Teaching and assessment methods in Dutch Computer Science curricula

Researching the difference in teaching and assessment methods between technical and non-technical universities

Ellie Bleeker

Supervisors: Sole Pera, Merel Steenbergen

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfillment of the Requirements
For the Bachelor of Computer Science and Engineering
June 20, 2026

Name of the student: Ellie Bleeker
Final project course: CSE3000 Research Project
Thesis committee: Sole Pera, Merel Steenbergen, Masoud Mansoury

Abstract

Developing teamwork skills is an essential part of studying during a Computer Science Bachelor. Though it is unclear which teaching and assessment methods are used in the different types of universities throughout the Netherlands. Thus, there is a knowledge gap in understanding the differences between universities for high-school graduates and program makers. This research compares technical and non-technical universities in the Netherlands based on how they teach teamwork skills throughout their curricula. The respective study guides serve as a basis for a comparative case study where the data is analysed through deductive and inductive coding. The technical university had a stronger focus on large group projects with a more in-depth description of assessment methods. In all other aspects the universities were comparable in terms of how they incorporated teaching methods, learning goals and which kinds of courses contained teaching or applying teamwork skills. These findings can serve as the basis for more research on the differences between technical and non-technical universities and to deepen the constructive alignment of Computer Science curricula's descriptions.

1 Introduction

Computer Science (CS) graduates entering the workforce need teamwork skills, as they rarely work individually. However, it is not a given that the relevant experience is taught during their education [1].

According to Craig et al. [6] there is an expectation gap between what soft skills universities focus on and what the industry expects from graduates. Universities have designated learning goals for teamwork, since most CS careers are team-based [1], which are accompanied by certain teaching and assessment methods. This earlier research finds multiple angles and dimensions in which graduates are not well prepared, but it remains unclear which exact teaching strategies are applied and how students are assessed on their teamwork skills. This knowledge gap applies to Dutch universities as well and can be extended to the differences between different types of universities.

Within the Netherlands, there are technical and non-technical universities. Research and legal documents on what sets these types apart is limited. The vision or mission statements of the four technical universities have in common that they mention a focus on educating students in the field of science & engineering with a strong focus on technologies [7, 17, 18, 20]. How this vision translates to differences in constructive alignment, the mapping of learning goals to teaching and assessment methods, has not yet been researched [5].

The identified research gap is the lacking research on the differences between technical and non-technical universities in regards to the methods used to teach and assess teamwork. The research will focus on the following research question:

How do the Computer Science bachelors of technical and non-technical universities in the Netherlands compare in teaching and assessment methods related to teamwork throughout the curricula?

- RQ1: Which courses explicitly incorporate teaching or applying teamwork skills?
- RQ2: What teaching and assessment methods are used to fulfil teamwork learning goals or teach teamwork skills?
- RQ3: To what extent do the universities differ in grounding their constructive alignment on teamwork skills in science?

The scientific contribution of this research is focused on gaining a better understanding on the differences between technical and non-technical universities their CS bachelors in terms of applied teaching and assessment methods. This gained knowledge could eventually be used by program makers to further improve their curricula or by students to make a more informed choice when deciding on which university to study at.

2 Related work

According to P. Laredo [11] universities have 3 important functions of which one is providing mass tertiary education (teaching bachelors). Since 70% of students stop studying after finishing their Bachelor, it is important for them to have a professional skills basis. Hewner & Guzdial [9] concluded that one of the most essential qualifications for employment was being able to work with others and check your own ego at the door.

Craig et al. [6] found six dimensions in which Computer Science studies lacked in providing a good basis for graduates. Adequate teamwork skills expected by industry do not align with the teamwork skills taught in universities. One dimension is the focus on smaller group projects in universities, whereas the industry expects graduates to function in large teams. This mismatch in expectations leads to graduates not obtaining the required skill throughout their studies.

Typically, CS students participate in multiple group projects during their studies with differing setups and grading strategies [1]. Most of these group projects are within software engineering courses, since they usually include a software development life cycle [4]. Interviews conducted by Aivaloglou et al. [1] show that almost all students encountered group projects throughout their studies, with varying experiences. The conclusion of the paper is that a variety of strategies for assignment setup, assessment methods and monitoring team contributions are used in CS courses. Regardless of the approach, the interviewees state the importance of having transparent and adequate methods to tailor the specific course or assignment.

Teamwork skills and what methods are used to teach them can be divided in categories. Steenbergen et al. researched how early-career computer scientists reflected on the certain teamwork skills and to what extent they were taught during their studies [15]. Most teamwork skills are taught implicitly, as a side effect of other learning objectives. The most common way of acquiring teamwork skills is through group projects. However, a teaching method that does explicitly

teach adjacent skills in *Scrum teaching* [15]. Other less common teaching methods that are used within universities are teaching students how to use Version Control Software in a team setting or do pair-programming [1, 2, 13].

The assessment methods referenced in literature for teamwork skills are closely related to the assessment for group projects. The most common assessment methods for group projects can be split in individual grading, group grading, peer grading or a combination of peer grading with either group or individual grading [1]. Peer grading mainly focusses on evaluating soft skills (like teamwork) [10], however trends in self-assessment show a certain level of inaccuracy. A. Tafliovich et al. [3] researched student preferences for assessment methods in group projects. They concluded that the preference is determined by the year students are in, the relation to their peers and how their peers perform.

The way students are assessed in group settings matters, because it could influence the extend to which social loafing exhibits itself in a project [8]. The negative effect of social loafing is a loss of productivity and students not obtaining the skills that are ought to be learned in a group project. Methods to reduce this phenomenon are using peer evaluations, providing individuals with feedback or increasing social cohesiveness.

Another lens to view teaching and assessment methods through, is the extent to which constructive alignment was applied. Constructive alignment is the process of identifying objectives, choosing and organising the learning experiences and then evaluating the results of learning, as described by A. Cain et al. [5]. For CS curricula, this commonly means evaluating to what extend teaching and assessment methods for teamwork skills are supported by learning objectives. The application of constructive alignment leads to better results [5] and therefore is relevant to how universities teach teamwork skills in their programmes.

3 Methodology

To answer how teamwork teaching and assessment methods in CS curricula differ between technical and non-technical universities, we performed a quantitive case study. Analysing the differences between specific universities has been done with a deductive and inductive coding strategy for available study guide data of the CS curricula. This chapter further explains the methods and choices that were made to conduct the research.

3.1 Setup

The universities that we selected are Leiden University (LU) and the Technical University of Eindhoven (TUe). Both universities are Dutch and have comparable Computer Science bachelors with adequate available study guide data. The motivation for choosing these universities is based on scope and representation. LU had a broad spectrum of bachelors whereas TUE had an Engineering focused set of bachelors. This allowed both universities to be representative for the non-technical and technical university categories [19, 16].

Coding of data happened inductively and deductively. The goal of coding the data was to create an understanding of the teaching and assessment methods that are used within the courses and to what extend they have learning objectives. Deductive codes were created based on the literature in the Related Works 2. After collecting the data and analysing it with the deductive codes, we created inductive codes to analyse common patterns which could not be adequately analysed with the deductive codes. The inductive codes are described in the relevant Results 4 sections.

Data collection was done by importing the study guide contents of the CS courses from both universities. The data was stored in a .xlsx file which separated the course contents on: University, Title, Description, Course objectives, Assessment method, Data source and possible remarks.

Data processing mainly consisted on assigning codes to the collected data. It started by applying the deductive codes to the data. This was done with the program Altas.ti, in which the courses were loaded split on university through .docx files.

Data analysis primarily focused on answering the research questions. For RQ1 both universities were compared in terms of how many courses contained elements of teaching teamwork skills. The data for RQ2 was analysed by identifying which teaching and assessment methods were actually used in courses and to what extent it occurred throughout the Bachelors. Last, RQ3 its data was analysed by using inductive codes to group learning goals and comparing the setup of group projects in both universities.

3.2 Deductive coding

Based on [1, 2, 3, 6, 9, 13, 15] the deductive codes were created for the study guide data analysis. The codes have been subdivided in categories.

Teaching methods: 'Git teaching', 'Group project', 'Pair programming' and 'Scrum teaching'.
Assessment methods: 'Git history', 'Group assessment', 'Individual assessment' and 'Peer review'.
Teamwork skills: 'Assisting', 'Communication' and 'Coordination/Leadership'.
Learning goals: 'Applying teamwork skills', 'Learning teamwork skills' and 'Participating in group projects'.
Course setup: 'Does not have teamwork' and 'Has teamwork'.

4 Results

The deductive and inductive analysis of the Leiden University's and Technical University of Eindhoven's Computer Science curricula are presented in the following subsections. Each chapter starts with the deductive coding results, being followed by the inductive coding results if present. Beside this, certain coding choices are explained to clarify how the results were obtained through the qualitative analysis. Text that was translated from Dutch to English is marked with a * for clarity.

4.1 Teamwork learning goals in courses

In Table 1, a comparison of how many courses did and did not contain teamwork as a part of the study guide can be seen.

LU	Count	Percentage of teamwork
Contains teamwork	12	35%
Lacks teamwork	22	65%

TUe	Count	Percentage of teamwork
Contains teamwork	9	31%
Lacks teamwork	20	69%

Table 1: Teamwork in courses

A course was considered to have teamwork when it included applying or teaching teamwork skills. Both LU and TUe their mandatory and CS elective courses were considered. Broader elective and minor courses were not considered, since they were not an integral part of either CS curricula. This resulted in 180ECTS from LU their courses and 150ECTS from the TUe their courses being used.

In total, 64 courses were analysed and coded. To better understand which kinds of courses contain aspects of teamwork, course categories were created. The categories are based on the categorization done by the universities themselves and merged to have alike courses grouped together. In Appendix A the assigned course categories can be reviewed. The categories that were used are: 'Mathematics', 'System/Hardware', 'Programming/Software development', 'Algorithmic & Compute theories', 'Data/AI', 'Research/Skills' and 'Security'.

In Table 2 the outliers were Mathematics courses, which did not contain any teamwork according to the study guide. The Research/Skills and Programming/Software development categories contained the most courses with teamwork for LU.

LU - Course category	Has	Total	Percentage
Mathematics	0	7	0%
Systems/Hardware	1	3	33%
Programming / Software Development	4	7	57%
Algorithmic & Compute theories	1	5	20%
Data/AI	2	6	33%
Research/skills	4	5	80%
Security	0	1	0%

Table 2: Teamwork per category for Leiden University

In Table 3 the category with the most Teamwork were Programming/Software development, while most other categories barely contained any. The TUe had courses 'Scop/e' that specifically focussed on developing professional. Though, it these courses did not have a study guide page with learning goals, resulting in them being marked as not containing teamwork elements.

In Table 4 LU and TUe their teamwork incorporation were compared for easier comparison.

TUe - Course category	Has	Total	Percentage
Mathematics	1	4	25%
Systems/Hardware	1	4	25%
Programming / Software Development	5	7	71%
Algorithmic & Compute theories	0	5	0%
Data/AI	0	3	0%
Research/skills	2	5	40%
Security	0	1	0%

Table 3: Teamwork per category for Technical University of Eindhoven

Course category	TUe	LU
Mathematics	25%	0%
Systems/Hardware	25%	33%
Programming / Software Development	71%	57%
Algorithmic & Compute theories	0%	20%
Data/AI	0%	33%
Research/skills	40%	80%
Security	0%	0%

Table 4: Teamwork comparison based on course categories

4.2 Applied teaching and assessment methods

Type	Codes	LU codes	LU courses	TUe	TUe courses
Assessment Method	Git history	0	0	0	0
Assessment Method	Group assessment	3	3	7	7
Assessment Method	Individual assessment	0	0	3	3
Assessment Method	Peer review	0	0	3	3
Teaching Method	Git teaching	2	2	2	2
Teaching Method	Group project	10	8	8	8
Teaching Method	Pair programming	0	0	0	0
Teaching Method	Scrum teaching	2	2	1	1

Table 5: Teaching and assessment methods from deductive coding

Group projects were the most common teaching method in both universities. They were present in 8 courses for both universities, since LU had one course with three explicit group projects. Group projects that could either be done individually or as a pair were also accounted for as a group project. The difference in group project setup between LU and TUe was mostly in group size, four out of eight courses in LU only had pair projects whereas in the TUe all projects consisted of more than two students.

Git teaching happened in 2 courses with each university. In both LU and the TUe Git was taught in a course in the first year and in the third year it came back as a learning goal which focussed more on the usage of Git within a group setting.

Scrum teaching appeared in 2 normal courses in LU and only occurred once in the TUe during the third year 'Software Project' course.

Assessment goals were most prevalent in the TUe's study guide in which multiple courses had a combination of group, individual and peer assessment. Peer and individual assessment were always combined with group assessment and might include all three of the options. LU's study guide provided less insight in the type of assessment, since 'group assessment' was only found thrice explicitly.

4.3 Differences between universities

The following results and highlighted differences were based on analysing the deductive and inductive codes between LU and TUe.

Group size of 4 out of 8 projects in LU were completed in pairs, whereas in the TUe 3 out of 9 projects required groups of more than 6 students. The larger group projects were present in the courses 'CBL Autonomous Systems Twinning' (year 1), 'Multidisciplinary CBL' (year 2) and 'Software Engineering Project' (year 3). The focus on the importance of teamwork was explicitly emphasized in the TUe's study guide from the course 'Impact of technology engineering ethics':

"Future engineers will work in multidisciplinary teams – whether they work in high-tech startups, large established companies, government agencies, or research institutions. They will be expected to cooperate and combine their technological expertise with ethical skills in order to realize successful and responsible innovations that improve society while at the same time addressing ethical concerns about new technologies."

Constructive alignment was considered to be present in a course, when applying or teaching teamwork skills was combined with learning objectives or further explanation on its relevance. Table 6 shows that both universities had 5 courses containing a rationale about teaching teamwork skills. The five referenced courses in LU were: 'Studying and Presenting', 'Oriëntatie Informatica' (Orientational Computer Science)*, 'Research Methods in Computer Science', 'Bachelor Thesis Project' and 'Software Engineering'. For the TUe these were: 'ITEC - Ethics of technology and engineering', 'Software Design', 'Autonomous Systems Twinning', 'Multidisciplinary CBL' and 'Software Engineering project'.

Teamwork & constructive aligment	LU	Tue
No teamwork, No constructive aligment	22	20
Teamwork, No constructive aligment	7	4
Teamwork with constructive alignment	5	5

Table 6: Constructive alignment in courses

During the deductive coding of the data, multiple learning goals concerning the application and learning of teamwork skills were found. To further explore the difference in learning goals, inductive codes were created to differentiate their intent. The inductive can be seen in Table 7 and are based on grouping learning goals together on similar words or objectives. The coded learning goals come from the courses that have some form of constructive alignment.

Three learning goals from LU are:

1. *"Completing this project requires more than just programming skills, as the team is also in charge of project management, version management, client communication, collaborations within LUdev (involving students from other courses) and product deployment."*

Inductive codes	LU	TUe
Giving or receiving feedback, like peer-reviewing	1	3
Project management: for example scrum or agile management	3	3
Participate in group work	5	6
Communicate with peers	0	1
Tools for development in teams, like VCS	3	2
Group presentations	1	1

Table 7: Inductive coding of teamwork related learning goals (LU=10, TUe=10)

2. *"Practical experience with project management and team-based software development."*
3. *"Interpersonal skills necessary for collaboration in a team, including the ability to give, receive and process feedback."*

TUe their similar learning goals are:

1. *"Apply personal management skills such as planning and organizing own activities in a project."*
2. *"Use tools for software development and team collaboration such as version control systems, continuous integration/development/testing and issue tracking systems."*
3. *"Develop professional skills like technical writing and presenting, group work, peer-reviewing, and project planning and management."*

5 Responsible Research

Reproducibility. The research relies on deductive and inductive coding to determine if certain aspects are present within the study guide. The methodology includes the used deductive codes and the data coding has been done in a structured and careful manner. Therefore the first part of the coding should be reproducible, though it should be noted that the subjectivity of the researcher could impact the outcome. The inductive coding is focused on the learning goals, since deductive codes were deemed insufficient. Another researcher could, given the codes, reproduce this part.

Study guides from both universities serve as the foundation for the research. All data that was collected came from open-sources and therefore is accessible to everyone.

An important note in reflecting on the reproducibility of this research and its findings, is in the difference between static data and what happens in real-life. Research conducted within the classrooms and based on contact with actual program makers could give different results, since the study guides do not necessarily include everything a course entails. The lack of assessment methods within LU's study guide serves as a good example for this.

Generalizability. Only Leiden University and the Technical University of Eindhoven were considered within this research. Since the Netherlands houses fourteen universities [14] it is important to keep in mind that the results of this research may not be entirely representable to all Dutch universities. Though general conclusions may apply since all universities are within the same country and culture.

The amount of analysed courses also matter in terms of generalizability. For LU 34 courses were considered with a

total of 180ECTS as opposed to TUe's 30 courses (of which one had no study guide data) counting 150ECTS. Every considered course is a core part of the universities' curriculum so to that extend they are representative.

Another concern is to what extend there actually is a difference between technical and non-technical universities within the Netherlands. No prior research was found on this subject within the Netherlands and therefore the upfront differences are unclear, which could impact the outcomes of this paper.

Ethical implications. The data and research does not include personal or sensitive data points. The choice to keep out references to people, which could have happened in analysing course specific study guide data, was made on purpose to not potentially harm people with the results of this research.

AI usage. No GenAI or other forms of Artificial Intelligence were used during this entire research process. Research by Lee et al. [12] in cooperation with researchers at Microsoft has shown that GenAI negatively impacts workers and conflicts critical thinking processes. As critical learning and experiencing academic paper writing was part of the learning objectives for the CSE3000 Research Project, it was decided not to utilize this technology.

6 Discussion

RQ1: "Which courses explicitly incorporate teaching or applying teamwork skills?" The types of courses that predominantly integrate teamwork as part of their content are in the category *Programming/Software development*. This makes sense since courses with a focus on software development often include a team-based development cycle to reflect the professional software engineering setting [4]. 80% of LU their *Research/Skills* courses integrated teaching teamwork. Most of these courses are preparatory for gaining academic skills and developing students as researchers, so most teamwork skills are likely taught with this context. These skills useful in academia, but are probably not sought after by industry positions [6].

RQ2: "What teaching and assessment methods are used to fulfil teamwork learning goals or teach teamwork skills?" *Group projects* were the most frequent used teaching method, though in LU the projects were in smaller groups whereas TUe focussed on group sizes better reflecting industry standards [6]. With this, the TUe better prepares students for industry like group-sizes. Smaller group sizes could be more relevant to continuing a career in academia. The other teaching methods that were found in both universities, teaching Git and Scrum, are sought after in most group-projects and useful to get experience with. The impact of this difference is that students, depending on which university they choose, graduate with different experiences in regards to team-based projects. Depending on what career path they want to take, it matters which type of university they choose to study at.

LU did not have elaborate explanation on assessment methods, whereas the TUe had more defined assessment methods and more variety forms where *Group assessment* was more often combined with *Peer reviews* or *Individual assessments*. More actively defining the used assessment methods is positive on the students their experience within a group project [3]. Next to positively affecting individual experiences, better assessment methods contribute to less social loafing [8].

RQ3: "To what extent do the universities differ in grounding their constructive alignment on teamwork skills in science?" The first identified difference was located within the group sizes and how the TUe their larger groups better resembled what will be expected from graduates. The larger group projects were spread over the three years of the curriculum ensuring students encounter industry like projects in every academic year. In Leiden this focus on structured large-group projects was missing in the curriculum, possibly not preparing students well enough for professional positions after graduation [6, 11].

The following difference in constructive alignment is to what extend courses that teach or apply teamwork skills had a foundation for doing such through explaining relevance or introducing learning goals. Based on the inductive coding, the TUe had a few more relevant learning goals and relatively applied it in more courses with a teamwork element. A reason why TUe had more codes in regards to learning goals, is likely because of the three team-based courses they have. Consolidating teaching teamwork skills more in actual methods leads to better results in students their learning experience [5].

The formulation and contents of learning goals are relatively similar. Based on the the results in Section 4.3, the learning goals describe learning how to give and receive feedback, do project management, project planning, collaborate with VCS (version control software) and be involved in team-base software development in similar manners. This signals that the courses in which constructive alignment was found, had a similar foundation of learning goals. Though in Leiden 7 out of 13 teamwork courses did not have learning objectives to support teaching teamwork skills, which likely decreases the effectiveness.

Limitations. Only two universities were compared based on the available study guide data. Thus it should be noted that the conclusions are based on static data. There is a chance that using different research methods on this matter, for example in-classroom research, could produce different conclusions. Such results would expose the differences between written methods and their execution, which in itself would be an impactful revelation.

Another aspect with limitations is the general difference between technical and non-technical universities in the Netherlands. With prior research missing, there could be different outcomes or insights to the extend in which these types differ. The differences between universities could be expressed through other demarcation lines which could be more applicable to the Dutch university ecosystem.

7 Conclusions and Future Work

The most prevalent difference between how technical and non-technical universities teach teamwork skills is the difference in group project setup. Technical universities prepare students better for industry positions after graduation, whereas non-technical universities have a stronger focus on

teaching teamwork skills that are likely relevant to an academic career. The implication is that students choose a different teamwork skill set when they decide to study CS at either a technical university or a non-technical university. Different career paths require different skill sets so this should be a rational decision, otherwise students may end-up without the teamwork skills they need. Another aspect regarding both university types, is that roughly half of the courses with teamwork elements did not contain any form of constructive alignment. A lack of constructive alignment or having a structured setup may lead to social loafing [8] and other types of unwanted behaviour. This affects how students experience teamwork and to what extend some develop behaviours which negatively impact the group.

Based on this research, program makers could evaluate on the process they go through when they create or evaluate a CS curriculum and to what extend certain skills have a foundation in constructive alignment. Future research should focus on two aspects. The first is researching into the general differences between technical and non-technical universities in the Netherlands and evaluate how this demarcation reflects on other possible categorizations. The second is gaining more knowledge on how study guide information translates to actual methods being used in courses and to what extend learning objectives regarding skill teaching are accomplished.

References

[1] Efthimia Aivaloglou and Anna van der Meulen. "An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments". In: 21.September 2021 (2021). P001, pp. 1–22. DOI: 10.1145/3440994. URL: https://dl.acm.org/doi/10.1145/3440994.

[2] Cristina Adriana Alexandru. "Group Assignments and Support Aimed to Develop Student Teamwork Skills and a Positive Attitude Towards Teamwork in Computer Science Higher Education". en. In: *Proceedings of the 9th Conference on Computing Education Practice*. P028. Durham United Kingdom: ACM, Jan. 2025, pp. 9–12. ISBN: 979-8-4007-1172-5. DOI: 10.1145/3702212.3702215. URL: https://dl.acm.org/doi/10.1145/3702212.3702215 (visited on 05/13/2026).

[3] Anya Tafliovich, Andrew Petersen, and Jennifer Campbell. "Evaluating Student Teams: Do Educators Know What Students Think?" In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. SIGCSE '16. P018. New York, NY, USA: Association for Computing Machinery, 2016, pp. 181–186. ISBN: 978-1-4503-3685-7. DOI: 10.1145/2839509.2844647. URL: https://dl.acm.org/doi/10.1145/2839509.2844647.

[4] Kevin Buffardi et al. "Chronicling Consistency and Parity of Contributions to Software Engineering Team Projects". en. In: *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. P033. Nijmegen Netherlands: ACM, June 2025, pp. 562–568. ISBN: 979-8-4007-1567-9. DOI: 10.1145/3724363.3729039. URL: https://dl.acm.org/doi/10.1145/3724363.3729039 (visited on 06/17/2026).

[5] Andrew Cain and Muhammad Ali Babar. "Reflections on applying constructive alignment with formative feedback for teaching introductory programming and software architecture". en. In: *Proceedings of the 38th International Conference on Software Engineering Companion*. P030. Austin Texas: ACM, May 2016, pp. 336–345. ISBN: 978-1-4503-4205-6. DOI: 10.1145/2889160.2889185. URL: https://dl.acm.org/doi/10.1145/2889160.2889185 (visited on 06/15/2026).

[6] Michelle Craig et al. "LISTENING TO EARLY CAREER SOFTWARE DEVELOPERS". In: 33 (2018). P002, pp. 138–149. ISSN: 1937-4771. URL: https://dl.acm.org/doi/10.5555/3199572.3199591.

[7] TU Delft. *Visie, missie en waarden van de TU Delft*. URL: https://www.tudelft.nl/over-tu-delft/strategie/visie-missie-en-waarden-van-de-tu-delft (visited on 04/30/2026).

[8] Ilenia Fronza and Xiaofeng Wang. "Towards an Approach to Prevent Social Loafing in Software Development Teams". In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. P032. Toronto, ON: IEEE, Nov. 2017, pp. 241–246. ISBN: 978-1-5090-4039-1. DOI: 10.1109/ESEM.2017.37. URL: http://ieeexplore.ieee.org/document/8170110/ (visited on 06/16/2026).

[9] Michael Hewner and Mark Guzdial. "What game developers look for in a new graduate: interviews and surveys at one game company". en. In: *Proceedings of the 41st ACM technical symposium on Computer science education*. P025. Milwaukee Wisconsin USA: ACM, Mar. 2010, pp. 275–279. ISBN: 978-1-4503-0006-3. DOI: 10.1145/1734263.1734359. URL: https://dl.acm.org/doi/10.1145/1734263.1734359 (visited on 05/12/2026).

[10] Kevin Buffardi. "Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. P013. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 650–656. ISBN: 978-1-4503-6793-6. DOI: 10.1145/3328778.3366948.

[11] Philippe Laredo. "Revisiting the Third Mission of Universities: Toward a Renewed Categorization of University Activities?" en. In: *Higher Education Policy* 20.4 (Dec. 2007). P024, pp. 441–456. ISSN: 0952-8733, 1740-3863. DOI: 10.1057/palgrave.hep.8300169. URL: http://link.springer.com/10.1057/palgrave.hep.8300169 (visited on 05/12/2026).

[12] Hao-Ping (Hank) Lee et al. "The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers". en. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. P031. Yokohama Japan: ACM, Apr. 2025, pp. 1–22. ISBN: 979-8-4007-1394-1. DOI: 10.1145/

3706598.3713778. URL: https://dl.acm.org/doi/10.1145/3706598.3713778 (visited on 06/16/2026).

[13] Robert Lingard and Shan Barkataki. “Teaching teamwork in engineering and computer science”. In: *2011 Frontiers in Education Conference (FIE)*. P023. Rapid City, SD, USA: IEEE, Oct. 2011, F1C–1–F1C–5. ISBN: 978-1-61284-469-5 978-1-61284-468-8 978-1-61284-467-1. DOI: 10.1109/FIE.2011.6143000. URL: http://ieeexplore.ieee.org/document/6143000/ (visited on 05/10/2026).

[14] Nationale onderwijsgids. *Hoeveel universiteiten in Nederland zijn er?* URL: https://www.nationaleonderwijsgids.nl/universiteit/hoeveel-universiteiten-in-nederland-zijn-er/ (visited on 06/10/2026).

[15] Merel Steenbergen. “Practicing Collaboration with Student Programming Projects”. en. In: *Proceedings of the 25th Koli Calling International Conference on Computing Education Research*. P022. Koli Finland: ACM, Nov. 2025, pp. 1–3. ISBN: 979-8-4007-1599-0. DOI: 10.1145/3769994.3770048. URL: https://dl.acm.org/doi/10.1145/3769994.3770048.

[16] TU/e. *Bacheloropleidingen*. URL: https://www.tue.nl/studeren/alle-opleidingen/bacheloropleidingen (visited on 04/30/2026).

[17] TU/e. *TU/e Strategy 2030*. URL: https://assets.w3.tue.nl/w/fileadmin/content/Our_University/About%20our%20university/Publications/TUE_Strategie_2030-LR.pdf?_gl=1*l7g3pz*_gcl_au*MTk5NjEwNzE4Ny4xNzc2OTQ5ODYy (visited on 04/30/2026).

[18] University of Twente. *SHAPING2030 - MISSION, VISION & STRATEGY*. URL: https://www.utwente.nl/.utwente_base/ws2016/download.shtmlf=shI6p7uGOy5WQbqXeAMm9upNT7iP8PtKMdVWPgU7mka4RyYQnO70_e7zQwLTLmqtuFATCGjTClOKemKFOXDtbBaRwkFc_v9TAK5RWw1L9wrP_0CSVqAfKoNuvao72DJ47SS9Yz3BtF5D39MMKYIiaExBtti4LUj_P7EucnmlE_tuSi7SUTPhv-9tjE73SRw1CFwq-rOq8cPKK6lRM1MAYqh7LNAktckw6CwwVEwVXrSAegC4Ba3XElY7l0iIgU-qpQhAA61PuIqEdMwTcNQbBZ0.m_5_5eVgdJoOlDMX.sq--qAzaHqkucJmxKQQZFQ (visited on 04/30/2026).

[19] Leiden University. *Bachelors*. URL: https://www.universiteitleiden.nl/onderwijs/bachelors (visited on 04/30/2026).

[20] WUR. *Shape responsible change - Strategic Plan 2025-2028*. URL: https://backend.wur.nl/sites/default/files/2026-03/WUR-strategic-plan-2026-UK.pdf (visited on 04/30/2026).

A Assigned course categories

University	Course name	Study year	Quarter	Has teamwork	My Categories
LU	Calculus 1	Y1	Q1	No	Mathematics
LU	Studying and Presenting	Y1	Q1	Yes	Research/skills
LU	Foundations of Computer Science	Y1	Q1-Q2	Yes	Algorithmic & Compute theories
LU	Fundamentals of Digital Systems Design	Y1	Q1-Q2	No	Systems/Hardware
LU	Programmeermethoden	Y1	Q1-Q2	No	Programming / Software Development
LU	Oriëntatie Informatica	Y1	Q1-Q2	Yes	Research/skills
LU	Linear Algebra for Computer Scientists 1	Y1	Q2	No	Mathematics
LU	Algoritmiek	Y1	Q3-Q4	No	Programming / Software Development
LU	Databases	Y1	Q3-Q4	Yes	Data/AI
LU	Programmeertechnieken	Y1	Q3-Q4	Yes	Programming / Software Development
LU	Logic 1	Y1	Q3	No	Mathematics
LU	Calculus 2	Y1	Q4	No	Mathematics
LU	Automaten	Y1	Q4	No	Algorithmic & Compute theories
LU	Probability Theory for Computer Scientists	Y1	Q4	No	Mathematics
LU	Formele Talen en Berekenbaarheid	Y2	Q1-Q2	No	Algorithmic & Compute theories
LU	Computerarchitectuur	Y2	Q1-Q2	Yes	Systems/Hardware
LU	Concepts of Programming Languages	Y2	Q1-Q2	Yes	Programming / Software Development
LU	Datastructuren	Y2	Q1-Q2	No	Programming / Software Development
LU	Statistics for Computer Scientists	Y2	Q1-Q2	No	Mathematics
LU	Artificial Intelligence	Y2	Q3-Q4	Yes	Data/AI
LU	Complexiteit	Y2	Q3-Q4	No	Algorithmic & Compute theories
LU	Operating Systems and Networks	Y2	Q3-Q4	No	Systems/Hardware
LU	Security	Y2	Q3-Q4	No	Security
LU	Research Methods in Computer Science	Y2	Q1-Q4	Yes	Research/skills
LU	Machine Learning	Y2	Q3-Q4	No	Data/AI
LU	Logic 2	Y2	Q4	No	Mathematics
LU	AI & Ethics	Y3	Q3	No	Research/skills
LU	Bachelor Thesis Project	Y3	Q2-Q4	Yes	Research/skills
LU	Software Engineering	Y3	Q3-Q4	Yes	Programming / Software Development
LU	Applied Data Science and Explainable AI	Y3	Q3-Q4	No	Data/AI
LU	Introduction to Reinforcement Learning	Y3	Q3-Q4	No	Data/AI
LU	Program Correctness	Y3	Q3-Q4	No	Algorithmic & Compute theories
LU	Requirements Engineering	Y3	Q3-Q4	Yes	Programming / Software Development
LU	Neural Computing	Y3	Q3-Q4	No	Data/AI

Table 8: Leiden university course categories

University	Course name	Study year	Quarter	Has teamwork	My Categories
TUE	Study Career and Orientation (SCOP/e Computer science)	Y1	Q1-Q4	No	Research/skills
TUE	Calculus Variant 2	Y1	Q1	No	Mathematics
TUE	Programming	Y1	Q1	No	Programming / Software Development
TUE	Logic and Set Theory	Y1	Q1	No	Mathematics
TUE	Linear Algebra and Applications	Y1	Q2	Yes	Mathematics
TUE	Discrete Structures	Y1	Q2	No	Algorithmic & Compute theories
TUE	Foundations of Data Analytics	Y1	Q2	No	Data/AI
TUE	Probability and Statistics for CS	Y1	Q3	No	Mathematics
TUE	Data Structures	Y1	Q3	No	Algorithmic & Compute theories
TUE	Computer Systems	Y1	Q3	No	Systems/Hardware
TUE	ITEC - Ethics of technology and engineering	Y1	Q4	Yes	Research/skills
TUE	Software Design	Y1	Q4	Yes	Programming / Software Development
TUE	Autonomous Systems Twinning	Y1	Q4	Yes	Programming / Software Development
TUE	Study and Career Orientation (SCOP/e Computer science)	Y2	Q1-Q4	No	Research/skills
TUE	Software Specification	Y2	Q1	No	Programming / Software Development
TUE	Networks	Y2	Q1	No	Systems/Hardware
TUE	Datamodeling and Databases	Y2	Q2	No	Data/AI
TUE	Automata and Formal Languages	Y2	Q2	No	Algorithmic & Compute theories
TUE	Operating Systems	Y2	Q3	No	Systems/Hardware
TUE	Algorithms	Y2	Q3	No	Algorithmic & Compute theories
TUE	Multidisciplinary CBL	Y2	Q4	Yes	Programming / Software Development
TUE	Security	Y2	Q4	No	Security
TUE	Statistics and Machine Learning	Y2	Q4	No	Data/AI
TUE	Study and Career Orientation (SCOP/e Computer science)	Y3	Q1-Q4	No	Research/skills
TUE	Software Engineering project	Y3	Q1-Q2	Yes	Programming / Software Development
TUE	Bachelor Final Project (BEP	Y3	Q3-Q4	No	Research/skills
TUE	Capstone Heuristic Algorithms	Y3	Q1	No	Algorithmic & Compute theories
TUE	Capstone Systems Integration	Y3	Q1	Yes	Systems/Hardware
TUE	Capstone Software Development	Y3	Q1	Yes	Programming / Software Development
TUE	TEC - Impact of technology: Engineering for Society	Y3	Q3	Yes	Research/skills

Table 9: Technical University of Eindhoven course categories