decided that the thesis course descriptions include a statement about the thesis *reflecting on the relevance for sustainability, starting with the UN Sustainable Development Goals*. The course description in question was modified to include versions of this statement.

Students and supervisors found the formulation to be rather vague, and the relevance of sustainability for specific types of projects was questioned. As a collaboration between IDI and Excited SFU a web site https://www.ntnu.no/excited/b%C3%A6rekraft-i-it-utdanning was launched in April 2024 to provide students and supervisors with 1) more background on sustainability in general (with relevant links to external resources) 2) motivation for why sustainability should be addressed in the thesis, and 3) a guide to how sustainability can be addressed by aid of SusAF. For the latter, a brief introduction to the framework was included, along with a set of examples of analysis based on prior master theses. In the first version of the website, the framework was modified to include only four dimensions (the individual and social being merged), whereas in the next version this was abandoned and the original SusAF used instead.

Evaluation and further development of the website is still ongoing. Focus here in this chapter is not so much on the website itself as on the insights we got from subsequent feedback from teaching staff in seminars, a focus group with students, and a set of in-depth interviews with supervisors. At the time of writing analysis of this data is not complete, but the ongoing work has led to useful insights.

4.8. Insights on sustainability integration in CS education

Our work to integrate sustainability in computing education at NTNU has resulted in some lessons learned. We share some of them in the final part of this chapter.

4.8.1. Sustainability as part of the profession and the scientific discipline

We strongly argue that to get the most out of PBL, including the development of students' sustainability competencies, sustainability should be addressed by the students as part of the work process (i.e. design/development). It belongs at the core of the experiential learning of *doing computing work* and should not be deferred e.g. to a requirement to add a section in the final project report.

A success criterion when integrating sustainability in computing education could be students' professional identity: having experienced the inclusion of sustainability as a part of practice/discipline and thus authentic and meaningful, the student should see themselves as *being a professional who is aware of sustainability challenges and can address them as part of their work*.

Our efforts to integrate sustainability in master and bachelor projects have brought forward different perspectives, both among teaching staff and among students. We note that to understand the issues and find viable solutions, it is necessary to distinguish between issues related to

- the needs of different computing disciplines (e.g. information systems, high performance computing)
- different types of project (sustainability being part of the project purpose vs. sustainability effects as a set of consequences following from the project)
- characteristics/norms related to the thesis genre (e.g. whether and where sustainability analysis/reflection belongs in a thesis)
- study program structure and progression (e.g. where should sustainability competencies be developed, and what competencies – this might include skills in using specific approaches of frameworks like SusAF)
- questions of constructive alignment (e.g. related to assessment criteria and the role of the external examiner)
- traditions of higher education (e.g. what is a university, what is a master/bachelor degree)

A preliminary insight from our work in this area is that to develop and successfully implement meaningful and more unified guidelines for addressing sustainability in bachelor and master projects, there is a need for transparency and discussion among the stakeholders. The above list of issues represents areas of diverging perspectives and could inform the discussion.

At NTNU we have successfully applied SusAF in several courses and study program and recommend it as an accessible tool. Here is our short, experience-based list of what to consider/include in a workshop:

- Read and understand the elements of the framework
- Determine what, in your context, is the purpose of the workshop
- Decide what solutions/technologies/designs are to be analyzed, whether they are the part of existing group projects or a case provided for the workshop
- Decide if you need assistants; train them by at least having a meeting going through an analysis example step by step
- Find a space: On-site is preferable but digital is also possible
- Make available a template for the diagrams; digital or on paper/cardboard (preferably at least A1). For the paper version, you also need post-its for the effects, and pens
- Divide participants into groups; they can be already established ones (e.g. student projects) or created for the occasion. Only one group is also fine in a workshop; max 7 participants.
- Prepare the schedule; aim for a 1,5 – 2-hour session
    - Start the session with a presentation of the framework, its elements and purpose along with an example or two
    - Let the groups' analysis work start with an individual step of identifying sustainability effects before turning to collaboratively filling in the diagram with effects and links between them
    - Provide the groups with the guiding questions, but let them start without them
    - Have some form of presentation of results. If there is more than one group, students benefit from other groups' reflections and results.

For a more elaborate discussion and example of how to apply the framework in a HCI education setting, see (Jaradat and Krogstie 2025; Jaradat et al. 2026).

The use of SusAF and similar frameworks implies systematically considering sustainability effects in various areas and with different time frames. It also implies acknowledging uncertainty: An effect may be possible without being certain. Furthermore, a lack of expertise in an area (e.g. business economics, human factors in the workplace) might impact on the possibility to identify reasonable sustainability effects on one or more sustainability dimension. We believe that the exercise of discussing, e.g. among team members in a software engineering student project, without having expertise in the areas in question, can still be fruitful. Speculating on possible scenarios of use of a solution can trigger reflection and support knowledge building and the development of shared understanding among the participants. The identification of a possible effect and the associated uncertainty can also be an inspiration to further exploration of weakly understood issues, e.g. by consulting relevant expertise, ultimately benefiting the identification of requirements. When applying SusAF in context of computing education, we thus recommend not to worry too much if the student teams lack deep knowledge of the application domain, any of the five sustainability dimensions, or competence in systems thinking. The analysis is likely to be beneficial and help trigger interest and motivation for further learning in areas where the students perceive a lack of competence. It should however be noted that all the dimensions of sustainability will not be equally relevant in all projects, In our experience, it makes sense to let students focus and elaborate on dimensions of sustainability most relevant to their project (especially in the case of a bachelor or master project) – possibly starting with a high level analysis to identify which dimensions are most relevant.

We also believe it is worth considering whether the students should become familiar with an analysis framework early in the study program so that they can easily apply it – possibly in a more advanced or elaborate way – later on, e.g. as part of a bachelor or master project.

Our experience with SusAF indicates that while the framework originates in Software Engineering, it is applicable in slightly different contexts, e.g. Information Systems or User Centered design. In any case, for the analysis it is essential to clarify what is at the center of the diagram.

4.8.2. Competencies and their progression

When considering intended learning outcomes on the study program level as well as the progression of competence development throughout the program, it is important to take into account not only what *new* to integrate, but what is *already there* – ideally reflected in the learning outcome descriptions. This was systematically addressed in the case of our bachelor programme in Computer Science (Case 1). A typical finding is that intended learning outcomes for project-based courses already cover several key sustainability competencies

A framework like the one by Redman and Wiek (see Figure 4) can facilitate the development of shared understanding among course staff of competencies targeted in a study program. The framework can be applied normatively as a means for examining an existing or planned curriculum to evaluate its coverage of sustainability competencies and identify new or modified program level learning objectives. Working top-down, a next question can be what courses in the program provide, or could provide, the competencies in question, again taking into account that existing learning objectives and course contents are likely to address many of the competencies found in the framework. As frequently commented in discussions among colleagues on these topics, not all courses need to have sustainability explicitly included for the study program as a whole to provide the relevant sustainability competencies. As with

other learning objectives on a study program level, there is a need for a coordinated effort where the study program lead and the course responsible teaching staff together design for common threads, progression, and good alignment between the study program and its constituent courses. This can also be considered as a way of supporting a holistic approach to curriculum development in line with one of the FTS principles.

The process of addressing sustainability in a study program might involve

- revision and quality improvement of learning outcome descriptions to include relevant competencies or make them more explicit
- clarification of the possible need for dedicated modules and courses on sustainability, weighed against the possibility for including sustainability by integrating it into existing courses
- identification of needs to clarify students' initial competence level when entering the course or study program. New students' "baseline competence" in the area of sustainability is likely to be significantly higher than it was a few years ago, and there may be a need to investigate this, e.g. by use of a survey among new students at the start of their first semester

4.8.3. Tools and methodologies

In this chapter we have particularly stressed two tools for addressing sustainability: The SusAF framework for analysis of sustainability effects and their interconnections, and the competence framework of Redman and Wiek. These frameworks are not the only ones that could serve similar purposes, but both are relatively established and, in our experience, comprehensible and applicable for stakeholders in sustainability analysis and curricular design. SusAF has furthermore proved to be relatively versatile in terms of course context and type of solution addressed. The application of the framework to projects that are not about the development of a technology is also useful, but requires some tailoring and agreement on what is being analyzed.

Some sustainability-related models, like the one on circular economy, may be particularly relevant for certain courses. Models capturing aspects of sustainability, with perspectives ranging from business value through user needs to power usage, are indeed already part of our curricula. The terminology varies, though; sustainability in a broad sense is partly being addressed by use of other terms. Often this is unproblematic – the students learn about important things whether we use one term or another - but it may also be a challenge to our collaborative work with computing education curricula xthat we use terminology inconsistently. One trap to avoid is to assume that those applying other terms, believe sustainability is unimportant!

From the research on integrating sustainability in master and bachelor theses we know that the use of terminology is important in the work to create shared understanding among academic staff. One type of comment from educators points to sustainability as a vague concept. Another is that training of teaching staff is needed in this area, which is sometimes followed by a comment on the lack of time, support and/or relevant course offerings. These viewpoints have a least two implications. First, there is a need among teaching staff for knowledge building and experience sharing in the area of sustainability in computing and computing education. This might be catered for through seminars and management support

Second, there may be some reluctance towards the concept of sustainability, possibly related to professional identity and even political stance. While incentivized training might result in a more positive attitude, it cannot be assumed that everyone will be on board with the idea of sustainability as meaningful and relevant to their teaching.

To address some of the above, Excited has developed a web resource[8] to support students and supervisors in addressing sustainability in master and bachelor theses. Included on the website are examples of SusAF-based sustainability analysis conducted as part of bachelor and master project, and examples of how sustainability as a topic has been successfully integrated in theses, e.g. as part of problem definition, requirements identification and/or reflection on the final result. The web resource is under regular revision based on feedback from students and scientific staff.

Finally, there is of course the use of AI and its impact on sustainability. A prominent sustainability-related issue is the environmental load of the rapidly growing number of huge data centers. Also, with the current global unrest, the use of AI in military systems and operations might have environmental, economic and social effects dwarfing most other concerns.

Leave the latter aside here, we observe two main angles on the use of AI for us in computing education:

First, there is the use of AI as part of the *domain of computing*. For instance, when conducting a sustainability analysis of an AI-based solution, the students should consider possible environmental, social and economic effects – intended and unintended ones - and dilemmas in balancing related concerns. Or, considering the design of an algorithm or the choice of a programming language in a solution, how might the choices made influence the resource consumption of the final, operating solution? Finally, how could AI be actively used for purposes of achieving sustainability objectives, e.g. in student projects, enabling processes and behaviour to the benefit of sustainability?

Second, there is our own use of AI as institution, as individual educators/scientists, and as students: Are we actively seeking to influence the technological development (e.g. choice of technology, areas of application, adherence to norms and regulations) or just trying to hang on to it? Do we actively consider sustainability as an important dimension when discussing current and future use of AI? The attitudes and above all actions demonstrated by computing education institutions and educators in this area should reflect what we want to teach our students.

4.8.4. Transformation

The issue of whether computing education should be transformative or not is not simple. Most of us in the sector would agree that university education needs to encourage innovation and enable our students to become “change agents” when they enter work life.

A university is an arena for research activity within established disciplines, each with their international communities, bodies of research, established norms and truths. There is both inertia and drivers for change inherent to the disciplines. Regarding teaching, the university builds on tradition and is, for better and for worse, a preserver of well-tried approaches while striving to adapt to updated knowledge about learning, changing industry needs, and the development of technology like AI.

The experience shared in this chapter is inclined towards incremental change. We nevertheless suggest putting transformative education higher up on the agenda, considering the key role of computing for sustainability and the current inertia in addressing sustainability in the computing field and in a society entirely reliant on digital solutions. A question we as educators should not avoid is whether sufficient change can happen within existing structures and systems or whether we need to challenge these structures – which includes challenging those who fund our activity, challenging the expectations of our students, and challenging the idea of what is the role of a university in our society. Who can make changes, who *should* make changes, and how can we make them happen? What does industry and the world around us need and expect from us?

We let these important questions serve as the concluding remarks of this chapter.

Copyright

Sustainability in Computing Education © 2026 by Birgit R. Krogstie and Bashira Jaradat is licensed under CC BY-NC-ND 4.0



References

Abreu, Cathy d', Clive Belgeonne, Douglas Bourn, and Jenny Hatley. 2025. *Transformative Learning for a Sustainable Future*. DERC Research Paper No. 24. Development Education Research Centre.

Becker, Christoph, Stefanie Betz, Ruzanna Chitchyan, et al. 2016. "Requirements: The Key to Sustainability." *IEEE Software* 2016 (January/February).

Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., & Venters, C. C. (2015). *Sustainability design and software: The Karlskrona manifesto*. In *Proceedings of the 37th International Conference on Software Engineering* (Vol. 2, pp. 467-476). IEEE. https://doi.org/10.1109/ICSE.2015.179

Bianchi, Guia, Ulrike Pisiotis, and Marcelino Cabrera Giraldez. 2022. *GreenComp - The European Sustainability Competence Framework*. Edited by Y. Punie and M. Bacigalupo. Publications Office of the European Union.

Biggs, John, and Catherine Tang. 2011. *Teaching for Quality Learning at University*. Fourth Edition. Open Univ Press.

Blumenfeld, Phyllis C., Elliot Soloway, Ronald W. Marx, Joseph S. Krajcik, Mark Guzdial, and Annemarie Palinscar. 1991. "Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning." *Educational Psychologist* 26 (3 & 4): 369–98.

Boud, David, Rosemary Keogh, and David Walker. 1985. *Reflection: Turning Experience into Learning*. RoutledgeFalmer.

Bygstad, Bendik, Birgit R. Krogstie, and Tor Morten Grønli. 2009. "Learning from Achievement: Scaffolding Student Projects in Software Engineering." *International Journal of Networked and Virtual Organizations* 6 (2).

Clear, Tony, and et al. 2001. "Resources for Instructors of Capstone Courses in Computing." Paper presented at ITiCSE '01.

Duboc, L., Betz, S., Penzenstadler, B., Kocak, S. A., Chitchyan, R., Leifler, O., Porras, J., Seyff, N., & Venters, C. C. (2019). *Do we really know what we are building? Raising awareness of potential sustainability effects of software systems in requirements engineering*. In *2019 IEEE 27th International Requirements Engineering Conference (RE)* (pp. 6-16). IEEE. https://doi.org/10.1109/RE.2019.00013

Eriksson, Elina, Anne-Kathrin Peters, Daniel Pargman, Björn Hedin, Minna Laurell-Thorslund, and Sandra Sjöö. 2022. “Addressing Students’ Eco-Anxiety When Teaching Sustainability in Higher Education.” Paper presented at ICT4S.

Hansson, Lon, Teresa Cerrato Pargman, and Daniel Pargman. 2021. “A Decade of Sustainable HCI.” Paper presented at CHI’21.

Hilty, Lorenz. 2012. “Why Energy Efficiency Is Not Sufficient—Some Remarks on ‘Green by IT.’” 13–20.

Holmén, Johan, and Johanna Lönngren. 2025. “Expanding Sustainability Learning in Engineering Education through Emotional Scaffolding.” *European Journal of Engineering Education*.

Jaradat, Bashira, and Birgit Krogstie. 2025. “Developing Sustainability Competencies through Design Work in an Introductory Human Computer Interaction Course.” Paper presented at SEFI.

Jaradat, Bashira, Birgit R. Krogstie, Georg Adrian Stoica, and Yngve Dahl. 2026. “A Contextualised Integration of Sustainability in a Human-Computer Interaction Course.” Paper presented at Accepted for EDUCON 2026.

Kolb, D. A. 1984. *Experiential Learning: Experience as a Source of Learning and Development*. Prentice Hall.

Krogstie, Birgit, and John Krogstie. 2020. “Introducing Sustainability in IT Education: The Case of a Course in User-Centered Design.” *Proceedings of FIE (Frontiers in Education)*.

Krogstie, Birgit R., and Monica Divitini. 2009. “Shared Timeline and Individual Experience: Supporting Retrospective Reflection in Student Software Engineering Teams.” Paper presented at Software Engineering Education and Training, 2009. CSEET ’09. 22nd Conference on.

Lago, Patricia, Nelly Condori Fernandez, Iffat Fatima, Markus Funke, and Ivano Malavolta. 2024. “The Sustainability Assessment Framework Toolkit: A Decade of Modeling Experience.” *Software and Systems Modeling*.

Lago, Patricia, Rogart Heldal, Stefanie Betz, et al. 2025. “A Roadmap for Integrating Sustainability into So□ware Engineering Education.” *ACM Transactions on Software Engineering and Methodology* 34 (5). https://doi.org/3708526.

Murugesan, San. 2008. “Harnessing Green IT: Principles and Practices.” *IT Professional* 10 (1): 24–33.

Operato, Luigi, Andrea Gallo, Erika A. E. Marino, and Daniele Mattoli. 2025. “Navigating CSRD Reporting: Turning Compliance into Sustainable Development with Science-Based Metrics.” *Environmental Development* 54.

Peters, Anne-Kathrin, Rafael Capilla, Vlad Constantin Coroama, et al. 2024. “Sustainability in Computing Education: A Systematic Literature Review.” *ACM Transactions on Computing Education* 24 (1).

Raworth, Kate. 2022. *Doughnut Economics: Seven Ways to Think like a 21st Century Economist*. Penguin Books.

Redman, Aaron, and Arnim Wiek. 2021. “Competencies for Advancing Transformations Towards Sustainability.” *Frontiers in Education* 6 (November).

Redman, Aaron, Arnim Wiek, and Matthias Barth. 2021. “Current Practices of Assessing Students’ Sustainability Competencies: A Review of Tools.” *Sustainability Science* 16: 117–35.

Sindre, Guttorm, Michael Giannakos, Birgit Krogstie, Robin Munkvold, and Trond Aalberg. 2018. “Project-Based Learning in IT Education: Definitions and Qualities.” *UNIPED* 41: 147–63.

SUSO Sustainable Software. 2026. “Sustainability Awareness Framework Overview and Guideline.” https://www.suso.academy/en/sustainability-awareness-framework-susaf/.

UNESCO. 2018. *Issues and Trends in Education for Sustainable Development: Education on the Move*.

Wiek, Arnim, Michael J. Bernstein, Rider W. Foley, et al. 2015. “Operationalising Competencies in Higher Education for Sustainable Development.” In *Handbook of Higher Education for SUstainable Development*, by M. Barth, G. Michelsen, M. Rieckmann, and I. Thomas. Routledge.

Wiek, Arnim, Lauren Withycombe, and Charles L. Redman. 2011. “Key Competencies in Sustainability: A Reference Framework for Academic Program Development.” *Sustainability Science* 6 (2): 203–18.

5. The Educational Production Gap

Ole Goethe

Nord University

5.1. Introduction

5.1.1. Context and Motivation

Before I started teaching, I spent 10 years making games in the industry, working on titles for PlayStation, Xbox and GameCube. In a studio, when a project started to slip, there was a producer, a schedule, a team that had shipped together before, and a contract that made sure everyone showed up on Monday. When I moved into higher education and began supervising student game projects, I found almost none of that structure in place. What I found instead were talented students running into problems that had very little to do with talent.

One early experience has stayed with me. At Hedmark University College (now Inland Norway University of Applied Sciences), I supervised a class of more than twenty animation students working on a production for an industry partner. Students earned a screen credit if the company's supervisor approved at least six of their animation assets. Five of them did. What separated those five, as far as I could see, was rarely raw skill. It was how they planned their work, how they used feedback, and whether they finished things. I have seen the same pattern in almost every cohort of game projects I have supervised since, at Inland University and later at Kristiania University of Applied Sciences. It becomes very visible when you are sitting across from a team at week eight and watching them realize, for the first time, that they have built half of everything and finished none of it.

Across these cohorts, technical limitations were rarely the main cause of a project going wrong. The breakdowns came from coordination, expectations and team process: missed deadlines, unclear roles, uneven contribution, creative disagreements, and the roller coaster of anxiety the night before submission. The teams that struggled were not short of effort. They had simply never been given the tools to organize themselves, resolve disagreements, or recover from the moment when the project feels heavier than the time left to finish it.

I think of this as the educational production gap, by which I mean the distance between what student teams are asked to deliver, a finished and playable game, and the production structure they are given to deliver it. In industry, that structure comes from producers, schedules, experienced colleagues and contracts. In education, students are expected to build the game and the structure at the same time, usually without ever having seen either done before. This chapter is about closing that gap.

Game development has become an established creative and technical discipline in higher education, and team-based game projects now appear in game design programs as well as in computer science, digital media and interaction design curricula. They are rich learning experiences. Students combine technical and artistic skills, collaborate across disciplines, manage time and resources, and deliver a working product to a real audience. They are also, as this chapter will argue, among the most demanding project types to supervise well.

This chapter combines that experience with existing research on project-based learning, agile-inspired educational practice and student project supervision. It addresses three related topics: how student game teams differ from professional production teams, how agile practices can be adapted to the constraints

of education, and what supervisors can do at each stage of a project to help teams finish well. Building on these insights, the Agile Learning Team Framework (ALTF) introduces a three-phase, supervisor-facing model that helps educators anticipate common challenges in student game projects and intervene at the moments when their guidance matters most. The chapter is written for educators, program leaders and industry mentors who supervise team-based game projects, and it assumes no deep background in either game development or project management.

5.1.2. Why Project Management Matters in Student Projects

A common view among educators and students alike is that project management belongs to industry, not to education: the messiness of the process is part of the learning, and students will figure it out. There is some truth in this, but it sells students short. Project-based learning works best when students are supported in organizing and sustaining their own work, not just given a problem and left alone with it (Blumenfeld et al., 1991; Thomas, 2000). The stakes are also higher than they may appear. Final-year game projects often carry a large share of a student's grade and become the portfolio pieces they show employers. A project that collapses because of poor planning or team breakdown affects more than a grade, and it can shake a student's confidence and shape their career.

Game projects are a natural fit for agile approaches. They involve many types of work, require repeated testing and refinement, and depend on both technical and creative problem-solving. Agile methods favor flexibility, collaboration and progress in small steps over rigid plans and heavy documentation (Beck et al., 2001). Transferring professional frameworks directly into education is, however, problematic. Student teams differ from professional teams in experience, motivation, assessment, available time and stability, and the educator's role is quite different from that of a project manager, balancing guidance, autonomy, feedback and assessment. Agile methods therefore must be adapted rather than copied. The ALTF is my attempt at that adaptation.

5.2. Agile Practice and Supervision in Student Projects

This section draws briefly on project-based learning and in more depth on two bodies of work that the ALTF builds on most directly: agile-inspired educational practice and the supervision of student project teams.

5.2.1. Project-Based Learning in Brief

Project-based learning (PjBL) has deep roots in Dewey's (1938) experiential philosophy and Kilpatrick's (1918) project method. Later work identified the features that make it effective: a driving question, sustained inquiry, authenticity, student voice and choice, reflection, critique and revision, and a public product (Krajcik & Shin, 2014; Larmer & Mergendoller, 2010; Thomas, 2000). Student game projects tick almost every box. They pose authentic creative and technical challenges, require sustained effort over a semester, and end in a playable product that others can experience and judge. Whitton (2010) and Squire (2011) argue that making games develops design literacy, systems thinking and collaborative skills at the same time.

The different forms of PjBL and their place in higher education are treated in their own chapter in this book. The point here is narrower. PjBL explains why game projects are valuable learning experiences, but it says relatively little about how a student team should organize its work from one week to the next. That is where agile practice comes in.

5.2.2. Agile-Inspired Educational Practices

The Agile Manifesto (Beck et al., 2001) placed individuals and interactions, working software, collaboration and responsiveness to change at the center of development work. Its influence has spread well beyond software engineering, and Scrum has become a common choice for capstone and team-based software courses (Schwaber & Sutherland, 2020). Reports from such courses describe benefits for coordination, estimation and student satisfaction, alongside clear warnings about workload (Mahnic, 2012).

Three lessons recur, both in the literature and in my own teaching. First, full Scrum is usually too heavy for student teams. Its roles, ceremonies and artefacts compete for time with the work itself, and a team of four students with three other courses cannot sustain the rituals of a professional studio. Kniberg's (2015) account of Scrum in practice makes a related point: the framework works best when teams adapt it to their own constraints rather than applying it by the book. Second, the value lies in agile's core logic rather than its formal machinery: short iterations, a visible backlog, regular review and honest retrospection. Third, the supervisor's role changes. In an agile-inspired project the supervisor works less like a lecturer and more like a coach, facilitating the team's own process and helping remove what blocks it.

Design matters as much as intent. Berkling and Thomas (2013) analyzed a gamified software engineering course that did not work as planned and showed that the way feedback and motivation structures are designed can decide whether students engage or disengage. The same holds for agile practices. Introduced as mechanical requirements, stand-ups and retrospectives quickly become performative. Introduced with a clear purpose, they give students a rhythm of planning, doing and reflecting that echoes the experiential learning cycle (Kolb, 1984) and structured reflection (Gibbs, 1988), but built into the work rather than added at the end.

Adaptations of Scrum for education already exist. The best known, eduScrum, translates Scrum's roles and ceremonies into the classroom, with students working in self-organizing teams and the teacher acting as product owner (Delhij et al., 2015). The ALTF differs in three ways. It is written for the supervisor rather than for the team. It is phase-sensitive, tying each intervention to the point in a project where a specific problem tends to appear. And it is built around a deliberate withdrawal of support over time, rather than a constant process repeated every sprint.

5.2.3. Supervising Student Project Teams

Supervision of student projects is best understood as a reflective and collaborative practice rather than a one-way transfer of expertise (Schön, 1983; Pearson & Kayrooz, 2004). Its central challenge is calibration: knowing how much to intervene, and when. The concept of scaffolding, temporary support that is gradually withdrawn as the learner becomes capable of working alone (Wood et al., 1976), is useful here, as is Vygotsky's (1978) zone of proximal development. A student team needs help with what it cannot yet do on its own, and that help should fade as the team matures. This principle sits at the heart of the ALTF.

Supervisors also shape the climate in which a team works. Psychological safety, the shared belief that it is safe to speak up, take risks and admit mistakes, is one of the strongest predictors of team learning and effectiveness (Edmondson, 1999; Frazier et al., 2017), a finding that also became widely known through Google's internal research on team effectiveness (Duhigg, 2016). In a game team, where creative ideas are exposed to criticism every week, a supervisor's tone strongly influences whether students raise problems early or hide them until it is too late.

Accountability is the third concern. Self-determination theory suggests that controlling external pressure can undermine the intrinsic motivation that makes students care about their games in the first place (Ryan & Deci, 2000). The supervisor's task is to create structures firm enough to keep progress moving but flexible enough to preserve autonomy, competence and a sense of belonging in the team. In my experience, accountability that runs between team members, through shared sprint goals and peer-facing reviews, does this far better than accountability that runs only to the supervisor. Autonomy, competence and relatedness also give the ALTF its theoretical spine, as Section 4 shows.

Finally, no supervisor is equally expert in programming, art, design and audio. A supervisor with a computer science background can give valuable feedback on code but may have little to say about the visual design of a game's interface, and the reverse is true for a supervisor from design. Multi-voiced supervision, in which several perspectives contribute to the guidance students receive (Dysthe et al., 2006), offers one answer: co-supervision, external jurors, and industry mentors or domain experts at playtesting sessions. Assessment raises a related problem. Because games are experienced through play, an embodied and partly subjective experience (Keogh, 2018), judging their quality is more complex than judging a functional system, and involving students in shaping the assessment criteria can make expectations clearer for everyone.

5.3. What Makes Student Game Projects Different

The previous section described what agile practice and supervision research offer. This section turns to the specific context they must work in, what student game projects look like in practice, and why they are harder to manage than they first appear.

5.3.1. Student Teams Are Not Small Studios

In industry, a game team usually works together for months or years. Membership is stable, roles are defined by hiring, and there are established routines for onboarding, communication and conflict resolution. When something goes wrong, there are producers, HR processes and contracts to fall back on (DeMarco & Lister, 2013). Student teams work very differently. Membership is often assigned or chosen based on friendship rather than complementary skills. The project ends when the semester ends, whatever state it is in. Students are learning the skills they need while they use them, alongside other courses and personal obligations, and the only accountability structures are academic ones.

Research on student software projects documents the recurring results: unclear task ownership, difficulty sustaining motivation, communication breakdowns, and strong individuals who either dominate the project or become its bottleneck (Fincher et al., 2001). Many student teams never reach the "performing" stage of group development (Tuckman, 1965) before the project is over. Figure 1 summarizes the contrast. A professional team aims for a product, while a student team must produce both a product and learning, with far less stability to do it with.



Figure 1. Professional vs. student game team maturity

5.3.2. Creative Ambiguity and the 80-20 Problem

What makes game projects especially tricky, compared with other student software projects, is creative ambiguity. In a standard software project, it is usually possible to define done: the feature works, the test passes, the requirement is met. In a game, done is much harder to pin down. A mechanic can function correctly and still feel wrong, a level can be completable and still be dull. Game design literature therefore treats playtesting and iterative, feel-based refinement as core practices rather than extras (Fullerton, 2018; Schell, 2008). Students are managing not just a technical build but an ongoing, emotionally charged judgement about whether what they are making is any good. Because they feel strong ownership of their art styles and mechanics, critique also lands differently. A code review is professional feedback, whereas being told your game is not fun yet can feel like a personal verdict, much as critique sessions in design studios often do (Dannels, 2005).

Closely related is what developers often call the 80-20 problem, a game-production version of the "ninety-ninety rule" that Bentley (1985) attributes to Tom Cargill: reaching a game that is 80 per cent complete takes roughly 20 per cent of the time, while turning it into a polished, bug-free and genuinely enjoyable experience takes the remaining 80 per cent. Students without prior production experience rarely appreciate this until they meet it themselves, and by then it is often too late. The pattern repeats with striking consistency. A team spends weeks six to nine adding features, convinced it is ahead of schedule, and then reaches week eleven, the point where it should be freezing features, with a game that

runs but does not yet feel like a game. Showing new students concrete examples from previous cohorts, including projects that failed, shifts their expectations in a way that no abstract warning quite manages. Game jam research is useful for the same reason: by compressing the whole arc of a project into 24 to 72 hours, jams make scoping failures and creative conflicts easy to see (Kultima, 2015; Fowler et al., 2013).

5.3.3. Interdisciplinary Friction

Games require a mix of technical and creative skills that rarely exists in one person, which is one of the strongest arguments for team-based projects. Interdisciplinary work is nonetheless difficult. Students from different disciplines bring different views of what counts as knowledge, evidence and good practice (Klein, 1990). A computer science student and an art student may disagree not only about what makes a good game, but about how decisions should be made and which work matters most at a given stage. Status perceptions add to this: programmers may underestimate how long art production takes, and artists may feel their work is valued less in a technically oriented program. These tensions rarely resolve themselves. They need to be made explicit early, and team charters that record agreed norms, roles and communication routines are an effective way to do so (Oakley et al., 2004).

5.3.4. Continuity and Motivation

Student teams also face a continuity challenge. Unlike industry teams, they assemble, work together and disband within a single academic period that is itself broken up by reading weeks, exams, holidays and deadlines in other courses. Momentum built over several productive weeks can disappear during a two-week break, and rebuilding it takes time that is rarely planned for. A team that loses two weeks to exams in week six seldom recovers its rhythm by week eight. What looks like a motivation problem in the final stretch is often a continuity problem that was never addressed when it first appeared. Version control, shared documentation and regular sprint reviews can bridge these gaps, but only if students are taught to use them consistently and understand why they matter. Many students arrive at university without ever having used version control, so these tools belong at the core of project practice, not as optional technical extras.

Motivation itself is both a strength and a vulnerability. Many students are genuinely passionate about games, and game projects tend to raise engagement in a course (Wang, 2011). But enthusiasm is rarely spread evenly across a team, and it tends to fade as the project becomes harder and pressure builds. Good project practice harnesses that passion early and protects it when the middle of the project gets heavy.

5.4. The Agile Learning Team Framework (ALTF)

The challenges described in the previous sections are not random. They appear at predictable moments in a project's lifecycle, and that predictability is useful, because it means educators can prepare for them rather than only react. The Agile Learning Team Framework (ALTF) is a supervisor-facing model that organizes a student game project into three phases, Formation, Production and Professionalization, each with three targeted interventions (Figure 2). The week ranges used throughout assume a sixteen-week project. For shorter or longer projects the phases scale proportionally: Formation takes roughly the first fifth of the project, Production a little under half, and Professionalization the remaining third or so.

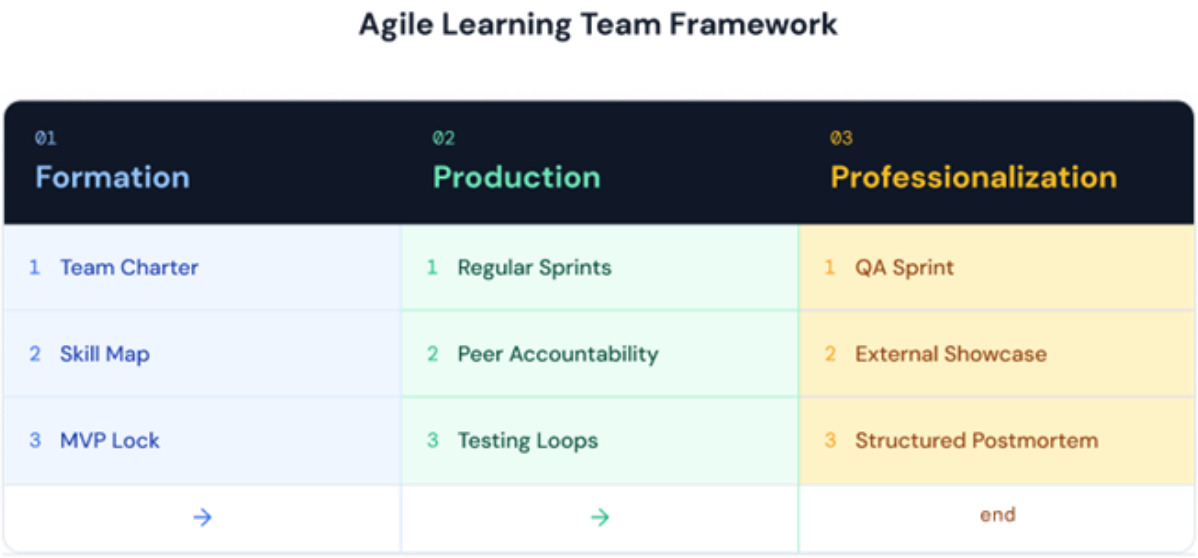


Figure 2. The Agile Learning Team Framework (ALTF): three phases, nine interventions

The ALTF is not Scrum for students. The goal is not to train students to perform professional ceremonies, but to extract agile's core logic, which I take to be iterative progress, regular reflection, scope discipline and team accountability, and strip it back to what a sixteen-week student project can sustain. It is a set of tools to be used selectively and with purpose, not a compliance exercise.

Underlying the framework is what might be called the supervisor intervention curve. Professional teams need little supervisory input at the start of a project, because their roles, tools and working relationships are already in place. Student teams are almost the inverse. They need the most support at the very beginning, when everything is being established for the first time, and that support should be reduced deliberately as the team develops its own rhythm and accountability. A supervisor who gives the same input in week twelve as in week two is either over-supporting a team that has found its footing or, more dangerously, under-supporting a team that is quietly falling apart. The ALTF formalizes this curve as scaffolding (Wood et al., 1976): intensive facilitation during Formation, coaching and monitoring during Production, and a more external, evaluative role during Professionalization.

The framework also has a motivational logic. Each intervention supports at least one of the three basic psychological needs described by self-determination theory (Ryan & Deci, 2000). The Team Charter, Peer Accountability and the External Showcase support relatedness, by making the team's shared commitments and its audience explicit. The Skill Map, MVP Lock, Testing Loops and QA Sprint support competence, by keeping goals achievable and progress visible. Regular Sprints and the Structured Postmortem support autonomy, by giving the team ownership of its own planning and reflection. Seen this way, the ALTF is not only a project-management aid but a way of protecting the motivation that makes students care about their games in the first place.

5.4.1. Phase 1: Formation (Weeks 1–3)

The opening weeks of a student game project are fragile. Energy is high and ideas are flowing, but underneath, the team is facing problems that will grow throughout the project if left alone. Roles are undefined, and students know they are a team but not what that means in practice. Ambition is high and experience is low, so the team has no way of judging how long its idea will take to build. Creative

conflict, which is a productive part of game making, feels threatening because the team has not yet built the trust to handle disagreement well. Phase 1 responds with three interventions.

Team Charter. A short document, written by the team, that records each member's role, working hours and preferences, and how the team will make decisions and resolve conflicts. Today it should also include an agreement on how AI tools will be used and who owns AI-generated output (see section "Agentic AI in Student Teams").

Skill Map. A simple overview that makes each member's strengths, and the gaps, visible to the whole team. It replaces assumptions and unspoken status dynamics with shared knowledge of who can do what.

MVP Lock. An agreement on the minimum viable product: the smallest version of the game that is complete and could be submitted. Prioritization techniques such as MoSCoW (must have, should have, could have, won't have) help teams separate the essential from the aspirational. This is the single most important thing for a supervisor to secure early.

In this phase the supervisor is a hands-on facilitator, running the charter and scoping sessions with the team rather than delegating them.

5.4.2. Phase 2: Production (Weeks 4–10)

Production is where the real work happens and where most crises appear. The excitement of the start fades somewhere in the middle of the project. This is normal and well documented, but it still catches teams off guard. Communication also breaks down, not because students stop caring, but because the informal coordination that worked at the start stops working once the project becomes complex and interdependent. Breaks in the academic calendar threaten continuity, and scope begins to creep. Phase 2 addresses these problems through three sustained practices, with one contingency intervention for teams that fall behind.

Regular Sprints. A short stand-up at the start of each week and a brief review at the end. The aim is not ceremony but rhythm: a predictable pulse that keeps work visible and makes drift noticeable early.

Peer Accountability. Sprint goals are shared with the whole team, not only with the supervisor. When students commit to each other, responsibility becomes shared rather than monitored, which is both more motivating and less stressful than reporting upwards. Where contributions are uneven, brief, structured peer evaluations at the midpoint and end of the project, using an instrument such as CATME (Comprehensive Assessment of Team Member Effectiveness; Ohland et al., 2012), can help identify imbalances early and provide the supervisor with a fair basis for addressing them.

Testing Loops. Every sprint ends with something playable to evaluate, not a checklist to tick. Playing the game together keeps attention on the experience and keeps the team's intrinsic motivation alive.

MVP Re-Lock (contingency). Not every team stays on course. If a member drops out, a break costs more momentum than expected, or the team is clearly behind by week seven or eight, the supervisor should call an explicit re-scoping session. The team returns to its MVP, decides what can realistically be finished in the time left, and moves everything else to "won't have". Treated as a normal, planned tool rather than an admission of failure, a re-lock lets a struggling team recover instead of quietly collapsing. Throughout Production, the supervisor now steps back into a coaching role, watching for early warning signals and running short retrospectives that ask what is slowing the team down, not only what it has built.

5.4.3. Phase 3: Professionalization (Weeks 11–16)

The final phase brings a different set of problems, ones many teams are unprepared for. The most persistent is the polish gap: the uncomfortable discovery that turning an 80 per cent game into a finished one takes more work than everything that came before. Teams that meet it unprepared tend to submit a game that is complete but has nothing that shines, with no time left to fix it. The second problem is portfolio readiness. Even teams that produce an excellent game often fail to document and present it in a way that communicates its quality to an outside audience. Phase 3 offers three interventions.

QA Sprint. A dedicated sprint, introduced together with a feature freeze, devoted to testing and improving what already exists rather than adding anything new. It helps the team resist the temptation to keep building when it should be finishing.

External Showcase. A jury session, public exhibition or industry playtest that raises the stakes productively. Knowing that people outside the course will play their game sharpens students' focus on quality and presentation. The showcase is also the natural deadline for portfolio material, such as a short trailer or gameplay capture, screenshots, and a one-page description of the game and each member’s contribution. Treating these as required deliverables rather than optional extras closes the portfolio readiness gap.

Structured Postmortem. A guided reflection on what worked, what did not and what the team would do differently. Postmortems are standard practice in the games industry, and they turn the project's experience into learning students can carry to the next one.

Here the supervisor takes a more external, evaluative role, and often must give teams explicit permission to stop building and start finishing.

5.4.4. Supervisor Quick Start Guide

Table 1 condenses the framework into a reference that a supervisor can return to at any point in a project without rereading the full description.

Table 1: ALTF supervisor quick start guide

Project Stage	Supervisor Question	Recommended Solution
Formation	Does this team know how to work together?	Facilitate a Team Charter session and complete a Skill Map
Formation	Is the idea realistic?	Run an MVP definition session and lock scope before week 4
Production	Are they progressing?	Establish a weekly sprint rhythm with stand-ups and end-of-week check-ins
Production	Are hidden problems emerging?	Run a short retrospective, ask what's slowing the team down, not just what they built
Production	Is the team falling behind?	Hold an MVP re-lock session and re-scope to what can realistically be finished
Professionalization	Is it finished or just functional?	Initiate a dedicated QA sprint, enforce a feature freeze
Professionalization	Can they explain their work?	Organize an external showcase or jury session followed by a structured postmortem

The questions in the middle column matter most, because a common supervision pitfall is diagnosing the wrong problem. A quiet, low-energy team in week nine is often the result of a scope problem that appeared in week two. The questions are designed to help supervisors look beneath surface symptoms and find the root cause early.

5.4.5. A Sixteen-Week Supervisor Timeline

For supervisors working within a standard semester, Table 2 translates the three phases into week-by-week focus areas. It is a pacing guide rather than a rigid schedule, and the point is to show what to pay attention to at each moment so that the right intervention happens at the right time.

Table 2: ALTF sixteen-week supervisor timeline

Week	Phase	Supervisor Focus
1	Formation	Team introductions, role discussion, charter session
2	Formation	Skill mapping exercise, begin MVP definition
3	Formation	Lock MVP and initial scope, establish sprint rhythm
4–5	Production	First playable prototype check, confirm sprint cadence is working
6–8	Production	Mid-project review, watch for motivation dips and scope creep, MVP re-lock if the team is behind
9–10	Production	Check continuity after any breaks, retrospective on process health
11	Professionalization	Feature freeze conversation, shift team focus to quality over quantity
12-13	Professionalization	Dedicated QA sprint, external playtesting if possible
14-15	Professionalization	Portfolio documentation, prepare for showcase
16	Professionalization	External jury or showcase event, structured postmortem

The most important weeks are weeks 2–3 and week 11. Weeks 2–3 are where scope is either locked or not, with consequences that play out for the rest of the project. Week 11 is where the feature freeze needs to happen, and where most teams need a supervisor's permission to stop adding and start finishing.

5.4.6. Common Supervision Pitfalls

Students are not the only ones who fall into predictable traps. Supervisors do too, and not through carelessness: these pitfalls often catch experienced supervisors precisely because the same habits work well in other settings. Being aware of them is the first step to avoiding them.

Pitfall 1: Reading enthusiasm as progress. The opening weeks are often the most energetic. Ideas are flowing, the team seems aligned, and it is easy to read the mood in the room as a sign that the project is healthy. Enthusiasm in week two says very little about that. What matters is whether the team has a locked MVP, a realistic scope and a shared understanding of what it is building. A supervisor who leaves week three without these in place, because the meeting felt good, is setting the team up for a painful week ten.

Pitfall 2: Waiting to be asked for help. Student teams are not good at recognizing when they are in trouble, and even less good at asking for help when they do. By the time a team raises a process problem, it has usually existed for weeks. Effective supervision is proactive and diagnostic. Missed sprint commitments, quieter stand-ups and one member doing most of the talking are all signals worth acting on before the team notices them itself.

Pitfall 3: Counting features instead of judging experience. It is tempting to measure progress by what exists: how many levels, mechanics or lines of code. A game with ten unfinished systems is almost always weaker than a game with one mechanic that has been properly designed, built and playtested.

Supervisors who show more excitement about new additions than about whether existing features are fun end up rewarding exactly the ambitious, unpolished projects that define the polish gap.

5.4.7. Adapting the ALTF

The ALTF is designed to be flexible. The week ranges assume a sixteen-week semester, and supervisors can shorten or extend the phases to fit their own program. What matters is not the timeline but the recognition that different challenges appear at different stages, and that support delivered when a challenge is most likely to occur is far more effective than generic advice. The framework also sits comfortably alongside Scrum or Kanban, and it is not meant to replace them but to help students apply them where they matter most.

A few practical choices make adoption more likely to succeed. Agile concepts should be introduced explicitly and early, with workshop time devoted to why iteration works and what makes a retrospective useful, rather than through a handout. Short, regular check-ins work better than long, occasional meetings that only happen when problems are already visible. Assessment should reward process as well as product, for example through marks for sprint reviews, retrospectives and individual reflection, so that students take the management side of the project seriously. And supervisors benefit from sharing experience with each other through regular debriefs, so that the framework is refined by the people who use it.

5.5. Discussion

5.5.1. Agentic AI in Student Teams

Agentic AI tools, which can plan and carry out multi-step tasks rather than simply answer prompts, are rapidly changing what a student team can produce in a semester. Coding assistants and AI asset generators let teams reach a playable prototype far faster than before, leaving more time for what matters most in game development: playtesting, iteration and refinement. In interdisciplinary teams they can also act as a levelling mechanism, letting a designer implement a mechanic they could previously only describe, or a programmer create placeholder art without waiting for an artist.

The risks map directly onto the challenges described in this chapter. The first is skill atrophy. AI assistants can already solve many typical programming assignments (Finnie-Ansley et al., 2022), so a finished-looking project no longer proves that its makers understand it. The second is an accountability gap. When large parts of a game are generated, ownership of creative and technical decisions becomes diffuse, and the reflective learning the ALTF is designed to support may not happen. The third concerns team dynamics, since a student who is fluent with AI tools can quietly become the dominant voice in the team. The fourth is scope. If AI makes features cheap to generate, scope creep gets worse, not better, because every generated feature still needs the same integration, polish and feel-based refinement, while the team's sense of how much is done is distorted by how quickly it appeared.

The ALTF does not prescribe whether AI should be used, since that depends on program context and learning goals. It does suggest where to address it: in the team charter during Formation, in sprint reviews that ask how something was built as well as what was built, and in the postmortem, where students can reflect honestly on what they learned versus what they generated. Treating AI as a new kind of team member, whose contributions must be evaluated, integrated and owned by the human students, fits naturally with the collaborative and reflective values the framework is built on.

5.5.2. Limitations and Cautions

Agile practices are no silver bullet. They take time, and in a twelve- to sixteen-week project even light ceremonies compete with production work. They can also become performative: students who do not understand why they hold a retrospective will produce generic observations that change nothing. The answer is not to drop the practices but to invest in explaining their purpose (Kniberg, 2015). Agile was also designed by and for software teams, and its emphasis on measurable output each sprint fits poorly with work that is hard to quantify, such as art production, narrative or game feel. Industry concepts need similar care. A vertical slice, a fully polished strip of the game built early to show its final look and feel, is standard in studios, but a student team that spends the first five weeks of a sixteen-week project building one is likely to end with an impressive but incomplete game. Finally, the ALTF is grounded in experience and in existing research rather than in systematic evaluation of the framework itself, and its value should be judged with that in mind.

5.5.3. Next Steps

That last point also sets the agenda for further work. The natural next step is to test the ALTF in practice: to examine more closely how student game teams differ from professional production teams, which agile adaptations work under which educational constraints, and which supervisory practices improve learning outcomes. Comparative studies across several programs and institutions would be especially valuable, since much of the current evidence on agile in student projects comes from single courses. There is also room for project tools designed for student teams rather than professional ones, and for drawing on gamification to make project management itself more engaging (Deterding et al., 2011).

5.6. Conclusion

Student game projects are among the richest learning experiences higher education can offer, and among the hardest to supervise. The challenges they bring, from overambitious scope and interdisciplinary friction to broken momentum and the polish gap, are not random. They arrive at predictable points in the project, which means supervisors can prepare for them.

The ALTF builds on that predictability. In Formation, the priority is team formation before task allocation: a charter, a skill map and, above all, a locked MVP, because scope managed early is a problem avoided later. In Production, the priority is rhythm and visibility: regular sprints, peer accountability and playable builds that keep the team focused on the experience rather than the feature count. In Professionalization, the priority is finishing well: a feature freeze and QA sprint, an external audience, and a postmortem that turns the project into learning. Across all three phases, the supervisor's role changes from facilitator to coach to evaluator, with support that is strongest at the start and fades as the team grows into its own process.

None of this requires students to become professional Scrum practitioners. It requires supervisors to bring a small set of agile practices to the right moment, explain why they matter, and pay attention to the human side of the project as closely as to the game itself. Teams that are helped in this way are far more likely to finish, and to leave the project knowing how to finish the next one. That, in the end, is what closing the educational production gap means.

Copyright

The Educational Production Gap © 2026 by Ole Goethe is licensed under CC BY-NC-ND 4.0



References

1. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development. http://agilemanifesto.org/
2. Bentley, J. (1985). Programming pearls: Bumper-sticker computer science. Communications of the ACM, 28(9), 896–901.
3. Berkling, K., & Thomas, C. (2013). Gamification of a software engineering course and a detailed analysis of the factors that lead to its failure. In Proceedings of the 2013 International Conference on Interactive Collaborative Learning (pp. 525–530). IEEE.
4. Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. Educational Psychologist, 26(3–4), 369–398.
5. Dannels, D. P. (2005). Performing tribal rituals: A genre analysis of 'crits' in design studios. Communication Education, 54(2), 136–160.
6. Delhij, A., van Solingen, R., & Wijnands, W. (2015). The eduScrum guide: The rules of the game. eduScrum.
7. DeMarco, T., & Lister, T. (2013). Peopleware: Productive projects and teams (3rd ed.). Addison-Wesley.
8. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining 'gamification'. In Proceedings of the 15th International Academic MindTrek Conference (pp. 9–15). ACM.
9. Dewey, J. (1938). Experience and education. Kappa Delta Pi.
10. Duhigg, C. (2016, February 25). What Google learned from its quest to build the perfect team. The New York Times Magazine.
11. Dysthe, O., Samara, A., & Westrheim, K. (2006). Multivoiced supervision of master's students: A case study of alternative supervision practices in higher education. Studies in Higher Education, 31(3), 299–318.
12. Edmondson, A. (1999). Psychological safety and learning behavior in work teams. Administrative Science Quarterly, 44(2), 350–383.
13. Fincher, S., Petre, M., & Clark, M. (Eds.). (2001). Computer science project work: Principles and pragmatics. Springer.

14. Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In Proceedings of the 24th Australasian Computing Education Conference (pp. 10–19). ACM.

15. Fowler, A., Khosmood, F., Arya, A., & Lai, G. (2013). The Global Game Jam for teaching and learning. In Proceedings of the 4th Annual Conference of Computing and Information Technology Research and Education New Zealand (pp. 28–34).

16. Frazier, M. L., Fainshmidt, S., Klinger, R. L., Pezeshkpour, A., & Vracheva, V. (2017). Psychological safety: A meta-analytic review and extension. Personnel Psychology, 70(1), 113–165.

17. Fullerton, T. (2018). Game design workshop: A playcentric approach to creating innovative games (4th ed.). AK Peters/CRC Press.

18. Gibbs, G. (1988). Learning by doing: A guide to teaching and learning methods. Further Education Unit.

19. Keogh, B. (2018). A play of bodies: How we perceive videogames. MIT Press.

20. Kilpatrick, W. H. (1918). The project method. Teachers College Record, 19(4), 319–335.

21. Klein, J. T. (1990). Interdisciplinarity: History, theory, and practice. Wayne State University Press.

22. Kniberg, H. (2015). Scrum and XP from the trenches (2nd ed.). InfoQ.

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

24. Krajcik, J. S., & Shin, N. (2014). Project-based learning. In R. K. Sawyer (Ed.), The Cambridge handbook of the learning sciences (2nd ed., pp. 275–297). Cambridge University Press.

25. Kultima, A. (2015). Defining game jam. In Proceedings of the 10th International Conference on the Foundations of Digital Games. Society for the Advancement of the Science of Digital Games.

26. Larmer, J., & Mergendoller, J. R. (2010). Seven essentials for project-based learning. Educational Leadership, 68(1), 34–37.

27. Mahnic, V. (2012). A capstone course on agile software development using Scrum. IEEE Transactions on Education, 55(1), 99–106.

28. Oakley, B., Felder, R. M., Brent, R., & Elhajj, I. (2004). Turning student groups into effective teams. Journal of Student-Centered Learning, 2(1), 9–34.

29. Ohland, M. W., Loughry, M. L., Woehr, D. J., Bullard, L. G., Felder, R. M., Finelli, C. J., Layton, R. A., Pomeranz, H. R., & Schmucker, D. G. (2012). The comprehensive assessment of team member effectiveness: Development of a behaviorally anchored rating scale for self- and peer evaluation. Academy of Management Learning & Education, 11(4), 609–630.

30. Pearson, M., & Kayrooz, C. (2004). Enabling critical reflection on research supervisory practice. International Journal for Academic Development, 9(1), 99–116.

31. Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist, 55(1), 68–78.

32. Schell, J. (2008). The art of game design: A book of lenses. Morgan Kaufmann.

33. Schön, D. A. (1983). The reflective practitioner: How professionals think in action. Basic Books.

34. Schwaber, K., & Sutherland, J. (2020). The Scrum guide. Scrum.org. https://scrumguides.org/

35. Squire, K. (2011). Video games and learning: Teaching and participatory culture in the digital age. Teachers College Press.

36. Thomas, J. W. (2000). A review of research on project-based learning. Autodesk Foundation.

37. Tuckman, B. W. (1965). Developmental sequence in small groups. Psychological Bulletin, 63(6), 384–399.

38. Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.

39. Wang, A. I. (2011). Extensive evaluation of using a game project in a software architecture course. ACM Transactions on Computing Education, 11(1), 1–28.

40. Whitton, N. (2010). Learning with digital games: A practical guide to engaging students in higher education. Routledge.

41. Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17(2), 89–100.

6. Educating for Generic Competencies

Rune Hjelsvold

Norwegian University of Science and Technology (NTNU)

6.1. Introduction

For more than a decade, research has shown that IT professionals need non-technical (generic) competences to succeed in their jobs. Aasheim et al. (2012) conducted a survey consisting of 48 competences that participants were asked to rate in terms of importance. There were more than three hundred respondents. The 12 most important competences were all generic competences, with the top three being *honesty/integrity*, *attitude* and *willingness to learn new skills*. Li et al. (2015) interviewed 59 experienced engineers at Microsoft to identify distinguishing attributes of great engineers. The interviews uncovered 53 attributes of great engineers. Some of the attributes were related to the qualities of the product that engineers create, but most of the attributes were generic, such as being *passionate* and *open-minded*, *handling complexity when making decisions*, and working well with team-mates. Stevens and Norman (2016) examined more than five hundred IT job advertisements and interviewed 26 industry participants to investigate what industry expects in IT candidates. Their research found that generic competences were considered extremely important by industry, with *communication*, *pro-active/self-motivated/autonomous*, *team player/collaboration* being the top three generic competences.

Given this history of research on the importance of generic competences in IT professions, it seems reasonable to expect that generic competences are well covered in current IT programmes. Groeneveld (2020) populated a database of nearly three hundred non-technical courses from more than one hundred computing programs spread over 30 European countries to investigate which competences are addressed and which still require more attention. They identified 50 different generic competences being addressed in the courses. 17 of these competences occurred in more than 5% if the courses. The top three competences were *communication*, *teamwork* and *ethics*.

In their work on constructive alignment, Biggs and Tang (2007) emphasize the importance of good alignment between intended learning outcomes (ILOs) at course-level and intended learning outcomes at programme-level and that the course level intended learning outcomes in total address the programme-level intended learning outcomes because a programme-level intended learning outcome often will be addressed by several courses, “from different and increasingly complex angles” (Biggs and Tag, 2007). The authors also highlighted the importance of intended learning outcomes, at both the course- and programme-levels, clearly specify “the appropriate level of understanding and performance intended” (Biggs and Tang, 2007).

In Excited, we investigated the intended learning outcome descriptions at the programme level for Norwegian bachelor programmes in IT (Brownridge et al., 2026). Using Bloom’s revised taxonomy (Krathwohl 2002) as an analytical lense, the research study found that 65% of the programme-level intended learning outcomes were aligned with lower-order cognitive processes (using verbs related to knowing, remembering, or understanding). Furthermore, the study found that only 2% of the programme-level intended learning outcomes explicitly specified the students’ expected degree of mastery. Even though the research study did not investigate generic competence learning outcomes specifically, these findings suggest that generic competences are mostly described at a low level. Also, the development of more advanced and complex generic competences is not well defined in the

programme-level intended learning outcomes and therefore not handled in a detailed and systematic way in these study programmes.

The abovementioned research study also showed that 84% of the programme-level intended learning outcomes were in the cognitive domain with as little as 12% of the programme-level intended learning outcomes in the affective domain, i.e., related to the development and internalisation of emotions, attitudes, values, and appreciations, which are important parts of generic competences. This suggests that generic competences related to the affective domain might receive more attention when IT study programmes are designed.

The main purpose of this chapter is to provide a useful foundation for addressing generic competences in a systematic way in study programmes. The main audience is the community of educators involved in the design of the study programme and of the individual courses/modules that make up the programme. The chapter explores the realm of generic competences, discusses the specification of generic competences in intended learning outcome descriptions, and investigates how generic competences can be progressively developed throughout a study programme. The aim is to provide a useful foundation for addressing generic competences in a detailed and systematic way in IT study programmes.

6.2. Exploring the Term Generic Competence

In EQF, the European Qualification Framework (European Union, 2017), competence is defined as *the proven ability to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development.* Similarly, the American Association for Computing Machinery (ACM), has defined a competency model consisting of four components (CC2020 Task Force, 2020): knowledge, skills, dispositions, and task (that represents the context for the competence). It is worth noting that personal dispositions and abilities are included in both definitions. Educators should, thus, pay special attention to what personal dispositions and abilities the students should develop when developing generic competence. It is also worth noting that both definitions highlight the importance of the context, the tasks, in which the students are expected to apply their competence. Educators should accordingly be deliberate about what context and tasks to select when designing learning activities for generic competence development to achieve proper alignment between the training context and target professional contexts and tasks.

There is no general agreement on what competences exactly are considered generic. Neither is there a general agreement of the naming of such competences. Other names being used may help promoting a better understanding of what generic competences refer to. For example, generic competences are often referred to by terms such as:

- *People skills*, *human skills*, and *soft skills*, which illustrates that generic competences often are used for competences related human aspects in contrast to science-specific competences.
- *Professional skills* or *employability skills*, highlighting the usefulness of such competences in a professional context – or to get a job in the first place.
- *Transversal skills* or *cross-cutting skills,* emphasizing that these competences tend to be useful across disciplines and in a variety of contexts and situations.
- *Transferable skills,* suggesting that these competences can be transferred to other contexts after being acquired in one context.

- *21st century skills* or, more generally, *future skills,* which is another related term focusing on the importance of competences that individuals will need in today's rapidly changing society and workplaces.

In summary, and in lack of an exact definition, generic competences can be thought of as

> *competences that can be useful in a variety of contexts and situations, that are possibly transferable from one context to another, that tend to focus on human aspects, and that are aimed at professional settings*.

There have been several efforts to define, describe and categorise generic skills. This chapter is reviewing and examining the work done by

- the CDIO (Conceiving, Designing, Implementing, Operating) initiative for engineering education (CDIO Office, 2023),
- the European Commission's ESCO (European Skills, Competences, Qualifications and Occupations) project (European Commission, 2025),
- the Jameel World Education Lab at MIT's HSX, i.e., the Human Skills Matrix (Stump, 2020),
- Ehlers' book chapter on future skills (Ehlers, 2024),
- UNESCO's Asia Pacific Education Research Institutes Network's work on transferrable skills (UNESCO, 2016), and
- NAIS', the National Association of Independent Schools, work on 21st Century Skills (Witt and Orvis, 2010)

plus research done on generic skills in IT Education by:

- Borges and Souza (2024),
- Caeiro-Rodriguez et al. (2021), and
- Sandstrak and Klefstad (2025).

These sources were found after an extensive, although not rigorous, search in databases for scientific papers as well as on Internet for the other sources, using terms such as "soft skills", "transversal skills", and "21st century skills". The IT papers were chosen because these are some of the more recent papers on generic competences in IT education. The other sources were chosen because these are some of the larger international efforts in defining generic competences. In the rest of this chapter, this set will be referred to as the *foundational sources* for the chapter.

As the basis for this chapter, a comparison matrix was created. In the matrix, columns were created for each of the sources listed above, while rows were created for what were considered unique competences - based on their name and descriptions. The row for the *collaboration* competence listed in the HSX (Stump, 2020), for instance, includes the names used in the other *foundational sources*: *teamwork*, *team collaboration*, *collaboration and teamwork – sometimes in multidisciplinary teams*, *cooperation competence*, *leadership and teamwork*, and *teamwork and collaboration*.

An examination of the comparison matrix suggests that the various efforts have been working in a largely independent and uncoordinated way as they have resulted in somewhat overlapping but also diverging lists/taxonomies, descriptions, and categorisations of generic competences. Comparing the three research papers on generic competences for IT professionals, for instance, show these differences: Borges and Souza (2024) identified 33 different generic competences, Caeiro-Rodriguez et al. (2021) identified 44 different generic competences, and Sandstrak and Klefstad (2025) identified 18 different generic competences, totalling to 71 different generic competences when ignoring difference in

competence naming. Only 7 out of these 71 different generic competences are included in all three papers (*critical thinking*, *ethics*, *problem-solving*, *communication*, *collaboration*, *learning*, and *autonomy*), while as many as 47 out of the 71 were included in only one of the papers (including competences such as: *decision-making*, *team establishment*, and *being positive*).

There may be several reasons why the three research papers on generic competences for IT professionals did not agree on a common list of generic competences. One may be that the importance of the various competences varies within different companies/types of IT business and types of jobs. The comparison matrix discussed above suggests two other causes for the differences observed:

1. **Different levels of abstraction**

   Generic competence is often composed of competences at a lower level of abstraction. Communication, for instance, is a generic competence that typically entails oral presentation, written communication, active listening, receiving feedback, and negotiation. In the ESCO taxonomy, there is only one communication competence defined (*T4.1 - communicating*), while the communications competence (*3.2 Communications*) in the CDIO taxonomy is further divided into nine lower-level generic competences.

2. **Difference in bundling**

   Lower-level generic competences may also be bundled into higher-level competences in various ways. The *taking a proactive approach (T3.2)* in the ESCO taxonomy, for instance, bundles *assuming responsibility*, *make decisions*, *show determination*, and *show initiative*, among others. In the CDIO taxonomy, *responsibility* and *determination* is bundled in the *perseverance, urgency and will to deliver (2.4.2)* competence, while *initiative taking* and *decisions, based on the information* at hand are bundled in the *Initiative and Willingness to Make Decisions in the Face of Uncertainty (2.4.1)* competence.

The net effect of this situation is that educators involved in the design of study programmes have no cohesive body of knowledge to consult when seeking input on what generic competences to be addressed in their study programme. The lack of such a body of knowledge of generic competences makes it a challenging task for educators to identify, assess and select generic competences to be addressed in the study programme. To make this a less challenging task, we provide an aggregation of generic skills based on the *foundational sources*. The objective is to give the reader a consistent and reasonably rich foundation and starting point for seeking information about generic competences. We believe this to be useful for educators when selecting generic competences to be integrated into their study programmes.

6.3. Commonly Recognised Generic Competences

As the objective of this chapter is to give an introduction and overview of generic competences and to assist a community of teachers in addressing generic competences in a detailed and systematic way when designing a study programme, this chapter will not list all and every generic competence mentioned in *foundational sources*. The chapter will rather present a set of generic competences that are commonly recognised in the *foundational sources*, i.e., competences that are included in more than half of these sources.

Generic competences can be divided into two groups, one for competences related to the individual (*intrapersonal competences*) and one for competences involved in relating to other people

(*interpersonal competences*). As shown in Table 1, intrapersonal competences can be further divided into thinking/cognitive competences and competences for individuals to manage oneself. The table lists the intrapersonal competences that are commonly recognised in the *foundational sources*.

Table 1: Commonly recognised intrapersonal competences

Thinking	**Managing oneself**
• Critical thinking • Creative and innovative thinking • Ethical thinking • Analytical reasoning and problem-solving • Systems thinking • Handling ambiguity and uncertainty	• Self-awareness and self-management • Initiative • Independence • Integrity • Adaptability and flexibility • Persistence • Professionalism • Planning and organisation • Life long learning

Table 2 lists the interpersonal competences that are commonly recognized in the *foundational sources*. Inspired by the HSX (Stump, 2020), this table is also divided into two parts, one for competences for interacting with other people and one for competences in leading other people.

Table 2: Commonly recognized interpersonal competences

Interacting	**Leading**
• Communication • Collaboration and teamwork • Empathy	• Leading others • Project management

6.3.1. Intended Learning Outcomes for Generic Competences

After exploring the realm of generic competences, the focus of this chapter will now be on how to specify generic competences in intended learning outcomes. This section is based on a general guide on writing intended learning outcomes that has been developed by Excited (SFU Excited, 2023).

Constructive alignment (Biggs and Tang, 2007) takes a competence-based approach to writing intended learning outcomes. A competence-based approach naturally leads to considering the observable result of learning as a demonstrable action. The competence-based approach overtly recognizes that the student is responsible for demonstrating his/her own learning and must evidence their attainment of key competences before being judged to have successfully completed a programme of learning. A

successful student will be able to perform a task, to a required standard, which evidences that they have the skills or knowledge needed to perform that task.

Intended learning outcomes should be described in statements that are specific and well defined. Intended learning outcome descriptions should explain the specific actions, in clear and concise terms, that would evidence that the student has satisfied the intended learning outcome. intended learning outcome descriptions should include performative actions that are observable.

The components of a learning outcome should be as follows:

Verb:	**Object:**	**Level:**	**Context:**
how the student will evidence learning	*what* the student will demonstrate	to what extent (not always necessary to write into the ILO)	a phrase qualifying the context of the *how* and the *what*

One important thing to consider when formulating course-level intended learning outcomes is the entry and exit point of the student. What is their expected prior learning before entering the course? To what level is the course expected to take them? The intended learning outcome(s) should describe the end goal of the learning experience; not a sequential and progressive list of stages that will get them to that end goal.

6.4. Designing Generic Competence Progression in Study Programmes

When designing study programmes and writing programme-level intended learning outcomes it is also important to make sure that the course level intended learning outcomes in total address the programme-level intended learning outcomes. Hence, when designing a study programme, educators need to agree on what course(s) will contribute to the progression of the various competences that students are expected to develop. As stated at the very beginning of this chapter, Excited's study of Norwegian bachelor programmes in IT (Brownridge et al., 2026) suggests that there is a lack of a detailed and systematic handling of generic competence progression throughout the study programmes. In this section, we will therefore explore how progression may be defined and handled.

Harden (2007) has developed a model for using intended learning outcomes as tools for defining progression. The model may help educators handle generic competence progression in a more systematic way. Harden's model describes four dimensions for competence progression:

1. Increased breadth
2. Increased difficulty
3. Increased application to practice
4. Increased proficiency

In the *increased breadth* dimension, progression happens through the extension of the area of competence to new topics or to new contexts. Using communication as a generic competence example, competence in negotiating compromises or resolving conflicts may not be a target competence to be developed in a course introducing communication competence but rather in later courses as part of communication competence progression.

In the *increase difficulty* dimension, progression happens through an increase in depth or difficulty. Using project management as a generic competence example, students being competent in managing single-discipline projects in stable and predictable environments (with regards to project goals and available resources) may have their competence progressed to managing projects in unstable and unpredictable environments or to managing inter- and multi-disciplinary teams, such as the Experts in Teamwork course at NTNU [1].

In the *increased application to practice* dimension, progression can move from a general context to a specific context, from theory to a professional context, or move to integration into the professional role. Using teamwork as a generic competence example, progression may move from student teams working on a "for learning only" project to a project working on work life problems in collaboration with work life stakeholders.

In the *increased proficiency* dimension, progression moves towards more efficient performance in terms of task organization, time spent, errors made, and quality achieved. Increased proficiency is also associated with less need for supervision, more initiative being taken, and increased justification of actions. Using teamwork as a generic competence example, increased proficiency may be seen in the quality of the team organization and project plan and the time spent when forming a new team.

The community of teachers designing a study programme can use these four dimensions as a means for developing a systematic handling of generic competence progression throughout the study programme.

6.4.1. Assessing Generic Competences

In constructive alignment, verbs used in the intended learning outcomes clearly specify what the student needs to perform during assessment (Biggs and Tang, 2007). Assessment should, therefore, be seen as a vital part of the initial design of the course. The intended learning outcomes serve as an important starting point for designing the assessment method. By designing the assessment methods based on the intended learning outcome, one gives purpose to the assessment. The action verb used in an effective learning outcome is usually a good indicator for the assessment method to be used.

There is an important distinction between assessment intended to identify how much has been learnt *(summative assessment)* and assessment which is intended to help the student learn (*formative assessment).* Summative assessment is the final evaluation of the students' performance. Formative assessment is most useful part way through a course or module and will involve giving the student feedback which they can use to improve their future performance (Gibbs and Simpson, 2005).

Designing summative assessment methods is especially challenging for interpersonal competences, as it often means assessing group project work. There are two distinct challenges to consider. Firstly, the assessment method should measure the qualities of the group work itself and not just the quality of the outcome being produced by the group. Secondly, it is the performance of the individual members of the team should be assessed, not just the performance of the team collectively. This means determining and assessing the contributions of each team member.

Designing formative assessment methods may include finding ways to involve the students in the assessment process. Self- and peer-assessment, where students assess themselves or their fellow students, respectively, may have great potential for improving students' academic performance (Carol 2013). Furthermore, self- and peer-assessment may also be considered as compositions of generic competences where students must judge and reflect on their own and/or fellow students' work against

expected performance criteria – maybe even participating in defining specific performance criteria, and where students must give, receive, judge and use performance related feedback.

One means for self-assessment is the use electronic portfolio (ePortfolio). An ePortfolio is a digital collection of academic work, produced artifacts, and self-reflections that allows students to document, track, and demonstrate their professional development as students. Collecting, tracking and, reflecting on their professional development, may help students recognize what they have learned and reflect on how the resulting competences can be used in a work life context. An Excited PhD project demonstrated how the use of ePortfolio could help students in this regard (Olstad et al., 2025).

6.5. Integrating Generic Competences into Programmes and Courses

As discussed earlier in this chapter, the lack of a cohesive body of knowledge of generic competences makes it challenging for educators to review how well generic competences are covered in the study programme and to improve the way generic competences are integrated into study programme. Curriculum development can be understood to be an iterative and continuous process of creation and renewal involving the community of educators contributing to the programme (DiPietro et al. 2022). DiPietro et al. (2022) describes four alternative methods for defining programme level intended learning outcomes. This chapter is based on their Method 1, brainstorming learning outcomes using a backward design approach. This method may look like a top-down sequential process, but the process may move back and forth between the activities to make it an iterative process.

6.5.1. A Structured Curriculum Development Process

The process is illustrated in Figure 1. The broken arrow drawn between Activity 5 and Activity 1 is included to show that Activity 5 may be run prior to Activity 1 to gather input about generic competences to be considered for integration into the program. The broken arrow drawn between Activity 5 and Activity 2 is included to show that aligning programme- and course-level intended learning outcomes might be an iterative approach.

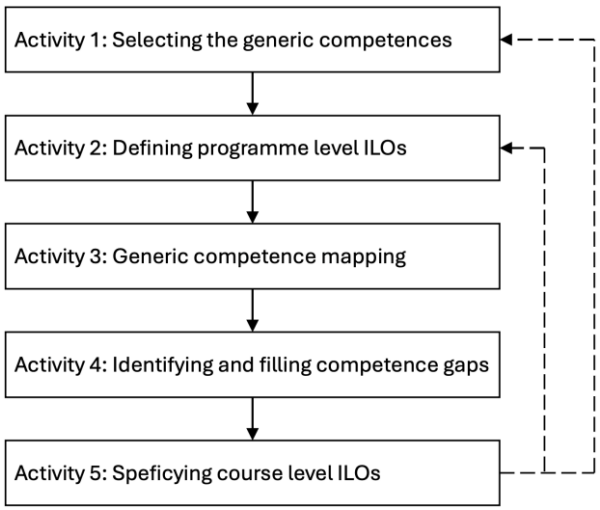


Figure 1 Structured curriculum development process for generic competences

The involved activities are:

Activity 1: Selecting the generic competences to be included in the programme

Table 1 and Table 2 are based on the comparison and examination of several lists of generic competences and may be used as starting points for brainstorming what generic skills to integrate into the programme. The educators could also explore the *foundational sources* for more inspiration and additional details during their brainstorming phase.

Activity 2: Defining the programme level ILOs

- As discussed previously, it is important that the intended learning outcome descriptions are specific and well-defined. Educators need to determine what the selected competences should entail to be able to develop such specific and well-defined intended learning outcome descriptions. When developing the intended learning outcome descriptions, the *foundational sources* may be consulted for in-depth information about what each selected competence could entail.

Activity 3: Generic competence mapping

- As described in DiPietro et al. (2022), curriculum mapping is a process that involves the collection and representation of information about a program. In the following case, we will focus on the curriculum mapping activities pertaining to program learning outcomes and the alignment with course outcomes and, therefore, refer to this as generic competence mapping. The goal of this activity is to decide how well the selected generic competences are covered in existing courses. A generic competence map would identify what courses contribute to the development of a given competence, and what the contribution would be. During this step, educators may find that some of the generic competences might be addressed in some of the courses already.

Activity 4: Identifying and filling competence gaps

- The generic competence map resulting from the previous activity, would form the basis for identifying alignment between courses offered in parallel (horizontal alignment), courses building on top of each other (vertical alignment), and the alignment between the programme level intended learning outcomes and the competences students develop in the various courses. This map can, then, be used to identify gaps in coverage, progression and alignment of generic competences in the entire programme. When the gaps are identified, the educators can discuss and agree on what courses should be redesigned – or what new courses should be developed – to fill these gaps.

Activity 5: Specifying course level ILOs

- The final activity is to develop course level intended learning outcomes that defines what the course is intended to contribute to the development of the selected generic competences such that students can develop generic competences at the intended level of *breadth*, *difficulty*, *application to practice*, and *proficiency* (Harden, 2007).

6.5.2. An Illustrative Case of Curriculum Development

This section will present a case illustrating how the information presented in this chapter can be used by a community of educators involved in the design of a study programme to integrate generic

competences into their programme and to define progression throughout the programme using the suggested process.

Activity 1, selecting generic competences

In the sample case, the educators decide to use Table 1 and Table 2 as starting points for discussing what generic skills to integrate into the programme. The *collaboration and teamwork* competence from the *interacting group* of *interpersonal* competences is one of the generic competences they choose to integrate into the programme.

Activity 2, defining programme level ILOs

The educators discuss what collaboration and teamwork competence should entail in the programme. They decide to visit the *foundational sources* for input on what collaboration and teamwork may entail. Collaboration and teamwork can be found in *working in teams (3.1.1)* in CDIO, *Working in teams (S1.8.1)* in ESCO, *cooperation Competence (C3)* in Ehlers, *leadership and teamwork (3)* in NAIS, *teamwork* in Borges and Souza, and *Collaboration and teamwork, sometimes in multidisciplinary teams* in Caeiro-Rodríguez. The following list generated from these *foundational sources* shows details of what *collaboration and teamwork* may entail:

- Team establishment
    - Forming teams, assigning roles and responsibilities (CDIO)
    - Setting goals and objectives, planning, scheduling the work (CDIO)
    - Setting norms (ground rules, respect and diversity, confidentiality, accountability) (CDIO)
- Teamwork distribution and contributions
    - Coordination and management of team process: meetings - physical and distance; communication - information, listening, feedback; decision-making; documentation and reporting; representing the team (CDIO)
    - Each team member doing their part in the service of the whole (ESCO)
    - Implement decisions and meet goals (NAIS)
    - Share the credit (NAIS)
    - Team-working ability (Ehlers)
    - Effectively work in a team and contribute to the common goal (Borges and Souza)
- Team member interaction
    - Handling diverse perspectives and conflicts (CDIO)
    - Understanding and respecting the roles and competencies of other team members (ESCO)
    - Social intelligence (Ehlers)
    - Enlist help (NAIS)
    - Ability to exchange information, cooperate, negotiate agreements, and make decisions with mutual respect (Borges and Souza)

    - Openly listen to the viewpoints of others (Caeiro-Rodríguez)
- Multidisciplinary teams
    - Intercultural competence (Organizational culture) (Ehlers)
    - Collaborate sensitively and productively with people of varied backgrounds (NAIS)
    - Collaborate effectively in multidisciplinary groups, integrate knowledge, and contribute to the synthesis of solutions (Caeiro-Rodríguez)
- Leadership
    - Team membership and leadership (delegating, facilitating, directing, supporting, coaching) (CDIO)
    - Creativity, empowerment and motivation (incentives, recognition) (CDIO)
    - Strategies for assessment and reflection to develop processes, team and members (CDIO)
    - Coordinate tasks, manage groups, and delegate responsibilities (NAIS)

Based on this, the educators may decide that the programme's emphasis on teamwork competence should be the individual team members' competence in contributing to the team, to the team process, and to the development of the other team members in a multi-disciplinary software development team. Components of intended learning outcomes at the study programme level could then be that each student can:

- demonstrate willingness and ability to implement decisions and meet goals defined by the team,
- make substantial and valuable contributions to the planning, scheduling, and coordination of work,
- demonstrate ability to adapt to needs and changes in the project,
- collaborate and communicate effectively and efficiently with professionals from other disciplines,
- assess and reflect on the qualities of the team work process,
- assess and reflect on their own contributions to the team and the team process,
- make substantial and valuable contributions to empowering and supporting other team members,
- assess and reflect on other team members' contributions to the team work and the team process and sharing this with the team in a way that serves the team well, and
- assess feedback received from team members and act on it in a constructive and productive way to the advantage of the team as whole.

Activity 3, mapping generic competences

The educators examine the courses currently in the programme to see where and how collaboration and teamwork is addressed in the individual courses. In the sample case, the educators find that collaboration and teamwork is mainly addressed in three different courses, one second semester and one fourth semester course on software engineering and one fifth semester course on multi-disciplinary project work course. When filling out the generic competence map, they find that the collaboration and teamwork competences are only vaguely described and not addressed in a systematic way in and between the courses.

Activity 4, identifying and filling competence gaps

The generic competence map created in Activity 3 is analyzed by the educators to identify competence gaps. In the sample case, the educators find that there are gaps in the competence map when it comes to the following programme level intended learning outcomes:

- make substantial and valuable contributions to the planning, scheduling, and coordination of work
- demonstrate ability to adapt to needs and changes in the project,
- make substantial and valuable contributions to empowering and supporting other team members,
- assess feedback received from team members and act on it in a constructive and productive way to the advantage of the team as whole

The educators then discuss how to modify the contents of the three courses to fill the competence gaps identified and decide to structure handling of generic competences in this way:

- Competencies in contributing to, and basic assessment of their own contributions to, teamwork and coordination in a software development project are allocated to the first software engineering course.
- Competencies in empowering and supporting other, in assessing the contributions made by other team members, and in giving and receiving feedback are added in the second software engineering course.
- Competencies in collaborating and communicating with professionals from other disciplines are added in the multi-disciplinary project work course.

Activity 5, specifying course level ILOs

The educators use the results of the generic competence map analysis to define course level intended learning outcomes for the three courses:

- First software engineering course:
    - demonstrate willingness and ability to implement decisions and meet goals defined by the team,
    - make substantial and valuable contributions to the planning, scheduling, and coordination of work,
    - demonstrate ability to adapt to needs and changes in the project,
    - assess and reflect on the qualities of the team work process,
    - assess and reflect on their own contributions to the team and the team process,
- Second software engineering course:
    - make substantial and valuable contributions to empowering and supporting other team members,
    - assess and reflect on other team members' contributions to the team work and the team process and sharing this with the team in a way that serves the team well, and
    - assess feedback received from team members and act on it in a constructive and productive way to the advantage of the team as whole

- Multi-disciplinary project work course:
    - collaborate and communicate effectively and efficiently with professionals from other disciplines

6.6. Conclusion

This chapter was motivated by an observed lack of detailed and systematic handling of generic competences in several study programmes in IT. The term generic competence has been explored and several existing research studies and efforts in identifying generic competences has been reviewed to identify commonly recognized generic competences. The exploration resulted in two main categories of generic competences, *intrapersonal* and *interpersonal* competences. *Intrapersonal* competences were further divided into the *thinking competences* and the *managing oneself* competences. *Interpersonal* competences were divided into *interacting competences* and *leading competences*.

This chapter has also discussed how progression in generic competences can take place through increased breath, difficulty, application to practice, and proficiency. It has also highlighted some of the issues to consider when designing methods for assessing generic competences.

Finally, this chapter has outlined a process for detailed and systematic handling of generic competences using the outcome of this chapter's explorations as a starting point for study programme and course design.

Copyright

Educating for Generic Competencies © 2026 by Rune Hjelsvold is licensed under CC BY-NC-ND 4.0



References

Aasheim, Cheryl, Jordan Shropshire, Lixin Li, and Christopher Kadlec (2012). "Knowledge and Skill Requirements for Entry-Level IT Workers: A Longitudinal Study". In *Journal of Information Systems Education*, vol. 23, no. 2.

Biggs, John and Catherine Tang (2007). *Teaching for Quality Learning at University*, 3rd Edition, Open University Press.

Borges, Giovana Giardini and Rogéria Cristiane Gratão de Souza (2024). "Skills development for software engineers: Systematic literature review". In *Information and Software Technology*, vol. 168.

Brownridge, Andrew Mark, Rune Hjelsvold, Svein Erik Bratsberg, and Gabrielle Hansen (2026). "Beyond Knowing: Evaluating Programme-Level Learning Outcomes in Norwegian IT Education". Accepted for publication in *Nordic Journal of Pedagogy & Critique*.

CC2020 Task Force (2020). *Computing Curricula 2020: Paradigms for Global Computing Education.* Association for Computing Machinery.

CDIO Office (2023). *CDIO Syllabus 3.0*.

Caeiro-Rodríguez, Manuel, Mario Manso-Vázquez, Fernando A. Mikic-Fonte, Martín Llamas-Nistal, Manuel J. Fernández-Iglesias, Hariklia Tsalapatas, Olivier Heidmann, Carlos Vaz De Carvalho, Triinu Jesmin, Jaanus Terasmaa, and Lene Tolstrup Sørensen (2021). "Teaching Soft Skills in Engineering Education: An European Perspective". In *IEEE Access*, vol. 9, pp. 29222–29242.

DiPietro, Cary, Patti Dyjur, Kathleen Fitzpatrick, Kimberley A. Grant, Carolyn Hoessler, Frances. Kalu, Jessie Richards, Allyson Skene, and Peter Wolf. (2022). "Educational Development Guide Series: No. 4. A Comprehensive Guide to Working with Higher Education Curriculum Development, Review & Renewal Projects". In (P. Dyjur & A. Skene, Eds), *Educational Developers Caucus*.

Ehlers, Ulf-Daniel (2024). "Towards a Future Skills Framework for Higher Education". In Ulf-Daniel Ehlers and Laura Eigbrecht (eds), *Creating the University of the Future. Zukunft der Hochschulbildung - Future Higher Education*, pp. 21–60.

European Commission (2025). *European Skills, Competences, Qualifications and Occupations (ESCO)*, ver 1.2.1.

European Union (2017). *Council recommendation of 22 May 2017 on the European European Qualifications Framework for lifelong learning*.

Evans, Carol (2013). "Making Sense of Assessment Feedback in Higher Education". *Review of Educational Research*, *83*(1), 70-120.Gibbs, Graham and Claire Simpson (2005). "Conditions Under Which Assessment Supports Students' Learning". In *Learning and Teaching in Higher Education,* Issue 1, pp. 3–31.

Groeneveld, Wouter, Brett A. Becker, and Joost Vennekens (2020). "Soft Skills: What do Computing Program Syllabi Reveal About Non-Technical Expectations of Undergraduate Students?" In *ITiCSE '20: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 287–293.

Harden, Ronald M (2007). Learning outcomes as a tool to assess progression. *Medical Teacher*, vol. 29, pp. 678–682.

Krathwohl, David R. (2002). "A Revision of Bloom's Taxonomy: An Overview". In *Theory Into Practice*, vol. 41, no. 4, pp. 212–218.

Li, Paul Luo, Amy J. Ko, and Jiamin Zhu (2015). "What Makes a Great Software Engineer?" In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, pp. 700-710.

Olstad, Hege Annette, Birgit Rognebakke Krogstie, Xiaomeng Su, and Rune Hjelsvold (2025). "Enabling computing students to recognize acquired competencies with ePortfolios". In *Computer Science Education*, vol 35 no 3, pp. 578–605.

Sandstrak, Grethe and Bjørn Klefstad (2025). "Empowering Employability through Transversal Skills and Competencies". In *UDIT Norsk konferanse for utdanning og didaktikk i IT-fagene*, vol. 37, no. 4.

SFU Excited, Cluster 1 (2023). *Writing Effective Intended Learning Outcomes (ILOs).*

Stevens, Matt and Richard Norman (2016). "Industry expectations of soft skills in IT graduates: a regional survey". In *ACSW '16: Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1–9.

Stump, Glenda (2020). “Human Skills: Critical Components of Future Work”. In *The evolllution*, Mar 2.

Uchiyama, Kay Pippin and Jean L. Radin (2009). “Curriculum Mapping in Higher Education: A Vehicle for Collaboration”. In *Innovative Higher Education*, vol. 33, pp. 271–280,

Witt, Robert and Jean Orvis (2010). *A Guide to Becoming a School of the Future*, National Association of Independent Schools.

UNESCO (2016). “School and Teaching Practices for Twenty-first Century Challenges: Lessons from the Asia-Pacific Region”. In *UNESCO Asia-Pacific education thematic brief.*

7. Peer Assessment in Project-Based IT Education

Trond Aalberg, Somayeh Bayat Esfandani and Mariusz Nowostawski

Norwegian University of Science and Technology (NTNU)

7.1. Introduction

Student peer review is a well-established learning activity in which students give feedback on each other's course deliverables or other outcomes of a learning activity. Topping (1998) defines peer assessment as "an arrangement in which individuals consider the amount, level, value, worth, quality, or success of the products or outcomes of learning of peers of similar status". Used well, it can strengthen critical thinking, give students formative feedback on their learning and progress, and help them gain skills in giving and receiving critique. The activity has traditionally been called "reviewing", as it partly is inspired by the review process in scientific publishing. Today the term "student peer assessment" is often preferred, because it signals a focus on formative assessment rather than judgement. The term "student peer feedback" is also common, and signals even more clearly to students that the process is intended for learning rather than evaluation. In this chapter we use peer assessment as the general term.

Peer assessment has been applied in many ways, and experience and best practice are reported in a number of overviews (Topping 2017; Cadima et al. 2024; Ortega-Ruipérez and Correa-Gorospe 2024). A meta-analysis of 58 studies found that students who took part in peer assessment performed better than students who did not, and that the effect was considerably larger when students were trained as assessors (Li et al. 2020). Most of this research, and most of the systems and recommendations that come with it, have been developed around text-based assignments. In IT education, peer assessment has traditionally had a weaker position, even though the discipline is in many ways well suited for it.

Peer assessment in student projects has been one of the activities of Excited, the Norwegian Centre for Excellent IT Education, in courses at both the NTNU campus in Gjøvik and the Gløshaugen campus in Trondheim. Excited also funds a PhD project which studies how peer assessment tools for IT education should be designed. Using design science research, the project has identified requirements for such tools and investigates how generative AI can support both giving and receiving peer feedback, for example through AI-generated summaries of feedback from several peers (Bayat Esfandani and Aalberg 2025). This chapter is written as a practical introduction for teachers who consider using peer assessment in project-based IT education, and focuses on recommended practice. To show how this practice plays out in real courses, the chapter ends with use cases from three courses taught at the two campuses, which differ in purpose and design, and in how students' use of generative AI has affected the activity.

The rest of the chapter is structured as follows. Section 2 explains why peer assessment is particularly relevant for IT education. Section 3 summarizes the pedagogical foundations, and Section 4 describes the peer assessment process and the features a supporting system needs. Section 5 shows how peer assessment can follow students' progression from small programs to large team projects. Section 6 sets out recommended practice for the main design decisions, and Section 7 discusses how artificial intelligence can support peer assessment. Section 8 presents use cases from three courses at NTNU, and Section 9 concludes.

7.2. Why Peer Assessment Suits IT Education

Perhaps the strongest argument for peer assessment in IT education is that review is an established part of professional software development. Code review, where developers systematically examine each other's contributions before they are integrated, is standard practice in industry (Chong et al. 2021). Practicing peer assessment is therefore not merely an academic exercise, but rehearses an authentic professional workflow (Chong et al. 2021). It helps students see evaluation as an integral part of the development process rather than something done to them at the end of it. The same holds for related practices such as design reviews, requirements reviews and usability evaluations.

Reading and understanding code written by others is also a core professional skill. Developers commonly move between projects and spend much of their time extending and improving existing code bases rather than programming from scratch. A curriculum organized around building programs from scratch rarely trains this skill directly. Peer assessment addresses this gap to some extent, since assessing a peer's solution requires the ability to read unfamiliar code, reconstruct its intent and judge its quality. It does not fully mimic professional maintenance tasks, however. A peer's solution to a course assignment is usually much smaller than a real code base, and the reviewer is asked to assess it, not to fix or extend it.

Peer assessment also exposes students to examples and alternative solutions (Indriasari et al. 2020). Seeing how others have approached the same problem, and comparing their design choices, data structures and implementation choices with one's own, builds a repertoire of patterns and an understanding of design trade-offs that is hard to acquire from a single solution or from lectures alone. How much students gain from this depends on how many submissions each student reviews, and on whether the submissions actually differ enough for reviewers to see solutions they had not thought of themselves. Beyond exposure to alternatives, studies of peer code review report the development of programming and code-review skills, problem-solving skills and knowledge of coding standards (Indriasari et al. 2020), and improved compliance with coding standards over a course has also been observed (Wang et al. 2012).

Finally, many software artefacts have well-defined technical requirements and quality characteristics. This makes it easier for fellow students to give concrete and useful feedback than when they critique textual explanations and arguments. This is particularly relevant for project-based learning (Sindre et al. 2018), where students learn by designing, implementing and evaluating software artefacts, and where the artefacts themselves provide a natural object for peer assessment.

7.3. Pedagogical Foundations

The use of peer assessment rests on well-established understandings of how people learn. This section summarizes the foundations most relevant to the design of peer assessment activities: the role of feedback, the social nature of learning, the learning that comes from assessing, self-regulation, and feedback and assessment literacy.

7.3.1. Feedback in learning

Timely feedback is a powerful way to influence learning and achievement, although the effect varies with how it is designed and used (Hattie and Timperley 2007; Black and Wiliam 1998). Effective feedback addresses three questions for the learner: where am I going, how am I going, and where to next (Hattie and Timperley 2007). This emphasizes feedback as formative information that feeds

forward into subsequent work (Black and Wiliam 1998; Shute 2008), rather than a summative evaluation delivered after learning has finished.

In large classes, a teacher can only provide a limited amount of feedback to each student, often only after a deadline has passed. Fellow students can produce feedback that is more detailed and varied, and deliver it while there is still an opportunity to act on it (Topping 1998; Nicol and Macfarlane-Dick 2006). Feedback from several peers also exposes students to a range of perspectives rather than a single authoritative judgement, which encourages them to compare differing views and reach their own conclusions (Nicol et al. 2014). Furthermore, feedback is increasingly understood as a dialogue rather than one-way transmission. For feedback to feed into learning, students need to actively interpret, discuss and use it (Nicol 2010; Winstone and Carless 2019). The availability of feedback alone does not guarantee that learning occurs.

7.3.2. Social and constructivist foundations

Peer assessment is a social learning activity, in which understanding is constructed through interaction with others rather than transmitted from an authority (Topping 2009; Dochy et al. 1999). This aligns with Vygotsky's notion of the zone of proximal development: learners can achieve more with guidance from a more capable other than they can alone (Vygotsky 1978). Students are close to the material and to each other's level of understanding and can sometimes explain difficulties in more accessible terms than an expert. When students present and justify their assessments, they engage in active and constructive processing that supports deep learning (Topping 2009; Berg et al. 2006). At its best, peer assessment creates a community of practice in which quality is discussed and the criteria for good work are made explicit and shared, rather than remaining the private knowledge of the teacher (Wenger and Snyder 2000; Merry and Orsmond 2020).

The social dimension also brings challenges. The reviewer is not necessarily more capable than the author and does not always bring a different perspective. Feedback may be of poor quality or contradictory, and most peer assessment is anonymous and gives little room for real dialogue. These challenges must be addressed in the design of the activity, as discussed in Section 6.

7.3.3. Learning by assessing

A distinctive element of peer assessment is that the reviewer often learns as much as the student being reviewed, and some studies have found that the reviewer learns more (Li et al. 2010). Producing useful feedback is in itself a demanding activity. The reviewer must interpret the criteria, inspect the deliverable closely, form a qualified opinion about its quality and express it constructively (Nicol et al. 2014; Liu and Carless 2006). In doing so, students engage with the standards of the discipline far more actively than when they simply receive a grade, and they encounter alternative approaches to a problem they have attempted themselves.

This is closely tied to the development of *evaluative judgement*, the capacity to judge the quality of work, whether one's own or that of others (Tai et al. 2018; Sadler 1989). Sadler argued that students cannot reliably produce high-quality work until they have a concept of what quality looks like and can monitor their own performance against it, and that they acquire this concept largely by evaluating work themselves (Sadler 1989). By repeatedly judging peer work against shared criteria, students internalise those criteria and become better able to regulate the quality of their own work (Tai et al. 2018). Evaluative judgement is also valuable in its own right, since the ability to assess quality and justify decisions is central to professional practice and lifelong learning (Boud and Soler 2016).

7.3.4. Self-regulated learning and scaffolding

Peer assessment also supports self-regulated learning, the active role learners take in managing their cognition, motivation and behavior (Zimmerman 2000). Self-regulated learning is commonly described as a cycle of forethought, monitoring, control and reflection (Pintrich 2000). The peer assessment process aligns well with these phases: students interpret criteria and activate prior knowledge, evaluate peer work and generate feedback, and reflect on their own work in light of what they have seen and written (Topping 2009, 2018; Liu and Carless 2006; Misiejuk et al. 2021). As self-regulation develops, students become less dependent on feedback from the teacher (Nicol and Macfarlane-Dick 2006).

Self-regulation is, however, constrained by cognitive resources. When the combined cognitive and metacognitive load of a task is too high, both learning and self-regulation suffer (Sweller 1988; Molenaar 2022). Peer assessment activities should therefore be designed so that students can devote their attention to assessing and being assessed, rather than to peripheral difficulties. Rubrics, worked examples and calibration exercises act as scaffolds that make an unfamiliar and demanding task manageable, with support fading as competence grows (Wood et al. 1976; Hmelo-Silver et al. 2007).

7.3.5. Feedback and assessment literacy

Giving and receiving feedback effectively is not an innate skill but must be trained (Carless and Boud 2018; Sutton 2012). Students who are new to assessing often struggle to be specific and constructive. They may be either too critical or too polite, and inexperienced reviewers tend to focus on details and concrete flaws rather than on the more important overall strengths and weaknesses of the work. Receiving feedback constructively, making sense of it, judging its relevance and deciding how to act on it, is equally a skill that develops with practice (Carless and Boud 2018).

These capabilities are described as assessment literacy and feedback literacy. Assessment literacy concerns students' understanding of assessment criteria and processes, and in particular their ability to judge work against criteria and standards (Smith et al. 2013). Feedback literacy refers to the understandings, capacities and dispositions needed to make sense of feedback and use it, and includes appreciating feedback, making judgements, managing affect and taking action (Carless and Boud 2018). Peer assessment can build both, because it places students on both sides of the feedback process (Nicol et al. 2014). Repeated participation is usually needed before students gain a deeper understanding of what makes feedback useful.

Table 1 summarizes the potential benefits of peer assessment discussed in this section, together with the challenges that the design of the activity must address.

Table 1: Potential benefits and common challenges of peer assessment.

Potential benefits	Common challenges
More, faster and more varied feedback than a teacher alone can give	Feedback quality varies, and feedback from different peers may be contradictory
Exposure to alternative solutions and design choices	Too few or too similar submissions limit what reviewers can learn from them
Development of evaluative judgement by assessing the work of others	The reviewer may be no more competent than the author
Training in giving and receiving feedback, close to professional code review	Novice reviewers are too polite or too harsh, and focus on details rather than the whole
Support for self-regulation and reflection on one's own work	Students may put in little effort, and may not read or act on the feedback they receive
Criteria and standards made explicit and shared among students	Peer scores are often not reliable enough to be used for grading

7.4. The Process and System Support

7.4.1. The peer assessment process

Peer assessment is not a single act of giving feedback but a structured process in stages, each with its own potential for learning (Figure 1). Students take two roles in turn: they are the creators of a piece of work and the assessors of the work of others. Learning happens both when receiving feedback on one's own submission and when producing feedback for someone else. Designing a meaningful process therefore requires insight into what is learned at each stage.

The process typically begins with *preparation*, in which the task and the criteria for judging it are made clear. This is where students form an initial concept of what quality looks like. Discussing or co-constructing the criteria with students may help them internalize standards they will later apply both to others and to themselves (Sadler 1989; Topping 2017; Fraile et al. 2017). Others report that leaving out predefined criteria can also be effective, since it motivates students to reflect on what quality is (Mirmotahari and Berg 2023). Students then develop and *submit* their own work. Knowing that peers will see the work may itself influence its quality. Submissions are *allocated* to peers, and the *reviewing* stage follows, in which each student reads, interprets and judges the work assigned to them and writes feedback. This is the cognitive heart of the activity, where evaluative judgement is chiefly developed (Nicol et al. 2014; Tai et al. 2018).

The feedback is then returned to the authors, who must *read and act on it*: make sense of the comments, weigh their relevance, decide what they mean for further work, and ideally revise their work or adjust their approach. Closing this loop is essential, because feedback that is never read or used yields little learning (Hattie and Timperley 2007; Carless and Boud 2018). Authors may also give feedback to the reviewers on how useful their comments were. A final, often implicit, stage is *reflection*, in which students consider what they have learned from both the feedback they received and the experience of assessing others. In a project, the cycle can be repeated at several points, so that the feedback from one round informs the work towards the next. Seen this way, the process maps onto the phases of self-

regulated learning, from forethought and planning, through monitoring and control during reviewing and revision, to reflection (Pintrich 2000).

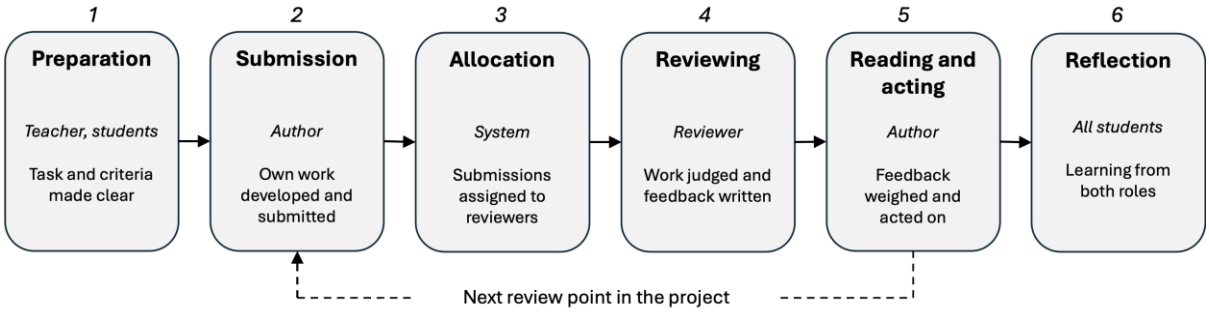


Figure 1. The peer assessment process. In project courses the cycle can be repeated at several review points during the project.

What makes these stages a process rather than a collection of independent steps is their sequencing in time. Submissions must be in before reviews can start, and reviews must be completed before they can be read, and each stage is bounded by deadlines. As the number of participants grows, ensuring that every submission is reviewed, that reviewers get what they need when they need it, and that the transitions between stages happen smoothly, becomes a substantial orchestration challenge. This is the main reason why peer assessment benefits from system support.

7.4.2. System support

Peer assessment can be organized manually or semi-manually for small groups, but a dedicated system is far better as the number of students grows. Major learning management systems such as Blackboard and Canvas offer built-in support, but it is rather limited and lacks essential features. The open-source learning management platform Moodle has somewhat better support, with optional plugins that add features. Dedicated third-party peer feedback systems are available for licensing (such as FeedbackFruits, used in IT2810; see Section 8), and some institutions have developed their own (such as CSAMS, an in-house system developed and used by the Intelligent Systems and Analytics group at IDI, Gjøvik; see Section 8). Research on peer feedback often refers to research prototypes, but most of these are not publicly available or have been discontinued.

The core of any system is the well-defined process of collecting submissions, distributing them to reviewers, and returning the feedback to the authors. A system that works well in practice, however, needs a number of additional features. Below we describe them in three groups: features that orchestrate the process, features that support the quality of the feedback, and features for oversight, group work and integrity. Table 2 gives an overview.

Table 2: Main features of a peer assessment system.

Group	Feature	Key questions for the teacher
Orchestration	User administration	Is the system integrated with the LMS? Does it support groups?
	Submissions	Can students submit code, repositories, links and running applications?
	Allocation	How many reviews per submission? Are own-group and reciprocal reviews avoided?
	Deadlines and stages	How are late submissions, extensions and late reviews handled?
	Notifications	Are students reminded when a stage opens or a deadline approaches?
Feedback quality	Rubrics	Structured rubric, open headings or no predefined criteria?
	Calibration and training	Do students practice on example submissions first?
	Authoring feedback	Can reviewers comment on specific lines, files or parts of the work?
	Reading feedback	Must students respond to the feedback? Are summaries provided?
	Grading and rating	Are scores used, and for what? Can authors rate the feedback?
Oversight and integrity	Anonymity	Single-blind, double-blind or open? Can identifying information be removed?
	Teacher overview	Can the teacher follow progress and spot missing or poor reviews?
	Group support	Can groups submit and review together? Is assessment within groups supported?
	Privacy and integrity	Is student data handled in line with the GDPR?

Orchestrating the process

User administration. The system must keep track of participants and their participation. Integration with the learning management system is a major advantage, but not a strict necessity if data can be transferred between systems. A system that meets a broad range of functional needs is generally preferable to a well-integrated but feature-poor one, which is a major reason for using third-party

systems. Project-based courses also need support for groups, since teams often submit and sometimes review collectively.

Submissions. The most traditional submission is a text document such a pdf file, and many peer assessment systems are built around this. IT courses need support for a range of other deliverables, such as source code files, code repositories or links to Git repositories, executable prototypes and web applications. Ideally, the system lets reviewers inspect the submission in its natural form rather than as a detached file.

Allocation. The main objective is to make sure that every submission is assigned to one or more reviewers, and that every reviewer receives the required number of submissions. Allocation can be done in one step, which lets reviewers see and compare all their submissions in parallel, or on the fly, where reviewers receive a new submission when they have finished the previous one, which more easily accommodates late submissions. Reviewers should not be assigned their own work or, in project courses, the work of their own group. It is often desirable to avoid reciprocal pairs, to distribute the load evenly, and to assign several reviewers per submission so that authors receive more than one perspective.

Deadlines and stages. A peer assessment activity moves through distinct stages, each with its own start and end. The larger the class, the more exceptions there are to handle, such as late submissions, extensions and late reviews, individually or for groups. The system should make it easy for the teacher to enforce the chosen course policy while handling the inevitable special cases. **Notifications** announcing that a stage has opened, that a deadline is approaching or that feedback is ready help keep students on track, especially in large classes.

Supporting the quality of feedback

Rubrics. Rubrics are the guides, instructions or headings used to structure the assessment. Some argue strongly for carefully constructed rubrics, even developed together with students, while others find that open headings or no rubric at all work better. The objective will vary between courses, and a system should not dictate how a review task is set up. Courses that emphasize consistent and comparable reviews need more structure, while courses that emphasize reflection and critical thinking may be better served by a more open feedback scheme.

Calibration and training. In calibration, students first practice on one or more example submissions and compare their judgements with a reference review before assessing their peers. This helps students internalize the criteria, improves the consistency of their feedback, and builds confidence in inexperienced reviewers.

Authoring feedback. How a system supports writing feedback strongly shapes its usefulness. For text, comments are often tied to rubric items or given as an overall assessment. For software, the ability to annotate specific lines of code, files or changes is particularly valuable, since it lets reviewers point precisely at what they comment on. This mirrors professional code review, and some courses build peer assessment directly on the pull request review features of platforms such as GitHub and GitLab.

Reading feedback. Reading the feedback is what makes students learn from the reviews they receive, so it can be included as a step in its own right. Systems can easily record that students have opened their feedback, but not that they have read it. Asking students to rate, flag or respond to the feedback they receive engages them with it, and motivates them to read it. When several peers review the same work, a summary of the reviews can also help, by presenting the feedback in a structured and coherent form

rather than leaving students to sort out contradictions themselves (Bayat Esfandani and Aalberg 2025; Pramudianto et al. 2016). A summary should complement the individual comments, not replace them.

Grading and rating. Reviewers may be asked to score the submission as a whole or for each rubric item, and authors may rate the feedback they receive. Such ratings can express an opinion about the level of performance, or serve as input to the teacher for follow-up.

Oversight, group work and integrity

Anonymity. Reviews can be single-blind, where the reviewer is hidden from the author, double-blind, where both are anonymous, or open. Anonymity can encourage more critical feedback and reduce bias, while openness supports accountability and dialogue, and mirrors professional practice where reviews are attributed. In IT courses, anonymity is harder to achieve than for text documents, since source code and its metadata often reveal the author, through names in comments and file headers, the commit history of a repository, or a distinctive coding style. A system that supports anonymity should help remove such information or warn about it.

Teacher overview. As classes grow, teachers need an overview of the activity: progress through the stages, students who do not participate, and outliers such as strongly disagreeing scores or unusually short reviews. Dashboards that summarize participation and the reliability of peer scores give the teacher the information needed to intervene.

Group support. In project-based courses, groups may submit and review together. Some systems also support assessment within groups, where members assess each other's contribution to a shared deliverable. This can address free-riding and inform the teacher about individual contributions, but it also raises questions of fairness and trust (see Section 6.8).

Privacy and integrity. A peer assessment system stores student work and the feedback exchanged around it. This raises questions of data protection, for example under the GDPR, about what is stored, who can access it and for how long. Related concerns include academic integrity, such as plagiarism and reuse of submissions.

7.5. Peer Assessment Across the IT Programme

A useful way to organize peer assessment in IT education is to follow the progression students make through a study programme. As students advance, both the artefacts they produce and the judgements they can make about the work of others grow in scope and sophistication. What a student can meaningfully assess is bounded by what they understand themselves, so peer assessment tasks are best pitched at, or a little below, the reviewer's own level of competence (Vygotsky 1978).

The choice of how to assess is as important as what is assessed. A software artefact can be assessed by *reading* it, inspecting the source to judge correctness, logic and clarity; by *running* it and observing its behavior; by *testing* it, with test suites supplied by the teacher or tests the reviewer writes; by *analyzing* it with tools, whether static analysis of style, complexity and potential faults or dynamic analysis of performance; and by *using* it as an end user would. These modes are not mutually exclusive, and a mature review often combines several. They also differ in how far they can be automated, as discussed at the end of this section.

7.5.1. Reading, running and testing small programs

In introductory programming, students produce small programs that solve well-defined problems, often a single file or a handful of functions with known correct behavior. Reviewers read the code to check that it is correct and follows the required standards, run it on sample inputs, and may apply premade tests that verify the required functionality (Ala-Mutka 2005). Because the criteria are clear and the right answer is known, the task is accessible even to novices. Reviewers can still be asked to comment on simple aspects of structure and design, such as the naming of variables, the division into functions and the readability of the code. The main value at this level is to establish the habit and skill of reading and reasoning about code written by someone else, and of articulating clearly whether it is correct and understandable (Hundhausen et al. 2013; Sitthiworachart and Joy 2004). Even here, exposure to alternative solutions begins to show students that a problem can be solved in more than one way.

7.5.2. Assessing code quality, security and performance

In intermediate courses on data structures, algorithms and software development, artefacts become larger and more open-ended, and the focus of assessment shifts from whether a program works to how well it is made. Reviewers can be asked to assess structure, naming, modularity and adherence to conventions, and to identify weaknesses a program may contain even when it passes its tests. Secure coding is an important part of this: reviewers can look for missing input validation, poor handling of errors and credentials, and other common vulnerabilities. Most programs at this level also rely on external libraries and frameworks, and reviewers can assess whether these are well chosen, up to date and free of known vulnerabilities. Tool support becomes prominent at this level. Static analysis can identify stylistic problems, excessive complexity, likely faults and security weaknesses, dependency scanners can flag known vulnerabilities in third-party packages, and profiling lets reviewers compare the performance of different algorithmic choices. There is usually no single correct answer, but a spectrum of better and worse solutions. The reviewer must form and justify a judgement about quality rather than simply check a result, which is precisely the evaluative judgement discussed in Section 3.3 (Tai et al. 2018; Sadler 1989).

7.5.3. Evaluating systems, workflows and users

In project and capstone courses, students build larger artefacts, often complete applications developed by teams over a substantial number of hours. The focus broadens from the code to the system as a whole and the people who use it. Reviewers may act as stakeholders or users rather than as code readers, running the application, evaluating its workflows and user experience, and judging whether it is fit for its purpose. Structured techniques such as heuristic evaluation and cognitive walkthrough can make this kind of feedback more rigorous (Nielsen and Molich 1990). Assessment may also extend to architecture and design, documentation, maintainability and the team's development process, including the use of version control and how the group has collaborated. When the artefacts are produced by groups, this is also the level at which assessment of individual contributions within the group becomes relevant. The judgement required is holistic and contextual, and draws on the reading, running and analytical skills developed at earlier levels.

7.5.4. Combining automated and peer assessment

At all levels, peer assessment works best in combination with automated assessment rather than in place of it. Premade tests, automated grading and static analysis are well suited to the objective and mechanical aspects of a submission, such as verifying that functionality is present and flagging clear

defects, and they scale easily to large classes (Ala-Mutka 2005). Leaving these tasks to automation frees peer reviewers to concentrate on what genuinely requires human judgement: the clarity of a solution, the quality of a design, the appropriateness of a trade-off and the experience of the user.

7.6. Designing Peer Assessment: Recommended Practice

Peer assessment can be designed in many ways. Topping (2017) lists a large number of design options, and a systematic review of 449 studies identified 27 design elements, while also finding that the designs used in practice are less diverse than they could be (Alqassab et al. 2023). In this section we go through the main design decisions a teacher must make and summarize recommended practice for each.

7.6.1. Clarify the purpose

The first decision is what the peer assessment is for. Most peer assessment is formative, intended to support learning rather than to produce grades (Alqassab et al. 2023), and we recommend this as the starting point. In IT education it is also useful to distinguish between peer assessment as a *topic* and as a *method*. Peer assessment can be a learning goal in itself, where students are trained in code review, design review or requirements review as professional practices. It can also be a method for reaching other learning goals, such as better programming or modelling skills, where the review is a means rather than an end (Sindre et al. 2003). The two purposes lead to different designs. A course that teaches code review as a practice should use the formats and tools of professional code review, while a course that uses peer assessment to improve students' own work should give priority to students acting on the feedback they receive. Being explicit about the purpose, also towards the students, helps avoid the unclear expectations that can undermine the activity (Sindre et al. 2003).

7.6.2. Plan timing and frequency

Peer assessment is most useful when there is still time to act on the feedback. In projects, this means reviewing during the project rather than only at the end, for example after an early prototype, at a midway milestone and before the final delivery. Several review points give students the chance to apply what they have learned from one round in the next, both as authors and as reviewers, and repeated participation is needed to build feedback literacy (Section 3.5). At the same time, each round adds workload for the students, so the number of rounds must be balanced against the size of the review task and the rest of the course.

7.6.3. Decide who reviews whom

Several decisions concern the reviewers. Feedback can be written by individual students or by groups. Group-written feedback is common in project courses and has clear advantages: the group must discuss and agree on its judgement, which itself is a form of the dialogue described in Section 3.1, the workload per student is reduced, and the feedback is often more considered. The disadvantage is that not every student necessarily takes part in the assessment.

The number of reviews per submission must also be decided. Several reviewers give authors more perspectives and make the overall feedback more robust, but each additional review increases the workload. In practice, two to four reviews per submission is often used as a compromise. Finally, the reviewer is not necessarily more competent than the author. Pitching the review task at the reviewers' level (Section 5), assigning several reviewers per submission, and complementing peer feedback with

feedback from teachers or teaching assistants, all reduce the risk that weak feedback undermines learning.

7.6.4. Make criteria and expectations explicit

Students need to know what to look for and what is expected of them. Structured rubrics direct attention to the aspects the course wants to emphasize and make reviews more consistent. Involving students in developing the criteria can increase both their understanding of quality and the accuracy of their feedback (Fraile et al. 2017; Topping 2017). Open criteria, or none at all, may instead be used deliberately to make students reflect on what quality is (Mirmotahari and Berg 2023). Whatever the choice, the criteria should direct reviewers towards the most important overall strengths and weaknesses of the work, not only towards details. Asking for both what works well and concrete suggestions for improvement helps reviewers avoid being only critical or only polite. Expectations about the effort required, such as the expected length and depth of a review, should also be explicit. In one early study, a large share of students put in so little effort that no learning effect could be confirmed, which the authors attributed mainly to low motivation and unclear expectations (Sindre et al. 2003). The IT2810 case in Section 8 shows a similar experience, and how making the reviews part of the portfolio assessment improved them.

7.6.5. Prepare and train the reviewers

Training is the design element with the clearest effect: in the meta-analysis by Li et al. (2020), reviewer training was the most influential factor for the learning gains from peer assessment. Training can be simple. Students can discuss examples of good and poor feedback, review an example submission together with the teacher, or take part in calibration, where they compare their own assessment of an example with a reference review (Section 4.2.2). Training should address the typical weaknesses of novice reviewers, such as a focus on details, vague comments, and comments that are either too harsh or too polite.

7.6.6. Choose an anonymity model

Anonymous reviewing lowers the threshold for honest criticism and reduces bias, and is a common choice. Open reviewing supports accountability and dialogue, and is closer to professional practice, where reviews are attributed. The choice can depend on the purpose: anonymity suits early rounds and inexperienced students, while open reviews suit courses that teach code review as a professional practice. If anonymity is chosen, students should be told how to remove identifying information from their code and repositories, since full anonymity is hard to achieve for software (Section 4.2.3).

7.6.7. Close the feedback loop

The value of peer assessment depends on students engaging with the feedback they receive. We recommend making this an explicit step: ask authors to go through each comment, decide whether to act on it, and briefly explain what they will change, or why they will not. This turns reading into active use of the feedback and develops the ability to judge the relevance of feedback (Carless and Boud 2018). Authors can also give feedback to their reviewers, sometimes called backward evaluation, by rating how useful the comments were (Misiejuk et al. 2021). This motivates reviewers to give better feedback and gives the teacher information about review quality. When students receive several reviews, a summary can help them get an overview, as long as they still engage with the individual comments

(Bayat Esfandani and Aalberg 2025). Finally, time must be set aside in the schedule for students to actually revise their work.

7.6.8. Handle grading and individual contributions with care

Peer scores are often not reliable enough to be used directly for grading (Sindre et al. 2003), and using them for grades can shift the students' focus from learning to judgement. In Norway, grading is also the responsibility of examiners appointed by the institution, which rules out letting peer scores count directly towards the grade. We therefore recommend keeping peer assessment formative. Participation can still be made compulsory, for example as a requirement for taking the exam, and the quality of the reviews a student writes can be assessed by the teacher. The same caution applies to assessment within groups. Asking group members to assess each other's contributions can make free-riding visible and support reflection on collaboration, but it is best used as input to the teacher and to the group's own discussion, not as a direct basis for individual grades.

7.6.9. Use system support and evaluate the activity

Computer-mediated peer assessment has been found to give greater learning gains than paper-based peer assessment (Li et al. 2020), and system support becomes indispensable as classes grow (Section 4.2). The workload must be realistic, for both students and teachers, and it is worth piloting the review task before using it at scale. Finally, the activity itself should be evaluated and improved from year to year. Useful sources are participation and completion rates, samples of reviews, the ratings authors give their reviewers, and students' perceptions of the activity.

7.7. Artificial Intelligence in Peer Assessment

Artificial intelligence (AI) can support peer assessment at several stages of the process, rather than replace it. Applications include allocating reviewers, improving individual reviews, deriving peer grades, analyzing the feedback, and supporting teacher oversight (Topping et al. 2025). Of particular relevance to learning, AI can analyze the quality of a peer review and give the reviewer immediate feedback on how to improve it, which can guide students' attention to important aspects of the assessment and help them become better assessors (Topping et al. 2025). AI can also aggregate and summarize feedback from several peers, and help teachers identify patterns, problems and possible bias in the process.

Students already use generative AI when they give peer feedback. In one study, 44% of students did so, mainly to make their feedback clearer and to generate constructive suggestions, and those who used it gave more suggestions addressing high-level issues (Noroozi et al. 2025). Other research shows that well-designed AI assistance for reviewers may still be little used by students (Pozdniakov et al. 2026). The question is therefore less what AI can do, and more how it should be introduced. Based on this, we recommend the following:

- **Start from the purpose of the activity.** AI support should strengthen the learning in the activity, not remove it. If the purpose is to develop evaluative judgement, an AI that writes the review for the student defeats the purpose, while an AI that comments on a draft review may support it.
- **Set clear rules for students' own use of AI.** Students should know whether and how they may use generative AI when writing feedback, and be asked to state if they have done so.

- **Consider the mode of assessment.** Our experience suggests that review tasks which require running, testing and observing a system are harder to delegate to AI than reviews based on reading code alone (Section 8).
- **Let summaries complement the individual comments.** AI-generated summaries can help students get an overview of feedback from several peers (Bayat Esfandani and Aalberg 2025), but students should still read the individual comments and decide for themselves what to act on.
- **Design for adoption.** AI support must be integrated in the review task and presented clearly, or students may not use it (Pozdniakov et al. 2026).
- **Consider privacy.** Sending student work and feedback to external AI services raises the same data protection questions as the peer assessment system itself.
- **Evaluate before scaling up.** AI support should be piloted, and its effect on the quality of the feedback and on student learning evaluated, before it is used at scale.

7.8. Use Cases from Three NTNU Courses

This section presents three examples of how peer assessment is used in courses at the Department of Computer Science (IDI), NTNU: IT2810 Web Development, taught in Trondheim, and PROG2006 Advanced Programming and PROG2005 Cloud Technology, taught in Gjøvik. The courses differ in level, size, type of artefact and in whether students work individually or in teams, and peer assessment has been developed independently by the teachers of each course. The purpose is not to compare the courses, but to exemplify the variation in intention, implementation and experience, and to relate it to the recommended practice in Section 6. Table 3 summarizes the main characteristics. Two themes recur across the cases: the importance of students seeing the value of the activity, and the growing influence of students' use of generative AI.

Table 3: Main characteristics of peer assessment in the three courses.

	IT2810 Web Development	PROG2006 Advanced Programming	PROG2005 Cloud Technology
Level and size	3rd year, about 200 students	2nd year, 35–45 students	80–100 students
Artefact reviewed	Team project (3–4 students): Git repository and running prototype at each milestone	Individual assignments in Rust and Haskell	Group project: deployed cloud service
Reviewers and allocation	Individual students review other teams' projects; each project receives up to eight reviews	Individual students; random allocation	Individual students from other groups; self-balancing allocation, two to four reviewers per project
System	FeedbackFruits	CSAMS; inline review in GitLab since 2024	CSAMS
Anonymity	Reviewers anonymous; projects known, since reviewers access the team's Git repository	Single-blind in CSAMS; open identities in GitLab	Single-blind
Review format	Short advice for early milestones; rubrics based on the project requirements in the last two rounds	Semi-structured review forms; inline code comments	Strongly structured checklist
Assessment mode	Reading the code and running the prototype	Reading and judging code	Running, testing, tool-based analysis and use of the system
Use of received feedback	Feedback on milestones informs later work; reviewing counts in the portfolio assessment	Revision for a final portfolio; authors rate the reviews they receive	No revision; value lies mainly in the reviewing itself
Experience with AI	AI summaries of reviews and AI guidance for reviewers explored	Declining interest in peer feedback; widespread delegation of reviews to AI in 2025–2026	Students largely carry out the reviews themselves

IT2810 Web Development is a course in modern web development for students in the third year of the bachelor's programme in Informatics. It was originally introduced to give students competences directly relevant for the first job many of them take after a bachelor's degree. The course has now been running

for ten years, and although it has changed every year, it is still founded on the original design. It is a project-based course in which students work in teams of three to four to build web applications. With around 200 participants, and thus around 50 projects, it is not feasible for the instructor to give each team substantial feedback during the course. The course is founded on learning through construction, where the primary learning activity is to create an artefact, learning about technologies and methods along the way. The lectures are mainly used for short introductions to technologies and practices, and students find their own learning resources for more detailed instruction. Over the years we have explored different setups for the project. The current version starts with a small initial project that establishes a basic level of experience with the technologies, after which students work on a larger project with a common set of functional and technology requirements. During the project, students submit milestone deliverables in the form of a Git repository and a running prototype on a virtual machine.

Peer assessment has always been a core learning activity in the course, with the purpose of training students in project inspection and quality assessment, and of enabling them to learn from each other. The reviews are organized in FeedbackFruits. They are single-blind: reviewers are anonymous, but the projects are not, since reviewers get access to the team's Git repository (cf. Section 4.2.3). One reason why peer assessment is relevant in this course is that the projects are reasonably synchronized in the state of the application, the technologies used and the functionality implemented. Reviewing someone else's project is therefore directly relevant for one's own learning: when comparing their own project with others, students have a good chance of either learning from the other project or giving relevant advice (Indriasari et al. 2020).

The first thing we learned was the importance of communicating the purpose of the activity well. Students choose strategically which activities to invest effort in, and without a proper justification we easily lose their engagement, and with it the learning (Sindre et al. 2003). We also learned that even students who agreed with the purpose did not necessarily find the task important, which led to many short reviews without substance. To make students invest in their reviews, we made reviewing a mandatory element of the portfolio assessment (cf. Section 6.8), which has substantially increased the quality of the reviews. Another lesson is that students need to find reading the reviews beneficial. Reviewing intermediate deliverables enables students to learn from and act on the comments they receive (Section 6.2). On the practical side, we have learned the importance of a well-defined setup with clear deadlines. We use rubrics with different levels of detail. For the early milestones, we ask for short reviews in the form of good advice; here the process is as much a way for teams to show others what they are planning and to get confirmation. The last two reviews are more substantial, with rubrics that rephrase the main requirements of the project.

Although students seem to find reviewing both interesting and valuable for learning, several challenges remain. Since students review individually, each project receives up to eight reviews, which may contradict each other (Section 6.7). We have run a project on summarizing the reviews with generative AI, which is a promising solution, although the language models available at the time did not produce sufficiently reliable summaries (Bayat Esfandani and Aalberg 2025). More recently, we have explored AI as guidance for students when they create and write their feedback (cf. Topping et al. 2025). The use of AI in project inspection may remove some of the learning we want to encourage, but it is also a way of reducing the cognitive load students often experience when they have to inspect an unfamiliar project (Sweller 1988). AI can also help students interpret the rubrics and shape their reviews. The future use of peer assessment in the course will have to be designed for this new landscape.

PROG2006 is a second-year Advanced Programming course with 35–45 students, built around a series of individual assignments in Rust and Haskell that emphasize the choice of abstractions and programming constructs rather than functional correctness alone. Peer review has been part of the course for five iterations (2022–2026), supported by CSAMS (Computer Science Assignment Management System), developed locally at NTNU Gjøvik (see Section 4.2), which lets teachers configure both submission and review forms and thereby supports semi-structured feedback. Submissions are randomly allocated, and reviews are single-blind: reviewers know whose work they assess, but authors do not know who reviewed them. Students may revise their assignments for a final portfolio, so received feedback can be acted upon, closing the loop on which the formative value of feedback depends (Hattie and Timperley 2007; Carless and Boud 2018). Since 2023, authors have rated each review they receive for accuracy, constructiveness and usefulness, a form of backward evaluation that encourages engagement with the feedback (Misiejuk et al. 2021). In 2024, form-based review was complemented by inline code review in GitLab with open identities, mirroring professional code review practice (Chong et al. 2021).

Reference group discussions and informal feedback from students were positive in the early iterations. Students particularly appreciated the use of GitLab and Git, and the inline code comments. In later iterations (2025–2026), however, we observed declining interest in the feedback students received from peers. We attribute this, tentatively, to students obtaining immediate feedback on their code from LLM-based coding assistants (Prather et al. 2023), which reduces the perceived value of slower and less polished comments from fellow students. Moreover, peer review takes place after students have finished and submitted their work, whereas coding assistants give immediate feedback while students are still working on the assignments (cf. Section 6.2). In 2025, some students also used coding assistants to write their reviews, and in 2026 this became widespread, with LLM-based agents commenting on code in place of the students themselves. Such delegation undermines the main rationale for the activity, since the learning gained from producing feedback (Li et al. 2010; Nicol et al. 2014) is lost when the judgement is outsourced. This contrasts with reports that GenAI support can improve the substance of peer feedback (Noroozi et al. 2025): the difference lies in whether the tool assists the reviewer's judgement or replaces it. The conceptual focus of the assignments makes delegation both easy to do and difficult to detect, because the review concerns judgements about design rather than observable behavior.

For 2026, we planned to shift the task from reviewing peers' work to reviewing LLM-generated feedback on that work, with the aim of retaining the exercise of evaluative judgement (Tai et al. 2018), including judgement of AI output (Bearman et al. 2024). Feedback was generated with locally hosted models (GLM, Qwen) on NTNU's IDUN computing cluster.[1] Staff pre-testing showed that the generated feedback was largely accurate for clearly strong or clearly weak submissions, in line with studies reporting that LLM-generated programming feedback is useful in many, but not all, cases (Kiesler et al. 2023). It failed to be constructive, however, where a student had a sound algorithmic idea but an incorrect implementation, or could not produce working code at all. These are precisely the cases where reconstructing the author's intent matters most and where formative feedback is most needed. Given this unreliability, the AI-feedback activity was cancelled and the course ran conventional peer review. How to design peer review for a conceptually oriented course in which students can readily delegate their reasoning to LLMs remains an open question.

PROG2005 Cloud Technology has 80–100 students who develop and deploy cloud services in project groups. Peer review has been used in this course since 2022 as well. Like PROG2006, the course uses CSAMS, and reviews are single-blind. Each group submission is reviewed by individual students from

other groups; the allocation is self-balancing and typically yields two to four reviewers per project. The review is strongly structured: reviewers work through a rubric of predefined checklist items (cf. Chong et al. 2021) that require deploying and running the reviewed service, executing tests, applying code-metric tools, interacting with the system, and documenting the outcomes. In the terms of Section 5, the review relies on running, testing and tool-based analysis rather than on reading alone. Because these judgements depend on observing the behavior of a deployed system, they are considerably harder to delegate to an LLM-based assistant than judgements made from source code; the write-up can be delegated, but the observations cannot. We observe that students largely carry out the reviews themselves. The structured format trades the open-ended reflection supported by less constrained formats for consistency and authenticity of the checks. Students cannot revise their submissions after review, so the formative value lies mainly in the reviewing activity itself and in transfer to later work, rather than in acting on received feedback (cf. Section 6.7).

Taken together, the PROG2006 and PROG2005 cases suggest that the robustness of peer review to LLM delegation depends on the assessment mode (Section 5): reviews requiring execution and observation of a running system resist delegation better than reviews based on reading and judging code. The courses also differ in artefact type, individual versus group work, cohort size and the opportunity for revision, so this should be read as a hypothesis for further investigation rather than a finding.

Across all three cases, some lessons recur. Students engage with peer assessment when they see its value, whether because the purpose is well explained, because the reviews count in the assessment, or because the feedback arrives while it can still be used. Generative AI also affects all three courses. It can support reviewers, as explored in IT2810, but it can also take over the judgement that the activity is meant to develop, as observed in PROG2006. The cases thus illustrate both the recommended practice in Section 6 and the open questions about AI raised in Section 7.

7.9. Concluding Remarks

Peer assessment is a well-founded learning activity, and it is particularly relevant for IT education, since reviewing the work of others is part of professional practice. It is not, however, a quick way to provide feedback with fewer teaching resources. Its value depends on careful design: a clear purpose, timing that leaves room to act on the feedback, criteria that direct attention to what matters, training of reviewers, a deliberate choice of anonymity, and a step where students actively engage with the feedback they receive. It also depends on system support that can handle the orchestration of the process, the variety of software artefacts, and the demands of group projects.

As students progress, peer assessment can grow with them, from reading and testing small programs to evaluating code quality, security and complete systems built by teams. Combined with automated assessment for the mechanical aspects, and possibly supported by AI, it lets students concentrate on the judgements that require human insight. The use cases in Section 8 show, however, that generative AI also changes the conditions for peer assessment. When students can get immediate feedback from coding assistants, or let an AI tool write their reviews, the activity must be designed so that the judgement remains the students' own, for example through review tasks that require running and observing a system. Teachers who start with a modest design, train their students, and improve the activity from year to year are likely to find that peer assessment becomes a valuable part of project-based IT education.

Declaration of AI Use

During the preparation of this chapter, the authors used Claude (Anthropic) to produce preliminary drafts of parts of the text, and to assist with language editing, restructuring of text and identifying relevant literature. All AI-generated text and suggestions were reviewed and revised by the authors, who take full responsibility for the content of the chapter.

Copyright

Peer Assessment in Project-Based IT Education © 2026 by Trond Aalberg, Somayeh Bayat Esfandani and Mariusz Nowostawski is licensed under CC BY-NC-ND 4.0



References

Ala-Mutka, Kirsti M. 2005. "A Survey of Automated Assessment Approaches for Programming Assignments." *Computer Science Education* 15 (2): 83–102.

Alqassab, Maryam, Jan-Willem Strijbos, Ernesto Panadero, Javier Fernández Ruiz, Matthijs Warrens, and Jessica To. 2023. "A Systematic Review of Peer Assessment Design Elements." *Educational Psychology Review* 35 (1): 18. https://doi.org/10.1007/s10648-023-09723-7.

Bayat Esfandani, Somayeh, and Trond Aalberg. 2025. "The Impact of Multi-Peer Feedback Summary Organization on Review and Implementation of Feedback." *Companion Proceedings of the ACM International Conference on the Foundations of Software Engineering (FSE Companion 2025)* (Trondheim, Norway).

Bearman, Margaret, Joanna Tai, Phillip Dawson, David Boud, and Rola Ajjawi. 2024. "Developing Evaluative Judgement for a Time of Generative Artificial Intelligence." *Assessment & Evaluation in Higher Education* 49 (6): 893–905. https://doi.org/10.1080/02602938.2024.2335321.

Berg, Ineke van den, Wilfried Admiraal, and Albert Pilot. 2006. "Designing Student Peer Assessment in Higher Education: Analysis of Written and Oral Peer Feedback." *Teaching in Higher Education* 11 (2): 135–47.

Black, Paul, and Dylan Wiliam. 1998. "Assessment and Classroom Learning." *Assessment in Education: Principles, Policy & Practice* 5 (1): 7–74.

Boud, David, and Rebeca Soler. 2016. "Sustainable Assessment Revisited." *Assessment & Evaluation in Higher Education* 41 (3): 400–413.

Cadima, R., D. Tavares, and H. Menino. 2024. "Assessment for Learning: A Systematic Review on Peer Assessment in Higher Education." *EDULEARN24 Proceedings* (Palma, Spain), 16th International Conference on Education and New Learning Technologies. https://doi.org/10.21125/edulearn.2024.0327.

Carless, David, and David Boud. 2018. “The Development of Student Feedback Literacy: Enabling Uptake of Feedback.” *Assessment & Evaluation in Higher Education* 43 (8): 1315–25.

Chong, Chun Yong, Patanamon Thongtanunam, and Chakkrit Tantithamthavorn. 2021. “Assessing the Students’ Understanding and Their Mistakes in Code Review Checklists: An Experience Report of 1,791 Code Review Checklist Questions from 394 Students.” *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 20–29.

Dochy, Filip, Mien Segers, and Dominique Sluijsmans. 1999. “The Use of Self-, Peer and Co-Assessment in Higher Education: A Review.” *Studies in Higher Education* 24 (3): 331–50.

Fraile, Juan, Ernesto Panadero, and Rodrigo Pardo. 2017. “Co-Creating Rubrics: The Effects on Self-Regulated Learning, Self-Efficacy and Performance of Establishing Assessment Criteria with Students.” *Studies in Educational Evaluation* 53: 69–76. https://doi.org/10.1016/j.stueduc.2017.03.003.

Hattie, John, and Helen Timperley. 2007. “The Power of Feedback.” *Review of Educational Research* 77 (1): 81–112.

Hmelo-Silver, Cindy E., Ravit Golan Duncan, and Clark A. Chinn. 2007. “Scaffolding and Achievement in Problem-Based and Inquiry Learning: A Response to Kirschner, Sweller, and Clark (2006).” *Educational Psychologist* 42 (2): 99–107.

Hundhausen, Christopher D., Anukrati Agrawal, and Pawan Agarwal. 2013. “Talking about Code: Integrating Pedagogical Code Reviews into Early Computing Courses.” *ACM Transactions on Computing Education* 13 (3): 1–28.

Indriasari, Theresia Devi, Andrew Luxton-Reilly, and Paul Denny. 2020. “A Review of Peer Code Review in Higher Education.” *ACM Transactions on Computing Education (TOCE)* 20 (3): 1–25.

Kiesler, Natalie, Dominic Lohr, and Hieke Keuning. 2023. “Exploring the Potential of Large Language Models to Generate Formative Programming Feedback.” *2023 IEEE Frontiers in Education Conference (FIE)*, 1–5. https://doi.org/10.1109/FIE58773.2023.10343457.

Li, H., Y. Xiong, C. V. Hunter, X. Guo, and R. Tywoniw. 2020. “Does Peer Assessment Promote Student Learning? A Meta-Analysis.” *Assessment & Evaluation in Higher Education* 45 (2): 193–211. https://doi.org/10.1080/02602938.2019.1620679.

Li, Lan, Xiongyi Liu, and Allen L. Steckelberg. 2010. “Assessor or Assessee: How Student Learning Improves by Giving and Receiving Peer Feedback.” *British Journal of Educational Technology* 41 (3): 525–36. https://doi.org/10.1111/j.1467-8535.2009.00968.x.

Liu, Ngar-Fun, and David Carless. 2006. “Peer Feedback: The Learning Element of Peer Assessment.” *Teaching in Higher Education* 11 (3): 279–90.

Merry, Stephen, and Paul Orsmond. 2020. “Peer Assessment: The Role of Relational Learning Through Communities of Practice.” *Studies in Higher Education* 45 (7): 1312–22.

Mirmotahari, Omid, and Yngvar Berg. 2023. “Utvikling av førsteårsstudenters tilbakemeldingskompetanse gjennom egen-vurdering.” *NIKT: Norsk IKT-konferanse for forskning og utdanning*. https://api.nva.unit.no/publication/0198cc725930-18cb78a1-d079-4e7b-9a59-b3d496151175.

Misiejuk, Kamila, Barbara Wasson, and Kjetil Egelandsdal. 2021. “Backward Evaluation in Peer Assessment: A Scoping Review.” *Computers & Education* 175: 104319.

Molenaar, Inge. 2022. “The Concept of Hybrid Human-AI Regulation: Exemplifying How to Support Young Learners’ Self-Regulated Learning.” *Computers and Education: Artificial Intelligence* 3: 100070.

Nicol, David. 2010. “From Monologue to Dialogue: Improving Written Feedback Processes in Mass Higher Education.” *Assessment & Evaluation in Higher Education* 35 (5): 501–17.

Nicol, David J., and Debra Macfarlane-Dick. 2006. “Formative Assessment and Self-Regulated Learning: A Model and Seven Principles of Good Feedback Practice.” *Studies in Higher Education* 31 (2): 199–218.

Nicol, David, Avril Thomson, and Caroline Breslin. 2014. “Rethinking Feedback Practices in Higher Education: A Peer Review Perspective.” *Assessment & Evaluation in Higher Education* 39 (1): 102–22.

Nielsen, Jakob, and Rolf Molich. 1990. “Heuristic Evaluation of User Interfaces.” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI ’90)*, 249–56.

Noroozi, Omid, Golnoush Haddadian, Xingshi Gao, Christian Schunn, Maryam Alqassab, and Seyyed Kazem Banihashem. 2025. “The Value of GenAI for Peer Feedback Provision: Student Perceptions and Impacts.” *International Journal of Educational Technology in Higher Education* 22 (1): 61.

Ortega-Ruipérez, Beatriz, and José Miguel Correa-Gorospe. 2024. “Peer Assessment to Promote Self-Regulated Learning with Technology in Higher Education: Systematic Review for Improving Course Design.” *Frontiers in Education* 9. https://doi.org/10.3389/feduc.2024.1376505.

Pintrich, Paul R. 2000. “The Role of Goal Orientation in Self-Regulated Learning.” In *Handbook of Self-Regulation*, edited by Monique Boekaerts, Paul R. Pintrich, and Moshe Zeidner. Academic Press.

Pozdniakov, Stanislav, Jonathan Brazil, Seyyed Kazem Banihashem, et al. 2026. “AI Assistance in Peer Feedback Provision: Pedagogically Sound, but Minimally Adopted.” *Computers & Education*, 105591.

Pramudianto, Ferry, Tarun Chhabra, Edward F. Gehringer, and Christopher Maynard. 2016. “Assessing the Quality of Automatic Summarization for Peer Review in Education.” *Proceedings of the Educational Data Mining (EDM) Workshops*.

Prather, James, Paul Denny, Juho Leinonen, et al. 2023. “The Robots Are Here: Navigating the Generative AI Revolution in Computing Education.” *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR ’23)*, 108–59. https://doi.org/10.1145/3623762.3633499.

Sadler, D. Royce. 1989. “Formative Assessment and the Design of Instructional Systems.” *Instructional Science* 18 (2): 119–44.

Shute, Valerie J. 2008. “Focus on Formative Feedback.” *Review of Educational Research* 78 (1): 153–89.

Sindre, Guttorm, Michail Giannakos, Birgit Rognebakke Krogstie, Robin Isfold Munkvold, and Trond Aalberg. 2018. “Project-Based Learning in IT Education: Definitions and Qualities.” *Uniped* 41: 147–63.

Sindre, Guttorm, Daniel L. Moody, Terje Brasethvik, and Arne Sølvberg. 2003. “Introducing Peer Review in an IS Analysis Course.” *Journal of Information Systems Education* 14 (1): 101–19.

Sitthiworachart, Jirarat, and Mike Joy. 2004. “Effective Peer Assessment for Learning Computer Programming.” *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE ’04)*, 122–26.

Smith, Calvin Douglas, Kate Worsfold, Lynda Davies, Ron Fisher, and Ruth McPhail. 2013. “Assessment Literacy and Student Learning: The Case for Explicitly Developing Students’ ‘Assessment Literacy’.” *Assessment & Evaluation in Higher Education* 38 (1): 44–60.

Sutton, Paul. 2012. “Conceptualizing Feedback Literacy: Knowing, Being, and Acting.” *Innovations in Education and Teaching International* 49 (1): 31–40.

Sweller, John. 1988. “Cognitive Load During Problem Solving: Effects on Learning.” *Cognitive Science* 12 (2): 257–85.

Tai, Joanna, Rola Ajjawi, David Boud, Phillip Dawson, and Ernesto Panadero. 2018. “Developing Evaluative Judgement: Enabling Students to Make Decisions about the Quality of Work.” *Higher Education* 76 (3): 467–81.

Topping, Keith. 1998. “Peer Assessment Between Students in Colleges and Universities.” *Review of Educational Research* 68 (3): 249–76.

Topping, Keith J. 2009. “Peer Assessment.” *Theory Into Practice* 48 (1): 20–27.

Topping, Keith J. 2017. “Peer Assessment: Learning by Judging and Discussing the Work of Other Learners.” *Interdisciplinary Education and Psychology* 1 (1): 7. https://doi.org/10.31532/InterdiscipEducPsychol.1.1.007.

Topping, Keith J. 2018. *Using Peer Assessment to Inspire Reflection and Learning*. Routledge.

Topping, Keith J., Ed Gehringer, Hassan Khosravi, Srilekha Gudipati, Kaushik Jadhav, and Surya Susarla. 2025. “Enhancing Peer Assessment with Artificial Intelligence.” *International Journal of Educational Technology in Higher Education* 22 (1): 3.

Vygotsky, Lev S. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.

Wang, Yanqing, Hang Li, Yuqiang Feng, Yu Jiang, and Ying Liu. 2012. “Assessment of Programming Language Learning Based on Peer Code Review Model: Implementation and Experience Report.” *Computers & Education* 59 (2): 412–22.

Wenger, Etienne C., and William M. Snyder. 2000. “Communities of Practice: The Organizational Frontier.” *Harvard Business Review* 78 (1): 139–46.

Winstone, Naomi, and David Carless. 2019. *Designing Effective Feedback Processes in Higher Education: A Learning-Focused Approach*. Routledge.

Wood, David, Jerome S. Bruner, and Gail Ross. 1976. “The Role of Tutoring in Problem Solving.” *Journal of Child Psychology and Psychiatry* 17 (2): 89–100.

Zimmerman, Barry J. 2000. “Attaining Self-Regulation: A Social Cognitive Perspective.” In *Handbook of Self-Regulation*, edited by Monique Boekaerts, Paul R. Pintrich, and Moshe Zeidner. Academic Press.

8. Self-Paced Mastery Learning

Guttorm Sindre

Norwegian University of Science and Technology (NTNU)

8.1. Introduction: Who is this chapter for?

You may be a teacher curious to try out mastery learning, either making a new course or transforming one of your existing courses to such an approach. You may be a student wondering whether you should urge your teachers to try out mastery learning. You may be an educational leader or involved in tech/admin or pedagogical support where you are a stakeholder in course improvement and redesign. Or, even if none of the above apply, you may simply be eager to know more about mastery learning: what is it, and what can we achieve by it? This is the tale of the course IT1001, with its self-paced mastery learning design, and the experiences and advice we have distilled from it.

IT1001, which teaches introductory Python programming to first semester STEM teacher students, was designed by NTNU and Excited personnel in 2022/2023, and offered for the first time in the autumn of 2023. Despite some "childhood diseases" in that first run, the course was highly successful in terms of student satisfaction, which increased further in subsequent years when some issues with the initial course design were resolved. This chapter presents mastery learning in general, the course experiences, and especially discusses design trade-offs when making such a course. The key difference between this chapter and more scientific accounts of the course – which can be found in conference and journal papers – is that it focuses less on analysis of course data, more on personal experiences and advice to others on how to make courses with mastery learning. Readers who are more interested in academic research articles about the course, would be recommended to look up (Sindre and Hansen 2024, Sindre 2025, Sindre and Hansen 2025) – or in Norwegian, (Hansen, Sindre et al. 2024, Sindre and Hansen 2025, Sindre and Hansen 2026).

The rest of the chapter is structured as follows: First we give a brief account of the history and key ideas of mastery learning. Then we present our experiences from the IT1001 course. The third and biggest section goes into Do-It-Yourself advice, outlining important design decisions that you will be facing in the development of such a course, as well as some process advice. Towards the end, some reflections are made on how such course designs can also work well, and be further enhanced, in the age of GenAI.

8.2. What is mastery learning?

The start of mastery learning is often considered to be the pioneering efforts of Bloom (Bloom 1968, Bloom 1971) and Keller (Keller 1968, Keller and Sherman 1974), although these again were inspired by yet earlier ideas. The main feature of mastery learning is that a course be divided into modules to be taken in sequence, and each student must pass the first module before proceeding to the second, etc. The modules should be small enough not to be overwhelming to the students – typically causing a quite big number of modules – and students should complete each of them with a high degree of mastery, not moving ahead to more advanced material before they have understood the basics. Since students learn at different paces, they must be allowed to spend different time on the tasks. In an ordinary course where all students move at the same pace, the idea is that all spend equal calendar time (e.g., from semester start to exam), but end up with widely different levels of mastery, ranging from the excellent to the barely passable – or less in case of failure. In a course using mastery learning, the pursued outcome is instead a high level of mastery for all – but instead maybe having to allow different usage

of calendar time for the students. Of the two approaches to mastery learning, our course is most similar to Keller's Personalized System of Instruction (PSI), with the following give principles:

- **High mastery requirement:** A student must master one module before moving on to the next, and all students should master all modules – typically at high thresholds like 90% to pass.
- **Use of proctors:** Teaching assistants played an important role in Keller's PSI courses, administering the module tests to students.
- **Self-pacing:** Students should be allowed to move at their own pace up the mastery ladder of modules. A quick learner should not be held back by other students needing more time, and those needing time should not be prematurely forced into new material just because others want to go faster.
- **Stress upon the written word.** Written material could be studied at any time and speed, thus giving students flexibility to do self-pacing.
- **Use lectures sparingly, just for motivational purposes, not for teaching new material.** Lectures, on the other hand, would not work well with self-pacing.

Both Keller's PSI and Bloom's mastery learning had promising empirical results (Kulik, Kulik et al. 1979, Kulik, Kulik et al. 1990, Hattie 2015). Yet, after a high-water-mark for mastery learning in the 1980's, the interest gradually declined through the 1990's (Eyre 2007), partly due to being labour-intensive for teaching staff, and partly due to organizational inertia in academia. Many courses also had problems with student procrastination due to self-pacing. In more recent years, e-learning has mitigated some of the problems with labour-intensiveness, and there has been a renaissance for mastery learning especially in programming courses (Purao, Sein et al. 2017, Garner, Denny et al. 2019, Toti, Chen et al. 2023, Szabo, Parker et al. 2025), as well as more generally in engineering and other disciplines (Perez and Verdin 2022, Mannion, Coyne et al. 2023).

8.3. Experiences from the IT1001 Course

Considering student satisfaction, the course IT1001 at the NTNU must be seen as a success. In each year 2023-25 an anonymous course survey has been distributed by the follow-up research team (teacher not being present) in the seminar in the 11th week of the 14-week semester. Figure 1 summarizes student responses to a 5-point question on overall satisfaction with the course (left) and a 3-point question whether it was worse, equal or better than other courses taken in the same semester (right). Even in the first run, when the course had some "childhood diseases", the student opinion was positive, with 60% satisfied or very satisfied, and 63% thought that the course was better than other courses they took. The subsequent years, when the main problems of the first run had been fixed, more than 70% considered the course better than other courses, and more than 80% were satisfied or very satisfied.

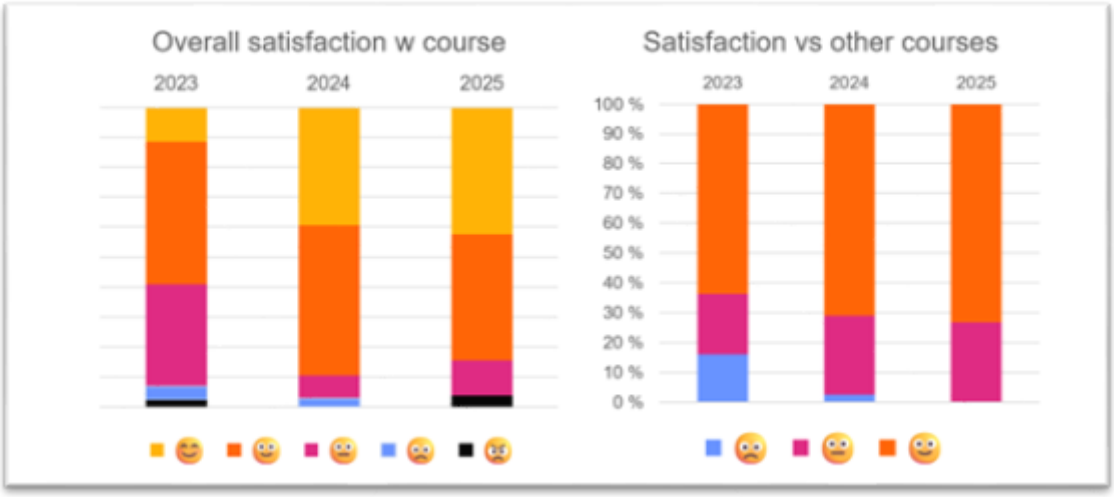


Figure 1: Satisfaction with the course, 2023-2025

A concern that some might have with switching to mastery learning is that a series of many tests during the semester, rather than just one exam at the end, might be stressful for the students. However, this is countered by the fact that students feel a much higher sense of own agency and control with a mastery learning approach. Figure 2 shows the results for the 2025 run on four of the questions in the survey. A clear majority of students agreed that the modules had clear objectives, that they had good control of what grade they could achieve, and that they had experienced mastery in the course. Although some students felt stressed (agree = orange in the top bar), this was a minority and focus group interviews revealed that they generally preferred many small tests during the semester rather than one big exam at the end. With less risk per test and possible retakes, failure had been made less dangerous. Formative practice tests pulling questions from the same item banks as the summative tests also meant that transparency was very high, so students would clearly see what they needed to learn or improve to pass a test (cf. very positive bar in Figure 2 about clear objectives).

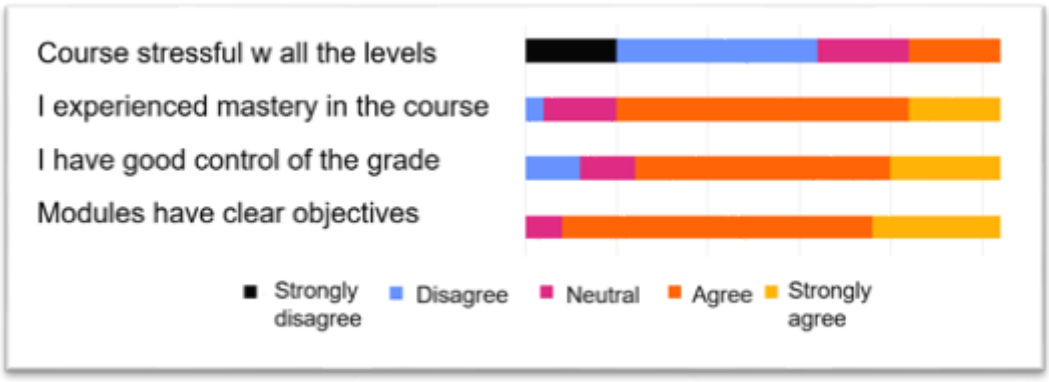


Figure 2. Student opinion on stress, mastery, control, and clarity, 2025

Figure 3 shows aggregated student response to questions about social aspects of the course.

Very few would have preferred a more traditional lectures + exam course – that question was hoped to be in the negative, with strongly disagree (black) and disagree (blue) being dominant. All students in 2025 were positive that the course gave a sense of belonging in the class and study program, and most agreed that it made it easier to get to know classmates and facilitated collaboration with peers. A

possible concern with self-paced mastery learning is that letting each student follow their individual pace would make it difficult to have meaningful group activities with student-active learning. As these results indicate, it is fully possible to have a course with meaningful group learning activities even with liberal self-pacing, and when we turn to the detailed design of the course, we will discuss further how this was achieved.

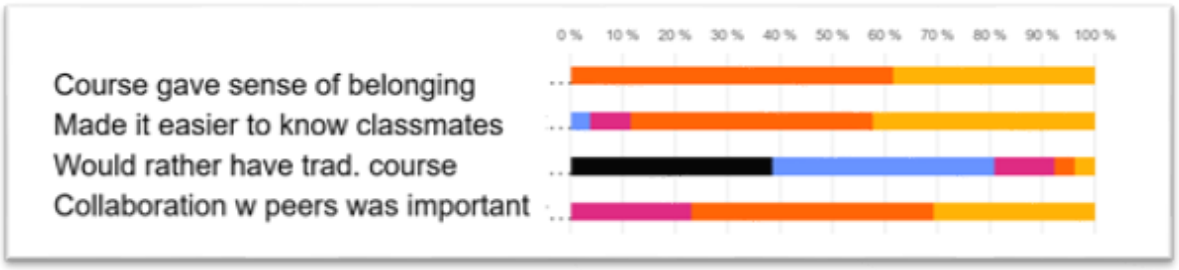


Figure 3. Student opinion on social aspects of the course, 2025

Figure 4 shows the results for student performance in the course, where the column 2022 is how STEM teacher students performed in the CS1 course they had before (a more traditional lecture + exam course, where they were together with several other study programs), while the columns 2023-25 show the results with the new course. As can be seen, grades are rather modest, with averages D+ (2022), D- (2023), C- (2024, 2025). Failure has dropped from 11% in 2022 to zero in 2024 and 2025, but the average became poorer in the first run of the new course in 2023 than it had been with the old course, because many students stopped at the lowest passing grade E in 2023. While the low average in 2023 can be considered negative, it demonstrates that students were satisfied with the course despite weak grades – i.e., the high student satisfaction has not been accomplished by giving away good grades easily. Concerning the grade variations from year to year, it must also be considered that the class is rather small, 48 in 2023, 40 in 2024, 31 in 2025, due to the sad fact that applications to teacher education in Norway has been declining. Hence, a rather small change to the number of students at various grades still makes a big visual impact in the bar chart.

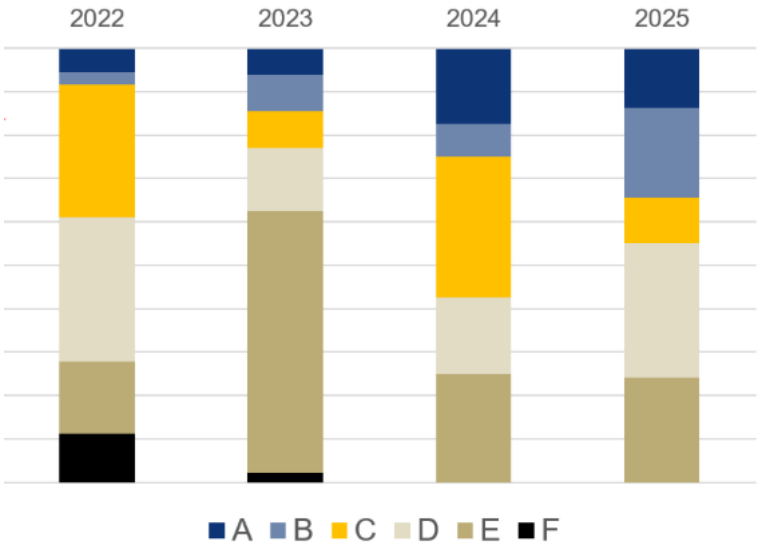


Figure 4. Grade distributions, 2022-25

The students reach their destination grades at widely different paces during the semester. The course has 9 modules named I, H, G, F, E, D, C, B, A – because they correspond to the grade thus taken. A student who wants A, must take all 9 modules, whereas one satisfied with the lowest passing grade E can stop at 5). Figure 5 shows the progression of the class through the series of module tests in 2025. The red "J" bar indicates the fraction of the class who had passed zero tests, then the lightest grey 1 (I), the next grey 2 (H), and so forth, up to the yellow indicating those passing all 9 tests. As an example, look at week 41 in the figure, when there were students at 6 different levels. The fastest few had passed 8 tests in 8 weeks (I, …, B) and were ready to take the A-test. The slowest few had just passed 3 tests (I, H, G) and were preparing to take the F-test. In 2024, the spread was even bigger on the midway mark, with students at 8 different levels (from passing 8 tests to just passing 1). Despite being free to choose their own pace, all landed on their feet and managed a passing grade, both in 2024 and 2025.

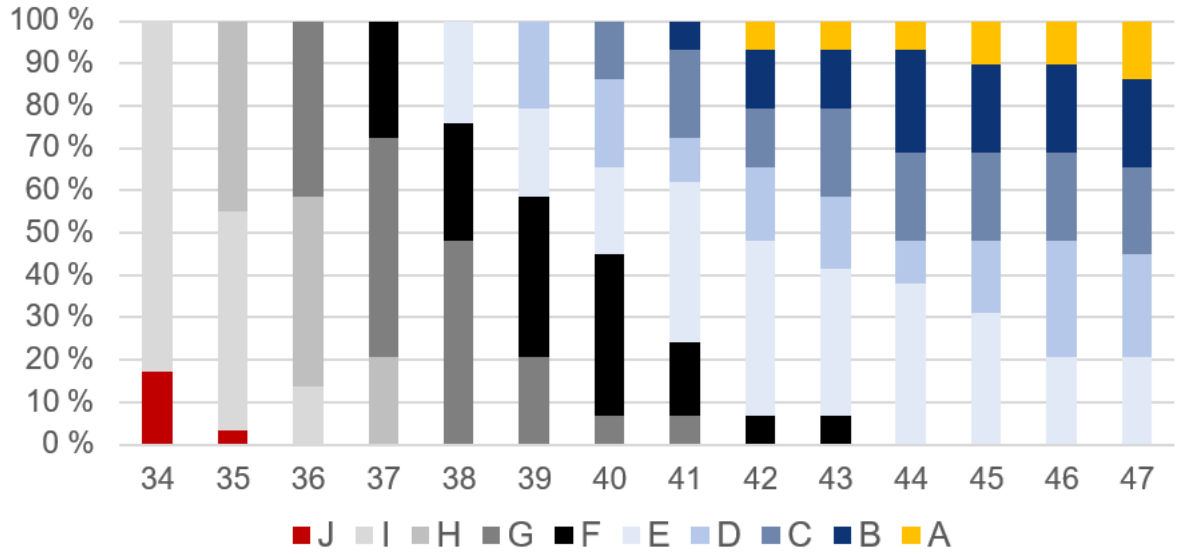


Figure 5. Test status of the class by week, 2025

Another thing that can be seen from Figure 5 is that the stacked bars are quickly changing in the first half of the semester, and much more stable towards the end, i.e., most students preferred to start quickly taking tests rather than waiting as long as possible. In many other courses with self-paced mastery learning around the world, a major challenge has been student procrastination (Purao, Sein et al. 2017, Ott, McCane et al. 2021, Szabo, Parker et al. 2025), often causing teachers to introduce intermediate deadlines during the semester as a countermeasure (Chiu, Moss et al. 2024). In the subsequent section on course design and trade-offs, we will discuss further how our design mitigated procrastination without any use of mid-term deadlines.

8.4. Make your own course: Design trade-offs

In this section, we go through the various design issues for a mastery learning course. Our design decisions and rationales for IT1001 are used to exemplify the issues, but we also point out possible alternatives. We start by presenting some overarching design choices concerning modularization and assessment. Then we look into learning activities and resources, and some more detailed design issues.

8.4.1. Overarching design choices

Some key design choices are illustrated in Figure 6, where we show rough sketches of the layout of various mastery-learning courses.

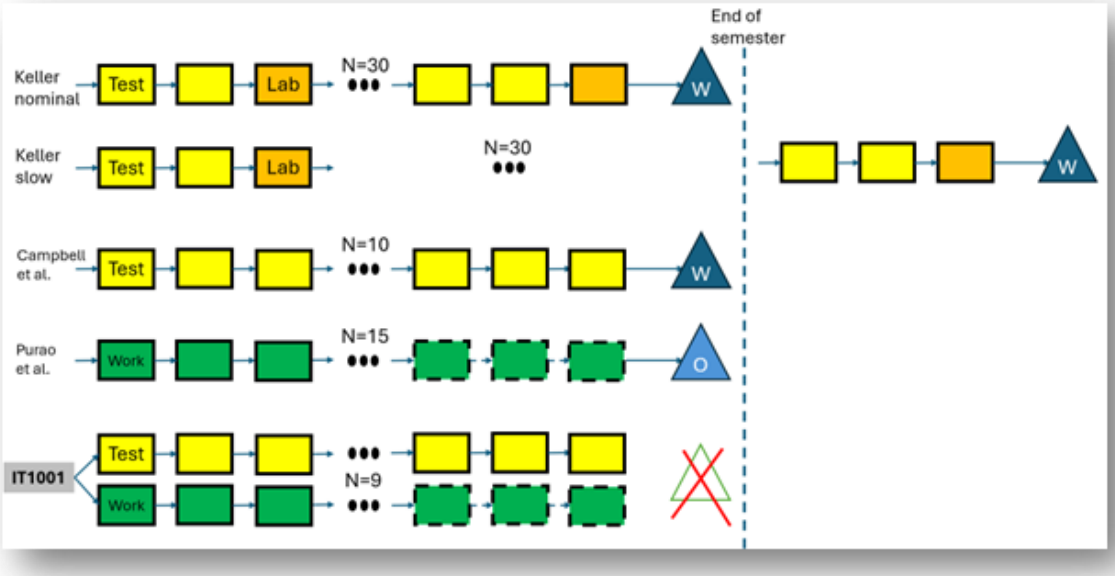


Figure 6. Various mastery learning course designs. Rectangles show modules, dashed voluntary, otherwise mandatory. Module contents are theory with tests (yellow), lab (orange), work assignments – here code writing (green). Triangles indicate exams, written (W) or oral (O).

The first two rows of Figure 6 illustrate Keller's psychology course PY112 from the 1960's. We are using two rows to capture the fact that students could continue into the next semester if they were unable to finish the modules in the nominal time (30 modules, nominal recommended pace 2 per week). All students had to take all modules to pass the course (cf. borders on rectangles), whereof 20 theory modules with tests, and 10 lab exercises. These were organized in a strict sequence, but each student could choose their pace. Failed tests or labs would have to be retaken before proceeding. Henceforth, we will classify this course design as **semester-flexible** (allowed to continue into the next semester), **all-take-all** (all students must take all modules), **dual type** (two different kinds of modules: test and lab), **single stream** (all modules in one sequence), and **partial exam** (the exam counts for part of the grade). In case of PY112 this was 25-30% depending on the exact year. With 30% for the exam, 50% would come from the module tests and 20% from the labs. Moreover, its modularization can be seen as quite **fine-granular**, with 30 small modules to be taken during the semester.

From the six properties of Keller's course identified in the previous paragraph, we can discuss six essential design trade-offs.

Trade-off #1: Semester-bound or -flexible? The ideal pedagogical idea of mastery learning is that every student can achieve a high level of mastery in a subject if given enough time, hence allow different pace, rather than force all students into the same time-window, which necessarily leads to varying the degree of mastery. There have been recent approaches to mastery learning that also used the semester-flexible approach, such as the CS1 course by Offutt et al. (2017). However, in most universities it will

not be acceptable to have a course where the slower tail of students drag into the next semester. Students are supposed to move on to their next courses, and the same with allocation of teaching rooms and staff. Hence, most newer approaches to mastery learning have decided that the course must finish when the semester ends, then grades must be reported even if some students did not complete. The choice in IT1001 was semester-bound, and we never really had serious considerations of something else.

Trade-off #2: All-take-all, or less? Several approaches to mastery learning have tried to maintain the all-take-all requirement within the bounds of the semester – one example is the course reported by (Campbell, Petersen et al. 2019) where students had to take 10 quiz modules and then sit an exam in the end. Typically, all-take-all implies that students who do not complete all modules will not get a grade, and a potential problem is that this may lead to bigger dropout or failure rates from courses. Going away from all-take-all does to some extent abandon the high ideals of mastery learning – the levels of mastery shown by a student taking all modules will necessarily be higher than one who takes fewer, who has then not learnt the content of the last modules. However, there are also pedagogical arguments in favor of a system where not all modules are mandatory. Mastery learning using module tests typically require 90% score to pass each module. If you demand that all students in a class should achieve a 90% mastery level over the entire curriculum of the course, you are actually demanding much more than in a typical lecture + exam course, where many exam answers may reveal limited mastery of especially the most difficult parts of the curriculum, yet still earn a passing grade. It is also totally legitimate for a student to have varying ambition levels for different courses, and, e.g., for some students intending to become high school STEM teachers, a modest passing grade in CS1 might be totally fine. For IT1001 we thus made the choice that the mandatory threshold to get a grade would be **5 / 9** modules, and modules beyond that would distinguish between lower and higher grades. Similar choices can be seen for the course of (Purao, Sein et al. 2017) in Figure 6, and for (Toti, Chen et al. 2023) which is not in the figure, though the exact number of modules will differ.

Trade-off #3: Single or dual type? While Keller's course had two different types of modules, many newer courses – like in CS1 – have had just a single type of modules. For most, these will be tests, such as (Offutt, Ammann et al. 2017, Campbell, Petersen et al. 2019), but it can also be programming assignments as in (Purao, Sein et al. 2017). Looking more broadly, it is also possible to find mastery learning courses where all modules have been assessed by practical performances, for instance a course in presentation technique by (Foss, Foss et al. 2014). The advantage of having just one single module type is that it tends to be simpler to organize. The disadvantage is if the learning outcomes of the course are such that one single form of assessment would not give sufficient coverage. This was the reason why IT1001 chose to have two different types of assessment – auto-scored tests done under supervision with locked-down laptops, and an individual programming project that students would gradually develop increment by increment during the semester. Our three main motivations for this choice were as follows:

- **Better coverage of learning outcomes.** The automated tests assess the understanding of programming concepts through multiple choice questions focused on code comprehension, as well as some problem-solving skills with program completion tasks such as dropdown fill-ins and Parsons problems (Ihantola and Karavirta 2011), but they do not contain any code-writing. This is instead addressed by the project, where students are working on the same program with gradually increasing functionality during the semester, so that it gradually becomes a program of several hundred code lines. Hence, the project also addresses more integrative learning outcomes related to programming (using many different programming concepts together) while

the smaller code snippets in test tasks typically deal with just a few programming concepts at a time. While the automated module tests might to a larger extent deal with specific concepts at a time, featuring tasks with rather short code snippets to be understood or completed, the semester-long project would expose students to using the various concepts together in a program that would typically grow to hundreds of code lines.

- **Increased authenticity and ownership.** The test tasks are fully set by the teacher, and necessarily rather generic. The project, on the other hand, is rather open so that each student can develop their own idea. The only requirement is that it must be a procedure-oriented pedagogical program written in Python, meant to help users (high school students) learn some topic within a high school STEM course. A student intending to major in Math should then make a program related to a math subject they might be likely to teach, while one intending to major in Biology would make something for that. Apart from this general rule, students are free to choose the topic (e.g., within the curriculum of high-school Biology) and the genre of pedagogical program (e.g., quiz, game, tutorial, simulator, data analysis for lab experiments or field trips, …)
- **Increased disciplinary and professional relevance.** Through the project, students are encouraged to reflect upon their future roles as teachers, thinking about what is difficult for students to learn in their high school subject of choice, and how a Python program might help with this. Sure, they are beginners and most of their programs will not reach a quality to be used in high school teaching sessions, but reflections about it can still be useful.
- **Gatekeeping for integrity.** From the bullets above, one might be tempted to ask why not go with only the project. However, this would present a much bigger challenge with potential cheating. With project work partly done at home, it is much more difficult to know that what is delivered really reflects the students' own competence – or that of helpful friends or AI. With project only, we might have had to put many more resources into assessing project deliveries, for instance having thorough face-to-face interviews with each student for each module to check that it was their own work, and that they had understood the underlying programming concepts otherwise assessed by tests.

Hence, even if tests + project makes it more complicated for us than, e.g., going tests only, we felt that this was the right choice with our course and our student group.

Trade-off #4: Single or double stream? This question only becomes relevant if going dual type. Most mastery learning courses have a single stream sequence of modules. Although Keller had two different types of modules (theory with test and lab) they were in one sequence, so you could not take a lab until you had learnt the theory relevant for that lab experiment. Our tests and project increments are topically connected, e.g. the first project increment shall demonstrate practical and purposeful usage of the same programming mechanisms as were in focus on the first test, and there are similar topical matches for the subsequent tests and project increments. Hence, it could have made sense for us to go, e.g.,

- Single stream, one specific order: Test #1 → Project #1 → Test #2 → Project #2 → …, since it might be natural to test first (understanding of programming concepts), then demonstrate practical usage in code writing.
- Single stream, free within level: (Test #1 & Project #1) → (Test #2 & Project #2) → …, meaning that you must have done both test and project at level #N before doing anything at level #N+1.

We chose not to require that students must finish project #N before taking test #N+1, and also not to require test #N before project #N+1, as we felt it would constrain the students too much for no good reason. Our only requirements are that tests must be taken in sequence, project increments must be delivered in sequence, and to take a certain grade, you must reach the corresponding level both for tests and project increments. Within these constraints, each student can compose their sequence to achieve their wanted grade in any way they want, taking all the tests first, then project, or the other way around, or in some intermingled way.

Trade-off #5: Exam – or not? The key idea of mastery learning is to divide the course into modules for which the students must demonstrate mastery. Hence, there are typically the following alternatives for grading:

- 100% based on modules, determined by the number of modules passed.
- 100% based on modules, aggregating points scored on modules. This approach is used for a Math course at UiA, with four term tests that the students can retake several times (Zakariya, Berg et al. 2025).
- X% modules + Y% something else. The most common "something else" is a final exam, and additional options might be, e.g., term tests at fixed dates or bonus points for participation.
- 100% exam. In this case, the modules have the function of mandatory exercises, but the key difference between a self-paced mastery learning course and an "ordinary" course is that the students can decide their own pace of delivery. This was the option chosen by (Purao, Sein et al. 2017), with an oral exam at the end.

In IT1001 we chose the first alternative, grading by the number of modules passed. This is clean and easily explained to the students and saves resources by avoiding additional assessment components – since each component you add will also demand some work. However, concerning other courses we have read about, it does not seem to be the most common approach. Keller's (1968) original PY112 course was graded by 50% module tests, 20% labs, and 30% final exam (first 25% but increased to 30%). Many recent CS1 courses have also included final exams to determine at least parts of the grade. For instance, Campbell et al. (2019) had a final exam counting for nearly 50% of the grade, in addition to results from their 10 module quizzes. Offutt et al. (2017) added points from modules rather than just having them as pass/fail. They used a lower pass threshold for their 10 modules (70%, while 90% is more common in mastery learning), and then used the points above 70 to distinguish between different final grades – also adding points for participation in seminars (10%) and a final exam (20%). Ott et al. (2021) had a final exam counting for 50% of the grade for the first two years of running their course but then dropped the exam. Subsequent offerings had mastery tests (60%) plus two term tests at fixed dates (40%). Most like our scheme is perhaps that of (Toti, Chen et al. 2023) in a CS1 course in Houston, Texas. They did not have any exam, and grades were determined almost exclusively based on modules passed, though students might earn up to one grade step in bonuses from preparation and participation, while we had no such bonuses.

A typical motivation for having a final exam might be that either the teachers or their leaders do not trust that the mastery tests give a correct account of the students' capabilities. However, our advice would be to **avoid the final exam if you can.** Giving a final exam is a lot of extra work for the teaching staff, without giving extra value to the students – indeed, the absence of a final exam is a key to reducing student stress and giving them a bigger sense of control. Having modules just as pass/fail also reduces stress, since then students do not have to carry with them throughout the semester the penalty of a "poor"

performance on an early test (e.g., passing a test just above the threshold, while others in the class scored 100%), and they only need to retake module tests that they failed, not those that they already passed.

On the other hand, it would not be realistic for slower learners to make it through all our 9 modules in a 14-week semester. Our conclusion was thus that it must be possible to take *fewer modules for a lower grade,* which also makes sense from the students' perspective. We still wanted to achieve the old ideal of high mastery for all but then limited to the modules that each student took. We decided **5 / 9** modules should be the minimum to pass the course. For end-of-course exams in other CS1 variants at the NTNU, the "percentage point rule" suggests a pass threshold at 41%. While 5/9 is 56%, the students in IT1001 need not pass these 5 tests at 100%. I, H, G can be passed by 90%, and F and E by 80%, which brings the 5/9 down to 48%. Still more than 41%, but the tests might also be made easier by higher transparency and multiple attempts. 4/9 would have been too little, since the pass margin at the individual tests would then have brought the minimum total down to just 38%.

Trade-off #6: Module granularity. From a pedagogical viewpoint, you want many small modules rather than a few big ones. Smaller modules avoid overwhelming the students and give them more fine-granular control of their progress. Keller's original PSI course in general psychology (PY112) at the Arizona State University had 30 modules, with a recommended pace of two per week. Of the 30, there were 20 theory modules with mastery tests, and 10 lab experiments. However, having so many modules might be a challenge to staff administering and assessing all those tests (Keller used TA's for that, but there may not be money for so many TA's), and frequent tests also take time for students, who are taking other courses in parallel. Hence, logistics may lead towards fewer modules. Looking at published experiences with mastery learning more recently, a wide range of choices can be seen, all the way from 4 rather big modules to 47 tiny ones, and a typical range for CS1 courses is 7-15, with e.g. (Campbell, Petersen et al. 2019, Ott, McCane et al. 2021) at the lower end, and (Purao, Sein et al. 2017, Toti, Chen et al. 2023) towards the higher end. Your decision will thus depend on several factors, like course weight and duration, staff capacity, etc.

For IT1001 we landed on **9 modules** – **if** you consider the test and project on a certain level to be the same module, as they are topically related. To communicate clearly to students the grade implications of the modules, they were simply named in descending alphabetical order: I, …, A as shown in Figure 7. However, as explained earlier, students are not required to sync their progress on tests and project increments, except they must reach the same letter by the end of the semester to take a certain grade. Hence, it might be more correct to say that we have 18 modules, whereof 9 test modules and 9 project increments. For the project work, there is no invigilation, thus no capacity problems, but the tests need to be invigilated – and our choice was motivated by the capacity challenge posed by this. We decided we could only organize invigilated module tests once a week, both in terms of staff time and what we could expect in terms of rooms reserved for scheduled course activities. We wanted students to have a chance to get through all modules even if they might be absent from a couple of test occasions, and maybe also failing once or twice so that the same test must be retaken next week. With a 14-week semester, 9 tests would mean that a student might be sick or away from 2 test offerings and fail at 3 yet still get through by passing all the other times.



Figure 7. The course modules, blue ones corresponding to passing grades.

Figure 8 shows three examples of fictitious students making it to the grade of C. “Cindy” takes 7 tests first, then delivering 7 project increments – the opposite would also be possible. “Carl” does it in a more intermingled way and reaches C already in week 9 of 14. He could have proceeded towards higher grades but was satisfied with C and chose to prioritize other courses that he found more motivating, or where he was struggling more, thus worried about failing an exam. “Cole” makes quicker progress with his project than with tests, and works the project all the way up to A. However – perhaps due to repeatedly failing the C-test – he only reaches C in the test series, which then becomes the grade. The same would apply in the opposite case of a student doing better at tests than project – the lowest grade applies in case they are not equal. This is very clearly communicated to the students at the start of the semester, so that it shall not come as a surprise.

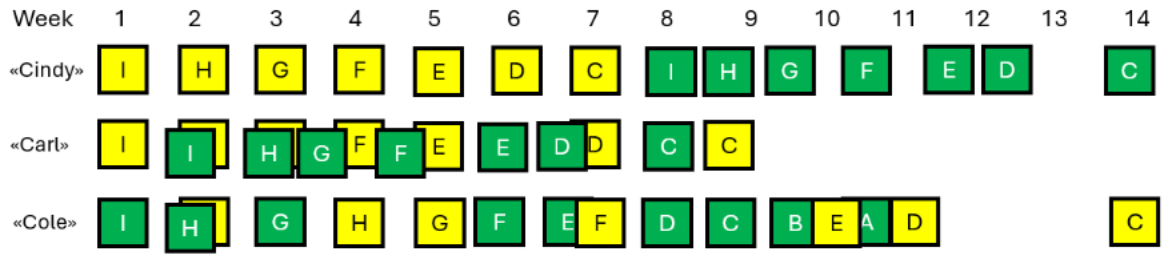


Figure 8. Three example trajectories reaching the grade of C in our course IT1001.

The choice of module granularity will obviously also have an impact on the contents for each module. A textbook or lecture plan might give good ideas for dividing course content into modules, since these plans tend to have a deliberate progression. For IT1001, the curriculum is quite similar to that of a pre-existing lecture course TDT4110. Still, we had to do some reshuffling of the content. With just 9 modules, they had to be larger than the weekly content in lecture plans. Moreover, with our course design it is possible for a student to just take the first 5 modules for an E and be satisfied with that, thus ignoring content in the remaining modules D-A. If you use a similar approach, it is important that the modules required to pass contain everything that the students **must** learn to deserve the course credits, and then the subsequent modules can contain what they **ought to** learn (for IT1001: D, C), and what they **may** learn, but which is only really expected of the best students (B, A).

“Clean” in terms of module content would mean to adhere to the principle that each module should cover one key concept. However, as has also been found in previous attempts at mastery learning for introductory programming, it is not necessarily easy to get such a perfect fit between modules and concepts (Purao, Sein et al. 2017). Some concepts are too complicated to deal with in just a single module, such as loops, functions and parameters, data structures, and for the students to be able to write meaningful code, it is often necessary to cover the basics of several different concepts early. Hence, we chose an *upward zigzagging* approach concerning the contents of modules, where new modules partly introduced new concepts, but also revisited previously seen concepts in a more advanced manner. Table 1.1 shows this in more detail. Starting from the bottom, the I-module introduces basics – however, there are lots of different concepts involved in this, such as numbers, strings, variables and assignment, arithmetic operators and their precedence, expression syntax, and simple i/o with input and print. The H-module could be seen as mainly about types, but also covers some things not usually seen at such an early stage, namely numpy arrays and plots with matplotlib. This was done to enable students to write programs that *show something* more than just text i/o, as their written code would be related to STEM subjects. Putting this in a late module, students would be going for a bigger part of the semester just writing rather boring programs, and those going for a low grade might not reach it at all. Moreover,

plotting is not particularly difficult, it is just to follow a recipe for the type of plot you want to achieve. The G-module mainly deals with branching – which is then revisited in the C-module with a more complicated branching problem. Loops are dealt with first in the F-module (single for and while loops), then in the E-module (double loops), and then revisited again in the two topmost modules looking at some more complicated looping problems (e.g. in A, requiring triple loops). The same can be seen for data structures, with lists and arrays first introduced in a very simple way in H (illustrating type differences, and to have some data to plot), then in F looking at iteration in one-dimensional lists and arrays, turning to 2d in E, looking more thoroughly into slicing and mutation in D, and finally revisiting mutation in a more advanced way for 2d lists and arrays in B. Similarly, functions are all over the place, introduced first in H, then in more advanced ways in further modules, ending with functions taking other functions as parameters in B and then recursion in A. Even in the modules for which functions are not mentioned, they are still there, as the code students are requested to understand in tests typically contains functions, and the code they write for their projects is typically required to be divided into functions.

Table 1: Module contents of the course

	Key conc.	More detailed contents
A	recursion	recursion, numerics, efficiency, dictionaries #2, loops #4
B	events	functions #3, callbacks / matplotlib widgets, 2d data #2, loops #3, accuracy #2
C	hash data	sets, dictionaries, random #2, branching #2, lists #4 (various methods)
D	exceptions	error types, exception handling, more advanced file i/o, strings #2, mutation #2
E	loops + 2d	loops #2 (nested), lists/arrays #3 (2d), file i/o #1, accuracy #1
F	loops	loops #1, lists & arrays #2, tuples, functions #2 (scope), mutation #1
G	branching	branching #1, bool, relation operators, logical expressions, random #1,
H	types	elementary types & conversion, functions #1, lists & arrays #1, plotting #1
I	basics	input, print, numbers, strings, variables, assignment, operators, sequence

On the surface, our upward zigzagging approach might look rather messy. However, repeated exposure to concepts at increasing levels of advancement can help students' learning, and some concepts are too demanding to fit into just one module. For instance, dealing with everything about functions in one module (e.g., definition, calling, arguments and parameters, return values, scope, side-effects, good decomposition, functions as parameters, recursion) would be way too demanding. Hence, we chose this pragmatic approach to chunking and laddering. By "zigzagging" we mean that each module introduces some new concepts (e.g., for C: sets, dictionaries) but at the same time also revisits some previous ones (more advanced than before about lists, branching, and random functions).

Analogous examples could be made with totally different subjects, say humanities or social sciences. Imagine "Introduction to European Philosophy" where the readings cover 10 important philosophers, or "Organizational Theory" with 8 key theories. Assuming half the modules are needed for a pass, one approach might be to say you only need to know about the first 5 philosophers or the first 4 theories – but often, that would feel like an unsatisfactory solution. Another approach might be to make a double

pass, e.g. 5 modules that cover the basics for the philosophers (then needing to cover 2 of them in each module), and then 5 more ambitious modules that cover more advanced analysis of their philosophies. Possibly, the 5 more ambitious modules might not even be structured philosopher by philosopher, but according to different philosophical concepts, and then comparing and contrasting various philosophers' contributions to the understanding of these concepts. Similarly, for the organizational theories, one might want to cover the basics of all the 8 theories through the first four modules if a student should really know about all of them to pass the course, then move into more advanced issues in subsequent modules.

Whatever course you are wondering to convert to mastery learning, there may also be someone somewhere in the world who have already tried mastery learning in that discipline, and who have published something about it, including an outline of their modularization. Hence, in addition to just reading our suggestions, a good idea might be to do a literature search to see if there is any possible inspiration to be found. If it is an existing course that you are planning to convert to mastery learning, the way chapters and sequenced in the textbook, or the way it is done in your current lecture plan, will also be a good starting point.

8.5. Learning resources and activities – and mitigating procrastination

One of Keller's five principles was focus on the written word, another was not to use lectures for conveying new course material, only for motivation. These two principles were again essential together with the principle of self-pacing: students could read textbooks or xeroxed booklets at their own pace, while lectures – necessarily given at specific points in time and with a certain pace of delivery – will contradict the principle of self-pacing, instead trying to force the entire class into the lecturer's pace. Hence, the 8 motivational lectures that were still included in Keller's PY112 course were not demanding – and not even hoping for – attendance by all students. Instead, participation in a lecture was only allowed if you had already passed the relevant module tests for the underlying theory, so that the lecture would not present new concepts. The main learning resources and activities in Keller's course would thus be:

- Xeroxed booklets for self-study, to read in preparation for the module tests
- Labs – and feedback from tutors in the lab
- Feedback and explanation from tutors in the individual interviews taking place after the module tests
- Lectures, but just as a voluntary and conditional bonus, as explained above

Modern implementations of mastery learning also focus on asynchronous learning resources to enable self-study at the student's own pace, though electronic resources like wikis and videos will largely have replaced xeroxed booklets or other sources on paper. However, several implementations of mastery learning have chosen to include lectures in a more traditional sense, i.e., explaining new concepts. For instance, (Toti, Chen et al. 2023) used lectures to set a nominal pace for the course that the students were recommended to follow, though they could lag some weeks behind in their module tests. (Purao, Sein et al. 2017) went the first year without lectures but later introduced sporadic lectures if requested by enough students. (Ott, McCane et al. 2021) removed lectures from the second year of running their self-paced mastery learning course, but some years later decided to re-introduce weekly lectures, stating

in their paper that this was seen as important for giving structure, setting a nominal pace for the course, and thus mitigating procrastination.

Although several other teachers of mastery learning courses have found lectures useful, our opinion is that **lectures should be avoided** as much as possible. With self-pacing students, a lecture will necessarily be of limited relevance for parts of the student group, either because the student is fast-paced and has already gone beyond the topic to be lectured, or slow-paced and thus lacking the basis to understand the lecture. Even for those just on track, it can be debated whether the lecture is a particularly good learning method (Kozanitis and Nenciovici 2023). Instead, for IT1001 we chose the following mix of learning resources and methods:

- Screencast videos and wikis (cf. Keller's focus on the written word) for self-study, structured module by module.
- Seminars with student-active learning, one double-hour per week, with mandatory attendance to at least 11 / 14. Students were grouped according to their current level, and their preferred activity at that point. Hence, a student wishing to prepare for their upcoming F-test would be sitting together with other F-test students, and a student wishing to work on the H-project, would be sitting together with others with the same intentions. Hence, unlike a lecture which would deal with topics beyond the progress of some students and behind the progress of others, every student could pursue learning activities for their own level. It is more defensible to have an activity as mandatory when it contains something useful for every student. In IT1001 the mandatory attendance also has a socializing purpose. It is the only course that all the STEM teacher students have together in the first semester, thus meant to help them get to know their classmates.
- Tests as a key learning resource, through automated feedback after delivery. In addition to the invigilated tests held once a week, the course had **practice tests** that the students could attempt at anywhere and anytime – though only having access to the tests they had qualified for attempting, not tests further up the ladder. Practice tests pulled random items from the exact same question banks as used for the invigilated tests, thus yielding high assessment transparency in the course. Indeed, working with practice tests became a dominant activity in the mandatory seminars, where students would discuss the solutions to the various tasks, then deliver and see their score and discuss in more detail the tasks where they had mistakes.
- Feedback from TAs and the teacher during seminars (on test tasks, or project code), and feedback from the teacher on delivered project increments.

Student **procrastination** has always been a challenge for self-paced courses, and many have seen the need for various mitigation efforts. The self-paced course of (Campbell, Petersen et al. 2019) was eventually abandoned, going back to a more teacher-paced format. Several other courses have found it necessary to limit student self-pacing. Typical measures include:

- Intermediate deadlines. For instance, (Purao, Sein et al. 2017) gradually introduced a requirement that students must have completed at least M of N modules by a certain date mid-semester. (Toti, Chen et al. 2023) allowed students to be no more than 3 weeks behind the nominal schedule of the course, thus having a last possible date for each module.
- Fixed-date assessments. (Ott, McCane et al. 2021) introduced two fixed-date term tests in addition to the self-paced modules. Hence, although students were in theory free to self-pace

for the modules, they would do poorly on the term tests if they had not yet covered the relevant curriculum for those tests.

- Bonuses, e.g., for participation and timely submission of assignments (Offutt, Ammann et al. 2017, Toti, Chen et al. 2023). Keller's (1968) scheme with motivational lectures that students were only allowed to attend if they had reached a certain level before the lecture could also be seen as a similar tool, although this did not give any points towards the grade.

In IT1001, it is only possible to take one test letter per week, and to get a passing grade, a student must take at least five modules. This means that there is a de facto deadline that the I-test must be taken in week 10 or earlier (of the 14-week semester), H-test in week 11 or earlier, and so on. In theory, this enables the student to procrastinate and do nothing for the 9 first of the 14 weeks, so it is much less constraining than the various mid-term deadlines found in many other courses. Despite lacking such deadlines, IT1001 has only had modest problems with student procrastination, instead observing that many students prefer to work more intensely early in the semester to have more leeway later (Sindre and Hansen 2025). The main reasons for this are believed to be the following:

- The absence of an end-of-course exam reduces the motivation to procrastinate. Without the exam, it becomes possible to finish the course early, and many students will prefer to start early to finish early.
- The increased transparency achieved by practice tests which draw questions from the same question banks as the invigilated tests, contributes towards reducing test anxiety for students. By doing practice tests they can see clearly which tasks they lose points on, whose concepts they therefore need to learn better, and with repeated practice they can see gradual improvement towards a level where they are likely to be able to pass the invigilated test.
- The seminars with mandatory attendance, although they do not have any deadlines for tests or project delivery, prevent students from detaching themselves from the course. If you anyway must spend two hours per week in a seminar, you might as well do something useful, and the group setting implies gentle social nudging. If the peers at your table are planning to take a test next Friday, a student otherwise likely to procrastinate might feel tempted to also take the test, to be grouped with the same peers also in the next seminar.

Figure 9. *Example table grouping, 6th week of the 2024 run of the course* shows as an example a snapshot of the table grouping for the seminar 6 weeks into the 2024 run of the course. The fastest students in the class had streaked the tests I, …, E in the five first weeks, thus being ready to practice for the D-test (red table, lower right). Other test practice tables were for students who had passed fewer tests. The dark grey trapeze symbols along the walls indicate wall screens where one student in each group could hook up their laptops so that they could easily work at one practice test together and discuss the problems along the way. Groups were dynamically changing from week to week, depending, e.g., on whether tests were passed or not, and whether a student wanted to work on their test or project during the seminar. Hence, students got to know a wider range of classmates than what would have been the case for teamwork with fixed groups, or with entirely voluntary seating which would more likely be based on whom they knew from before.

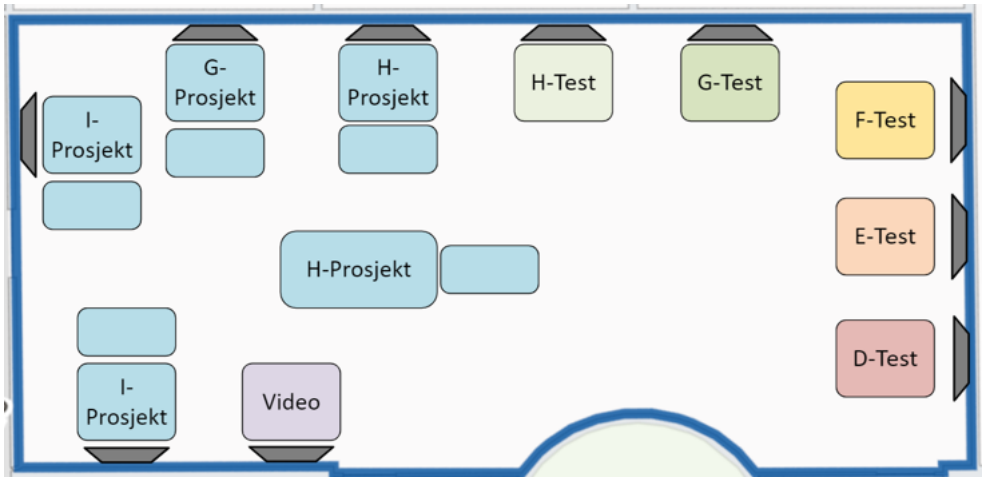


Figure 9. Example table grouping, 6th week of the 2024 run of the course

The video table was specifically set up for students who, at some point during the seminar, felt a need to watch one of the teacher's videos to understand a concept better. It was observed that if someone put on headphones and started watching a video at another groupwork table (either for test or project) this would tend to inhibit interaction in the group, since the student watching a video would easily become an obstacle between other students who might otherwise have interacted. Hence, moving video watchers to a separate table was a better solution – and gradually it also caused less video watching in the seminars. After all, watching videos was something that might as well be done alone at home.

8.6. Detailed design considerations

Module pass criteria. In most mastery learning approaches – including Keller's original PSI – a high degree of mastery is required to pass the module tests, typically 90%. At the same time, this was sometimes found too rigorous, as students might be stuck rereading for the same test because of one lacking point. An typical workaround, also suggested by Keller himself (1968) was that students who were just short of the pass threshold might amend this in the subsequent interview with the tutor correcting the test. If the tutor could ascertain that the student had shown understanding of the concepts during the interview, the tutor could mark a test as passed despite being marginally below the threshold. This was found to give smoother test progression and reduce dropouts, as well as enhancing the role of the tutors. However, there are also caveats. In the paper "Dangers of PSI" (Caldwell 1985), there is criticism that tutors tended to become too friendly with (some) students, allowing soft and potentially unfair passes. In our course, we do not have capacity for such oral interviews after tests – but have found another but fully automated way to achieve some smoothness in test passing. Each student has three "80% passes" that they might use on tests from F onwards. So, if a student scores, e.g., 82% on the F-test, they may use an 80% pass to proceed even if their score was below 90%, but then they only have two left for further tests. Students with high ambitions will typically prefer to save these passes for as long as possible, at least having one left for the most difficult A-test.[1]

For delivered work – like our project increments – it is challenging to come up with clear-cut, objective pass/fail assessments, and you must somehow adapt this to the number of students, the number and size of deliverables, and the staff capacity including assistants. For the first run of the course, we had a good number of TA's due to extra pilot funding, but in subsequent years it has been mainly the teacher assessing and approving project increments. There will typically be 200-300 project increments

delivered during a semester, so obviously, the teacher needs to be able to make an approve / revise decision rather quickly. Our tactics to enable quick and simple reviews of project deliverables were as follows:

- Clear and concise rubrics for each project increment, with a brief bullet-point list of requirements that the code must satisfy. Since each student chooses their individual topic and genre for their pedagogical Python program, no specific features could be required for the program. Instead, for each increment, the program should run without error and demonstrate purposeful usage of certain listed programming mechanisms – the same that were in focus for the corresponding module tests.
- Employing student self-assessment in terms of a checklist that they fill in before delivering a project increment, declaring for each requirement that it has been used.
- For the current run of the course, GenAI has also been employed for this project increment assessment. The teacher has written prompts asking the AI to go through the code and check the requirements one by one, plus provide some improvement suggestions appropriate for the level. Students will copy-paste these prompts into the university chatbot and then hand together with their code the AI assessment of the compliance with the requirements, plus their own reflection upon the AI response, arguing whether they agreed the AI's assessment, and whether they learnt something from it. This has helped the teacher to speed up the checks since students will have discovered and fixed more of the issues before delivering.

If you plan to have delivered work as part of your modules, and the nature of the work is such that it must rely on some degree of manual assessment for the pass/revise/redo decisions, you need to find a way to do this within the bounds of available staff resources. Having clear requirements and employing student self-feedback – possibly enhanced by GenAI – could be a way to achieve it, but your exact approach must depend on the type of work to be assessed.

8.7. Make your own course - Process suggestions

Your exact process to design a mastery learning course will depend on a lot of factors, such as the nature of the subject, size of the class, the placement of the course in the degree program(s), the capabilities of the students, and the staff resources you have available. Also, the starting points may be different depending on whether you are redesigning an existing course to use mastery learning or making a course from scratch. Here we will only make some rather general advice concerning the process.

Advice #1: Involve students. The students who are actually going to take the course may not be available – for IT1001 which is a course for first-year students, it cannot be known at the time of course design who will enroll in the study program in question and thus be supposed to take the course. However, it is possible to involve older students in the same study program, who know the context of the students, and – especially in case of redesign of an existing course – will have relevant experiences of the strengths and weaknesses of the old version of that course. In IT1001 we involved older students in the STEM teacher degree program as assistants in the design phase, and they provided a lot of useful input (Sindre, Hansen et al. 2023).

Advice #2: Involve colleagues. The design of a mastery learning course may require considerable effort up front. If the main form of assessment is module tests, there may be a need to make a huge number of test tasks to facilitate student retakes (and even more if also allowing practice tests). Although previous exam and assignment tasks may be used to some extent, the tasks may need to be adapted

somewhat to the mastery learning setting, and you may need many more tasks, or variant of tasks, than you have available. Similarly, there may be a need to make new learning resources, although many courses already have wikis, screencast videos and other resources suitable for self-study. Involving several colleagues could distribute the effort across several people or at least ensure that there are people to discuss with when uncertain about various design choices.

Advice #3: Contact support staff early. Especially if mastery learning is previously untried in your university, it is a good idea to contact the exam office / study administration to find out exactly what is within university regulations concerning course designs and forms of assessment. It can also be smart to contact tech / admin support personnel to investigate what level of support can be expected, especially if there is something about the mastery learning course that might be more complicated to support than an ordinary lecture and exam course.

Advice #4: A rough recipe for a course design workshop. One of the key issues of a mastery learning course is the breakdown of the content into suitable modules. If going with an all-take-all approach to the modules, a lecture plan or textbook chapter structure may be readily transformed into a list of modules if you are lucky – though some adaptations may be needed depending on your module granularity. With the all-take-all design, a bigger problem is likely to come up with a way to handle the challenge that the weaker half of students will have trouble completing the course within the semester.

If instead allowing students to take fewer modules for poorer grades, like we do in IT1001, some more adaptations may be needed, since you must decide on the detailed breakdown of learning outcomes into those which are absolutely necessary to pass the course, and other outcomes that are not considered necessary for the lowest passing grade. Figure 10 illustrates a possible way to explore this in a workshop. Write detailed learning objectives on post it notes and try to sort them into MUST (even the lowest passing grade must achieve these objectives), OUGHT (mid-range students should achieve these objectives), and MAY (only expected of the better students). After collecting and grouping the learning objectives, try to group them into reasonably sized modules, starting at the bottom and working upwards to achieve good cohesion and progression. Figure 11 sketches a way to go into further details.

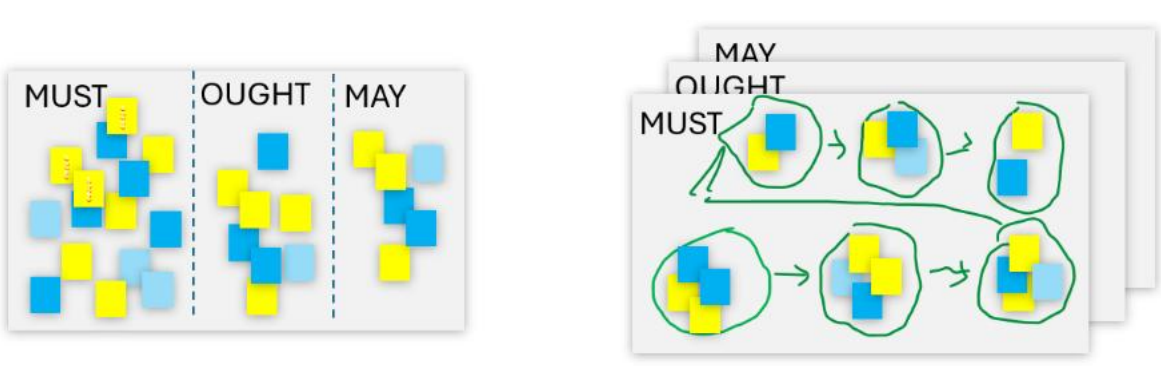


Figure 10. Sketch of workshop materials. (Left) Have sectors MUST, OUGHT, MAY on a wall or large sheet of cardboard or paper, write detailed learning objectives on post-it notes. (Right) Subsequent sequencing of the outcomes.

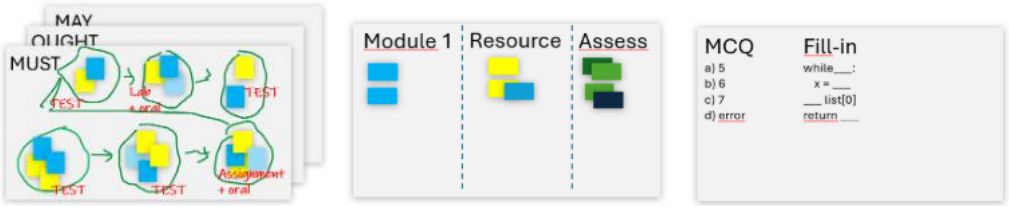


Figure 11. Further workshop steps, going into assessment types and tasks for each module (left), then learning resources and assessment tasks for specific objectives in the modules.

Of course, there may be many other ways and work formats for such workshops, depending on your needs and preferences.

When specific materials have been prototyped, such as learning resources and module tests, it could also be a good idea to have some pilot users try them out – which is one of the possible tasks that student assistants could contribute to.

8.7.1. Into the future – the project and AI

Apart from the ambition to propagate our mastery learning approach onwards to other courses and bigger classes, a current challenge is how our teaching and assessment approach relate to the disruption posed by AI in higher education. The course was planned in 2022 when generative AI had not yet caused a sense of crisis in higher education, and AI was not such a big concern in 2023 either when the first run took place. It was first towards the end of 2023 / beginning of 2024 that the discussion really took off that AI was impacting students' learning (possibly in a negative way, if they outsourced their assignments to AI rather than doing them through own efforts), and that some unsupervised forms of assessment like take-home exams, term papers, etc. might be too easy to cheat on with AI.

Hence, the course IT1001 with its laddered form of assessment was not developed with AI in mind – it was meant to address quite different problems, like students' sense of mastery and control of their progress, continuous effort during the semester, and in particular for the mandatory seminars: the social learning environment of the class. However, even if AI was not a concern taken into account in the design of the course, it has turned out to be rather robust against AI, for the following reasons:

- **The module tests**, taken under supervision using a lock-down browser, actively prevent the students from using AI, thus encouraging them to solve those problems by their own reasoning. While preparing for a test, using formative practice tests, they can of course use AI – and sometimes do, for instance to have AI explain in more detail the concepts underlying a certain test task, why their answer to a certain question was incorrect, and why the solution is correct. However, knowing that the supervised test will be without aids, they will normally try to see how much they can score by their own effort, and especially run their last practice tests without any aids. Just asking AI to give them the answers to a practice test will not be productive use of time.
- **The project** allows the use of AI, and students use it to some extent. Bad usage of AI would be if a student just copies the assignment text or project requirements into the prompt field and lets AI solve everything. However, the way the project in IT1001 is designed, this is not straightforward.
    - First, the **open** nature of the assignment means that there is no assignment text. Each student has to determine for themselves what kind of program they want to develop. For a student

planning to major as a biology teacher, it could be a tutorial about genes and inheritance, a simulation program about population ecology for predators and prey, a quiz program about microorganisms, or a program for students to collect and analyze data from a field excursion, or a game where you can play the role as a country's chief responsible in fighting an epidemic. For students to major in chemistry, physics, math it might be other topics, the only requirement being that it has to relate to competency goals in that topic's high school curriculum, as given by the course web pages established by the Ministry of Education.

- Second, this open nature of the project also means that each student can choose a topic and program genre that they are interested in, and that they can program something that they are able to program – reducing the temptation to outsource the work to AI.
- Third, the incremental delivery of the project means that each increment is rather small, so will more easily feel within reach for a student to do mainly with own efforts rather than outsourcing the job to AI.

Indeed, the levelled and highly transparent assessment design of the course means that the students at any point know where they stand in the course in terms of grade and what to do next, and retake opportunities for tests also contributes to reduced anxiety – which also likely reduces the motivation for cheating.

On the other hand, AI can be a very useful learning resource in the course. In a self-paced mastery learning course, where students are at many different stages, it is meaningless for the teacher to provide any plenary instruction, and the teacher cannot be everywhere helping everyone at once. Moreover, the openness of the project assignment in IT1001 means that there is no model solution that the teacher can refer to, and students may have problems not only with the programming, but also with the disciplinary topic (e.g., understanding the formula for the movement of a ballistic object with air drag) and how to translate that into code. Here, AI may have a valuable function as an extra tutor, making it possible to run such a flexible course without the high number of TAs that were traditionally needed in PSI the way Keller designed it.

Somewhat surprisingly, students' usage of AI seems to have dropped from 2024 to 2025, at least based on the responses to the questionnaire survey. Figure 12 indicates that the percentage of students who used AI to some degree in their projects dropped from around 80% to 60%, while on the other hand especially usage of the teacher's videos increased. Part of the reason may be that the teacher had made some new screencast videos specifically giving advice about the project – and this may have reduced the need for AI (and – maybe somewhat less positive – also for seeking help from peers related to the project, as this seems to have dropped from '24 to '25).

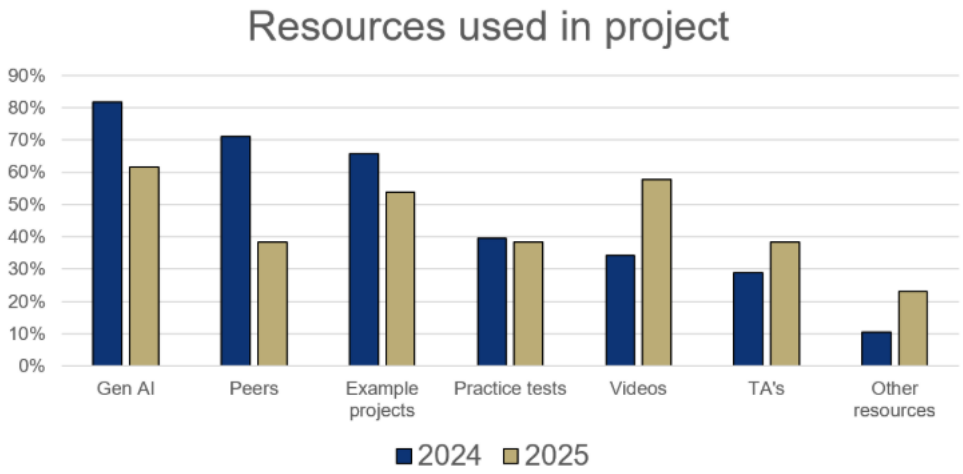


Figure 12. Learning resources used in the project (possible to select several)

Figure 13 sums up the students' responses to questions about the usefulness of various learning resources – here for the whole course, not just for the project. The formative practice tests are seen as the clearly most useful learning resource in both years – but this is unsurprising, given that they give quick and direct feedback on where the student stands in preparation for the next test. GenAI was in second place in 2024 but dropped to third behind the teacher's videos in 2025, thus indicating a similar decline as shown in Fig. 5. There are several possible explanations for this. One could be that the teacher had made some improved learning resources for the project, thus reducing the need for AI. Another could be that the students progressed slightly faster on tests in 2025 while instead lagging somewhat more behind on the project. Since the questionnaire survey was done three weeks before the end of the semester, relatively more students had only made modest progress on their projects at the time of the survey in 2025. Hence, it could be that they reached a more challenging stage of their project in the weeks after the survey, and then made more use of AI, which would not have been captured by the survey. A third possible explanation might however be that students in 2025 had become more skeptical towards AI than they were in 2024. This was raised as a possible explanation by the student reference group, and there are factors that may have contributed to this. In the spring of 2025 there had been several scandals in Norway concerning bad usage of AI in the public sector, some of them featuring prominently in the news, such as a case of bad AI usage in Tromsø municipality[2] and also, NTNU had developed clearer guidelines about AI usage than what had been present in 2024, including a 40 page slide presentation that study program leaders would give to the first year students during the start-up week, discussing good and bad usages of AI and warning against pitfalls.

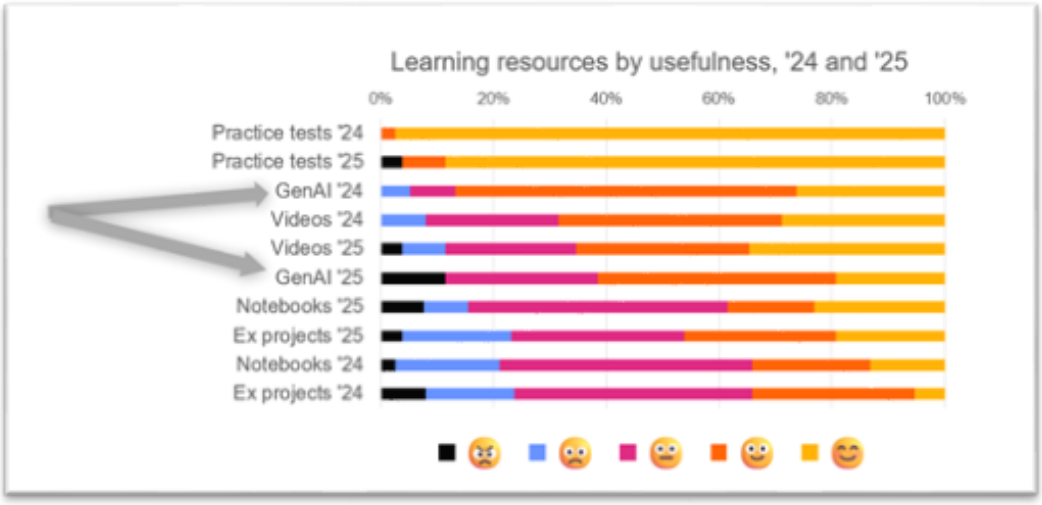


Figure 13. Student opinion on usefulness of learning resources in the course, '24 and '25

Table 1 summarizes student responses to a free-text question "To the extent that you used GenAI in the course, what did you use it for?" As can be seen, error finding was the most common usage, followed by idea generation. The third most common usage was not in the project but rather related to the test, namely explanation of test tasks. It is quite encouraging that the usage which would typically be considered "bad", e.g. outsourcing the whole code writing process to GenAI, does not seem to have been prevalent. Rather few students report having used GenAI for code generation, and even for these it may just as well have been only parts of the code, not the whole thing. My impression after looking through the students' code was also that most of it was hand-written, as it was too messy to look like typical GenAI code.

Table 1: What students used GenAI for in the course

	2023	**2024**	**2025**	**Total**
Respondents (N=…)	44	38	29	111
AI users (%)	22 (50%)	34 (89%)	21 (81%)	78 (70%)
Error finding	8	18	9	35
Idea generation	5	18	5	28
Test task explanation	3	7	6	16
Code generation	2	3	5	10
Code improvement	3	3	2	8
Code comprehension	2	2	1	5
Prog concept explanation	1	1	3	5
Requirement check / explan.	1	4	0	5
Unspecified	3	1	0	4
Data / content generation	0	1	1	2
Discipline knowledge	0	1	0	1

For students who are to become teachers in high school, it is important to have competence about AI, not just because it can be a useful tool to themselves both as students and later as professionals, but also because they as in-practice STEM teachers in school must relate to their students' AI usage. For the upcoming run of the IT1001 course we will thus make modifications to the project so that the students are required to use some AI in the project and deliver a reflection note about their experiences with AI in the project – while at the same time also demanding that they write a sufficient part of the code themselves. This means that the learning resources of the course must also be updated, to provide advice on how to use AI, and how *not* to use it, and how to critically evaluate the output that it delivers. With these changes, we hope that the course will appear even more interesting and satisfying to the students than it has been so far.

Copyright

Self-Paced Mastery Learning © 2026 by Guttorm Sindre is licensed under CC BY-NC-ND 4.0



References

Bloom, B. S. (1968). "Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolinas and Virginia, Topical Papers and Reprints, Number 1." Evaluation comment **1**(2): n2.

Bloom, B. S. (1971). Mastery learning. Mastery learning: Theory and practice. J. H. Block. New York, Holt, Rinehart & Winston**:** 47–63.

Caldwell, E. C. (1985). "Dangers of PSI." Teaching of Psychology **12**(1): 9–12.

Campbell, J., A. Petersen and J. Smith (2019). Self-paced mastery learning CS1. Proceedings of the 50th ACM Technical Symposium on Computer Science Education**:** 955–961.

Chiu, M.-C., E. Moss and T. Richards (2024). Effect of deadlines on student submission timelines and success in a fully-online self-paced course. Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1.

Eyre, H. L. (2007). "Keller's Personalized System of Instruction: Was it a Fleeting Fancy or is there a Revival on the Horizon?" The Behavior Analyst Today **8**(3): 317.

Foss, K. A., S. K. Foss, S. Paynton and L. Hahn (2014). "Increasing College Retention with a Personalized System of Instruction: A Case Study." Journal of Case Studies in Education **5**.

Garner, J., P. Denny and A. Luxton-Reilly (2019). Mastery learning in computer science education. Proceedings of the Twenty-First Australasian Computing Education Conference**:** 37–46.

Hansen, G., G. Sindre, G. S. Korpås and J. P. Vassbakk (2024). "La kvar student velje sitt eige tempo: Meistringslæring i intro programmering." Læring om læring **11**(1): 1–13.

Hattie, J. (2015). "The applicability of visible learning to higher education." Scholarship of teaching and learning in psychology **1**(1): 79.

Ihantola, P. and V. Karavirta (2011). "Two-dimensional parson's puzzles: The concept, tools, and first observations." Journal of Information Technology Education **10**: 119–132.

Keller, F. S. (1968). "Good-bye, teacher…." Journal of applied behavior analysis **1**(1): 79–89.

Keller, F. S. and J. G. Sherman (1974). PSI, the Keller Plan Handbook: Essays on a personalized system of instruction, WA Benjamin Advanced Book Program.

Kozanitis, A. and L. Nenciovici (2023). "Effect of active learning versus traditional lecturing on the learning achievement of college students in humanities and social sciences: A meta-analysis." Higher Education **86**(6): 1377–1394.

Kulik, C.-L. C., J. A. Kulik and R. L. Bangert-Drowns (1990). "Effectiveness of mastery learning programs: A meta-analysis." Review of educational research **60**(2): 265–299.

Kulik, J. A., C.-L. C. Kulik and P. A. Cohen (1979). "A meta-analysis of outcome studies of Keller's personalized system of instruction." American Psychologist **34**(4): 307.

Mannion, A., R. Coyne, C. Ferrari, M. Neşeli, C. McGee, S. Mollaoglu and G. Leader (2023). "Personalized System of Instruction in Higher Education: A Systematic Review." Journal of Behavioral Education: 1–41.

Offutt, J., P. Ammann, K. Dobolyi, C. Kauffmann, J. Lester, U. Praphamontripong, H. Rangwala, S. Setia, P. Wang and L. White (2017). A novel self-paced model for teaching programming. Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale**:** 177–180.

Ott, C., B. McCane and N. Meek (2021). Mastery learning in cs1-an invitation to procrastinate?: Reflecting on six years of mastery learning. Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1**:** 18–24.

Perez, C. and D. Verdin (2022). Mastery learning in undergraduate engineering courses: a systematic review. 2022 ASEE Annual Conference & Exposition.

Purao, S., M. Sein, H. Nilsen and E. Å. Larsen (2017). "Setting the Pace: Experiments With Keller's PSI." IEEE Transactions on Education **60**(2): 97–104.

Sindre, G. (2025). Mastery Learning in CS1 With High Transparency Tests: Challenges for Fairness Among Task Variants. IEEE Global Engineering Education Conference (EDUCON). London, IEEE.

Sindre, G. and G. Hansen (2024). Mastery Learning in Introductory Programming: Running a Project Alongside a Test Ladder. 20th International CDIO Conference. Tunis, Tunisia.

Sindre, G. and G. Hansen (2025). A High-Transparency Approach to Mastery Learning in CS1: Mitigating Procrastination Despite Liberal Self-Pacing. 30th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2025). Nijmegen, The Netherlands, ACM. **V. 1:** 7.

Sindre, G. and G. Hansen (2025). "Meistringslæring: Korleis kan du som faglærar utforme og gjennomføre eit emne med studentstyrt framdrift?" Læring om læring **13**(1).

Sindre, G. and G. Hansen (2026). "Frå «robust mot KI» til «robust og konstruktiv med KI»." Læring om læring **14**.

Sindre, G., G. Hansen, G. S. Korpås, A. Kirknes, J. X. Skøien and J. L. Magnussen (2023). "Meistringslæring i intro programmering: plan for eit nytt emne." Læring om læring **10**(1).

Szabo, C., M. C. Parker, M. Friend, J. Jeuring, T. Kohn, L. Malmi and J. Sheard (2025). Models of Mastery Learning for Computing Education. Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1**:** 1092–1098.

Toti, G., G. Chen and S. Gonzalez (2023). Teaching CS1 with a Mastery Learning Framework: Impact on Students' Learning and Engagement. Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1**:** 540–546.

Zakariya, Y. F., M. T. Berg, T. Gjesteland, R. E. Umahaba and M. O. Abanikannda (2025). "From single to multiple assessments in a foundational mathematics course for engineering students: what do we gain?" Frontiers in Education **10**.

9. Gamification in Higher Education From Research to Educational Practice

Ole Goethe

Nord University

9.1. Context and Motivation

Anyone who has taught computing knows the moment: a student who was enthusiastic in week two has gone quiet by week six. It usually isn't a lack of ability. An early misunderstanding has quietly made everything after it harder. Software development asks students to handle abstract concepts, research on their own, build knowledge across topics, and stay committed through task after task, and each gap compounds the next. Sustaining engagement through that process is one of the central challenges in higher education, and it is especially visible in computing.

I came to this question from two directions: years of building games, where keeping players engaged is a design discipline, and years of teaching, where the same problem looks very different. That combination makes gamification attractive to me. It promises to make progress visible and participation more compelling, exactly what a struggling student needs. It has also made me skeptical of how the term is often used, as a layer of points and badges added to a course rather than as a set of intentional design decisions that shape how students' progress, make choices, and collaborate.

9.2. Beyond Points, Badges and Leaderboards

Gamification is frequently reduced to a short list of surface features: points, badges and leaderboards. The assumption is that adding them is enough to improve learning. The research is much less tidy. Critical reviews report mixed results and repeatedly stress that design and context matter more than the mechanics themselves. Gamification is not a treatment that works automatically (Dichev & Dicheva, 2017; Hamari, Koivisto & Sarsa, 2014; Koivisto & Hamari, 2019).

Part of the problem is how we understand engagement. It is more than attendance or time-on-task. Fredricks, Blumenfeld and Paris (2004) distinguish three forms, behavioral, emotional and cognitive. In his discussion of Game Lab, Munkvold (2017) uses a similar three-part view. He separates participation and involvement from willingness to do the work, and both from the motivation to learn beyond the minimum requirements. This distinction matters for gamification. A well-built interactive system can make students click more, collect more points or spend more time in it without deepening their learning. The real question is not whether a design produces activity, but what kind of activity and motivation it produces.

9.3. Three Cases, One Spectrum

This chapter argues that gamification is most useful when understood as a spectrum of design choices rather than a fixed set of features. Three cases illustrate that spectrum.

The first, Heimdall's Quest, from vocational ICT education, serves as a contrast case, an explicitly game-like system built around experience points, powers, roles, guilds and competition (Lorås, 2017). The second, Game Lab, sits at the opposite end. It uses no points or badges at all. Its game-like qualities come from roleplay, authentic company structures, visible milestones, real customers and meaningful

stakes (Munkvold, 2017). The third, IT1001 Information Technology, Introduction, goes further still. It was never presented to students as a gamified course. Yet its progression from I to A levels, its repeated attempts, rapid feedback and visible progress can all be analyzed through game and learning mechanics (Sindre & Kolås, 2025). Taken together, the cases suggest that what a mechanic does motivationally matters more than whether it looks like a game.

The chapter builds on work connected to Excited, the Centre for Excellence in IT Education, which brings together educational development and research in IT education at NTNU and Nord University. The three cases serve as a practice-oriented evidence base, supported by broader gamification research.

The chapter is written for educators, course designers and program leaders, and it is organized around three questions. What have we learned about gamification from these cases? What can educators take into practice? And what should they be careful about? The aim is not a definitive model but a set of practical tools: eight design principles, a reusable design framework and a checklist of warning signs. Together, these are meant to help educators decide where gamification is useful, how to align it with learning, and when a simpler intervention would serve students better.

9.4. Understanding Gamification in Education

9.4.1. Definitions and Distinctions

Gamification is commonly defined as using game design elements in non-game contexts (Deterding et al., 2011). This definition distinguishes gamification from a serious game. In a serious game, the learning activity is itself a game, and a commercial game can likewise be used for educational purposes. In gamification, by contrast, game elements are added to an activity that is not itself a game. This distinction is important, as not all Excited examples look game-like in the same way. For instance, Game Lab asks students to make a game, but it is not itself designed as a serious game. Students are learning through construction, project work, and simulated professional roles. Its gameful quality comes from the structure around that work (roles, stakes, progression, and feedback) and a shared company narrative.

A second distinction concerns gaming and play. Drawing on Ask (2016), Lorås (2017) associates gaming with acting within defined rules and goals and play with creating meaning. This distinction helps explain how two students can respond differently to the same point system. One may play it safe and focus on the visible rewards, while another experiments with the choices, identities and freedoms the system allows. In educational settings, the latter is especially valuable, because it lets learners not only pursue rewards but also make meaningful choices within the system.

9.4.2. Surface and Structural Gamification

This chapter distinguishes between surface and structural gamification. Surface gamification consists of visible game-like elements added to an activity. These include points, XP, badges, leaderboards, avatars, or cosmetic rewards. Structural gamification defines how a learner moves through the activity. It can be based on levels, progression paths, roles, narratives, stakes, meaningful choices, repeated attempts, or social units like teams and guilds. A design can use both, as Heimdall's Quest does, or it can rely on structural features with few or no visible game elements, as Game Lab and IT1001 do.

The distinction also helps explain why individual game elements do not have a uniform impact. Different configurations of game elements affect different psychological needs. Gamification is not effective simply because it is present (Sailer et al., 2017). Costa's (2023) multilevel meta-analysis finds

an overall positive association between gamification and learning in programming education. Particularly relevant here is Costa's element-level result, as reported by Sindre and Kolås (2025): levels showed a positive learning effect, whereas badges, points, avatars and leaderboards did not. Because this observation comes through a secondary source, it should be treated as an indication rather than as a direct reanalysis of Costa's data.

9.4.3. Motivation: From Rewards to Psychological Needs

Self-determination theory offers a way to analyze why a mechanism might help or harm. Autonomy, competence, and relatedness are considered basic psychological needs associated with intrinsic motivation (Ryan & Deci, 2000). In an educational setting, a point or level does not motivate simply because it is a reward. Its value depends on what it communicates and enables. A level can enhance competence. A meaningful choice supports autonomy. A team structure can support relatedness. In contrast, a leaderboard may support competence, but at the same time it may also make other students feel publicly behind.

Munkvold's (2017) analysis of Game Lab uses a four-part framework from Blumenfeld, Kempler, and Krajcik (2006), comprising value, competence, relatedness and autonomy, which makes the case easy to read through this lens. Game Lab's social factors map onto relatedness, freedom and specialization onto autonomy, and the finished product and industry relevance onto value. However, Heimdall's Quest adds a useful complication. Lorås (2017) finds that points can give learners an extra push, but richer engagement appears when students use roles, powers and possibilities playfully rather than focusing only on points. Extrinsic mechanics can therefore act as an entry point toward intrinsic engagement, but that transition is not guaranteed.

Human-Engaged Computing (HEC) offers complementary lens. Ren (2016) argues that computing should be designed to engage and strengthen people's own capacities rather than simply capture their time and attention. For gamification, this sharpens the distinction drawn in Section 1.2: a mechanic is valuable not when it keeps students in the system longer, but when it helps them become more capable and self-directed learners.

9.4.4. Affordances, Feedback and Flow

Another way to analyze mechanisms is to ask what a design can afford. Hamari, Koivisto and Sarsa (2014) describe gamification in terms of relationships between motivational affordances, psychological outcomes and behavioral outcomes. Applied to education, this model raises three questions. What action or experience is provided by the mechanic? What does a learner feel or understand through it? What behavior follows? A progress bar, for example, is not educationally valuable simply because it moves. It is valuable if it clarifies the distance to a meaningful goal, builds a sense of competence and supports autonomous, productive study decisions.

Feedback supports this process. Game structure works best when goals are realistic, progress can be visualized, and the next action is clear. In the Game Lab example, inconsistent and negative feedback was among the main sources of demotivation, and the discussion recommends constructive feedback and clearer support for week-to-week progress (Munkvold, 2017). Munkvold also draws on Steele and Fullagar's (2009) account of flow, in which a challenge should be balanced with skill, goals should be clear, feedback should be unambiguous, and learners need a degree of autonomy. IT1001 incorporates several of these factors through small modules, practice tests, repeated attempts, and quick feedback.

Such a structure can also support self-regulation by providing learners with checkpoints and information about what to work on next.

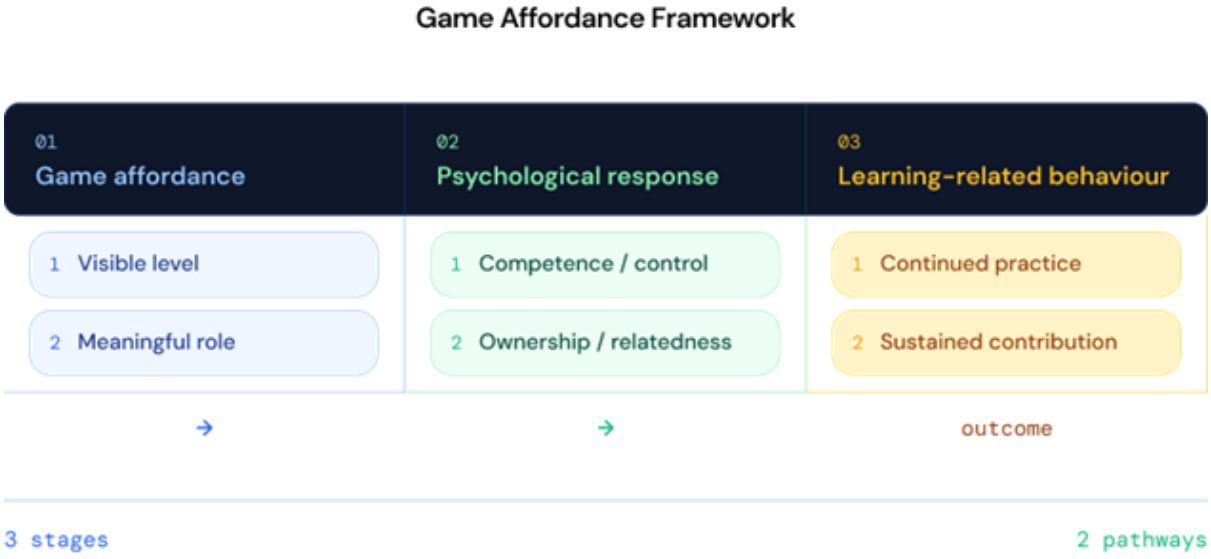


Figure 1. A practical model for reading gamification mechanics

Design check: Which need is being supported (value, competence, relatedness or autonomy) and is the mechanic surface, structural, or both?

9.5. The Excited Context

Excited's work spans learning outcomes and structures, pedagogy, tools, and infrastructure. The three cases used in this chapter are not identical projects and should not be treated as a single experiment. They were developed for different learners and purposes, in different educational settings. Their value for this chapter lies in comparison. Together they show how similar motivational functions like visible progress, ownership, and social structure can be implemented in very different forms.

The rest of the chapter is based on three connecting themes. First, progress is made visible through XP in Heimdall's Quest, weekly executive reporting in Game Lab, and levels in IT1001. Second, learners are provided with ownership through powers and identities, choice of role and game direction, or choice of pace and project topics, respectively. Third, the social layer is designed rather than left to chance: guilds and groups, company teams and level-based seminar tables all shape how learners interact.

The evidence must, however, be interpreted with caution. Heimdall's Quest is a context-specific qualitative case. Game Lab relies heavily on self-reported perceptions from a relatively small sample. IT1001 is also based on a small first-year cohort, and comparison across years involves different student groups and assessment arrangements. In addition, a separate experiment with 1,780 adults in distance education found no effect of gamification on performance, pass rate, or dropout rate when compared with a control group, although participants were positive toward the design (Jedel et al., 2021). Therefore, this chapter treats these cases as sources of design insight, not as proof of general causal effect.

Table 1. Three cases along the explicit-to-implicit gamification spectrum

Case	Context and data	Type	Key elements	Core Finding
Heimdall's Quest (Lorås, 2017)	Vocational ICT students, qualitative case study	Explicit: surface and structural	XP/points, characters and powers, guilds, competition	Points could provide an extra push, richer engagement appeared when students played with choices and identity. Generalizability is limited by the all-male, gamer-heavy case.
Game Lab (Munkvold, 2017)	University game-development students, 44 student respondents plus teacher/industry responses	Structured through role-play and construction	Company roles, real customers, weekly executive meetings, stakes, final product	Social factor, freedom and the finished product were the most cited motivators, group problems and inconsistent feedback the most cited demotivators. 72.1% of participants reported learning more in this course than in other courses.
IT1001 (Hansen et al., 2024, Sindre & Kolås, 2025)	First-year STEM teacher students in introductory Python, surveys and test-log data	Implicit structural gamification	I-to-A levels, visible progress, retries, rapid feedback, three 80% passes, self-chosen project	Students described control and continuous work positively, test participation was strongest early, and practice attempts generally rose with level. The design was not presented as gamification.

9.6. Three Gamification Designs from Excited

The three discussed cases can be seen as a movement from explicit to implicit gamification. This framing is useful because it shifts attention from whether a design looks like a game to whether it fulfills an educational function.

9.6.1. Heimdall's Quest: Explicit Gamification

Heimdall's Quest was created for a vocational ICT classroom where motivation and attention were the main concerns. The system is based on recognizable game structures. Students accumulate points or XP, work with characters and powers, participate in a guild-like social structure, and encounter competition. The learning environment was analyzed through an iterative qualitative case study (Lorås, 2017) involving observations and interviews with both students and teachers. The case therefore offers a detailed account of how learners experienced the system, not an experimental estimate of learning gain.

The most interesting finding is the difference between following rewards and playing. Points worked as an additional push to complete the work. In one sense, this is a successful extrinsic motivating factor. The mechanic prompts students to act. However, Lorås's analysis shows that the quality of motivation improved when students moved beyond point-chasing. Characters, powers, and freedoms gave them something to interpret and use. They could identify with the role, discuss possibilities, and treat the system as a space for play. In this sense, the reward was not the sole reason behind engagement. It served as an entry point towards more meaningful engagement with the learning environment.

Competition illustrates the same ambivalence. Here it was received positively, but by a class of experienced gamers. Lorås notes explicitly that the class consisted of male students with substantial commercial gaming experience. This makes it difficult to generalize the findings to learners with different backgrounds. The point is not that competition should be avoided in every course. It is that a mechanic that feels motivating and legible to one group may feel alienating, irrelevant, or unfair to another group. Therefore, competition should be treated as a design choice that needs an alternative route, not as a default feature of gamification.

9.6.2. Game Lab: Gamification through Role-Play and Construction

Game Lab offers a different setting. Students in the bachelor program in Game and Entertainment Technology at Nord University work as if they were operating a real company. Teams have leads, work with real customers, report to an executive committee each week, document progress and working hours, and finally are assessed through a portfolio. This portfolio is based on process documentation, reflection, and the final digital game. Projects can be rejected. Within the simulation, the students who fail to perform their responsibilities in time can be fired and asked to move to a new team (Munkvold, 2017). These stakes make the activity rely on consequences rather than points or badges.

Seen through a gamification lens, the company is the narrative frame, professional roles promote a sense of identity, weekly executive meetings create milestones, and the finished game makes progress explicitly visible. The design also maps closely onto the motivation determinants Munkvold uses. The 44 student respondents gave 113 motivating statements. The most frequent categories were the social factor, mentioned 34 times, freedom 22 times, the finished product 18 times, specialization 15 times, feedback 10 times, and industry relevance 9 times, with the remaining statements in smaller categories. Asked whether they learned more in Game Lab than in other courses, 72.1% of those who answered this item said yes [check n: 72.1% equals 31 of 43, not of 44]. This is perception data rather than an objective learning comparison, but the pattern is informative: students repeatedly valued belonging, choice, and a meaningful end-product.

The negative data cannot be ignored, however. There were 99 demotivating statements. The largest categories were negative aspects of group work (28), negative or inconsistent feedback (23), slow progress (15), restrictions (9) and pressure (8), with the remainder in smaller categories. The discussion

suggests that competence is clearly less served than relatedness, autonomy, and value (Munkvold, 2017). This makes Game Lab a suitable example of why a social mechanic cannot be evaluated simply by checking whether teamwork is present. A functional group can be motivating, whereas an unbalanced or poorly structured group can become the largest source of demotivation. Likewise, authentic executive feedback can add meaning, but inconsistency can make the same structure feel arbitrary.

The core point is that role-play needs pedagogical support to work. The teachers need to switch between the executive role and the facilitator role. Munkvold recommends more consistent and constructive feedback, and clearer help with week-to-week progress. The match between the project challenge and the skills available in each group must also be assessed before the final milestone is set. These supports are not decorative additions to the simulation but the conditions that make it educationally usable.

9.6.3. IT1001: Gamification without Visible Game Elements

IT1001 is an introductory Python course for first-year STEM teacher students at NTNU. It is the most implicit case. The course consists of nine modules, named I, H, G, F, E, D, C, B and A, and the more modules a student passes, the higher the grade. Students prepare through video lectures and formative practice tests, take supervised graded tests, and develop an individual project in parallel. In the 2024 design, repeated attempts were unpenalized. Automated tests supplied quick feedback. Students had three '80% passes' that could smooth progression below the usual 90% threshold. The open project also allowed a topic connected to the student's own subject area (Sindre & Kolås, 2025).

The course was analyzed with the Learning Mechanics–Game Mechanics framework (Arnab et al., 2015), which identified levels, behavioral momentum, assessment, feedback, tokens and ownership as relevant mechanics. Importantly, the course was not presented to students as gamified, and it did not rely on trophies or badges. Students' feedback emphasized control over progress and continuous work. The 2024 log data showed strong early participation in the weekly Friday test, and practice activity increased across much of the level sequence. The underlying pedagogy is mastery learning associated with Bloom (1971) and Keller and Sherman's (1974) Personalized System of Instruction. However, the pedagogy is not the focus here. What matters here is that the level structure, repeated attempts and transparent progress can perform a gamification function without game-themed decoration.

Table 2. Making progress visible: same function, different form

Same function, different form	
Heimdall's Quest	XP makes progress visible through an explicit reward system.
Game Lab	Weekly executive reports make progress visible through a professional role-play structure.
IT1001	I-to-A levels make progress visible through the assessment architecture itself.

Design lesson: begin with the function (visible progress), then choose the least intrusive form that fits the learning context.

9.7. Lessons Learned: What Works?

Table 3: Eight design principles for gamification in computing education. HQ = Heimdall's Quest

	Principle	**Main Excited evidence**
1	Make progress visible and structure it as achievable levels	HQ: XP, Game Lab: weekly reporting, IT1001: I-A levels
2	Reward competence and progress, not mere activity	HQ: risk of point-chasing, IT1001: code-line requirement removed
3	Give frequent, consistent and actionable feedback	Game Lab: inconsistent feedback demotivated, IT1001: practice-test feedback
4	Offer meaningful choice inside a clear structure	HQ: powers, Game Lab: freedom, IT1001: pace and project topic
5	Use roles, identity and narrative to build ownership	HQ: characters/powers, Game Lab: company roles
6	Design the social layer and support groups actively	HQ: guilds/groups, Game Lab: social factor and dysfunctional groups
7	Let rewards serve meaning, and leave room for play	HQ: extra push versus play, IT1001: levels without badges
8	Calibrate challenge and iterate	Game Lab: flow, IT1001: threshold and code-line adjustments

9.7.1. Make Progress Visible and Structure It as Achievable Levels

Visible progress helps a learner to keep track by repeatedly showing where they stand and what comes next. This mechanism appears in all three cases, though its form differs: XP in Heimdall's Quest, weekly reporting in Game Lab, and lettered levels in IT1001. In IT1001, the sequence itself guides the learner to the next step. Similarly, in Game Lab, weekly executive meetings turn a long project into recurring moments of accountability.

In practice, a course or project supervisor can break a long-term project into smaller, meaningful milestones. Each can be described by what the learner can now do rather than by time spent. The current milestone and the next one should be visible in the course environment or assessment rubric. The caveat is that visibility can become surveillance if every action is measured. Track only what helps learners interpret their progress.

9.7.2. Reward Competence and Progress, Not Mere Activity

A reward system teaches students what the course values. If the reward system is set in a way that students can get more points based on clicks, posts, or lines of code, they may rationally optimize those quantities. The 2023 IT1001 course implementation initially required additional lines of code at successive project levels. The requirement was removed after teachers observed longer but poorer code (Hansen et al., 2024). This case is a concrete example of a misaligned metric.

Two practical steps can be followed. First, attach points to evidence of mastery, improvement, explanation, or quality rather than raw volume. Second, review the metric after the first few weeks, looking for behavior that technically satisfies it while undermining the learning goal. The caveat is that competence is harder to measure than activity. A smaller set of well-thought-out criteria is usually safer than a dense points system built around activity data.

9.7.3. Give Frequent, Consistent and Actionable Feedback

Feedback is the bridge between visible progress and learning actions. Game Lab shows the effects of weak feedback clearly. Negative or inconsistent feedback was the second most frequently mentioned source of demotivation. IT1001 shows a different model: practice tests gave rapid information about gaps before a graded attempt. The point is not that all feedback must be automated, but that students should be able to interpret and track what happened and what to do next.

To make this approach practical, the feedback rhythm should be defined before adding game mechanics. Decide in advance what can be immediate, what requires teacher judgement, and what students can obtain from peers. Then a stable rubric or response pattern can be used. In this way, similar performance will produce similar guidance across weeks. The caveat is workload. Frequent feedback that cannot be sustained will eventually become delayed or inconsistent. This can be worse than a simpler but dependable system.

9.7.4. Offer Meaningful Choice inside a Clear Structure

Freedom of choice supports autonomy when it changes something learners care about. Heimdall's Quest offered powers and freedom, Game Lab allowed teams to make design decisions and students to specialize, and IT1001 allowed self-pacing and a self-chosen project topic. In each case, autonomy operated within defined boundaries. Game Lab also shows that restrictions can demotivate, yet teachers reported that completely unrestricted projects could lose direction or eventually become over-ambitious.

A useful approach is to define non-negotiable learning outcomes, deadlines, or quality constraints. Then two or three places can be identified where students can choose a path, topic, role, or pace. The consequences of each choice must be defined. The caveat is that choice without scaffolding can overwhelm novices. Autonomy should expand as learners gain the knowledge needed to use it.

9.7.5. Use Roles, Identity and Narrative to Build Ownership

Role-play can make an abstract course structure more meaningful. Heimdall's Quest uses characters and powers, and Game Lab uses professional identities such as team lead and a proper company-like structure. These mechanics are not merely decorative, because they can change responsibility, perspective, or action. A role matters by giving a learner a reason to contribute in a particular way.

To apply this, first ask whether a real or plausible role naturally fits the learning outcome, be it reviewer, analyst, designer, consultant, or team lead. Settle this before adding any fantasy layer. Second, give that role genuine responsibility. The caveat is authenticity. A narrative that does not connect to the work may feel fake, childish, or artificial, especially for learners who are not familiar with game culture.

9.7.6. Design the Social Layer and Support Groups Actively

Social structure is one of the most explicit cross-case themes. In Game Lab, the social structure was frequently reported to boost motivation. Yet dysfunctional groups were also the most frequently mentioned source of demotivation. Teamwork alone does not create sustainable engagement. Success depends on group composition, communication, leadership, conflict support, and task interdependence. These all shape whether relatedness becomes a resource or a burden.

In practice, define group roles and expectations in advance, and agree on a lightweight way to address imbalance before a project reaches crisis. Then plan how teachers or teaching assistants will support team leads and conflicts. The caveat is equity. A mechanic that heavily relies on peer interaction can be a serious burden for students who remain isolated, feel underconfident, or are placed in a dysfunctional group. Individual routes to demonstrate learning remain important.

9.7.7. Let Rewards Serve Meaning and Leave Room for Play

The Heimdall's Quest findings suggest maintaining a balanced relationship between intrinsic and extrinsic motivation. Points can provide an initial push. However, deeper engagement appears when learners use the system playfully and invest their own meaning in it. The takeaway is not to remove rewards, but to place them around an experience worth having (competence, choice, identity, contribution, or progress) rather than letting them become the experience itself.

In practical terms, keep the reward layer light enough that students can ignore it without penalty once the learning activity becomes meaningful. Provide optional choices, playful variations, or alternate strategies rather than a single route to gain points. The caveat is that some students will always optimize the visible system. If point optimization and good learning diverge, the scoring model needs redesign rather than a lecture about motivation.

9.7.8. Calibrate Challenge and Iterate

Gamified structures make difficulty visible, which is useful only if the progression is calibrated. Game Lab links motivation to flow conditions: challenge must be aligned with skill, and goals should be clear and feedback unambiguous. IT1001 shows iteration in practice. Modules F and E proved too difficult with a strict 90% threshold, so the course introduced three 80% passes. Similarly, the lines-of-code rule produced quantity over quality and was removed (Hansen et al., 2024).

Educators should therefore treat the first implementation as a trial that can be adjusted with time. Decide in advance what evidence would indicate a bottleneck. These may include repeated failure at one level, a sudden drop in participation, stress, or large queues for help. A mid-course review should also be scheduled. The caveat is fairness. Changes must be communicated transparently and applied so that students who progressed earlier are not disadvantaged.

Above all, the principles work as a system. The first three help to make the learning structure legible: progress, competence, and feedback. Principles four to six serve to create meaning and social structure through choices, roles, and supported groups. The final two link rewards to learning and treat calibration

as an ongoing responsibility. A course based on just visible rewards, neglecting these supporting conditions, is likely to achieve activity without the quality of engagement that educators look for.

Table 4. Common gamification pitfalls and how to mitigate them. HQ = Heimdall's Quest

Pitfall	Typical symptom	Evidence used here	Mitigation
Over-reliance on extrinsic rewards	Students ask what earns points rather than what they need to learn	HQ: points as an 'extra push'	Tie rewards to meaningful progress, allow the learning activity to become rewarding.
Gaming the system	Learners optimize the metric while quality stagnates	HQ: point-chasing, IT1001: code-line rule produced quantity over quality	Reward quality/competence, inspect unintended strategies, revise metrics mid-course.
Competition	Low-ranked students withdraw or feel forced into comparison	HQ: positive competition in an experienced-gamer group	Offer cooperative and self-referenced progress routes, avoid making public ranking compulsory.
Different player types and motivations	Some learners ignore or dislike the game layer	HQ context, broader player-type research	Keep optional mechanics lightweight, vary routes to motivation rather than assuming one player profile.
Equity and inclusion	Game language or mechanics systematically fit some groups better than others	HQ: all-male, gamer-heavy sample, broader diversity research	Pilot with diverse learners, test terminology, accessibility, and competitive mechanics.
Complexity	Rules require explanation and the game layer competes with the subject	HQ was a context-specific, relatively complex system	Start with one function and one or two mechanics, add only when a learning need is clear.
Teacher workload and sustainability	Feedback slows, content and question banks become expensive to maintain	IT1001: major question-bank preparation and about 700 assistant hours in the 2023 pilot, Game Lab: weekly executive work	Reuse assets, automate where pedagogically safe, budget staff time, and simplify before scale-up.
Technology dependence	The platform cannot support the intended retry/progression model	IT1001: Inspera did not directly support repeated use of the same tests in the needed way	Test the workflow in the actual platform before launch, maintain a low-tech fallback.

9.8. When Gamification Goes Wrong

Five themes cut across Table 4. First, extrinsic rewards can narrow attention to what is counted. Misaligned incentives are described as a core gamification predicament (Palmquist, Munkvold & Goethe, 2021). Learners can act rationally within the system while their attention shifts from the main educational goal. Second, competition can be encouraging for some learners while discouraging for others. Competitive mechanics can be especially discouraging for students with low self-efficacy or little familiarity with competitive gaming culture (Goethe & Palmquist, 2020). Third, complexity has a cost. A mechanic that needs lengthy explanation, continuous maintenance, or specialist technology may consume the attention it was meant to direct towards learning. Fourth, activity should not be confused with beneficial engagement. Gamified systems can generate unintended activity and even encourage the learners to 'game the system' for recognition rather than knowledge (Palmquist et al., 2021). Fifth, inclusion is not a final-stage add-on. Culture, age, and gender can shape how gamified systems are perceived, as argued by Goethe and Palmquist (2020). Educators must therefore move beyond a one-size-fits-all use of traditional game elements.

The ethical question is not whether points or rewards are good or bad. It is about the design's purpose: whose interests it serves, what behavior it promotes, and what costs it imposes. This concern is organized into four predicaments by Palmquist et al. (2021). These are unintended activities, misaligned incentives, compulsory behavior, and sludging. In education, the practical mitigation is simple: keep mechanics transparent and aligned with learning outcomes, and monitor disengagement. Educators should be willing to remove a feature that produces bad behavior, even if students initially find it entertaining.

9.8.1. Warning Signs Checklist

1. Are students chasing points, ranks, or completion markers rather than improving the target competence?
2. Who is not participating in the game layer, and do they still have a fair route to learn and demonstrate achievement?
3. Is competition motivating the whole class, or mainly students who are already confident and experienced?
4. Do students understand why each mechanic exists and how it relates to the learning outcome?
5. Has any metric created obvious shortcuts, low-quality output, or strategic behavior?
6. Is feedback still timely and consistent, or is the design exceeding available teacher/assistant capacity?
7. Can the platform reliably support retries, progression, and record-keeping without administrative workarounds?
8. What evidence, including stress, disengagement, and inequity, would make us change or remove the mechanic?

9.9. Practical Framework for Educators

The cases suggest a simple sequence for designing gamification. Starting with a mechanic lead to a search for somewhere to use it and starting with a learning goal gives a reason to decide whether a mechanic is needed at all. This goal-first stance reflects a broader design mindset in which the learner's needs, not the mechanics, are the point of departure (Goethe, 2019).

Table 5. A goal-first design sequence for gamification

Stage	Questions to ask
Learning goal	What should students be able to do afterwards? Would a simpler intervention reach the same goal?
Learner motivation	Which of value, competence, relatedness, and autonomy is weakest today? Who may be put off by a game layer?
Game mechanic	Surface or structural? Which learning outcome does each element serve? Where is there a meaningful choice or play?
Progression	Can every student see where they are and what comes next? Is the next level achievable with available support?
Feedback	How fast, specific, and consistent is it? Which feedback can be automated, peer-supported, or teacher-provided? What does it cost to sustain?
Reflection	What data will show whether the design works - including stress and who disengages? What will be adjusted mid-course?

The framework also helps decide whether to gamify, simplify, or leave alone. If the problem is only that students need more frequent feedback or reminders, a checklist, practice quiz or clearer rubric may solve it with less complexity. Structural gamification is most useful when the learning challenge genuinely involves progression, ownership, repeated practice or social engagement, and when the mechanics make these features easier for students to understand and act on.

Heimdall's Quest illustrates this. The learning problem was low motivation and attendance, and the gamified layer responded with visible progress and a system of roles, choices and social interaction. The framework would now add two safeguards. First, test whether the point system is aligned with meaningful work. Second, provide a non-competitive or self-referenced route for students who are less familiar with game-style competition. Rather than accepting a mechanic simply because one group enjoyed it, evaluate it against motivation, inclusion, and learning behavior.

In Game Lab, the learning goal is professional game-development practice, so the most useful gamification is structural rather than cosmetic. Roles, real customers, weekly executive reports, and a finished product make progress visible and meaningful. The framework further points to the weak points identified in the data. It suggests that teachers' feedback must be consistent, team problems need active support, and challenges must be calibrated to the level of competence available in each group. Improving the structure would solve these problems. Adding badges would not.

In IT1001, the learning goal is basic programming competence for future teachers, and the challenge is sustaining steady practice. Here the framework points away from a game layer. Levels, retries and rapid feedback already make progress visible, so adding badges would add complexity without a clear function. The main lessons concern calibration and cost: thresholds and metrics had to be adjusted after the first run, and the question banks, assistant hours and platform workarounds show that even a structural design needs resourcing from the start.

9.10. Discussion

Across the three cases, the strongest benefits are described in terms of engagement, ownership, social cohesion, control and relevance. Game Lab students frequently valued collaboration, freedom, a finished product, and industry relevance. IT1001 students described control over pace and continuous work positively. Heimdall's Quest shows how explicit game elements can create an initial motivational push and, for some learners, a space for playful engagement. These are meaningful educational outcomes, but much of the evidence is perceptual or behavioral rather than causal evidence of improved learning. The 1,780-participant distance-education experiment by Jedel et al. (2021), which found no performance, pass-rate, or dropout advantage, is a useful counterweight to overly broad claims.

The evidence is also highly context dependent. The three core cases involve different target groups: vocational ICT students, university game-development students, and first-year STEM teacher students learning introductory programming. The research designs are mostly case-based, qualitative, survey-based, or descriptive, with small cohorts in the core Excited examples. Self-report can tell us what learners experienced and valued. However, it cannot by itself establish that a mechanic caused better learning. For that reason, the design principles in this chapter should be treated as hypotheses to test in a local course rather than universal rules.

Scalability is another constraint. Structural gamification may be less visually elaborate than a full points-and-badges system, but it can still demand substantial design effort. The 2023 IT1001 case documents several hundred hours of preparation for large question banks and learning resources, around 700 assistant hours during delivery, and administrative overhead because the assessment platform did not support the required repeated-test workflow efficiently (Hansen et al., 2024). Game Lab similarly depends on regular executive meetings and feedback. These costs do not invalidate the designs. They mean sustainability must be part of the design from the beginning.

Diversity also limits one-size-fits-all solutions. As noted, the Heimdall's Quest sample was all-male and unusually experienced with games. Broader work on ethics and diversity argues that culture, age, and gender can influence the reception of gameful design, particularly competitive mechanics (Goethe & Palmquist, 2020). Future designs should therefore make room for hybrid approaches: an explicit game layer where it adds identity or play, combined with structural progression that remains understandable to students who do not identify as gamers. Learning analytics can help locate bottlenecks, but longitudinal follow-up is needed to distinguish short-term novelty from sustained learning value.

9.11. Conclusion

What have we learned about gamification? First, it is more than a collection of points, badges, and leaderboards. The Excited cases show that visible progression, roles, meaningful stakes, autonomy, feedback, and social structure can do the motivational work even when a course does not look like a

game. Second, mechanics do not operate independently of pedagogy. The same feature can support competence and ownership in one context, distract from learning in another, or benefit one group while alienating another.

What can educators take into practice? Of the eight principles, five have the most direct support across the cases: make meaningful progress visible, align rewards and metrics with competence rather than activity, give feedback that is timely, consistent, and actionable, offer real choice within a clear structure, and design the social layer with active support rather than assuming groups will function on their own. These recommendations favor structural gamification because it begins with the learning experience and uses game elements only where they serve a clear purpose.

What should educators be careful about? Point-chasing, metric gaming, compulsory competition, exclusion, workload and platform constraints can all turn a promising design into a burden. A useful rule is to choose the simplest mechanic that makes the intended learning behavior easier to understand, practice or sustain. If a quiz, clearer rubric or better feedback solves the problem, there is no need to gamify it.

The next research step is to test these design principles across larger and more diverse cohorts with longitudinal measures that connect student experience and behavior to learning outcomes, while reporting the workload and infrastructure required to sustain the design.

Copyright

Gamification in Higher Education – From Research to Educational Practice © 2026 by Ole Goethe is licensed under CC BY-NC-ND 4.0



References

1. Arnab, S., Lim, T., Carvalho, M. B., Bellotti, F., de Freitas, S., Louchart, S., Suttie, N., Berta, R., & De Gloria, A. (2015). Mapping learning and game mechanics for serious games analysis. British Journal of Educational Technology, 46(2), 391-411.
2. Ask, K. (2016). The value of calculations: The coproduction of theorycraft and player practices. Bulletin of Science, Technology & Society, 36(3), 190-200.
3. Bloom, B. S. (1971). Mastery learning. In J. H. Block (Ed.), Mastery learning: Theory and practice (pp. 47-63). Holt, Rinehart & Winston.
4. Blumenfeld, P. C., Kempler, T. M., & Krajcik, J. S. (2006). Motivation and cognitive engagement in learning environments. In R. K. Sawyer (Ed.), The Cambridge handbook of the learning sciences (pp. 475-488). Cambridge University Press.
5. Costa, J. M. (2023). Using game concepts to improve programming learning: A multi-level meta-analysis. Computer Applications in Engineering Education, 31(4), 1098-1110.

6. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining 'gamification'. In Proceedings of the 15th International Academic MindTrek Conference (pp. 9-15). ACM.

7. Dichev, C., & Dicheva, D. (2017). Gamifying education: What is known, what is believed and what remains uncertain: A critical review. International Journal of Educational Technology in Higher Education, 14, Article 9.

8. Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. Review of Educational Research, 74(1), 59-109.

9. Goethe, O. (2019). Gamification mindset. Springer.

10. Goethe, O., & Palmquist, A. (2020). Broader understanding of gamification by addressing ethics and diversity. In C. Stephanidis et al. (Eds.), HCI International 2020 - Late Breaking Papers: Universal Access and Inclusive Design (LNCS 12425, pp. 688-699). Springer.

11. Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work? A literature review of empirical studies on gamification. In Proceedings of the 47th Hawaii International Conference on System Sciences (pp. 3025–3034). IEEE.

12. Hansen, G., Sindre, G., Korpås, G. S., & Vassbakk, J. P. (2024). La kvar student velje sitt eige tempo: Meistringslæring i intro programmering. [Letting each student choose their own pace: Mastery learning in introductory programming]. Læring om læring, 11(1).

13. Jedel, I., Palmquist, A., Munkvold, R., Goethe, O., Jonasdottir, H., & Olsson, E. (2021). An industry experiment of academic performance and dropout in gamified distance education. In Proceedings of the 5th International GamiFIN Conference 2021. CEUR Workshop Proceedings.

14. Keller, F. S., & Sherman, J. G. (1974). PSI, the Keller Plan handbook: Essays on a personalized system of instruction. W. A. Benjamin.

15. Koivisto, J., & Hamari, J. (2019). The rise of motivational information systems: A review of gamification research. International Journal of Information Management, 45, 191-210.

16. Lorås, M. A. (2017). The gamified learning environment - pedagogical possibilities and pitfalls. NIK: Norsk Informatikkonferanse.

17. Munkvold, R. I. (2017). Game Lab: A practical learning approach for game development. In M. Pivec & J. Grundler (Eds.), Proceedings of the 11th European Conference on Games-Based Learning (pp. 472-479). Academic Conferences and Publishing International.

18. Palmquist, A., Munkvold, R., & Goethe, O. (2021). Gamification design predicaments for e-learning. In X. Fang (Ed.), HCI in Games: Experience Design and Game Mechanics (LNCS 12790, pp. 245-255). Springer.

19. Ren, X. (2016). Rethinking the relationship between humans and computers. Computer, 49(8), 104–108.

20. Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist, 55(1), 68-78.

21. Sailer, M., Hense, J. U., Mayr, S. K., & Mandl, H. (2017). How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. Computers in Human Behavior, 69, 371-380.

22. Sindre, G., & Kolås, L. (2025). Mastery learning as gamification: Level-up through high-transparency assessments. Proceedings of the European Conference on Games-Based Learning, 19(2), 789-797.

23. Steele, J. P., & Fullagar, C. J. (2009). Facilitators and outcomes of student engagement in a college setting. The Journal of Psychology, 143(1), 5-27.

10. The Learning Environment

Guttorm Sindre

Norwegian University of Science and Technology (NTNU)

10.1. Introduction

Many study programs in higher education have substantial problems with dropouts and low student satisfaction. A good learning environment is seen as essential for the retention and progress of students, and Excited had a focus on this from the outset. Main objectives were to improve the first-year experience of students and increase the focus on project-based learning – both in terms of learning effectiveness and social belonging in the study program. This chapter will present a selection of efforts by Excited related to the learning environment, with advice for others who might want to do something similar.

Learning environment can be defined as "the diverse physical locations, contexts, and cultures in which students learn"[1]. Abualrub et al. (2013) argue that it should be considered a multifaceted concept, covering both organizational aspects and student perceptions.

In Norway, there is an annual national survey investigating student satisfaction with the various degree programs in higher education, called *Studiebarometeret*[2]. This survey contains two question groups related to the learning environment, one being "Study environment and infrastructure" focusing on physical environment and equipment, and the other one just called "Study environment" which focuses more on the social issues. Each group has three questions on a 5-point scale, inquiring how satisfied the students are with various aspects of the study environment. The former category has questions about (i) the rooms for teaching and studying, (ii) the library services, and (iii) the ICT learning tools and related support, and the latter about (i) the social environment among students in the program, (ii) the academic environment among students in the program, and (iii) the relationship between the students and the academic staff. The score on these questions is not too bad for Computer Science students, both typically averaging just below 4 of 5 (e.g., 3.9 in 2025). While such scores still leave room for improvement, there is reason to believe that they also fail to capture some serious challenges related to the study environment not just in Computer Science, but more generally in Norwegian universities.

One notable trend is that students' attendance and time-investment in courses are decreasing. This is also asked about in the national student survey, which has shown sinking numbers for the students' self-reported time investment in learning activities since 2016. Hence, the decline started well before the Covid pandemic, and the trend has continued in recent years. Thus, the latest survey in 2025 showed the lowest figure ever measured for student self-reported study effort nationwide. On the other hand, effort on paid work outside of the studies has increased somewhat, but much less than the study effort has dropped. Within technology and natural science studies, the average weekly hours of attending organized learning activities dropped from 14.4 to 12.0 hours from 2022 to 2025, while self-study was evener (24.5 to 23.6), so the organized activities seem to be prioritized the least, for instance with lower attendance at lectures. This is not just observed for weak students – for instance Industrial Economy at the NTNU, with one of Norway's highest admission thresholds, saw self-reported time on organized study activities drop from 15.0 to 11.8 hours in this period, and in Computer Science the drop was from 12.8 to 9.9. Similar trends are found in other countries, for instance indicated in the large longitudinal study by Dekker and Theelen (2025).

Lower attendance would not be a big problem as such, if students otherwise learnt as much as before, qualified for good jobs, and were generally happy with their lives. However, Norway's national *Students' Health and Well-being Study (SHoT)*[3] indicates a strong increase in challenges such as mental distress, anxiety and loneliness from the first survey in 2010 and through the second to last one (2022), and although the numbers improved somewhat for the most recent one (2026), it is not a satisfactory situation. Similar challenges with student wellbeing have been observed also in other Western countries (Abelson, Lipson, & Eisenberg, 2023). To some extent this might be related to the lower attendance – more students sit at home, studying alone, rather than coming to campus to attend lectures and group sessions. Since the response rate to the national student survey is modest, it might be plausible to assume that students with higher attendance are also more likely to respond to the survey, for instance because teachers or admin personnel advertise the survey in lectures to encourage a bigger response rate. This could create a biased, overly positive score for the social and academic learning environment among students. However, the learning environment in higher education may not be the root cause of these increasing problems with student mental health. Indeed, many have pointed to social media that students have been exposed to already from their earlier adolescence, e.g. (Keles, McCrae, & Grealish, 2020), for instance causing reduced ability to concentrate and lower self-esteem versus the seemingly "perfect" facades posted by others. Regardless of the cause, these trends change the world that higher education operates within, so that higher education itself must change to address the challenges in an appropriate manner.

Project-based learning has been seen as one way to increase student motivation and attendance. However, it cannot be the only solution. A project team does not work well if some of the students struggle with social anxiety, or if some drop out or do not contribute to the teamwork. Moreover, most degree programs are unable to offer project-based learning in every course. Typically, even programs known for a strong focus on projects will also have courses taught in a more ordinary lecture + exam format, and these will tend to be placed in the early years of study when topics are more generic – although it is the first year which is also key to establishing good study habits and a sense of belonging for the students. Hence, many students may struggle during the first semesters and end up dropping out before reaching the project courses which were supposed to give them extra motivation.

The rest of this chapter is structured as follows: In the next section we discuss general interventions to improve the learning environment, followed by a section presenting the specific interventions and research that Excited has conducted related to learning environment. Finally, the last section of the chapter sums this up with some concluding advice on how to improve learning environments elsewhere.

10.2. Interventions for the learning environment

In higher education in Norway, the most generally applied intervention to combat loneliness among first-year students and ensure motivation and a sense of belonging is the so-called *Start-up Week,* which is initiated by the matriculation ceremony on a Monday in the middle of August. During this week, there is no ordinary teaching. Instead, the study program leadership has some time to introduce the students to the program and how it is organized, and many programs have a workplace excursion to illustrate the program's job relevance. The students also get some tutorials about organizational issues, study techniques, and teamwork, the latter exemplified by a little group project. Maybe most importantly, second-year students work as mentors for the first-year students, introducing them to the university and surrounding town as a social arena, with various parties and get-together activities. While many students report positively about getting new friends through the start-up week – possibly friends

for life – it also has its problems. It is a one-week stint, set apart from the everyday study activity, and the tiny team project does not count towards the credits or grade of any of the courses they will be taking in the first semester. Students who do not enjoy partying may prefer to abstain from many of the get-together activities and thus find it more difficult to make friends. Those who arrive a week or two late, for instance through supplementary enrollment, will miss the start-up week entirely. In spite of the existence of start-up weeks in almost every study program, the study effort and mental health has been steadily decreasing as described above. Hence, although the start-up week did not cause these problems, which might have been even worse without it, it is also clear that it does little to solve the problems.

In the light of the above, the social and disciplinary onboarding of students in a degree program should be a more consistent effort throughout the entire first semester or longer, integrated in the every-week learning activities, rather than just a one-week stint set apart from the normal mode of operation. At KTH in Sweden, the degree programs have been systematically designed with this in mind. For instance, their Computer Science degree includes the course DD1390 Programme Integrating Course in Computer Science Engineering[4]. The course is mandatory for, and exclusively offered to, CS students and its 6 credits are distributed across the three first years of study – hence with low intensity in each semester. Other engineering study programs at KTH have similar courses specific to their disciplines. The aim of this course, according to the KTH website, is to "make the course participants into professional students, among other things by showing how the program's courses are interconnected and giving everyone an opportunity to reflect on their studies and current courses. It will establish contacts between the years and with teachers. It is also a part of the program's quality development process."

In Norway there is no tradition for such longitudinal courses to foster reflection and belonging in the study program. Both Nord and NTNU tend to have courses set within a single semester, and NTNU also has a stricter structure that each course must be a multiple of 7.5 credits – typically meaning that each semester consists of four courses, with no openings for longitudinal courses to "steal" one or two of those credits. This means that other interventions must be sought. One obvious candidate is *the signature course,* as recommended by the FTS project at the NTNU: each degree program should have a first semester a course exclusively for their own students, specifically targeting their discipline and future profession. This recommendation by the FTS project is a response to a challenge that many study programs in technology tend to start with some generic courses in math and physics which give students a limited view of their own discipline. Study programs within the scope of Excited at Nord and NTNU have this to a varying degree. For the Games and Entertainment Technology bachelor at Nord, all first semester courses could be said to be clearly related to the discipline (Game Design, Programming, Art Design, CG Art Fundamentals, Sound for Creative Media, Animation Fundamentals), and the bachelor in Digital Enterprise Development at the NTNU has the first year course DIFT1001 Web Technology and Teamwork which is exclusive for that program and clearly related to the discipline, and two of the other three courses are also exclusive to the program. For the 5-year integrated Master's in Computer Science, the situation is somewhat different. Out of 4 courses taken in the first semester, three are generic and offered by other departments (Calculus, Discrete Math, Philosophy), and the fourth (CS1, introduction to Python programming) – although clearly related to the discipline – is taken in a rather big class of 500 students together with four other study programs. This makes it much more challenging to build a discipline-social environment by means of this course. An obvious solution might be to split the CS1 course into separate classes to have one such course exclusive to the Computer Science students, but this also has challenges in terms of increased costs (e.g., more teachers) and other constraints, such as available learning areas.

The learning area can also play an important role for the social learning environment. Some programs offer *identity areas* to their students, so that teachers of various courses come there to teach (given that the courses are exclusive to those students) rather than having the students go to different rooms for different courses. The advantage of this is also that students are likely to find an available spot for self-study at campus outside of classes, with good likelihood of meeting classmates if showing up at campus. If, on the other hand, students struggle to find a place to sit, and the likelihood of meeting friends or classmates is anyway low, it will feel more tempting to stay at home to study rather than going to campus. Programs offering an identity area often have higher student satisfaction than those who do not – for example, the 2-year Master's in Digital Transformation at the NTNU has had very high scores in the national student survey several years in a row (in 2025, 4.8 of 5 for learning environment, 5.0(!) of 5 for overall satisfaction), but this is a rather small class (20-30 students) mainly taught in an identity area. Many other study programs could learn something from this program, but there would not be enough rooms at campus to offer similar identity areas for all study programs.

10.3. Excited interventions and research

In this section we will present insights and experiences gained from Excited interventions and research, specifically related to student identity and sense of belonging and how the learning environment may contribute to this. Table 1 gives an overview of what will be presented. In the "What?" column, an "I" means we made an intervention, "R" means we conducted research, and "I+R" that we both made an intervention and did research about it. The four rightmost columns indicate which aspects of the learning environment were addressed: Social between students, Academic between students, Student-staff relationship, and Physical learning environment.

Table 1: Summary of various interventions and research related to learning environment

Effort	What? (Research/ Intervention)	Social student-student	Acad. student-student	Student-staff relations	Physical learning environm.
CS1 for STEM teacher stud., NTNU	I + R	X	X	X	X
Workdays at NTNU and Nord	I + R	X	X	?	X
Conflict management (learning game)	I	X	X		
Students as teachers at NTNU and Nord	I + (R)	X	X		?
Hackathons and creative media jams	I	X	X	X	
CatchIDI department symposium	I			X	
Hybrid and online learning activities	R	X		X	X
Student identity and belonging	R	X	X		

In the following subsections, we will give a little more information about each of the initiatives listed in the table.

10.4. CS1 as a signature course for the STEM Teacher Students, NTNU

This intervention is described in its own chapter (Self-Paced Mastery Learning), so we will not go into detail about the course design here, only say a little bit about the way it addressed four various aspects of the learning environment as indicated in Table 1. The course has replaced large auditorium lectures that previous cohorts of STEM teacher students received in CS1 with a course exclusively for the STEM teacher students, taking place in a smaller seminar room. Instead of the compulsory exercises typical of other CS1 courses, it has mandatory attendance in the seminars. Here students engage in group learning activities. This both contributes to the social environment among students (getting to know their classmates) and the academic environment through their joint problem-solving activities. Rather than providing frontal lectures, the professor walks among student group tables offering feedback and explanations – alike two TAs who are also present. This offers much closer contact with the professor than students are used to from most other courses in their first year of study. The seminar room is equipped with wall screens where students can share within their roundtable group the problems they are working on (e.g., practice tasks for a test), thus also utilizing special infrastructure.

Figure 1.1 summarizes student responses to some questions related to the social learning environment for the 2025 offering of the course. For the most part students agree or strongly agree that the course contributed positively to a sense of belonging in the study program, made it easier to get to know classmates, and encouraged peer collaboration. Of the four questions shown in this diagram, the only one where there was more disagreement was the one where this would be hoped for – namely asking whether the students would rather have preferred to have CS1 together with other study programs in a traditional big lecture and exam format.

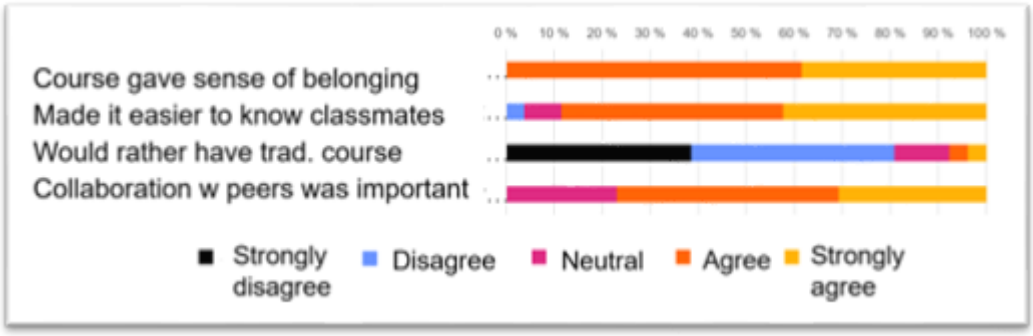


Figure 1.1. Student opinion on the course's contribution to the social learning environment

In post-course focus group interviews, several students also expressed that they appreciated the increased accessibility of and closeness to the teacher compared to lecture-driven courses. It is, however, unlikely that this course by itself will have much impact on the score for the question on relationship between students and staff in the national survey. The course takes place in the first semester, while the survey is run when the students are nearing the end of their third semester, so the CS1 course is then just 1 of 12 courses they will have been exposed to, and not the most recent one.

There are several other study programs both at Nord and IDI/NTNU that have signature courses in the first semester. However, these courses did not emerge as direct interventions by Excited, so we will not claim credit for them here.

10.5. Workdays at NTNU and Nord

The Excited / IDI Workday. Unlike the STEM teacher study program as mentioned in the previous subsection, the Computer Science students at NTNU do not have a course of their own in the first semester, as their CS1 course is together with several other study programs, with a class size of almost 500. The Bachelor Informatics is one of the other programs in that class, and they also lack a course just for them. To give these first-year students better motivation to come to campus – especially for a morning slot (8-12) that happened to be void of lectures, Excited PhD student Madeleine Lorås came up with the idea for the *workday* in 2017, in collaboration with *Sit* (The Student Welfare Society in Trondheim) who offered funding for breakfast for the students in the first run of the intervention. Still ongoing, this is one of the longest running interventions conceived by Excited, and the department has signaled that they will continue the scheme after Excited ends.

The "workday" lasts from 8-12 once a week during each teaching term and is an offer to first year students in Informatics and Computer Science. Students are offered a simple self-serve breakfast (sliced break, toppings, milk, coffee, …), and there are TA's present who may be able to assist with all the four courses that these students are taking in the current semester. In addition to seeking help from assistants, students may help each other, as they are seated in roundtable fashion and have the same courses with some assignments that must be completed. Hence, it addresses both social and academic interaction between students, in addition to physical aspects in terms of providing a room in which to sit and work for 4 hours.

Initial experiences from the first workday intervention in 2017 were reported in (Lorås & Aalberg, 2018). A more recent research article about the intervention was published by Lorås & Aalberg (2021) including an evaluation of the workday. Figure 1.2 shows aggregated results from a questionnaire survey among attendants, using a 5-point Likert scale. Female students were slightly more positive than males, but no significant difference, and both groups were clearly better than the lukewarm average 3. As can be seen, the workday was considered to contribute positively both to the social and academic learning environment, motivating the students to put in more effort than they would otherwise have done. We have received similar results on more recent evaluations.

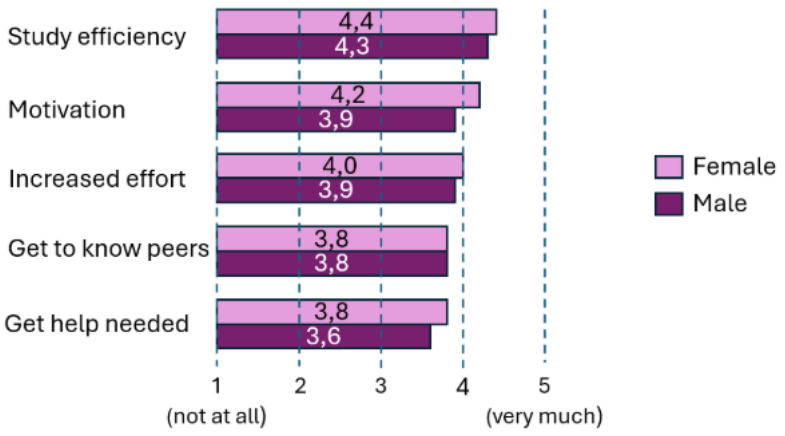


Figure 2. Average student responses, workday evaluation, adapted from Lorås & Aalberg (2021)

The workday is an example of how one can mend constraints in the study program design (in this case the lack of a first semester signature course) by an intervention going across or outside courses. It is positive that the department is committed towards continuing the workday. On the other hand, it is important to keep in mind that it was an ad hoc intervention to a more structural problem in some study programs. Hence, the workday might not be an ideal solution, rather a partial fix in the absence of more systematic solutions like a program-integrating course (cf. KTH) or a stronger string of exclusive signature courses taught in a learning area dedicated to the study program. However, it might be worth noticing that the social onboarding in a study program will be more robust if having a workday scheme in addition to signature courses. Relying on just the first semester signature course to facilitate the social-academic onboarding of students in the program would mean that, e.g., students who are exempt from the signature course due to previous studies elsewhere, may miss the most social learning activities in the class. Here, the fact that the workday scheme is *not a course,* rather a learning activity serving all courses the program has in that semester might be helpful to some students who would otherwise be excluded from social learning activities specifically within the program.

An example of using a workday type of intervention in a smaller, tightly knit study program with several signature courses, is the social workshops for students in Game Development and Entertaining Technology at Nord University. At Nord, this was done in the afternoon, serving pizza rather than breakfast. Initially called *Afternoon Workshops* but later renamed *Excited Social Workshop*, these gatherings have contained a mixture of presentations by industry representatives, teachers, and students, combined with the students' own problem-solving activities. Initially it was only for one study program but was then extended to three study programs (Games and Entertainment Technology; CG Art and Animation; Film and TV Production). This provided good opportunities for the students to get to know each other across related study programs.

10.6. Conflict management in project work (game-based learning)

Both at Nord and NTNU, project-based learning plays a central part in the study programs. At Nord, the GameLab course (Munkvold, 2017) makes groups of students work as a small, simulated game development company, trying to develop a game for an external customer (ideally role-played by a person from the game industry, or with such industry experience). It has been observed that project work is often good both for motivating students to learn more, and for the social learning environment because they get to know their peers better when they work together on a joint challenge. At the same time, project work may also be a source of conflict. In courses where the assessment is individual (e.g., assignments during the semester, the final exam) each student may organize their workweeks to their personal preference and choose their own studying approach and ambition level. Project work often requires several group members to be at the same place at the same time, with a mutual understanding of ambition level and fair distribution of effort. If there are serious conflicts in a project team, this can be detrimental to both the social and academic learning environment of the students involved.

Of course, one might argue that such conflicts are a good thing because students may acquire important experiences and competencies from them – after all, they must expect to face conflicts in their future jobs, too. However, from such a perspective, just having been accidentally exposed to conflicts is hardly the best learning approach. Students should be given some knowledge and tools on how to handle conflicts – which would give a better learning experience with potential conflicts in student projects and make them even more prepared for the future. See the chapter "The Collaborative Learning Challenge" by Munkvold in this same book for a more thorough discussion of these issues.

Rather than just lecturing or offering textbook material about conflict handling, people at Nord University came up with the idea of a game called *Cards of Conflict* (Munkvold, Kolås, & Palmquist, 2023). In this game, groups of students draw cards outlining various conflict scenarios in project teams and then try to rate and discuss what is the best solution to the conflict. In addition to being fun and social in itself, and thus contributing to the social learning environment, the game prepares students mentally for typical conflicts that may occur in their project teams, and the discussion experience makes it easier for students next time around, in the more serious setting of a full semester project, to address conflicts early rather than letting them grow until they poison the team's working environment. The Cards of Conflict game is flexible in the sense that it is easy to introduce new cards, not only to address different project settings, but also beyond just project-based learning. For instance, some educators in Sweden have adapted the game to address conflicts related to AI (Mozelius, Sällvin, & Johansson, 2026).

10.7. Students as Teachers at NTNU and Nord

Both in the abovementioned workdays and in large courses in early study years, teaching assistants (TA) play an important role as first-line educators. Especially in large lecture classes, where the students only see the professor as a somewhat distant figure at the podium (if they attend), their main contact with teaching staff will be with TAs. The social skills and pedagogical competencies of TAs will thus be essential to students' learning experience.

The Excited intervention related to TA competence development was to redesign the discipline specific part (6 h) of the NTNU 20 h pedagogical course for teaching assistants. The structure was 2 h for an introductory workshop, then 2 h for a home assignment, and finally 2 h for a second workshop – more details can be found in Lorås (2020). The intervention was positively evaluated by the TAs who participated in the course, as most of them found the discipline-specific part more useful than the generic parts. In a follow-up study with international collaboration across three countries (Sweden, Norway, Czech Rep) challenges of TAs were elicited from their reflective essays from such pedagogical courses (Riese, Lorås, Ukrop, & Effenberger, 2021). Although there were some differences, the main issues appeared to be quite similar between countries, and the article provided four key recommendations related to the pedagogical training of TAs (p.552): *(1) Best practice needs to be defined and spread*. It is not necessarily obvious to TAs (or even to professors) what the best practice is. *(2) Acknowledge threats to best practice and address them.* Course coordinators should make informed decisions to minimize these threats. For instance, TAs cannot be expected to give detailed feedback on student work if they have too little time available. *(3) Dare to discuss ethical dilemmas and provide guidelines.* Dealing with, e.g., plagiarism or deadline extensions may feel problematic to TAs and they need info on how to handle such situations to act based on knowledge and guidelines rather than feelings. *(4) Recognize the student-TA relationship as unique.* TAs are age-wise and socially very close to their students, sometimes having close personal relationships with the students they assist, thus needing guidelines to handle potentially awkward situations.

Another Excited intervention with students as teachers took place at Nord University, where third year students in the program for Game and Entertainment Technology developed a mini course for first-year students on how to be a team lead in their game development project. Whereas the above intervention with improved TA training had a more top-down nature, related to the training of students who are hired for the whole semester to do a teaching job, the intervention at Nord was student-driven. The idea was conceived by one third year student because many freshmen experienced the role as Team lead in the first year GameLab project as quite challenging. The student thus applied for some small funding from

Excited and designed a series of workshops to coach those first-year students who might be interested. The feedback from the students who received the coaching was generally positive, indicating that they felt more comfortable in the team-lead role after the coaching.

10.8. Hackathons and Creative Media Jams (Nord and NTNU)

Hackathons are time-limited collaborative events – for instance lasting for an evening, a day or a weekend, where students solve problems related to a real-world challenge. In IT education it will often involve prototyping of software. A hackathon need not be related to one specific course, but could be related to several topics, and is often a voluntary extra-curricular activity. It can be a good opportunity for students to meet and collaborate across cohorts and study programs. Hackathons can also be attended by teachers, either as supervisors or even as participants in the problem-solving tasks – in both cases, this can give opportunities for a more social-academic contact between students and teachers than what tends to be achieved in a normal lecture and exam course.

At NTNU, one of the Excited mini projects was related to a course in software security and data privacy. A previous study (Li, Meland et al., 2023) had investigated how Generative AI might impact the exercises in such a course, finding that many exercises related to identifying and fixing security vulnerabilities – which previously had been quite demanding for the students to solve – could now be quickly solved by GenAI with little student effort beyond pasting the question text into a prompt field. Hence, the 2024 run of the course explored the deliberate use of GenAI, which would likely be an important tool in the students' work-life when they graduate anyway, while still also teaching them the basics. The particular intervention of the mini project was a hackathon where students worked in teams of two or three to find security vulnerabilities in a quite large software artefact. While the primary target group for the hackathon were the students taking the course, it was also open to other students interested in the topic, demonstrating how hackathons can enable learning beyond and between courses. The task that the students got would have been quite demanding without GenAI – similar-sized exercises would have three week deadlines in the course some years earlier. With GenAI, students were able to perform the task in a couple of hours, with good quality. In post-task interviews, they still demonstrated a healthy skepticism about GenAI – some expressing that they did not quite trust the findings of GenAI, and some worried that they did not learn as much as they would have if doing the task in a more tedious way with simpler tool support. The experience indicating that hackathons could be a promising ingredient in software security and data privacy courses, and also that students should still learn the basics themselves to be well positioned to evaluate the detailed findings of GenAI.

At Nord University, there have been Creative Media Jams for the entertainment and media technology students, as discussed by Kolås et al. (2026). A Creative Media Jam is a specific type of hackathon focusing more on digital storytelling, art and media production rather than hardcore programming. This activity was very popular among the students. One of the questions in the post-activity survey was "Why do you attend intensive project events (hackathons, game jams etc.)?" Enjoyment was the most frequently cited reason for participation. Many participants simply described these events as "fun" or "enjoyable", often emphasizing the excitement of creative challenges. One respondent explained: "Fun to be challenged", while another noted: "I think they are fun and an enjoyable challenge". These comments suggest that intrinsic enjoyment and the thrill of testing limits are central motivators. Learning opportunities were another dominant subtheme. Participants highlighted game jams as a chance to develop skills and gain experience, describing them as opportunities to "learn new things", "test skills", and "improve myself and try out more things". Some emphasized the intensity of learning: "An extremely good way to learn quickly, and to get to know others, and yourself better as a

developer". These statements indicate that intensive creative projects (ICPs) are valued as spaces for rapid skill development and creative growth. Social aspects also featured prominently. Several respondents valued the chance to meet new people and expand their networks: "Meet new people and have fun testing new ideas or tools" and "Networking, trying and learning new skills, break from other projects etc.". Others highlighted the sense of community: "Game jam is fun and social" and "It's fun, social and a rich learning experience". Interestingly, one participant reflected on personal challenges: "I'm introverted and appreciate the spaces where I am forced to socialize with strangers", suggesting that ICPs can also serve to overcome social barriers. A smaller number of participants mentioned external motivators, such as school-organized events or perks: "Our school provided a game jam once" and "free pizza and beer". Some responses reflected deeper motivations, such as using the events to "immerse myself in work I enjoy" or "refresh your mind and just work on something simple". These comments show that ICPs are not only about fun and learning but also about personal well-being and creative immersion. Overall, the findings indicate that intensive project events are perceived as multifaceted experiences that combine enjoyment, skill development, and community interaction, making them attractive to students and aspiring developers.

10.9. Catch IDI Department Symposium, NTNU

Whereas most interventions focus on the social and academic learning environment among students, the Catch IDI Department Symposium had an idea more directly targeting the relationship between staff and students, and the students' sense of belonging to an academic department rather than just a study program. The Department of Computer Science at NTNU is a rather large department, being the main provider of courses in several different study programs (e.g., 5-year Integrated Master's in Computer Science, 3+2 Bachelor Engineering + Master in CS, 3+2 Bachelor + Master Informatics, 3+2 Bachelor + Master's in Digital Transformation, etc.). It was observed that especially first- and second-year students had limited knowledge of what professors in the department were doing apart from lecturing and setting exams (or, in case of project-based learning, setting and grading project assignments). To mend this, a one-day symposium called *Catch IDI* was conceived in 2018 and organized for the first time in January 2019. The bulk of the program was professors presenting their research in a popularized way, to be understandable also to first year students, but the event also had social coffee breaks, lunch, and an evening banquet with dinner, speeches, and student bands playing.

The first symposium was a great success, lowering the threshold for students and employees to connect socially, and creating better awareness among students concerning the advanced research going on at the department. Hence, the symposium has been held every year since. In 2025, the economy was tighter than before, so the event had to be organized on campus rather than in a hired venue, and with much cheaper food than before. However, this did not hurt the evaluations of the event, which were no less positive than before.

10.10. Hybrid and online learning activities

After a merger in 2016 (which was part of a bigger national trend), NTNU were offering studies in three different towns (Trondheim, Gjøvik, Ålesund), sometimes with similar courses in all three places. From the Government and top management there was a wish to get some demonstratable synergy out of these mergers, hence there were initiatives to offer some courses across multiple campuses, for instance with a hybrid mode of teaching. A typical setup might be to have a teacher lecturing on campus in town A, and live streaming this to a teaching room in town B where other students might be watching it – maybe with a local teacher also present, maybe not. Our investigations of some such cases indicated that it was

not necessarily a good idea as both students and teachers had many negative experiences with this delivery format (Bahmani, Hjelsvold, & Krogstie, 2019). In many ways, hybrid delivery carried a risk of making it worse for both student groups. The students being on-site might feel that the teacher was somewhat less focused on them than s/he might have been if only addressing in-room students, i.e., the hybrid component caused poorer flow of the lecture. Invariably, though, the online students were the ones who suffered most, as to them it seemed that the teacher was mainly focused on the onsite students, while the streaming was more of a by-product – thus giving a poorer learning environment especially considering staff-student contact. A key insight from our studies was that it is challenging to repurpose an onsite course to hybrid delivery, and that more careful planning would be needed to accomplish good multi-campus courses. Further research along these lines led to a conceptual framework to provide advice for teachers planning multi-campus courses (Bahmani & Hjelsvold, 2020) which has subsequently been applied in case studies (Bahmani & Hjelsvold, 2024).

During Covid lockdowns, on-campus lectures and other learning activities were mostly switched to online webinars – out of necessity rather than by choice. Lorås, Haugset et al. (2021) conducted a mixed-methods study to find out how students were affected by this change, based on interaction tracking data, student weekly reports, and interviews. The switch to online was found to reduce student participation in organized learning activities (e.g., lectures) without correspondingly increasing their independent efforts, and they missed the campus-based activities due to the social arena they provided (e.g., meeting other students in the lecture breaks), as well as for their scaffolding contribution to their study efforts. The switch to online was found to be especially hurtful to students with poor self-regulation, thus increasing the learning gap between stronger and weaker students.

10.11. Student identity and belonging

Two of our PhD projects have investigated student identity formation in various ways. Gunhild M. Lundberg defended her thesis at NTNU in 2021. She addressed the topic of graduate employability. While this is mostly considered from the angle of students acquiring the "right" competence, Lundberg took a more processual perspective, considering how students build a professional identity during their studies – through *alignment* with the discipline, *engagement* in learning activities, and *imagination* of a future career. The paper (Lundberg, Gaustad, & Krogstie, 2018) emphasized that employability emerges from a combination of curricular and extra-curricular activities, and that extra-curricular activities may be especially important on the CV because they indicate that the candidate has a personal interest in the field and a drive towards technological exploration, beyond just passing exams. The importance of the physical learning environment is stressed, both in terms of appropriate IT equipment and infrastructure, so that students will be familiar with tools used in work life, and by means of an identity area for the class, so that students get to know their peers and have a closer relationship with teaching staff than they might have had in a bigger lecture hall setting. The study (Lundberg & Krogstie, 2020) confirmed the importance of project-work for the students' social and academic learning environment (getting to know their peers better), but also found some students being uncertain whether they really fit in with the other students in the class, or with the study program and future work roles in general. Lundberg & Ness (2020) focus on the importance of enabling students to imagine their future dream job, emphasizing that it is not only important to foster good relationships among students, or student-staff, but also with employers providing realistic yet diverse examples of future careers and what a typical workday in industry might look like.

Justyna Szykiewicz, defending her thesis at Nord in 2025, looked especially at the contribution of project-based courses to the identity formation of Norwegian computing students. Identity formation

was found on three levels: individual, collective and disciplinary – and project-based learning contributed uniquely to all three (Szynkiewicz, 2025b). The social and academic environment between students would be especially important for the collective identity formation, while a good relationship with teachers and practitioners would be important for the disciplinary identity formation. At the same time, her PhD research reveals a tension between developing industry-ready students who have the competence needed to be effective as software developers and the wish to develop more overarching societal responsibilities related to ethics and sustainability (Szynkiewicz, 2025a).

10.12. Concluding advice

Excited's efforts related to the learning environment have included many different initiatives, and Table 1 does not give a complete picture. Here, we sum up some final takeaways from the research and interventions described above:

- **Study programs should contain a string of signature courses** already from the first semester. It is important to give students good learning experiences which contribute to **building their disciplinary identity**. Signature courses may help give the students a better idea of what kind of study they have embarked upon, rather than just giving them more generic courses taught in large classes during the first year. To help the students in developing a disciplinary identity, it is important to look not just at the design of specific courses but at the study program as a whole – and beyond, as extracurricular activities are also essential for student identity.

- **A workday intervention can be useful to offer social small-group learning activities**, and cheap, simple breakfast can further motivate students to get up and go to campus in the morning. This can be especially important in programs that lack a sufficient string of signature courses in early semesters, but as discussed in this chapter, the social-academic onboarding of students in a program will be more robust if having both signature courses and a workday scheme.

- We believe that **face-to-face contact between students and teaching staff is important**, for direct feedback and discussion related to the students' learning. In large classes, this contact will mainly be with TAs rather than the professors. It is important that the TAs are well prepared for their role. Just like most professors can become better educators through pedagogical courses and engaging in communities of practice, TAs can also become more secure in their roles through suitable preparation, some of which needs to be specific for the course or discipline. Good TAs can contribute a lot to how students perceive the social-academic learning environment.

- **Hybrid course delivery is challenging**, due to the different modes of contact with two student groups in the course (some being present in the room, others participating online). Simply taking a traditional single-campus course and adding streaming of teaching sessions to other campuses tends to give poor results. **Rather, a multi-campus course should be designed anew to fit this kind of delivery,** and Excited has provided a model and recommendations to support this.

- **Do not be afraid to introduce learning activities with mandatory attendance**, especially in signature courses in the first semester. Making students come to campus and work together on course activities will contribute to the social-academic learning environment and increase the students' sense of belonging. However, sessions which have mandatory attendance should be dominated by student-active learning where students interact with each other, e.g., working

together to solve problems. **Don't require mandatory attendance at sessions which are mainly lectures or similar activities with the teacher in the spotlight**.

- **Project-based learning can contribute positively to the learning environment** in several different ways. Working in teams, students get to know each other better, and with a job relevant project theme, a project can also help the students gradually build their disciplinary identity. Projects can also give the students a closer relationship with their teachers, as there tends to be much more feedback during the semester than is typical of lecture-and-exam courses.
- **However, projects may also be a challenge to the learning environment due to conflicts in the teams.** Hence, it is important that students are given knowledge and tools on how to handle conflicts before setting out on a huge team project. This can be addressed in many ways. As seen in Excited, the Cards of Conflict game is one interesting intervention that can be tried out here, and it has already been successfully adapted elsewhere. We hence recommend others to try this out, either with the existing cards or making their own.

Despite challenges with potential conflicts, project-based learning has been seen mainly to contribute positively to the learning environment, and most study programs should strive to have more of it.

Copyright

The Learning Environment © 2026 by Guttorm Sindre is licensed under CC BY-NC-ND 4.0



References

Abelson, S., Lipson, S. K., & Eisenberg, D. (2023). What works for improving mental health in higher education. Retrieved from Washington DC, USA:

Abualrub, I., Karseth, B., & Stensaker, B. (2013). The various understandings of learning environment in higher education and its quality implications. Quality in Higher Education, 19(1), 90–110.

Bahmani, A., & Hjelsvold, R. (2020). Work-in-progress: towards a conceptual framework for multi-campus course development. Paper presented at the 2020 IEEE Global Engineering Education Conference (EDUCON).

Bahmani, A., & Hjelsvold, R. (2024). Designing for Diversity: Applying the Conceptual Framework for Multi-campus Course Development in a Case Study. Paper presented at the Proceedings of the Future Technologies Conference.

Bahmani, A., Hjelsvold, R., & Krogstie, B. R. (2019). ICT-based challenges of repurposing a single-campus course to multi-campus settings: A pragmatic case study. Paper presented at the Conference on e-Business, e-Services and e-Society.

Dekker, I., & Theelen, H. (2025). The vanishing classroom: Attendance trends in higher education.

Keles, B., McCrae, N., & Grealish, A. (2020). A systematic review: the influence of social media on depression, anxiety and psychological distress in adolescents. International journal of adolescence and youth, 25(1), 79–93.

Li, J., Meland, P. H., Notland, J. S., Storhaug, A., & Tysse, J. H. (2023). Evaluating the impact of ChatGPT on exercises of a software security course. In 2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-6). IEEE.

Lorås, M. (2020). From teaching assistants to learning assistants–lessons learned from learning assistant training at Excited. Læring om læring, 4(1).

Lorås, M., & Aalberg, T. (2018). Informatics study day: helping CS students become better students. Paper presented at the Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education.

Lorås, M., & Aalberg, T. (2021). Creating learning environments within the constraints of higher education-A case study of a first-year computing program. Paper presented at the 2021 IEEE Global Engineering Education Conference (EDUCON).

Lorås, M., Haugset, B., & Trætteberg, H. (2021). The Importance of the Campus-A Study on the Effects of the COVID-19 Pandemic in a CS2 Course. Paper presented at the 2021 IEEE Global Engineering Education Conference (EDUCON).

Lundberg, G. M., Gaustad, A., & Krogstie, B. R. (2018). The employer perspective on employability. Paper presented at the 2018 IEEE Global Engineering Education Conference (EDUCON).

Lundberg, G. M., & Krogstie, B. R. (2020). Employability through imagination, alignment, and engagement-Students' prospects and change during their first year in computing education. Paper presented at the Proceedings of the 20th Koli Calling International Conference on Computing Education Research.

Lundberg, G. M., & Ness, I. J. (2020). First year students' imagination of future employment: Identity as an important employability aspect. Paper presented at the Proceedings of the 9th computer science education research conference.

Mozelius, P., Sällvin, L., & Johansson, J. (2026). Reshuffling the Cards of Conflict-A Game-based Approach for Dealing with Ar ficial Intelligence Related Conflicts. Paper presented at the I-LEAD 2026.

Munkvold, R. I. (2017). Game lab: a practical learning approach for game development. Paper presented at the European Conference on Games Based Learning.

Munkvold, R. I., Kolås, L., & Palmquist, A. (2023). A conflict management game in project-based learning. Paper presented at the European Conference on Games Based Learning.

L. Kolås, O. M. Lindgaard, V. M. P. Colado and R. I. Munkvold (2026). Intensive Game Development Projects as a Learning Arena for AI Literacy and Critical Thinking. Paper presented at the IEEE Global Engineering Education Conference (EDUCON), Cairo, Egypt.

Riese, E., Lorås, M., Ukrop, M., & Effenberger, T. (2021). Challenges faced by teaching assistants in computer science education across europe. Paper presented at the Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1.

Szynkiewicz, J. (2025a). Doing yourself, doing the collective, and doing the discipline: the model of computing students' identity development in project-based learning. Computer Science Education, 1–28.

Szynkiewicz, J. (2025b). Exploring the Envisioned Identity of Computing Students: Insights from Norway. ACM Transactions on Computing Education, 25(3), 1–29.

Appendix: Book chapter abstracts

1) Learning Theories and their Use in Computing Education Research

Birgit R. Krogstie

Norwegian University of Science and Technology (NTNU)

This chapter introduces learning theories and demonstrates their relevance for educators and researchers in computing education. It presents learning theories as perspectives that help explain learning phenomena and support the design, evaluation, and improvement of teaching practices. The chapter is organized around three broad orientations to learning: behaviorist, cognitive, and social. Behaviorist theories emphasize reinforcement, conditioning, motivation, and engagement, illustrated through examples such as gamification and initiatives to encourage student participation. Cognitive perspectives focus on mental processes, including information processing, cognitive load, metacognition, Bloom's taxonomy, computational thinking, threshold concepts, self-directed learning, self-regulated learning, and mastery learning. Social perspectives view learning as situated within interaction, culture, and practice, and include theories such as social learning, social constructivism, communities of practice, experiential learning, reflective learning, sociocultural theory, and activity theory. The chapter also discusses connectivism and related perspectives that acknowledge the growing role of digital technologies and artificial intelligence in distributed learning networks. Examples from computing education research illustrate how different theories can inform pedagogical design, curriculum development, and research. The chapter concludes with a set of recommendations to educators in terms of selecting a learning theory to support their teaching and research.

2) Designing Projects for Learning

Line Kolås, Helga Dís Ísfold Sigurðardóttir, Aamna Rais Ahmed

Nord University

Project-based learning (PjBL) is widely used in higher computing education because it enables students to work on authentic problems, collaborate with others, and develop competencies relevant for professional practice. However, successful project-based learning requires more than assigning students a project; it demands deliberate pedagogical design. This chapter examines how teachers can design projects that support both project outcomes and student learning.

Drawing on literature, we compare four complementary frameworks for project-based learning: CDIO, the Industry-Enriched Project-Based Learning (I-PBL) framework, the Aalborg model of problem- and project-based learning, and the Comprehensive PBL Design Model (CPBL). Using the CDIO phases Conceive, Design, Implement and Operate as an organizing lens, we discuss how the frameworks contribute different perspectives on project architecture, stakeholder involvement, student inquiry, collaboration, and cognitive and socio-emotional development.

The chapter further presents our experiences from higher computing education related to blended project-based learning and the use of external mentors. We argue that both should be understood as

pedagogical design resources that can strengthen authenticity, collaboration, reflection, and professional development when aligned with intended learning outcomes.

Throughout the chapter, planning questions are provided to support teachers in designing project-based courses. We conclude that effective PjBL depends on intentional design that integrates project processes, learning goals, stakeholders, and learning environments into a coherent whole.

3) The Dynamics of Group Work

Robin Isfold Munkvold

Nord University

Teamwork competence is increasingly recognized as a critical competence in project-based learning (PjBL), as student teams frequently experience challenges related to communication, coordination, participation, responsibility sharing, and conflicting management. There are several aspects related to these challenges and issues that arise during a semester where students work in teams may develop into interpersonal, task-related, or process-oriented conflicts that negatively affect both learning outcomes and project progression.

This chapter explores how game-based learning can be used to promote students' understanding of conflict management in project-based settings. Drawing on theories of teamwork, social interdependence, experiential learning, and conflict management, the chapter presents Cards of Conflict (CoC), a serious game developed to support teaching and learning about conflict resolution. The game is grounded in the Thomas-Kilmann Conflict Mode Instrument (TKI) and challenges students to collaboratively analyze authentic conflict scenarios and evaluate alternative response strategies.

Building on previous research that emphasizes the importance of authentic scenarios, collaborative learning, and game-based educational approaches, the chapter presents two instructional designs for implementing the game in higher education. The first focuses on facilitated gameplay using predefined scenarios, while the second engages students in the creation and evaluation of their own conflict cases.

The chapter offers practical guidance for educators seeking to strengthen teamwork competence and conflict-management skills through engaging and reflective learning activities.

4) Sustainability in Computing Education

Birgit R. Krogstie and Bashira Jaradat

Norwegian University of Science and Technology (NTNU)

Severe sustainability challenges face society. Information technology is part of almost every aspect of modern life and plays a key role for sustainability, both as part of the problem and as part of possible solutions. IT professionals need the competence to work with sustainability-related issues, and computing education must provide such competence. The practical integration of sustainability in computing education meets some challenges, including how sustainability can be made a meaningful part of courses and study programs rather than an "add-on". Another key question is how much current curricula need to change – is it sufficient with incremental updates, or is a full transformation required? This chapter presents a set of sustainability models and principles useful as a basic sustainability

vocabulary for educators and students. We then address sustainability competencies relevant for defining learning outcomes. The chapter further discusses pedagogy for teaching sustainability, in computing education and in general, drawing on examples from teaching and research. This includes the use of a framework for sustainability analysis, which originates in requirements engineering. Use of the framework can be integrated into design and software development courses, e.g. in the context of project-based learning. We further discuss issues related to the inclusion of sustainability in master theses and provide advice on how to conduct a sustainability analysis workshop.

5) The Educational Production Gap

Ole Goethe

Nord University

Team-based game projects are among the richest learning experiences in higher education, yet student teams often struggle for reasons that have little to do with talent. This chapter identifies an educational production gap in how the distance between what student teams are asked to deliver, a finished and playable game, and the production structure they are given to deliver it. It examines how student teams differ from professional studios, including unstable membership, creative ambiguity, the 80-20 polish problem, interdisciplinary friction and broken continuity across the academic calendar. In response, the chapter introduces the Agile Learning Team Framework (ALTF), a supervisor-facing model that adapts agile's core logic rather than its formal machinery. The ALTF organizes a project into three phases, Formation, Production and Professionalization, each with targeted interventions such as a team charter, MVP lock, regular sprints, a QA sprint and a structured postmortem. Grounded in scaffolding and self-determination theory, the framework treats supervision as an intervention curve, with support strongest at the start and fading as teams mature. The chapter also discusses agentic AI in student teams, outlines limitations, and offers practical guides for educators.

6) Educating for Generic Competences

Rune Hjelsvold

Norwegian University of Science and Technology (NTNU)

For more than a decade, research has shown that IT professionals need non-technical (generic) competences to succeed in their jobs. In study program design, it is important to achieve good alignment between intended learning outcomes at course-level and intended learning outcomes at programme-level important so that the course level intended learning outcomes in total address the programme-level intended learning outcomes because a programme-level intended learning outcome often will be addressed by several courses. Research done by Excited on Norwegian bachelor programmes in IT, however, suggests that generic competences are not well defined in the programme-level intended learning outcomes and may therefore not be handled in a detailed and systematic way in these study programmes.

The aim of this chapter is to provide a community of teachers involved in the design of study programmes in IT a consistent and reasonably rich foundation for good programme-course level alignment regarding generic competences. The chapter explores the realm of generic competences and

presents a core set of generic competences that are commonly recognised in research on generic competences for IT professionals. In the chapter, we discuss the specification of generic competences in intended learning outcome descriptions. We also discuss how generic competences can be progressively developed during a study programme along four different dimensions. The chapter finally presents a process for achieving good programme-course level alignment during study programme design and describes a case for how the process can be run in practice, using the outcomes from this chapter's discussions.

7) Peer Assessment in Project-Based IT Education

Trond Aalberg, Somayeh Bayat Esfandani and Mariusz Nowostawski

Norwegian University of Science and Technology (NTNU)

Peer assessment, where students give feedback on each other's work, is a well-documented learning activity. It trains critical thinking and evaluative judgement, offers formative feedback at a scale teachers cannot match, and builds the capacity to give and receive critique. In IT education it has particular relevance, since code review and other forms of peer inspection are standard practice in professional software development. This chapter gives a practical overview of peer assessment in project-based IT education. We first explain why peer assessment suits the discipline and summarize its pedagogical foundations. We then describe the peer assessment process and the features a supporting system needs, from allocation of submissions and anonymity to feedback authoring, reading of feedback and teacher oversight. We show how the artefacts and modes of assessment change as students' progress from small programs to team-based projects, and we set out recommended practice for the main design decisions: purpose, timing, who reviews whom, criteria, reviewer training, anonymity, closing the feedback loop and grading. We then discuss how artificial intelligence can support peer assessment and what teachers should consider before introducing it. Finally, three use cases from courses at NTNU illustrate how the purpose, design and experience of peer assessment vary in practice, and how students' use of generative AI is beginning to change the conditions for the activity.

8) Self-paced mastery learning

Guttorm Sindre

Norwegian University of Science and Technology (NTNU)

This chapter presents Do It Yourself (DIY) advice for readers who want to know more about self-paced mastery learning, possibly to try it out in courses that they are involved in. It starts by presenting our experiences with IT1001, an introductory programming course for STEM teacher students at the NTNU. It explains the background and principles of self-paced mastery learning and summarizes some key results from follow-up research on the course, in terms of student performance and satisfaction. The second half of the chapter outlines the key design decisions and trade-offs to be made when developing such a course, using the design rationale of IT1001 as illustration, but also pointing out alternative designs. The chapter also gives some advice on the course design process, highlighting the benefits of student involvement in course design.

9) Gamification in Higher Education - From Research to Educational Practice

Ole Goethe

Nord University

Gamification in higher education is often reduced to points, badges and leaderboards, yet research shows mixed results and stresses that design and context matter more than mechanics. This chapter argues that gamification is better understood as a spectrum of design choices, ranging from surface elements to structural features such as levels, roles, stakes and feedback. Drawing on work connected to Excited, the Centre for Excellence in IT Education, it compares three computing education cases: (1) Heimdall's Quest, an explicitly game-like system in vocational ICT, (2) Game Lab, a company role-play in game development with no points or badges, (3) and IT1001, an introductory programming course whose mastery-based levels function as gamification without being presented as such. Read through self-determination theory, the cases show that the same motivational function, such as visible progress, can take very different forms, and that what a mechanic does matters more than whether it looks like a game. The chapter offers eight design principles, a goal-first design framework and a warning-signs checklist addressing point-chasing, metric gaming, competition, exclusion, workload and platform constraints. Because the evidence is largely qualitative and context-dependent, the principles are presented as hypotheses for local testing, favoring the simplest mechanic that serves the learning goal.

10) The Learning Environment

Guttorm Sindre

Norwegian University of Science and Technology

This chapter presents various interventions and research efforts that have been conducted in Excited to understand and improve the students' learning environment, both covering social aspects, physical aspects (e.g., learning areas), and the academic learning environment. The chapter distills main findings and lessons learned and ends with some advice for others who may be interested in trying out similar interventions. A special focus within Excited has been to do research about student identity and belonging and develop interventions that contribute to an improved sense of belonging in courses and study programs, through learning activities such as groupwork and project-based learning.




Excited
Centre for Excellent IT Education
NORD University
NTNU