

Learning Strategies for Programming-Heavy and Theory-Heavy Assessments: A Shared Role for Critical Thinking

Sebastian Iacovino Spitaleri[1]
Supervisor(s): Xiaoqi Feng[1], **Martin Skrodzki**[1], **Rafael Bidarra**[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
July 5, 2026

Name of the student: Sebastian Iacovino Spitaleri
Final project course: CSE3000 Research Project
Thesis committee: Xiaoqi Feng, Martin Skrodzki, Rafa Bidarra, Jan van Gemert

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Assessments are a crucial, high-stakes part of the learning process, and students consolidate much of their learning during the considerable effort they make preparing for them. Computer science curricula typically include both theory-heavy and programming-heavy assessments. This study investigates what learning strategies final-year computer science students adopt when preparing for theory-heavy assessments compared to programming-heavy assessments. Using semi-structured interviews with three final-year Computer Science and Engineering students at TU Delft, transcripts were analysed through an inductive open-coding pass followed by deductive coding using an established framework of cognitive and metacognitive learning strategies. Results show that rehearsal and elaboration strategies — reviewing lecture slides, note-taking, and creating cheat sheets — were predominantly associated with preparation for theory-heavy assessment, while metacognitive self-regulation, through consulting assignment specifications, was predominantly associated with preparation for programming-heavy assessment. Critical thinking was the one strategy reported across both assessment types, expressed through the use of practice exams. These findings suggest that practice exams function as a strategy supporting critical thinking regardless of assessment type. The relevance and diversity of practice exam material may therefore be an important, underexplored lever for supporting effective learning in computer science education.

1 Introduction

Understanding how students learn effectively is an important concern for researchers, educators, and students themselves. In computer science education, assessments serve not only to evaluate student performance but also to shape learning. Different forms of assessment place different demands on students, which may influence the learning strategies they adopt (Joughin, 2009).

Research on learning has frequently been concerned with the distinction between theory and practice (Thuné & Eckerdal, 2019), and computer science education is no exception (Eckerdal et al., 2024); in this study, I make a similar distinction between two types of assessments: theory-heavy assessments, whose primary learning outcomes involve mathematical reasoning, formal proofs, and abstract concepts, and programming-heavy assessments, which focus on the design, implementation, and evaluation of code. While a body of work has examined learning and teaching strategies within introductory programming courses specifically (Blikstein et al., 2014; Boudia et al., 2019; Isong, 2014), there is limited work investigating whether the strategies students use for programming-heavy assessments differ from those used for theory-heavy assessments more broadly across a curriculum. Understanding these differences could help educators design assessments and accompanying support that better match how students actually study for them, and could help students themselves prepare more effectively for the different types of assessment they encounter. This study addresses that gap by investigating the following research question:

What learning strategies do final-year computer science students adopt during programming-heavy assessments compared to theory-heavy assessments?

This study examines what strategies students report using, independent of whether those strategies proved effective. To answer this question, I conducted semi-structured interviews with three final-year Bachelor students in Computer Science and Engineering at TU Delft and analysed the transcripts using a deductive coding framework drawn from Pintrich et al.'s

(1991) taxonomy of learning strategies. By examining the experiences of these students, this work aims to provide insight into what learning strategies students use across theory-heavy and programming-heavy assessments.

This paper reports that final-year Bachelor students in Computer Science and Engineering use different strategies depending on the type of assessment they are preparing for. Rehearsal and elaboration strategies — reviewing lecture slides, note-taking, and creating cheat sheets — were predominantly associated with preparation for theory-heavy assessment, while metacognitive self-regulation, through consulting assignment specifications, was predominantly associated with preparation for programming-heavy assessment. Critical thinking was the one strategy reported across both assessment types, expressed through the use of practice exams. These findings suggest that practice exams function as a strategy supporting critical thinking regardless of assessment type.

2 Literature Review

This section reviews the concepts and prior work underpinning this study. First the distinction between theory and practice in Computer Science education research is examined. Then, assessment as it is understood in this study is defined. Finally, three influential frameworks for classifying learning strategies are discussed.

2.1 Theory and Practice in Computer Science Education

Thuné and Eckerdal (2019, p. 772) note that "research on learning has frequently been concerned with the concepts 'theory' and 'practice'". Eckerdal et al. (2024, p. 333) agree that there is a "distinction between theory and practice" in Computer Science and argue that the concepts may have different definitions in different contexts. Eckerdal et al. (2024) and Thuné and Eckerdal (2019) both agree that the distinction between theory and practice is fuzzy and that theory is about general principles and relations while practice is about special cases.

Boudia et al. (2019) further argue that, in computer science, teaching strategies and methods should be different for programming courses, proposing that collaborative learning combined with early, hands-on use of current technologies leads to more effective outcomes than conventional lecture-based instruction. If teaching approaches are expected to differ across theory-heavy and programming-heavy contexts, the learning strategies students themselves adopt in response may differ as well — motivating the comparison this study undertakes.

2.2 Definition of Assessment

Definitions of assessment in the literature emphasize different aspects of the assessment process. Joughin (2009, p. 16) centers his definition on the interpretive act performed by the assessor: "To assess is to make judgements about students' work, inferring from this what they have the capacity to do in the assessed domain, and thus what they know, value, or are capable of doing." Here, assessment is framed abstractly, as an epistemic act of judgement and inference about a student's underlying capability, independent of any particular institutional context or assessment format.

Delft University of Technology (n.d.), the institution at which this study's participants are enrolled, instead defines assessment in terms of its institutional function within a course:

"an educational activity during which individual students demonstrate how well they master the learning objectives of a course, and during which their performance is measured. This information is shared with students to steer and/or evaluate their learning." This definition reflects the principle of constructive alignment (Biggs, 1996; Biggs & Tang, 2011), which holds that in a well-designed course, learning objectives, teaching activities, and assessment tasks are deliberately aligned. Learning objectives specify what students are expected to know, understand, or be able to do upon completing a course; at TU Delft, every course has an explicit set of learning objectives formulated according to Bloom's taxonomy (Bloom, 1956; Krathwohl, 2002). Where Joughin's definition centres on the assessor's interpretive judgement, this definition centres on assessment as a structured, course-embedded activity with a stated downstream purpose: measuring performance against predefined learning objectives and feeding that information back to the student.

This study draws on both perspectives: like Joughin, it treats assessment outcomes as something from which broader conclusions about a student's capability can be inferred, and like the TU Delft definition, it is concerned specifically with assessment as it occurs within a structured curriculum — namely, the exams, programming assignments, and projects through which participants' learning is formally evaluated.

These definitions also contextualise the distinction this study draws between the two assessment types. Theory-heavy assessments are those whose primary learning outcomes involve mathematical reasoning, formal proofs, and abstract concepts — where students are assessed on their ability to understand, derive, and apply formal models. Programming-heavy assessments are those whose primary learning outcomes involve the design, implementation, and evaluation of code — where students are assessed through programming assignments, projects, or implementations. In Joughin's terms, the difference between the two lies in what the student's work entails; in TU Delft's terms, it lies in which learning objectives are being measured.

2.3 Learning Strategies

The literature offers several frameworks for conceptualizing learning strategies, differing in scope and purpose. This section reviews three influential frameworks — those of Weinstein and Mayer (1983), Pintrich et al. (1991) and Hattie and Donoghue (2016) — before identifying which framework this study adopts as its coding scheme, and why.

Weinstein and Mayer (1983) propose a taxonomy of five categories of learning strategies, intended as a conceptual framework to guide teachers in explicitly teaching students how to learn, rather than focusing solely on content. These five categories encompass rehearsal strategies such as copying, underlining, or shadowing; elaboration strategies such as paraphrasing or summarizing; organizational strategies such as outlining or creating a hierarchy; comprehension monitoring strategies such as checking for comprehension failures; and affective strategies aimed at managing anxiety and motivation.

Pintrich et al. (1991) developed the Motivated Strategies for Learning Questionnaire (MSLQ), a self-report instrument designed to assess college students' motivational orientations and their use of cognitive and metacognitive learning strategies within the context of a specific course, providing researchers and instructors with a modular, empirically validated tool for measuring how students actually approach their learning. They introduce a framework of five cognitive and metacognitive learning strategies: Rehearsal Strategies, Elaboration Strategies, Organization Strategies, Critical Thinking and Metacognitive Self-Regulation. Rehearsal Strategies are those that involve simple repetition. For example,

copying notes or rereading text. Elaboration Strategies are those that involve building internal connections between items to be learned. For example, Paraphrasing, summarizing, and generative note-taking. Organization Strategies are those that involve building an internal overview. For example, clustering and outlining. Critical Thinking involves applying previous knowledge to new situations. Metacognitive Self-Regulation involves planning, monitoring and regulation activities. For example, tracking of one's attention and questioning. Pintrich et al. (1991) share three learning strategies with Weinstein and Mayer's taxonomy — rehearsal, elaboration, and organisation — but differ in two important ways: they separate affective strategies out into a distinct motivation component, and they split comprehension monitoring into two more precisely defined constructs, critical thinking and metacognitive self-regulation.

Hattie and Donoghue (2016) propose a model of learning with three progressive phases of learning - surface learning, deep learning, and transfer - each comprising processes of acquisition (initial encoding of information) and consolidation (strengthening and stabilising learning over time). Surface learning focuses on the initial acquisition of basic knowledge and is supported by strategies such as rehearsal, memorisation, note-taking, summarising, and the use of mnemonics. Deep learning involves building meaningful understanding and relationships between ideas, and is supported by elaboration strategies (e.g., explaining concepts in one's own words, questioning, and making connections), organisational strategies (e.g., structuring and mapping knowledge), and metacognitive monitoring. Transfer refers to the ability to apply knowledge in new and unfamiliar contexts, and is supported by strategies that emphasise abstraction, comparison, and problem-solving across contexts. Overall, Hattie and Donoghue (2016) argue that the effectiveness of learning strategies is determined by the learner's stage of understanding and the intended learning outcome, than being universally good or bad.

These three frameworks differ both in scope and in purpose: Weinstein and Mayer (1983) offer a broad, behaviourally descriptive taxonomy that includes affective strategies alongside cognitive ones; Hattie and Donoghue (2016) propose a more theoretical model organized around phases of learning rather than discrete, codable categories; and Pintrich et al. (1991) provide a framework explicitly designed to classify self-reported learning behaviour, making it more directly applicable as a deductive coding scheme for interview data. This study adopts Pintrich et al.'s (1991) five cognitive and metacognitive categories as its coding framework, since the set of categories is well-defined, focuses specifically on the cognitive and metacognitive dimensions of learning strategies and is directly suited to classifying participants' own descriptions of their study habits.

3 Methodology

This study employed a qualitative research design using semi-structured interviews. While the same research question could in principle have been addressed through a quantitative survey distributed to a larger student population, a survey would have constrained responses to predefined strategy categories, preventing participants from describing strategies in their own terms or elaborating on the conditions under which they use them. A qualitative approach, by contrast, enables open-ended, exploratory responses (Patton, 2015).

3.1 Sampling

A two-step sampling approach was used. First, purposive sampling was applied to define the eligible population: participants were required to be final-year Bachelor students in Computer Science and Engineering at TU Delft. This criterion was chosen because, by their final year, students have accumulated experience across a wide variety of both theory-heavy and programming-heavy assessments and by recruiting from a single institution, all participants have followed a shared core curriculum — making their retrospective comparisons across assessment types more informative.

Within this eligible population, participants were then recruited through convenience sampling via personal networks. This approach was chosen for its practicality given the limited time available for recruitment, though direct close friends were deliberately excluded to limit the potential introduction of bias from pre-existing shared assumptions about studying (Miles & Huberman, 1994).

Three students participated in the study. Recruitment was limited to three participants primarily due to the time constraints of a 10-week bachelor thesis project, which restricted the time available for both recruitment and in-depth analysis of each transcript. Given this sample size, the study aims for analytical generalisation — using the findings to refine theoretical understanding of how assessment type relates to strategy use rather than statistical generalisation to the broader population of Computer Science students (Patton, 2015).

3.2 Interview Guide

A semi-structured interview guide was used to ensure consistency across participants while allowing flexibility to explore relevant topics in greater depth. A semi-structured format was chosen over a fully structured format because it allowed follow-up questions when participants raised specific strategies or examples not anticipated in the guide, and over an unstructured format because it ensured all participants were asked about the same core topics, supporting comparison across interviews. The guide was developed based on the research question and through iterative refinement (Patton, 2015), and was organised into four parts. The interview guide can be found in Appendix A.

The pre-interview section introduced the study's purpose, obtained informed consent, and requested permission to record the interview for transcription. Part I served to establish participants' general background and to ease them into the interview by beginning with topics they find comfortable and familiar before moving to more substantive questions (Patton, 2015); asking about favourite courses and general experience with CS serves precisely this function, putting participants in the right frame of mind to reflect candidly on their study habits before the more focused questions of Parts II and III. Before Part II began, both theory-heavy assessment and programming-heavy assessment were defined explicitly to participants, to ensure a shared understanding. Part II and Part III then focused on learning in programming-heavy and theory-heavy assessments respectively, asking participants to recall specific assignments and exams and describe in detail what they did first, what they did next, and how they approached preparation overall. Part IV placed the greatest cognitive demand on participants (Patton, 2015), inviting them to compare the two assessment types directly, reflect on differences in their overall approach, and draw broader conclusions about their own learning.

3.3 Analysis

Interview recordings were first transcribed using Microsoft Word's transcription feature. Then I read the interview transcripts to familiarise myself with the data. The interview transcripts were then uploaded to Atlas.ti 26 (Friese, 2019) and an initial round of open coding was conducted. Although no initial codes survived after the second coding phase, the inductive first pass served to further familiarise myself with the data and to ensure the deductive pass was applied without unconscious retrofitting.

In a second phase, I re-coded the interview transcripts using the five categories from Pintrich et al. (1991) as a deductive coding framework. This deductive step allowed the interview data to be interpreted through an established theory. Finally, a co-occurrence analysis was carried out to examine how often each learning strategy code occurred within theory-heavy versus programming-heavy assessment contexts, providing a basis for comparing the distribution of strategies across the two assessment types.

4 Responsible Research

4.1 Ethical Considerations

As this study involved human participants, approval from the TU Delft Human Research Ethics Committee (HREC) was sought and obtained prior to data collection, in line with standard practice for research involving human subjects. The participants involved did so voluntarily and were informed both verbally and through the informed consent form which can be found in Appendix B, of the purpose of the study, the use of the collected data, and their right to withdraw from the study at any time. To protect participant privacy, interview transcripts were anonymised before analysis and no personally identifiable information is reported in this thesis.

4.2 Positionality

The researcher was a student in the same TU Delft Computer Science and Engineering bachelor programme as the participants. This shared background can be a double-edged factor in qualitative interviewing: on one hand, a shared context can make participants more comfortable describing their study habits candidly, since they can assume a baseline understanding of the courses and assessments being discussed; on the other hand, it creates a risk that the researcher's own assumptions or experiences as a student in the same programme could shape data collection or interpretation in ways that diverge from what participants actually intended to convey. To reduce the risk of bias from over-identification with participants, the researcher avoided recruiting close acquaintances.

4.3 Reproducibility and Transparency

Due to the qualitative nature of the study and the confidentiality of interview data, exact replication is not possible. However, methodological transparency was promoted by documenting the participant selection process, interview guide, coding procedure, and analytical framework. The interview guide is included in Appendix A to facilitate understanding and potential replication of the study design.

4.4 Use of Artificial Intelligence

Artificial intelligence tools were used to support the writing process by assisting with language refinement, improving clarity, and providing feedback on the structure of the thesis. All substantive research activities, including literature review, interview design, data collection, coding, analysis, interpretation of findings, and final editorial decisions, were carried out by the author. AI-generated suggestions were critically reviewed before inclusion in the thesis.

5 Results

This section presents the findings of the analysis of deductively coded transcripts, organised into five themes. Three strategies — reviewing lecture slides, taking notes, and creating cheat sheets — were predominantly associated with preparation for theory-heavy assessment. One strategy — consulting assignment specifications to determine required knowledge — was predominantly associated with preparation for programming-heavy assessment. One strategy — working through practice exams — was reported across both assessment types.

5.1 Looking at the Lecture Slides for Activation of Information in Working Memory

All three participants reported using lecture slides when preparing for theory-heavy assessments, often as the first step before any other study activity. Participant 1 stated that they ''would first go over the slides". Participant 2 similarly described that ''first I would go back to the lecture [slides]" and Participant 3 mentioned that ''I looked at all the lectures [slides]".

These statements were deductively coded as 'Rehearsal: Looking at Lecture Slides' since they involve the review of material rather than active transformation of it. For teachers, this indicates that lecture slides are not merely a complement to spoken lecture content but an active, standalone study resource that students return to. Their clarity and completeness as self-contained documents may therefore have a significant effect on theory-heavy assessment. For students, this finding suggests being aware of whether lecture slides function as standalone resources or primarily as accompaniments to spoken lecture content. If the slides alone are insufficient relying on them as the primary study resource may leave significant gaps, and seeking or creating supplementary materials early in the preparation process would be advisable.

5.2 Note-Taking as an Effective Strategy for Conceptual Understanding

Participants 1 and 2 described taking notes about lectures and both mentioned that they find this effective for theory but not for programming. Participant 1 mentioned that ''for theory courses I try to write a bit down in a notebook. On my computer [I] just write some notes about it and I feel like for programming I don't really do that at all". Participant 2 mentioned how ''I think if you go to every lecture and make lecture notes, then for theory that would be way better.". Participant 2 also described going back through their notes. Together, these accounts suggest that note-taking is used by participants as a theory-specific

strategy for building the kind of conceptual understanding that abstract or formal material requires.

These statements were deductively coded as 'Elaboration: Taking Notes' since they involve the active transformation of lecture content into a personally organised form, through paraphrasing and reorganising ideas rather than simply re-reading them. For teachers, this suggests that explicitly modelling or encouraging note-taking in theory-heavy courses may reinforce a behaviour students already find valuable. For students, the finding is a prompt to examine whether their note-taking is genuinely generative — paraphrasing and connecting ideas — or whether it has collapsed into copying, which would place it closer to rehearsal than elaboration and likely be less effective for building the deep understanding that theory-heavy exams typically require.

5.3 Cheat Sheets as an Effective Strategy for Summarising

Participants reported creating summaries or cheat sheets to revisit and condense course material. Participant 1 explained that ''making cheat sheets always helps because you see the content one more time" and noted that the examination subsequently ''wasn't really that hard." Similarly, participant 2 described making ''a summary of everything that has been said." The act of compiling a cheat sheet appears to function less as a reference tool actually used during the exam itself and more as a study activity in its own right.

These statements were deductively coded as 'Elaboration: Cheat Sheet' since they involve selecting, condensing, and reorganising course material into a personally meaningful form, requiring conceptual judgement about what is essential rather than simple repetition. For teachers, this suggests that explicitly encouraging cheat sheet creation as an exercise may support preparation for theory-heavy assessment. One concrete way to incentivise this is to allow students to use a self-made cheat sheet during assessments, rewarding the preparation process. For students, the finding suggests that the effort invested in making a cheat sheet is the point — producing a thorough one is worthwhile even when the assessment does not permit its use.

5.4 Using Assignment Specifications to Identify Required Knowledge

Participants commonly described beginning their preparation for programming-heavy assessments by examining assignment descriptions. Participant 1 stated that they ''looked at the assignment first". Similarly, participant 2 explained that they ''started reading the assignments to know what they expect from me". Participant 3 described how the assignment page outlines the required knowledge, ''They give you kind of an assignment page where it goes over like everything that you kind of need to know."

These statements were deductively coded as 'Metacognitive Self-Regulation: Determining What Knowledge Is Needed' since they reflect a planning activity in which students assess the demands of the task before engaging with the material, regulating what they study based on what the assessment requires. For teachers, this places a particular importance on the clarity of assignment descriptions: if students are using the specification as their primary planning tool, ambiguous or incomplete descriptions may directly undermine their ability to self-regulate effectively. For students, the finding suggests that reading the assignment specification carefully and systematically before beginning work is not merely good practice but the central strategic act that guides their subsequent preparation.

5.5 Practice Exams for New Relevant Situations

Unlike the strategies described above, practice exams were reported as a preparation strategy across both theory-heavy and programming-heavy assessments. Participant 1 explained ‘‘I was doing a lot of exams, and over time you just get quicker in it.” Participant 2 described independently attempting old exams, ‘‘I made a lot of old exams. I just tried to do it my own, try to find answer behind it.” Participant 3 similarly stated, ‘‘I did all of the previous exams, the past exams that they had available”. Working through practice exams appears to give participants exposure to problem formats and question types they are likely to encounter, allowing them to apply previously learned material to new but structurally similar situations — regardless of whether the underlying assessment emphasises theoretical reasoning or programming implementation.

These statements were deductively coded as ‘Critical Thinking: Practice Exams’ since they involve applying previously learned material to new and unfamiliar problems, approximating the conditions of the actual assessment rather than reviewing known content. For teachers, this suggests that the availability and quality of practice exam material is impactful for student preparation. If practice material does not vary sufficiently across instances it risks becoming rehearsal rather than critical thinking. For students, the key insight is in how practice exams are approached: attempting to construct answers independently before consulting solutions is essential for practice exams to serve as a critical thinking exercise rather than rehearsal.

6 Discussion

The results show that final-year CS students at TU Delft draw on different learning strategies depending on the type of assessment they are preparing for. Table 1 provides an overview of how each strategy identified in the results section maps onto Pintrich et al.’s (1991) categories, and Figure 1 illustrates their distribution across theory-heavy and programming-heavy assessments.

Results Section	**Learning Strategy**
5.1 Looking at the Lecture Slides for Activation of Information in Working Memory	Rehearsal
5.2 Note-Taking as an Effective Strategy for Conceptual Understanding	Elaboration
5.3 Cheat Sheets as an Effective Strategy for Summarising	Elaboration
5.4 Using Assignment Specifications to Identify Required Knowledge	Metacognitive Self-Regulation
5.5 Practice Exams for New Relevant Situations	Critical Thinking

Table 1: Mapping of results to Pintrich et al. (1991) learning strategy categories

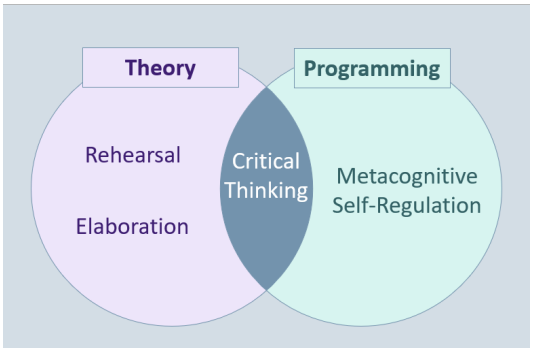


Figure 1: Distribution of learning strategies across assessment types

6.1 Rehearsal and Elaboration in Theory-Heavy Preparation

Participants' use of lecture slides as a starting point for preparation for theory-heavy assessment (Section 5.1) maps onto rehearsal strategies, which involve the repeated review of material rather than active transformation of it. That this strategy appeared early in the preparation process aligns with the surface learning phase described by Hattie and Donoghue (2016), in which students first acquire and consolidate foundational knowledge before moving toward deeper engagement with material.

Note-taking and cheat sheet creation (Section 5.2 and Section 5.3) both map onto elaboration strategies, as they involve transforming and condensing material into a personally organized form rather than simply repeating it. A pattern in which the process of creating study material — rather than the resulting artefact itself — appears central to learning, is consistent with Erbe (2007) finding that the act of preparing a cheat sheet was often sufficient for learning, regardless of whether the sheet was used during the exam itself.

6.2 Metacognitive Self-Regulation in Programming-Heavy Preparation

Participants' use of assignment specifications to determine the knowledge needed for programming-heavy work (Section 5.4) maps onto metacognitive self-regulation, since it reflects a planning step in which students assess task demands before beginning the work itself. This indicates that, for programming-heavy assessments, students' first strategic act is not to revisit taught material directly (as in the theory-heavy case) but to extract requirements from the task itself and regulate their subsequent study accordingly. This aligns with Biggs and Tang's (2011) concept of backwash, which describes how assessment determines what students focus on learning more than the curriculum itself does.

6.3 Critical Thinking Across Both Assessment Types

The use of practice exams (Section 5.5) was the one strategy reported across both theory-heavy and programming-heavy, and was coded as critical thinking in both contexts. Pintrich et al. (1991) define critical thinking as the application of previous knowledge to new situations — and this is precisely what working through a practice exam requires, regardless of whether the "new situation" is an unfamiliar proof or an unfamiliar programming problem. This suggests that what differs between theory-heavy and programming-heavy preparation is not whether critical thinking is required, but what students are applying their existing

knowledge to. In context of the Hattie and Donoghue (2016) three-phase model, it aligns with the transfer phase. That a single, generalisable strategy — independent practice on novel but representative problems — supports transfer-level critical thinking in both highly different assessment formats suggests that practice exams may be one of the more domain-general tools available to computer science educators.

6.4 Limitations on Interpretation

This study has several limitations that should be considered when interpreting its findings. First, the 10-week time frame of this bachelor thesis constrained both the time available for participant recruitment and the depth of analysis possible for each transcript; a longer study could have allowed for a larger sample and a more extensive analytical process. Second, the study was based on interviews with only three participants; while this is consistent with the study's aim of analytical rather than statistical generalisation, it limits the extent to which the reported patterns can be assumed to hold across computer science students more broadly. Third, all participants were drawn from a single institution, TU Delft; since these students share a common educational system, grading culture, and curriculum structure, the learning strategies reported here may reflect norms specific to this institution rather than patterns that would generalise to computer science students at other universities. Finally, the specific mix and structure of theory-heavy and programming-heavy assessments at TU Delft, may not be representative of how these two types of assessment are balanced or designed in computer science curricula elsewhere, meaning that the strategies associated with each assessment type here may not transfer directly to a different institutional context.

7 Conclusions

This study investigated what learning strategies final-year computer science students adopt when preparing for theory-heavy assessments compared to programming-heavy assessments. Through semi-structured interviews with three final-year Computer Science and Engineering students, and a deductive coding analysis based on Pintrich et al. (1991) framework, this study found that rehearsal and elaboration strategies — reviewing lecture slides, note-taking, and creating cheat sheets — were predominantly associated with theory-heavy preparation, while metacognitive self-regulation, in the form of consulting assignment specifications to determine required knowledge, was predominantly associated with programming-heavy preparation. Critical thinking was the one strategy reported across both assessment types, manifested through the use of practice exams.

The main contribution of this study is the observation that practice exams function as a strategy supporting critical thinking, regardless of whether the assessment being prepared for is theory-heavy or programming-heavy. Working through practice exams appears to exercise the same underlying skill — applying existing knowledge to unfamiliar but representative problems — across fundamentally different types of computer science assessment. This suggests that the design of practice exam material, specifically its relevance and diversity, may be a more impactful lever for supporting effective learning than has previously been recognised, since poorly varied practice material risks training rote rehearsal rather than the transfer-oriented critical thinking that students in this study associated with genuine exam preparedness. Future research could build on these findings by testing, with a larger sample, whether increasing the abundance and diversity of relevant practice exam questions measurably improves students' performance on novel, previously unseen problems.

References

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education*, *32*(3), 347–364. https://doi.org/10.1007/BF00138871

Biggs, J., & Tang, C. S.-k. (2011). *Teaching for quality learning at university: What the student does* (4th ed.). McGraw-Hill; Society for Research into Higher Education; Open University Press.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, *23*(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

Bloom, B. (1956). Bloom's taxonomy.

Boudia, C., Benguedach, A., & Haffaf, H. (2019). Collaborative Strategy for Teaching and Learning Object-Oriented Programming course: A Case Study at Mostafa Stambouli Mascara University, Algeria. *Informatica*, *43*(1). https://doi.org/10.31449/inf.v43i1.2335

Delft University of Technology. (n.d.). *Reading guide* [TU Delft Teaching Support]. https://www.tudelft.nl/teaching-support/educational-advice/assess/guidelines/assessment-framework/reading-guide

Eckerdal, A., Berglund, A., & Thuné, M. (2024). Learning programming practice and programming theory in the computer laboratory. *European Journal of Engineering Education*, *49*(2), 330–347. https://doi.org/10.1080/03043797.2023.2294953

Erbe, B. (2007). Reducing test anxiety while increasing learning: The cheat sheet. *College Teaching*, *55*(3), 96–98. https://doi.org/10.3200/CTCH.55.3.96-98

Friese, S. (2019). *Qualitative data analysis with atlas.ti*. SAGE Publications.

Hattie, J., & Donoghue, G. (2016). Learning strategies: A synthesis and conceptual model. *npj Science of Learning*, *1*, 16013. https://doi.org/10.1038/npjscilearn.2016.13

Isong, B. (2014). A Methodology for Teaching Computer Programming: First year students' perspective. *International Journal of Modern Education and Computer Science*, *6*(9), 15–21. https://doi.org/10.5815/ijmecs.2014.09.03

Joughin, G. (2009, January). *Assessment, learning and judgement in higher education*. https://doi.org/10.1007/978-1-4020-8905-3

Krathwohl, D. R. (2002). A revision of bloom's taxonomy: An overview. *Theory Into Practice*, *41*(4), 212–218. https://doi.org/10.1207/s15430421tip4104_2

Miles, M., & Huberman, A. (1994). *Qualitative data analysis: An expanded sourcebook*. SAGE Publications. https://books.google.nl/books?id=U4lU_-wJ5QEC

Patton, M. Q. (2015). *Qualitative research & evaluation methods: Integrating theory and practice* (4th ed.). SAGE Publications.

Pintrich, P. R., Smith, D. A. F., Garcia, T., & McKeachie, W. J. (1991). *A manual for the use of the motivated strategies for learning questionnaire (mslq)* (tech. rep.). National Center for Research to Improve Postsecondary Teaching and Learning, University of Michigan. Ann Arbor, MI.

Thuné, M., & Eckerdal, A. (2019). Analysis of Students' learning of computer programming in a computer laboratory context. *European Journal of Engineering Education*, *44*(5), 769–786. https://doi.org/10.1080/03043797.2018.1544609

Weinstein, C. E., & Mayer, R. E. (1983). The teaching of learning strategies. *Innovation abstracts*, *5*(32), n32.

A Interview Guide

Bridging computer science and learning science: an interdisciplinary research project

Pre-interview (3min)

- Thank you for your time. Self-introduction. The purpose of this interview is to understand your perspectives on [TOPIC]. - What learning strategies do final-year CS students adopt when studying for programming-heavy assessments compared to theory-heavy assessments?
- There are no right or wrong answers I'm interested in your perspectives.
- Sign the consent form!
- With your permission, I'll record this interview for transcription.

Part I: Getting to know the participants (5min)

1. I know you are a final-year student, can you tell me more the courses you liked the most? Why? What variant and electives?
2. How would you describe your experience with CS so far — do you consider yourself more of a "programmer" type, a "theory" type, or somewhere in between?

Theory-heavy assessments are CS assessments are those whose primary learning outcomes involve mathematical reasoning, formal proofs, and abstract concepts — where students are assessed mainly on their ability to understand, derive, and apply formal models rather than build software.

Programming-heavy assessments are CS assessments whose primary learning outcomes involve writing, designing, and evaluating code — where students are assessed mainly through programming assignments, projects, or implementations.

Part II: Learning in Programming-Heavy Courses (15 min)

1. Think about one of the programming heavy assignments. What was the first thing you looked at when studying for it?
   What did you do after that?
   How did you study for it? Interleaving theory and practice. Or first all theory and the practice?
2. Same level of theory?
3. Think about one of the programming heavy exams. What was the first thing you looked at when studying for it?
   What did you do after that?
   How did you study for it?
4. Ask for another example.

Part III: Learning in Theory-Heavy Courses (15 min)

1. Think about one of the theory heavy assignments. What was the first thing you looked at when studying for it?
   What did you do after that?
   How did you study for it?
2. Think about one of the theory heavy exams. What was the first thing you looked at when studying for it?
   What did you do after that?
   How did you study for it?

3. Ask for another example.

Part IV: Comparing the Two (8 min)

4. Which type of course do you find harder to study for?
   Why?
5. Which type of assessment did you spend the most time on? How much?
6. Do you notice any difference in the way you study for the courses?
7. Do you think it can be done better?
8. What do you think of the lack of theory-heavy assignments.

Closing off (2min)

- We're looking to understand learning strategies that final-year students have used. Can you think of fellow students that we should talk to?
- That's all from our side, is there anything you would like to add, or do you have any questions for me?

Course	Programming-Heavy	Theory-Heavy
CSE1100 Object-oriented programming	The exam	
CSE1105 OOP Project	The project	Oral presentation.
CSE1110 Software Quality and Testing	The exam and assignments	
CSE1200 Calculus		The exam
CSE1205 Linear Algebra		The exam
CSE1210 Probability Theory and Statistics		The exam
CSE1300 Reasoning and Logic	The project	The exam
CSE1305 Algorithms and Data Structures	The exam	The exam
CSE1400 Computer Organisation	The projects	The exam
CSE1405 Computer Networks		The tutorials. The exam
CSE1500 Web and Database Technology	The project	The exam.
CSE1505 Information and Data Management	The assignments.	The exam
CSE2000 Software Project	The project.	
CSE2115 Software Engineering Methods	The projects	The exam.
CSE2120 Concepts of Programming Languages	The projects. (The exam)	The exam
CSE2215 Computer Graphics	The projects	The exam
CSE2220 Signal Processing		The exam
CSE2225 Image Processing	(The assignments)	(The assignments) The exam.
CSE2230 Multimedia Analysis	(The	The exam
CSE2310 Algorithm Design	The exam.	The exam.
CSE2315 Automata, Computability and Complexity		The exam
CSE2420 Digital Systems		The exam
CSE2425 Embedded Software	The project	The exam
CSE2430 Operating Systems	The projects	The exam
CSE2510 Machine Learning	The assignments . (The exam)	(The exam)
CSE2520 Big Data Processing	The projects. (The exam)	(The exam)
CSE2525 Data Mining	The project	The exam
CSE2530 Computational Intelligence	The project	The exam
CSE3100 Functional Programming	The project. (The exam.)	(The exam)
CSE3130 Introduction to Quantum Computer Science		The assignments and the exam.
CSE3210 Collaborative Artificial Intelligence	The projects	The exam
CSE3220 Computer Security	(The project)	(The project) The exam
CSE3300 Algorithms for NP-Hard problems	The project	The exam
CSE3500 Human Computer Interaction		The project. The exam

B Informed Consent Form

Informed Consent Form

You are being invited to participate in a course-related project titled “Bridging computer science and learning science: an interdisciplinary research project”. This study is being done by Sebastian Iacovino Spitaleri and supervised by Dr. Xiaoqi Feng, Assistant Professor Martin Skrodzki, and Associate Professor Rafael Bidarra from TU Delft.

The purpose of this project is to deepen our understanding of how the best learning strategies differ between theory and programming within Computer Science. Insights will be used for research thesis as well as to inform teaching and learning practices as well as potentially for academic publications.

This research project includes interviews (approx. 30 minutes) on their perceptions of curriculum design, learning experiences, and support needs. The data collected will be used primarily for learning purposes and potentially publication.

As with any online activity, the risk of a breach is always possible. To the best of our ability, your answers in this study will remain confidential. We will minimize any risks by:

1. *Keeping the data only accessible for the project researchers, and ensuring that participating in this research will not have an impact on student’s course evaluation in any way.*
2. *Removing identifiable information from student interview data as much as possible by the corresponding researcher. In the case of direct quotation, we will attribute it to a general term such as ‘a student’.*
3. *Removing interview recordings after transcription is done, and storing survey and transcripts securely in TU Delft approved SURFdrive, accessible only to the project researchers.*

Your participation in this study is entirely voluntary. For the interview, you are free to refuse answering any questions. You may request removal of your data up to two weeks after the interview. After this point, anonymization may make withdrawal impossible. All interview recordings will be transcribed securely, after which the recordings will be deleted. Any identifying information that may be stated in the audio recordings will be removed from the transcription.

If you have any questions about the study, please contact.

PLEASE TICK THE APPROPRIATE BOXES	Yes	No
A: GENERAL AGREEMENT – RESEARCH GOALS, PARTICPANT TASKS AND VOLUNTARY PARTICIPATION		
1. I have read and understood the study info, or it has been read to me. I have been able to ask questions and have my questions answered to satisfaction.	☐	☐
2. I consent voluntarily to be a participant in this study. I understand that I can refuse any question. I can withdraw at any point without any consequences..	☐	☐
3. I understand that taking part in the interview involves answering questions on my perceptions on the curriculum and teaching/learning experiences.	☐	☐
B: POTENTIAL RISKS OF PARTICIPATING (INCLUDING DATA PROTECTION)		
4. I understand that taking part in the study involves collecting personally identifiable information (PII), specifically my name, signature and/or email address. I understand that my name and signature will only be use for the consent form and administrative purposes, such as scheduling, and will not be used for the research output.	☐	☐
5. I understand that taking part in the study also involves collecting personally identifiable research data (PIRD), specifically my speech in an audio recording. I understand that this audio recording will not be shared outside of the research team, and will be deleted within ten weeks of the recording being made, after a transcription is created. I understand that the transcription of the recording will be anonymized, and that the transcription procedure will be secure.	☐	☐
C: RESEARCH PUBLICATION, DISSEMINATION AND APPLICATION		
6. I understand that after the research study, the de-identified information I provide will be used in research thesis and potentially academic publications.	☐	☐
D: (LONGTERM) DATA STORAGE, ACCESS AND REUSE		
7. I give permission for transcriptions to be stored securely on a private SURFdrive directory hosted by the TU Delft. I understand that this directory will only be accessible to the responsible and corresponding researchers.	☐	☐

Signatures

_____	_____	_____
Participant name [printed]	Signature	Date

I, as researcher, have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

_____	_____	_____
Researcher name [printed]	Signature	Date