A Method for Describing Declared Teamwork Instruction in a Computer Science Bachelor Curriculum

With a Single-Case Demonstration

Mohammed Shomis[1]

Supervisor(s): Dr. Sole Pera[1]**, Merel Steenbergen**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Mohammed Shomis
Final project course: CSE3000 Research Project
Thesis committee: Dr. Sole Pera, Merel Steenbergen, Dr. Masoud Mansoury

Abstract

Computer science (CS) programmes are expected to develop students' teamwork skills, yet there is no established, repeatable way to check whether a programme actually declares teamwork instruction across its full mandatory curriculum rather than leaving it to one or two courses. This paper contributes such a method. The method takes a programme's publicly declared course syllabi, codes them against a teamwork framework that spans professional dispositions, non-technical skills, and collaborative teaching methods, and aggregates the course-level coding into a programme-level picture: which teamwork categories the curriculum declares, which it omits, and what it declares that the framework does not anticipate. The method is demonstrated on one CS bachelor programme. Doing so reveals patterns that a single course could not show: teamwork appears in only a few of the 25 mandatory courses, the final capstone project is individual rather than team-based, and an identical ethics paragraph is copied into every course whether or not it involves group work. This is evidence that the method captures how a whole curriculum does — or does not — build teamwork. The paper provides the method, an operationalised codebook, and a worked single-case application that other programmes can replicate.

1 Introduction

Computer science (CS) programmes are formally expected to develop teamwork skills [1, 5, 12, 15]. Yet in practice, industry reports that CS graduates often lack the teamwork skills they need at work [6].

A reported skill gap of this kind could originate at one of three levels. It could lie in industry's own expectations and hiring, where the point of intervention lies with employers. It could lie in the transition from what students are taught to what they carry into practice — the enacted curriculum (what happens in class) and the achieved curriculum (what students retain) — where the relevant factor is how instruction is delivered and assessed. Or it could lie in what programmes formally set out to teach: the declared curriculum. Each is a plausible origin, and diagnosing which one is at issue requires evidence at that level.

For the declared curriculum, that evidence is difficult to obtain, because there is no established way to gather it. Programmes are formally expected to develop teamwork [5, 15], but whether a programme declares it across the curriculum cannot readily be determined. Teamwork is widely treated as a capability built across a whole programme through repeated practice rather than acquired in a single course, so a course-by-course inspection is insufficient; a curriculum-level view is needed of where teamwork is declared, practised, and assessed. Existing work does not provide one. Studies examine individual courses or single capstone projects, or — like Groeneveld et al. [9] — sample non-technical courses across many programmes at the *course* level; none offers a repeatable way to describe how teamwork is declared across a single programme's full mandatory curriculum.

This is a methodological gap as much as an empirical one. This paper therefore asks: *how is teamwork represented across a CS bachelor curriculum's full mandatory load?* To answer this in a repeatable, programme-level way, it develops a reusable method — the paper's main contribution — and demonstrates it on one programme. The question breaks into three sub-questions: (SQ1) which categories of teamwork instruction the curriculum declares; (SQ2) which categories are absent; and (SQ3) which declared content current categorisations do not capture.

The main contribution is the method itself — a data requirement, a coding instrument, and an aggregation procedure that take a programme's declared syllabi to a programme-level account of its teamwork instruction. A secondary contribution is the demonstration: applying the method to one CS bachelor programme (KTH's Datateknik) yields a programme-level description and shows that the method surfaces curriculum-level patterns that a course-level study would miss.

2 Background and Related Work

The work this paper builds on falls into three groups: studies of what industry expects from CS graduates, studies of how teamwork should be taught, and studies of what programmes actually do.

Garcia et al. [6] provide the most recent comprehensive evidence of the gap between what industry expects from CS graduates in non-technical skills and what programmes currently deliver, reviewing 48 academic studies and surveying 124 industry professionals. Teamwork is among the skills they identify as expected by industry yet under-developed in academic preparation. The same work also offers an integrated framework that describes teamwork-related preparation through three lenses — professional dispositions, non-technical skills, and collaborative teaching methods. This study adopts that framework as its coding vocabulary; it is used here as the *instrument*, not the contribution. What this paper adds is the method that turns such a vocabulary into a programme-level analysis.

Sancho-Thomas et al. [14] describe the NUCLEO framework, a computer-supported collaborative-learning environment for teaching teamwork in software-engineering courses. NUCLEO is an example of how teamwork instruction *can* be designed and delivered, not a description of what programmes declare; it serves here as a reference point for what good teamwork instruction looks like, against which the patterns the method surfaces can be read.

Groeneveld et al. [9] is the closest study in terms of method. They manually code 278 non-technical course syllabi drawn from 110 European computing programmes and identify

*Supervisors: Dr. Sole Pera and Merel Steenbergen, EEMCS, TU Delft.

which soft skills are mentioned; teamwork is among the most common. Two things differ from the present work. First, they sample only non-technical courses, so they cannot speak to how teamwork appears in technical courses or how it is structured across a programme. This matters because teamwork in engineering is often embedded in technical work itself — group labs, programming and design projects — so technical courses are plausible sites of it. Not every technical course is expected to carry teamwork (a pure-mathematics course may reasonably have none); the point of coding the full mandatory load is to show where teamwork is and is not declared, not to assume it. Second, they look at many programmes shallowly, whereas this study's method looks at one programme deeply, including its technical courses. Their non-technical skills inventory is one of the components of the framework used here, so this work extends rather than competes with it.

A few other studies are nearby but address different questions. Garcia, Treude and Valentine [7] systematically map 14 papers on collaborative-learning paradigms in software-engineering education, so their unit is the research literature, not the curriculum. Miedema et al. [13] perform a programme-level analysis but for Data Systems courses, not teamwork. Arafeh [3] proposes a curriculum-mapping methodology applied at doctoral level. Aivaloglou and van der Meulen [2] study student perceptions of group programming assignments, so the unit is the assignment, not the curriculum. None offers a repeatable, programme-level way to describe how a CS bachelor curriculum declares teamwork across its full mandatory load. That is the gap this paper addresses.

3 Method

This section describes the method independently of any particular programme; Section 4 applies it to a case.

3.1 Data requirements

The method operates on the *declared* curriculum: the content a programme publicly commits to in its structured course syllabi, as opposed to the enacted curriculum (what happens in class) or the achieved curriculum (what students retain). The declared layer is the appropriate target because it is what a programme officially commits to teach, practise, and assess; it is what prospective students see before they enrol; and it is what accreditation bodies evaluate.

To apply the method, a programme must expose, for its full mandatory course load, syllabi with structured fields — at minimum intended learning outcomes, content, and examination. Programmes whose syllabi omit these fields, or expose them inconsistently, limit what the method can observe; this is a boundary condition returned to in the case study and the limitations.

3.2 The teamwork framework and the categories we code

The instrument is a teamwork framework that describes teamwork-related preparation along three axes: professional dispositions, non-technical skills, and collaborative teaching methods. This study uses the framework of Garcia et al. [6], which integrates eleven dispositions drawn from the CC2020 curricular recommendations, twelve non-technical skills drawn from the Groeneveld inventory [9], and a palette of collaborative-learning teaching methods. The method is not tied to this particular framework: any scheme that categorises teamwork along comparable axes could serve as the instrument.

The three axes are applied as follows. The **dispositions** (e.g. *Collaborative*, *Responsible*, *Self-directed*) are attitudes a programme aims to develop, read from a course's intended learning outcomes and content. The **non-technical skills** (e.g. *Communication*, *Conflict resolution*, *Leadership*, *Role awareness*, *Teamwork/dynamics*) are concrete abilities, read the same way. The **teaching methods** (e.g. group project, pair programming, peer review, scrum) are ways of delivering teamwork, read wherever a course names one.

Because teamwork is a single named skill in the framework (*Teamwork/dynamics*) but is better understood as a composite capability, the method codes a defined *teamwork-relevant subset* rather than the single keyword: the dispositions *Collaborative* and *Responsible*; the skills *Communication*, *Conflict resolution*, *Leadership*, *Role awareness*, and *Teamwork/dynamics*; and any of the collaborative teaching methods. A course is treated as declaring teamwork instruction when it declares any category in this subset. Categories that are intrinsically individual (e.g. *Self-directed*, *Creativity*) are recorded when present but not counted as teamwork instruction.

The method's own components — the data requirement above, the teamwork-relevant subset, the coding rule and misfit typology below, and the course-to-programme aggregation — are what this paper contributes; the category vocabulary is borrowed from the framework.

3.3 Codebook and coding rule

The underlying analysis technique is *directed* (deductive) content analysis [10]: coding proceeds from the framework's categories and is extended where the text does not fit (a Type-2 misfit, below). What this paper assembles on top of that technique — the data requirement, the teamwork-relevant subset, the misfit typology, and the course-to-programme aggregation — is the contribution. For each category, the codebook gives a clear rule for what counts as a match, plus a worked example for the main teamwork codes. I call this rule *liberal fit*: a passage counts when its meaning clearly fits the rule, even if it does not use the exact words — that is, I code the *latent* meaning of a passage rather than only

its *manifest* wording, a standard distinction in qualitative content analysis [8, 10]. For example, "students learn to coordinate their contributions to a shared deliverable" counts as *Teamwork/dynamics* even though it never says "teamwork".

Two distinct kinds of misfit are tracked separately and reported as substantive findings rather than as data-cleaning leftovers:

- **Type-1 misfit:** a framework category for which no instance is declared anywhere in the curriculum — the framework expects something the curriculum does not deliver.
- **Type-2 misfit:** content in the curriculum that bears on teamwork in a way no framework category captures — the curriculum delivers something the framework did not anticipate.

3.4 From course-level coding to programme-level analysis

The unit of coding is the individual course; the unit of analysis is the programme. For each course, the syllabus fields are read in order (intended learning outcomes, content, examination), every passage bearing on teamwork is extracted verbatim, and each passage is mapped to one or more codebook categories or flagged as type-2 misfit.

Programme-level claims then come from patterns across the full set of courses, not from any single course. The method reports three: the distribution of declared categories (which appear, and in how many courses); structural absences (categories with no instance anywhere, i.e. type-1 misfit); and recurring unmapped content (type-2 misfit). Beyond counts, the distribution is read against the *structure* of the curriculum — where in the degree teamwork is and is not declared (which years, and which kinds of courses, such as dedicated project courses versus the technical core or the capstone) — so that the analysis describes the curriculum as a designed whole rather than a set of separate courses. Claims are restricted to counts and structural position across the coded courses; nothing is inferred beyond what the coded passages support.

3.5 Reliability and transparency

Reliability rests on instrument transparency rather than on inter-rater agreement. Because the codebook operationalises an externally validated framework [6] into explicit fit criteria, the relevant question is whether another coder applying the same codebook would reach the same coding. Three features support this. First, every code is defined by an explicit fit criterion with worked examples for the principal teamwork codes, so decisions are checkable rather than tacit. Second, every coded passage was verified against the codebook by the author, who is responsible for the final coding. Third, the coding instrument, codebook, and procedure are documented in full, so the analysis can be reproduced from the public syllabi or applied to another programme. To check coding consistency, eight of the 25 courses (about 30%), spanning teamwork-rich, explicitly individual, and technical-core courses, were re-coded a second time after a gap of about four weeks, without reference to the first pass. The two passes agreed on 90% of coding decisions; the remaining disagreements were resolved by re-checking each passage against the codebook criterion.

4 Case study: applying the method

4.1 Case selection

The method is demonstrated on *Datateknik* (translated by the host institution as Computer Engineering), the three-year bachelor of engineering at KTH Royal Institute of Technology, for the Autumn 2025 cohort. KTH was chosen because it satisfies the method's data requirement well: every mandatory course publishes a complete, structured, English-language syllabus with the needed fields. The case is selected for data availability and as a structured, research-intensive European CS bachelor; no claim is made that it is statistically representative.

4.2 Dataset

The dataset is the public syllabi of all 25 courses that are mandatory somewhere in the programme. Fifteen are common to every student (Years 1 and 2). The remaining ten are Year-3 courses that depend on specialisation: the programme splits into two tracks, Software Engineering (DPU2) and Internet of Things (SAIN). Three Year-3 courses are mandatory in both tracks (including the degree project II142X), two are mandatory only in Software Engineering, three only in IoT, and two are conditionally-elective in Software Engineering (each student takes one). The 25 courses are therefore the *union* of both tracks' required load; no single student takes all 25 — a Software Engineering student and an IoT student each take 21. The two tracks share 18 of those 21 courses — including every course that declares teamwork — so a separate per-track analysis would largely repeat the same courses and reach the same teamwork findings. The union is analysed because the question concerns what the curriculum, as designed, declares; track-specific findings are flagged where they arise. Each syllabus was scraped from the public KTH course catalogue and stored as one structured JSON record, preserving the original section structure.

5 Results

Applying the method to the KTH case yields the patterns below. Table 1 gives the headline counts, and Figure 1 maps every mandatory course by year.

5.1 Categories present (SQ1)

The curriculum declares the disposition *Collaborative* and the skills *Communication*, *Teamwork/dynamics*, *Leadership*,

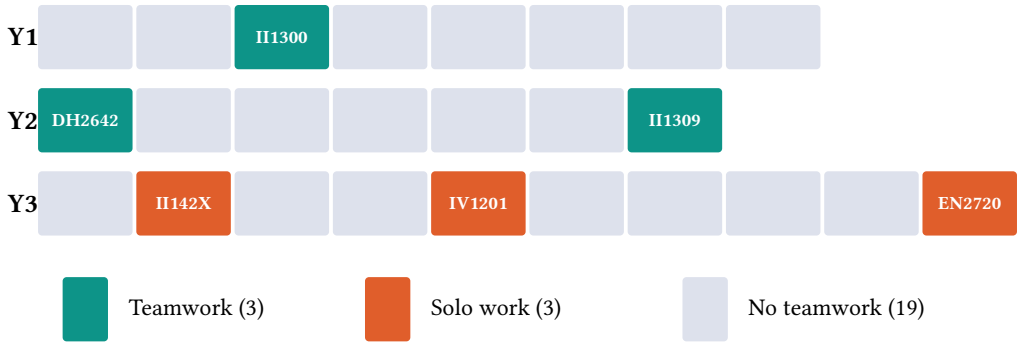

Figure 1. Curriculum map of all 25 mandatory courses, by year. Teamwork is declared only in three early project courses (teal); the three explicitly individual courses (orange), including the II142X capstone, are all in year 3; the remaining courses (grey) declare no teamwork. The concentration in years 1–2 is visible only at the curriculum level.

Table 1. Programme-level summary of fit and misfit findings across the 25 mandatory courses of the case programme.

Question / pattern	Finding	Count
SQ1 – categories present	Courses declaring teamwork substantively in ILO or Content	3 / 25
SQ2 – categories absent (type-1)	Courses with no substantive teamwork-related ILO/Content	19 / 25
SQ2 – explicit individual framing	Courses declaring solo work in ILO or Content	3 / 25
SQ2 – schema-level absence	Courses with no dedicated Teaching Methods field in their public syllabus	25 / 25
SQ3 – type-2 misfit	Recurring unmapped category: *Individual accountability within group*	2 / 25
Pattern A (discovered)	Ethical-approach boilerplate mismatched against course content	21 / 25
Pattern B (discovered)	Teamwork courses assessed by "Project", but individual-vs-group grading not declared	3 / 3

and *Role awareness*, together with the collaborative teaching methods *group project* and *agile*. These declared categories are concentrated in just three of the 25 mandatory courses: II1309 *Projects and Project Methods* (Y2), II1300 *Engineering Skills* (Y1), and DH2642 *Interaction Programming and the Dynamic Web* (Y2).

All three are dedicated project or engineering-skills courses rather than technical-core subject courses: teamwork is declared where a course is explicitly *about* working on a project, and not elsewhere. *Collaborative*, *Communication*, and *Teamwork/dynamics* appear across these courses, all three declaring *group project* as the method of delivery and II1309 additionally *agile*; *Leadership* and *Role awareness* appear only in II1309, the dedicated project-methods course. A further small set of courses mentions only individual communication (presenting solutions orally or in writing) or carries teamwork-related content only in non-substantive locations such as the examination procedure; these are recorded but not counted as substantive teamwork declarations.

Curriculum-level reading. Taken as a whole, the declared teamwork content sits entirely in years 1 and 2 and in project-type courses; the technical core and year 3 declare no collaborative teamwork — the only year-3 collaborative element is a peer-review step inside the otherwise individual capstone. If teamwork is meant to build across the degree, this curriculum front-loads it into a few early project courses rather than developing it across the programme — a pattern visible only at the curriculum level, not from any single course.

5.2 Categories absent (SQ2)

At the category level, the teamwork-relevant skill *Conflict resolution* is declared nowhere in the curriculum (a type-1 misfit). At the course level, three patterns of absence together account for all 25 courses.

Most courses declare no teamwork at all. Of the 25 mandatory courses, 3 declare teamwork substantively (SQ1) and 3 declare individual work explicitly (below); the remaining 19 declare no substantive teamwork content in their intended learning outcomes or content sections — predominantly the technical-core courses across the three years. Some of these 19 mention only individual communication (presenting solutions orally or in writing), which is recorded but not counted as teamwork.

Three courses declare individual work explicitly. IV1201 *Design of Global Applications* (Y3 Software Engineering specialisation) has an ILO that students will "alone solve complex problems". EN2720 *Ethical Hacking* (Y3 conditionally-elective) has content in which "students independently attack a network". II142X *Degree Project in Computer Engineering* (Y3, 15 ECTS) is the bachelor capstone, and its Content section declares "an individual independent project". In a curriculum view this last one is the most notable: the capstone is the largest single course and the final synthesis of the degree, yet it requires no teamwork at all — the programme's culminating mandatory experience is solo (discussed in Section 6.2).

No dedicated teaching-methods field. The public KTH syllabus schema contains no Teaching Methods or Learning Activities field, so a method of delivery can be read only where a course names one in its Content or ILO text. Where methods are named they cluster in the same few courses: *group project* in II1300, DH2642, and II1309, *agile* in II1309, and *peer review* in the II142X capstone. For the other courses the method of delivery is not declared.

5.3 Content unmapped by the framework (SQ3)

One recurring pattern is declared that no framework category directly anticipates: *individual accountability within group work*. II1309 and II1300 both declare that each member of a group must be able to present and answer questions about the entire group product, separately from the group grade — recurring content aimed at preventing free-riding that does not fit cleanly under any single disposition, skill, or teaching method. An *Individual accountability within group* code was added to capture it. At the curriculum level this pattern is localised: it appears only in the dedicated project courses where group work is actually declared (II1300 and II1309), not across the programme — so the curriculum's one explicit safeguard against free-riding sits inside the same small project cluster that carries all of its teamwork.

5.4 Two additional patterns observed during coding

Pattern A. The *ethical approach* field appears in every course's syllabus and contains a verbatim-identical template paragraph across all 25 courses, referring to group responsibility ("all members of a group are responsible for the group's work…in an oral assessment, every student shall be able to present and answer questions about the entire assignment and solution"). It appears unchanged even in courses whose substantive sections describe explicitly individual work, including the three explicit-individual courses above. In 21 of the 25 courses, the ethical-approach text and the substantive content do not match in this way. The field's passages were not coded as fits.

Pattern B. The three teamwork-declaring courses do carry project-based assessment components (II1300, DH2642, and II1309 each examine a "Project"). However, the syllabus does not state whether these projects are graded individually or as a group, so whether *teamwork itself* is assessed — as opposed to the project deliverable — cannot be determined from the declared data.

6 Discussion

The findings above are reported not for their own sake but as evidence that the method surfaces real, curriculum-level structure that a course-level study would miss. Each subquestion, and the two discovered patterns, illustrate a different thing the method can reveal.

6.1 What SQ1 shows

Teamwork is substantively declared in only 3 of 25 mandatory courses, and the declared content covers a narrow subset of the framework's dispositions and skills; teaching methods are declared only where a course names them in its text, because the schema has no dedicated field for them. Teamwork preparation in the declared curriculum is therefore concentrated in a small number of dedicated courses, with the rest of the programme declaring none in its substantive sections. If teamwork is meant to develop across a programme through repeated experience, the declared curriculum at this case does not visibly reflect that. This matters because teamwork is not an optional aim here: the Swedish Higher Education Ordinance sets it as a required degree outcome [15]. The mandated *graduate* outcome is thus barely visible in the *declared course-level* syllabi — exactly the visibility gap the method is built to surface.

Against Garcia [6], who reports teamwork as expected by industry but under-developed in academic preparation, finding it substantively declared in only 3 of 25 courses is notable. Against Groeneveld [9], who finds teamwork among the most commonly mentioned skills in a course-level sample across many programmes, the present absence across most of one programme is consistent with non-technical content being concentrated in a few courses and missing from the technical core — but it is a kind of evidence (programme-level structural absence) that course-sample studies cannot produce.

6.2 What SQ2 shows

Most of the 25 courses (19 of 25) contain no substantive teamwork-related ILO, 3 courses declare individual work explicitly, and the schema exposes no dedicated teaching-methods field. The type-1 misfit across most courses shows that the declared curriculum does not require students to engage with teamwork through their formal ILOs in most of the programme, so its declared structure of teamwork is sparse rather than continuous.

The most striking result is the 3 explicit-individual courses, one of which is II142X, the 15-ECTS bachelor capstone — the largest single course and the final synthesis of the degree.

The course intended to demonstrate what a graduate can do is declared as an individual independent project. Whether this is deliberate design or inherited convention is beyond what the declared data can answer, but the data does indicate that the programme's culminating mandatory experience is positioned as solo — the kind of design-level observation that only a curriculum-level method makes visible.

The absence of a dedicated teaching-methods field is itself a finding about the method's reach: the framework's teaching-methods axis is only partially observable on this data source, visible only for the few courses that name a method in their text. A complete account of declared teaching methods would need a data source with an explicit methods field (course memos, formal handbooks, instructor interviews).

6.3 What SQ3 shows

The curriculum declares one recurring content pattern the framework does not directly capture: *individual accountability within group work*. The framework, while comprehensive, may not yet describe this institutional response to free-riding; the declared response in II1309 and II1300 — requiring each member to defend the entire group product individually — is neither a pure disposition nor a pure skill but a structural assessment choice that crosses both. One reading is a possible extension to the framework: an explicit *individual accountability* category could improve coverage without changing its three axes. The concept is recognised in collaborative-learning research (under names like "positive interdependence with individual accountability"), so the finding is not new to the field; what is new is that a programme-level method can surface category gaps in an otherwise comprehensive framework.

6.4 The two additional patterns

Pattern A (the boilerplate) must be read separately from the per-course findings. The identical institution-level ethical-approach paragraph in all 25 syllabi indicates two things: the field is not informative as evidence of any course's declared teamwork content (it is an institutional template, not a course-level declaration), and the existence of such a template is itself a signal that teamwork is treated as a general institutional expectation rather than a course-level commitment. A practical consequence for the method: descriptive work using such syllabi must detect and discount boilerplate before coding, or it will over-count.

Pattern B (the assessment of teamwork) reads against constructive alignment theory [4], which holds that intended outcomes, activities, and assessment should line up. The three teamwork-declaring courses do carry a project-based assessment component, so on first inspection outcome and assessment align. But the declared examination does not say whether the project is graded individually or as a group, so it cannot be determined from the syllabus whether *teamwork itself* is assessed or only the deliverable. This is a finding about *declared* alignment only, and marks another boundary of the declared layer: it can show that a project is assessed, not whether the teamwork within it is.

6.5 Limitations

Single demonstration. The method is demonstrated on one programme. No claim is made that the KTH *findings* generalise; the contribution that generalises is the method, and a single application is one tile in a wider mosaic of descriptions that the method could produce.

One coder. The case was coded by a single coder. Reliability rests on the transparency of the codebook and on an intra-rater re-code of about 30% of the courses, which reached 90% agreement (Section 3), rather than on inter-rater agreement. Full inter-rater coding by a second independent coder would strengthen the demonstration further; the public syllabi and documented codebook make that possible.

Declared layer only. The method, by design, reads the declared curriculum. The enacted curriculum (what happens in class) and the achieved curriculum (what students retain) are out of scope; all claims apply to the declared layer.

Schema dependence. The method is only as complete as the syllabus schema it reads. Where a programme exposes no dedicated teaching-methods field, that axis is only partially observable — a limit the case made concrete.

Framework dependence. The description is only as complete as the framework that supplies its vocabulary, and that framework also motivates the study. A different framework might surface categories this one does not; the type-2 misfit reported here is one sign that the vocabulary is not exhaustive.

Coding rule. The coding rule is liberal fit. A stricter rule requiring explicit terms would likely lower the fit counts, so a figure such as 3-of-25 should be read as the result of a deliberately inclusive rule, not a hard floor; a strict re-code is a natural robustness check.

7 Conclusion

The research question — how teamwork is represented across a CS bachelor curriculum's full mandatory load — is answered, for this programme, by the description above. The contribution, however, is the reusable method that produces such an answer for any programme: a data requirement (structured, publicly declared syllabi), a coding instrument (a teamwork framework reduced to a teamwork-relevant subset), and an aggregation procedure that turns course-level coding into a programme-level account of what is declared, what is absent, and what the framework misses. The three sub-questions (SQ1–SQ3) are the specific outputs the method produces, and a single application is how its value is shown.

Demonstrated on one CS bachelor programme, the method surfaced patterns invisible at course level: teamwork substantively declared in only 3 of 25 mandatory courses, the whole technical core and year 3 declaring none, a 15-ECTS capstone framed as individual work, an institution-wide ethics template that does not match course content, and project assessment whose individual-versus-group grading is undeclared. These show the method reveals real curriculum-level structure rather than restating course-level facts.

For the original problem, the demonstration shows the declared curriculum cannot be ruled out as a source of the teamwork gap at the programme studied, and points to concrete uses: a programme can apply the method to its own curriculum to check whether teamwork is built as a thread across courses or confined to a few, and whether its capstone matches that intent; accreditation bodies can ask programmes to show, across the full mandatory load, where teamwork outcomes are declared and assessed. Future work would apply the method to further programmes to build the cross-programme mosaic, add a second coder for inter-rater reliability, and extend it to teamwork-relevant elective courses.

8 Responsible Research

8.1 Ethics

The study uses only public, non-personal data. The KTH public course catalogue contains syllabi for mandatory courses with no personal information about students, instructors, or other individuals. No human subjects were involved at any stage. The data collection complies with the Netherlands Code of Conduct for Research Integrity [11] and required no ethics review under the TU Delft EEMCS standards for bachelor-project research.

8.2 Reproducibility

The study can be reproduced from public sources. The course syllabi are openly available in the KTH course catalogue, and the coding instrument and procedure are described in full in Section 3: the teamwork-relevant subset, the codebook (each category operationalised as a fit criterion, with worked examples for the principal teamwork codes), the coding rule, and the course-to-programme aggregation. A researcher can therefore re-collect the same syllabi and re-apply the documented method to reproduce the analysis, extend it to other programmes, or revise the codebook.

8.3 Generative AI disclosure

I used generative AI during this project for limited tasks consistent with the TU Delft GenAI policy for BSc/MSc end projects [16]: literature search assistance; first-pass coding suggestions that I subsequently verified row by row; writing-quality support (proofreading and structural feedback on prose); and assistance with build tooling (LaTeX, JSON pipelines). AI is not listed as an author and was not the sole driver of any research decision. The coding decisions, the framework choice, and the analytic argument of this paper are mine. A per-session disclosure log is maintained as a supplementary artefact.

References

[1] ABET Computing Accreditation Commission. 2025. Criteria for Accrediting Computing Programs, 2025–2026. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2025-2026/. Student Outcome 5.

[2] Efthimia Aivaloglou and Anna van der Meulen. 2021. An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments. *ACM Transactions on Computing Education* 21, 3 (sep 2021), 1–22. doi:10.1145/3440994

[3] Sousan Arafeh. 2016. Curriculum Mapping in Higher Education: A Case Study and Proposed Content Scope and Sequence Mapping Tool. *Journal of Further and Higher Education* 40, 5 (2016), 585–611. doi:10.1080/0309877X.2014.1000278

[4] John Biggs. 2003. *Teaching for Quality Learning at University: What the Student Does* (2nd ed.). Society for Research into Higher Education & Open University Press, Buckingham, UK.

[5] ENAEE. 2021. EUR-ACE Framework Standards and Guidelines. https://www.enaee.eu/eur-ace-system/standards-and-guidelines/. First Cycle programme outcome, Transferable Skills.

[6] Rita Garcia, Andrew Csizmadia, Janice L. Pearce, Bedour Alshaigy, Olga Glebova, Brian Harrington, Konstantinos Liaskos, Stephanie J. Lunn, Bonnie K. MacKellar, Usman Nasir, Raymond Pettit, Sandra Schulz, Craig D. Stewart, and Angela M. Zavaleta Bernuy. 2025. An International Examination of Non-Technical Skills and Professional Dispositions in Computing – Identifying the Present Day Academia-Industry Gap. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 124–174. doi:10.1145/3689187.3709610

[7] Rita Garcia, Christoph Treude, and Andrew Valentine. 2024. Application of Collaborative Learning Paradigms within Software Engineering Education: A Systematic Mapping Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3626252.3630780

[8] Ulla H. Graneheim and Berit Lundman. 2004. Qualitative Content Analysis in Nursing Research: Concepts, Procedures and Measures to Achieve Trustworthiness. *Nurse Education Today* 24, 2 (2004), 105–112. doi:10.1016/j.nedt.2003.10.001

[9] Wouter Groeneveld, Brett A. Becker, and Joost Vennekens. 2020. Soft Skills: What do Computing Program Syllabi Reveal About Non-Technical Expectations of Undergraduate Students?. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2020)*. Association for Computing Machinery, New York, NY, USA, 287–293. doi:10.1145/3341525.3387396

[10] Hsiu-Fang Hsieh and Sarah E. Shannon. 2005. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research* 15, 9 (2005), 1277–1288. doi:10.1177/1049732305276687

[11] KNAW, NFU, NWO, TO2-federation, Vereniging Hogescholen, and VSNU. 2018. Netherlands Code of Conduct for Research Integrity. https://doi.org/10.17026/dans-2cj-nvwu.

[12] Robert Lingard and Shan Barkataki. 2011. Teaching Teamwork in Engineering and Computer Science. In *2011 Frontiers in Education Conference (FIE)*. IEEE, F1C–1–F1C–5. doi:10.1109/FIE.2011.6143000

[13] Daphne Miedema, Toni Taipalus, Vangel V. Ajanovski, Abdussalam Alawini, Martin Goodfellow, Michael Liut, Svetlana Peltsverger, and

Tiffany Young. 2025. Data Systems Education: Curriculum Recommendations, Course Syllabi, and Industry Needs. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 95–123. doi:10.1145/3689187.3709609

[14] Pilar Sancho-Thomas, Rubén Fuentes-Fernández, and Baltasar Fernández-Manjón. 2009. Learning Teamwork Skills in University Programming Courses. *Computers & Education* 53, 2 (sep 2009), 517–531. doi:10.1016/j.compedu.2009.03.010

[15] Swedish Government. 1993. The Higher Education Ordinance (1993:100), Annex 2 — System of Qualifications. https://www.uhr.se/en/start/laws-and-regulations/Laws-and-regulations/The-Higher-Education-Ordinance/Annex-2/. Engineering degree qualitative target: capacity for teamwork and collaboration with various constellations.

[16] TU Delft. 2026. Instruction on the Use of Generative AI in BSc/MSc End Projects. Internal course document, CSE3000 Research Project, 2025/26 Q4 Brightspace.