Improving Tutors' Practice by Integrating Explicit Instructional Programming Strategies in Introductory Programming Courses at University

Olivier Goletti

Thesis submitted in partial fulfillment of the requirements for the Degree of Doctor in Engineering Sciences and Technology

January 2025

ICTEAM
Louvain School of Engineering
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Thesis Committee:

Prof. Kim **Mens** (Supervisor)	UCLouvain/ICTEAM, Belgium
Prof. Felienne **Hermans** (Supervisor)	VU Amsterdam, The Netherlands
Prof. Mariane **Frenay**	UCLouvain/IPSY, Belgium
Prof. Virginie **März**	UCLouvain/IACS, Belgium
Prof. Jan **Vahrenhold**	Universität Münster, Germany
Prof. Charles **Pecheur** (Chair)	UCLouvain/ICTEAM, Belgium
Prof. Efthymia **Aivaloglou**	TU Delft, The Netherlands

Improving Tutors' Practice by Integrating Explicit Instructional Programming Strategies in Introductory Programming Courses at University

by Olivier Goletti

ICTEAM
UCLouvain
Place Sainte-Barbe, 2
1348 Louvain-la-Neuve
Belgium

Abstract

Context. Often, university-level introductory programming courses (CS1) make use of undergraduate teaching assistants (UTAs). Such UTAs generally are responsible for most of the students' interactions with the CS1 course. UTAs usually have no pedagogical background. While some universities provide pedagogical training and research documents these trainings, challenges and interactions, pedagogical content knowledge specific to the teaching of programming has not been extensively researched.

Objective. In this dissertation, we explore how UTAs' teaching practices can be improved by training them with dedicated evidence-based explicit instructional programming strategies. Moreover, we want to understand and characterise how UTAs use and adapt these strategies, what hinders and helps them.

Methods. Based on learning transfer processes and cognitive load theory, we selected four strategies: explicit tracing, subgoal learning, Parsons problems and explicit problem solving. We used a design research approach to iterate on our interventions on the instructional design of the course's setup in three different years. This iterative process allowed us to gradually tailor explicit tracing and subgoal learning to the studied CS1 course. We progressively increased the number of UTAs participating in our study, while refining our intervention up to a full integration of the subgoal learning strategy throughout the course. We collected data on tutors' usage of the instructional strategies by mixing methods: semi-directed interviews, surveys and observations. The data was analysed using thematic analysis with both inductive and deductive coding, and the concept of fidelity of implementation.

Findings. This dissertation reports on the results of this iterative process. We found that integrating subgoal learning throughout the course pushed UTAs to apply explicit instructional programming strategies in their practice. We also report on triggers for strategy uses by UTAs, adaptations made by UTAs and UTAs' fidelity of implementation of explicit tracing and subgoal learning.

Implications. Since UTAs are widely used in CS courses, this work provides research-informed actionable advice to practitioners willing to integrate explicit instructional strategies in their own setup. We also discuss the feasibility of applying these strategies to school teachers.

Acknowledgments

Prof. Kim Mens was one of my co-supervisors for this thesis. He embarked with me on an unknown journey towards computing education research which was not his main field of research. Nevertheless, depite this being a bit out of his comfort zone, he was very eager to learn with me, bought and read all books I threw at him, and never failed to ask me the right itchy questions on my research. His vast research experience made him an invaluable source of feedback through innumerable comments on my countless drafts. His teaching experience and interest made him genuinely interested in my work and he defended it and applied it when and where he could. We share the love of beer, sports, friendship and a common taste for bad jokes. Kim, thank you for letting me chose my way forward and for all the help provided.

Prof. Felienne Hermans was my other co-supervisor for this thesis. Felienne and I first discussed explicit instruction in 2008 at Educode, Brussels and this definitely made me fall for her research approach and her personality. Felienne is a field expert and shared with me her research experience, her honest feedback and provided the right references. She guided me to always self-reflect on my research assumptions and decisions, but also on how and where to publish. She has strong opinions and demanded a high standard of writing quality but all her comments made my work better. She never hesitated to accept me as a PhD, included me in her team and welcomed me in her house. We share the love of reading, playing boardgames and arguing around good food and drinks. Felienne, I am really thankful and proud for counting you as a friend now.

I would also like to thank my supervising committee members: Prof. Mariane Frenay who, however busy she was, could find the time to meet me, offer me her opinions on constructivism, learning transfer and observation methods when I was still a baby researcher; Prof. Virginie März who pushed me to situate my research in the broader perspective of education literature, teachers and UTA identity and motivation; Prof. Jan Vahrenhold who advised me to not go too broad, focus more on what I wanted as a final result and who would always give me (scary) honest feedback.

I thank the entire CS1 teaching staff and especially Charles, Kim and Siegfried who were always supportive of my ideas. They provided constructive feedback on our design interventions. More broadly, I want to thank

all TAs and UTAs of the course. Without them, the course could not run so smoothly.

Prof. Efthymia Aivaloglou and Prof. Charles Pecheur, thank you for accepting to read my thesis by joining the jury.

Thank you Chantal Poncin and Prof. Olivier Bonaventure, who recruited me in 2016 by offering me a position at the intersection of CS and education. This started my career in the INGI departement and in computing education research. I also want to thank Julie Henry who motivated me to pursue a PhD in this field.

Thank you to the whole INGI department, all my colleagues during the five years of my thesis and three more before that. I was privileged to play cards, sports or crossword puzzles with you all. I made good friends, watched movies, role played, and practiced yoga. Citing names would be too long, but thank you to the entire administrative staff, the (ex-)PhD students, the (past and actual) sysadmin team and all professors. Special thanks to those who shared my office and supported me! A dedicated thank you to Gorby and his drawing skills showcased on the next page at the occasion of the "My thesis in 3 minutes" competition. Special mention here to Fantasia, the best place in town for a fresh pasta takeaway.

I also want to thank Felienne's research team. Being alone on a computing education research topic has been challenging at times, and being able to share ideas, give an opinion, help others, read related work made by colleagues and even contribute to others' research has really been a treat for me. Thank you all so much for making me feel welcome in your team!

Special thanks to my family who always were supportive. Merci maman, merci papa! Thanks to you, I've had the privilege to be able to experience this successful educational and professional journey. Thank you sis', thank you bro' and your wonderful families!

My dearest thanks to my beloved partner Delphine. She is the best! She always supported me even through the late nights of work or my grumpiness when stressed before a deadline or a talk. Thank you Marion for keeping me young in my head and teaching me the last cool slang. I love you both so much! Special mention to Rufio, my good dog.

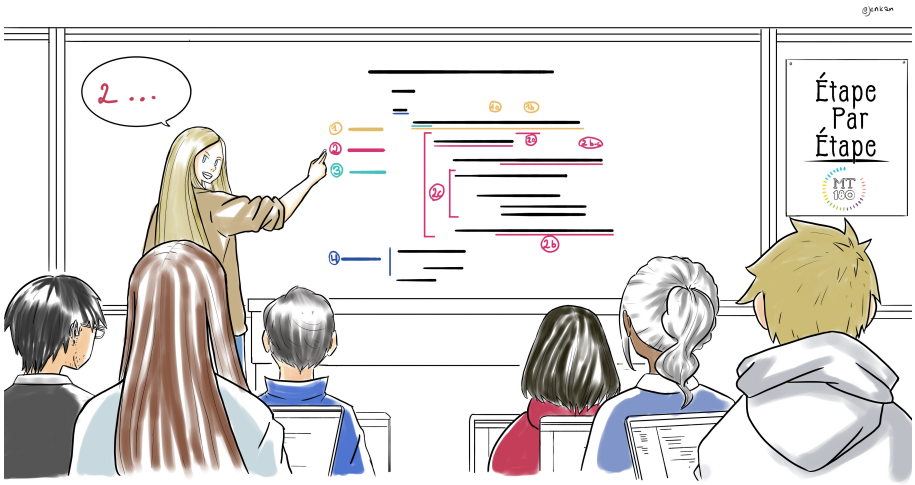


Illustration of a tutor using subgoal learning drawn by Gorby – AKA jenkan – at the occasion of the "My thesis in 3 minutes" competition.

Table of Contents

List of Figures

List of Tables

Preamble 1

Motivation

As a teaching assistant of a CS1 course, my interest in the pedagogical methodology used when teaching programming, and in particular in the balance between explicit teaching strategies and the course's project-based approach led me to the research done in this thesis. The origin of this reflection comes from reading various critiques of constructivist approaches during my educational studies conducted after my engineering studies. Notably, work by Kirschner et al. [KSC06] stating that less guided approaches to learning are less effective. Various responses have been proposed to them [HDC07; Sch+07] insisting on the strong presence of scaffolding in problem-based learning (PBL), as well as suggesting the possibility that limitations of problem-based approaches might be due to a lack of research. Their effectiveness also depends on the discipline, as suggested by Galand et al. in their own analysis of the civil engineering setup effectiveness at UCLouvain [GFR12]. This last answer seems to suggest that the approach could still be effective in the context of engineering studies.

Positionality

Since part of the work presented in the dissertation is qualitative and hence subject to some interpretation based on the researcher's background, it is relevant to provide some information about my background.

I identify as a white young researcher in computing education (CEd). I graduated as MSc in Computer Science (CS) and have started my PhD after having worked as a developer, as a science and math teacher, and as an in-service teacher trainer for CS in primary and secondary schools. Both my interest in education and computer science led to my return to university as a research assistant. Before starting my research, I first followed two master-level classes on pedagogical design analysis and qualitative research.

I also have been a teaching assistant myself for the CS1 course that I studied during my research and even before. I followed that same CS1 course (still in Java at the time) as a student in 2006, was a tutor for the course in 2010 and have been a teaching assistant for the course since 2016

(it shifted to Python in 2018).

In my work, I tried to bring together as much as I could of both worlds of qualitative and quantitative research and of both worlds of less and more guided learning. A more elaborate and deeper reflective exercise on my journey and positionality as a CEd researcher can be found in Section 3.1.

1.1 Research Goals

Introductory programming courses (CS1) have been described as challenging [McC+01; Lis+04]. Literature describes programming as a hard to learn topic but it does not have to be that way. Luxton-Reilly suggests it might be attributed to unrealistic expectations or unquestioned teaching practices [Lux16]. Since a lot of CS departments use (under)graduate teaching assistants - (U)TAs - as teaching staff for their courses [For+17; DDL17], we decided to assist and train them with more pedagogical content knowledge ("subject matter knowledge *for teaching*" [Shu86]).

The goal of this thesis is to investigate how and if integrating explicit instructional strategies in the training of course tutors could improve the effectiveness of the course setup. In short, far from pedagogical disagreement, in this thesis we aim at combining the best of both worlds [deJ+22] by bringing research, here evidence-based explicit instructional programming strategies, in the practice [Sen21] of tutors in a problem-based learning setup.

Based on the learning transfer processes and cognitive load theory, this thesis aims at identifying explicit instructional strategies that will effectively help undergraduate teaching assistants when tutoring their students learning how to program.

To understand how to effectively use these strategies in CS education, here are the main questions we address:

RQ1 How and what instructional programming strategies to select to integrate in a CS course, to enhance UTAs' teaching practice?

RQ2 What makes explicit instructional programming strategies easily usable and applicable by UTAs? What are their obstacles to use these strategies?

RQ3 How to train and support UTAs to integrate explicit instructional programming strategies in their teaching practice?

RQ4 How do tutors use and adapt explicit instructional programming strategies in their teaching practice?

This dissertation studies **how interventions in the instructional design of an introductory programming course can encourage undergraduate teaching assistants to make use of evidence-based strategies in their practice**. More details on our findings can be found in Chapter 10. Since our goal is to bring research to practice, we propose a series of actionable advice to practitioners based on our research results in Section 10.3.

Research Statement. Interventions on the instructional design of an introductory programming course can push tutors to make use of evidence-based explicit instructional programming strategies in their practice, in particular subgoal learning. The interventions must include tutors' training and follow-up as well as strategy integration in the course material. This will enhance tutors' teaching practice since they often lack didactic knowledge specific to programming teaching.

1.2 Approach

Table 1.1: For each chapter, the research questions studied, the used data sources, type of research and outputs are provided. * indicates when material to answer the main RQ comes from the analysis of the instructional intervention presented in the chapter but not through a specific RQ.

Chap.	RQs	Data source(s)	Type of research *(output/themes)*
5	1	Course material	document analysis *(three proposals for instructional intervention)*
6	1*,2,4	semi-structured interviews; recordings of focus groups	coding *(UTAs' perception)*
7	3	experts' live solutions	Task analysis *(subgoals for selected Python concepts)*
8	3*,4	recordings of tutors' lab sessions	thematic analysis; fidelity of implementation *(triggers, adaptations, fidelity of implementation)*
9	2,3*,4	recordings of tutors' lab sessions; surveys	thematic analysis, fidelity of implementation *(UTAs' perception, fidelity of implementation)*

To answer the research questions, the research carried out in this dissertation used mixed methods. Table 1.1 shows an overview of the different studies and research methods used. For readability, the main research questions of this dissertations use the notation RQX while research questions of a specific chapter use the notation RQY.Z where Y is the number of the chapter and Z the number of the question. Table 1.2 presents the mapping between the main RQs and more specific RQs of the different chapters which will be introduced in more detail later. For example, RQ1 is mainly answered through RQ5.1 as wells as an analysis of the various instructional strategies explored in Chapter 6. This section details the research approach we followed and gives a label (in bold) to each study for clarity of the main thread of the dissertation.

Table 1.2: Mapping between the main RQs of this dissertation and the specific RQs of each chapter. * indicates when material to answer the main RQ comes from the analysis of the instructional intervention presented in the chapter but not through a specific RQ.

Chapter	RQ1	RQ2	RQ3	RQ4
5: Analysis	RQ5.1			
6: Exploration	*	RQ6.2,3		RQ6.1
7: Creation			RQ7.1,2	
8: Characterisation		*		RQ8.1-3
9: Integration		RQ9.3	*	RQ9.1-3

We started this work with an **analysis** of a CS1 course based on two processes of learning transfer. The first is the encoding of new knowledge or the initial learning phase of a specific bit of knowledge. The second is the access to this knowledge in a situation of transfer, ie. when the initially learned bit of knowledge has to be retrieved and reused based on a new task to solve, maybe leading to new knowledge being learned. This analysis was done to better understand the structure of the course to assess if and how well the course material supported the chosen educational theory of learning transfer and to identify potential area of improvements. The conceptual model and analysis are presented in **Chapter 5**. Based on this analysis, we made three proposals to improve the course to foster learning transfer by using explicit recall strategies and equipping tutors with recall strategies for students.

Next, we started a process of design research to study how best to integrate a selection of explicit strategies into tutors' practice. We conducted a **first exploratory iteration** presented in **Chapter 6**. During this first iteration, we trained four undergraduate teaching assistants with four strategies and interviewed them at the end of the course. The coding and analysis of

the interview transcripts led us to introduce four criteria to select instructional strategies: the strategy should be easy to understand, straightforward to apply, useful on the long term and supported by literature.

In a second iteration of the design of the course, those criteria and our analysis of the first iteration led us to select only two strategies in order not to overload our UTAs: explicit tracing and subgoal learning (SL). We decided to introduce both strategies earlier in the semester and created dedicated training material to help UTAs integrate the different aspects of the strategies. While in the first iteration, we used subgoals from the literature, in the second iteration, we developed our own subgoals for the concepts and language of our own CS1 course. The process of **creating these subgoals** is presented in **Chapter 7**. For this **second iteration**, we focused on the **characterisation** of the adoption of the strategies by the tutors. By characterisation, we mean describing and categorising based on observation the triggers, adaptation and fidelity of implementation of the strategies by the tutors. We measured the fidelity of implementation of the two strategies by the UTAs by observing, recording and coding recordings of their teaching. We identified three categories of triggers to strategy use: students' triggers (like a question or a mistake), tutors' triggers (like an important concept being presented before a difficult exercise) and exercises themselves as triggers when then contained already existing explicit prompts to trace code. We also listed adaptations made by UTAs when using both strategies. The second iteration is presented in **Chapter 8**.

The most mentioned issue by UTAs in the focus groups and follow up discussions during the second iteration was the time it took them to properly present a specific worked example and to introduce the corresponding subgoals. UTAs argued that they would benefit from an introduction of the labels done in the course material. Moreover, we observed that for the other strategy used in the second iteration, namely explicit tracing, an important trigger for strategy use was to have prompts in students' exercises statements. Based on this, for our **third iteration** focusing only on the strategy of subgoal learning (SL), the main design focus was **the integration** of the strategy globally throughout the course. Our design for the integration of SL throughout the course for the third iteration takes this into account by providing a mandatory training for all UTAs and presenting the labels during lectures. The third iteration is presented in **Chapter 9**. This study was an observational study where we analysed UTAs' strategy use by observations and surveys. The main results of this third iteration are that SL integration through exercises is a major trigger of strategy use. We also found that training and follow-up of the UTAs has impact, since we reported frequency of use correlates positively with reported familiarity by UTAs. More frequent stronger uses of the strategy were observed by UTAs self-reporting

more frequent uses. Finally, UTAs express that subgoals are best suited in order to introduce concepts and especially the more structured ones.

1.3 Supporting Publications

Following the main thread (in bold) presented in the previous approach section, here is a list of the publications this PhD dissertation builds upon. All these chapters but the analysis have been accepted and soon to be published in peer-reviewed conferences. I deliberately chose to publish to a large variety of venues in order to get feedback from a broader part of the computing education research community. A label (in bold) represents its place in the main thread of this work and the corresponding chapter presenting it.

Analysis: the analysis of the CS1 course which bootstrapped this thesis is available as an unpublished report and serves as basis for **Chapter 5**. This work was done in French in the scope of my educational training before my PhD and its title can be translated as "How does the problem-based learning approach implemented in the engineering CS1 course support the processes of learning transfer: knowledge encoding and knowledge accessibility?".

[Gol19] O. Goletti. *En quoi le dispositif mis en œuvre dans le cours d'introduction à l'informatique en BAC1 ingénieur civil basé sur l'apprentissage par problèmes soutient les processus du transfert des apprentissages : l'encodage et l'accessibilité aux connaissances ?* Tech. rep. `http://hdl.handle.net/2078.1/245579`. UCLouvain, 2019

Exploration: The first iteration of our intervention in the design of the CS1 course was the topic of a published ITiCSE paper and serves as basis for **Chapter 6**.

[GMH21] O. Goletti, K. Mens, and F. Hermans. "Tutors' Experiences in Using Explicit Strategies in a Problem-Based Learning Introductory Programming Course". In: *ITiCSE '21*. Virtual Event, Germany: ACM Press, June 2021. DOI: `10.1145/3430665.3456348`

Creation: The creation of subgoals and subgoal labeled worked examples adapted to our own CS1 course is published and presented in **Chapter 7**. This paper was published in the French-speaking venue Didapro and can be translated as "Creation of subgoal labeled worked examples for teaching programming with Python".

[GDM22] O. Goletti, F. De Pierpont, and K. Mens. "Création d'exemples résolus avec objectifs étiquetés pour l'apprentissage de la programmation avec Python". In: *Didapro 9–DidaSTIC*. 2022

This publication was realised with the help of Florian De Pierpont whose master thesis I co-supervised with Kim Mens:

F. De Pierpont. "Intégration des stratégies Subgoal Labeled Worked Exemples (SLWEs) et Explicit Tracing aux travaux pratiques du cours d'introduction à la programmation". MA thesis. UCL - Ecole polytechnique de Louvain, 2022. URL: http://hdl.handle.net/2078.1/thesis:35668

Characterisation: Our second iteration on the integration of explicit tracing and subgoal learning was published at Koli Calling and presented in **Chapter 8**.

[GMH22] O. Goletti, K. Mens, and F. Hermans. "An Analysis of Tutors' Adoption of Explicit Instructional Strategies in an Introductory Programming Course". In: *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research*. 2022, pp. 1–12. DOI: 10.1145/3564721.3565951

Integration: Finally, our last iteration was the integration of subgoal learning throughout the whole course. This led to a SPLASH-E publication and serves as basis for **Chapter 9**.

[GMH24] O. Goletti, K. Mens, and F. Hermans. "An Observational Study of Undergraduate Teaching Assistants' Use of Subgoal Learning Integrated in an Introductory Programming Course". In: *Proceedings of the 2024 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E '24)*. Pasadena, CA, USA: ACM Press, Oct. 2024. DOI: 10.1145/3689493.3689986

This publication was realised with the help of Antoine Demblon whose master thesis I co-supervised with Kim Mens:

A. Demblon. "Intégration de l'apprentissage par étapes dans les ressources d'un cours d'introduction à la programmation". MA thesis. UCL - Ecole polytechnique de Louvain, 2024. URL: http://hdl.handle.net/2078.1/thesis:48837

During this thesis, I participated in an ICER doctoral consortium and a Didapro doctoral consortium. Being selected and attending these two venues allowed me to reflect on my overall research plan and led to published extended abstracts:

[Gol21] O. Goletti. "Promoting Learning Transfer in Computer Science Education by Training Teachers to Use Explicit Programming Strategies". In: *ICER '17*. Virtual Event USA: Acm, Aug. 2021, pp. 411–412. DOI: 10.1145/3446871.3469776

[Gol24] O. Goletti. "Subgoal Learning Integration in a CS1 Course". In: *Colloque Didapro 10 Sur La Didactique de l'informatique et Des STIC*. 2024, pp. 131–135

Other interesting but less related works were published by the author. The first one is published at ICER as a collaboration on teachers' learning transfer strategies when they switch between programming languages and the two following publications were done in the scope of an Erasmus+ project — "Communauté d'apprentissage de l'informatique" — on the development of a community of practice for CS school teachers[1]:

[Tsh+21] E. Tshukudu, Q. Cutts, O. Goletti, A. Swidan, and F. Hermans. "Teachers' Views and Experiences on Teaching Second and Subsequent Programming Languages". In: *Proceedings of the 17th ACM Conference on International Computing Education Research*. ICER 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 294–305. ISBN: 978-1-4503-8326-4. DOI: 10.1145/3446871.3469752

[Cor+20] P. Corieri, M. Romero, T. Massart, O. Goletti, K. Mens, M. Rafalska, T. Viéville, L. Meziane, J. Christophe, S. Hoarau, et al. "Enjeux dans la création d'une communauté d'enseignants engagés dans l'apprentissage de l'informatique". In: *Didapro 8-DidaSTIC-Colloques francophones de didactique de l'informatique*. Poster. 2020

[Kom+22] V. Komis, S. Bachy, O. Goletti, G. Parriaux, M. Rafalska, and K. Lavidas. "Connaissances du contenu et connaissances technologiques des enseignants en Informatique en milieu francophone". In: *Review of Science, Mathematics and ICT Education* 16.2 (2022), pp. 105–133. DOI: 10.26220/rev.4080

1.4 Structure

This dissertation is organised in three parts. **Part I** covers the related work, context and background knowledge of my research. Chapter 2 introduces the theories and conceptual frameworks which form the bases of my choice of intervening in an introductory programming course. This background

[1] https://cai.community/accueil/qui-sommes-nous/partenaires/

also served as the basis for the selection of the instructional strategies. The chapter also presents what literature says on undergraduate teaching assistants, our target audience. Chapter 3 presents the pragmatic research paradigm, methodologies and tools used in the studies presented in this dissertation. It also presents the iterative design research method that guided the succession and impact of the different studies presented and the qualitative tools used in these studies. Chapter 4 presents in detail the setup of the introductory programming course in which the interventions took place.

Part II presents the different studies as detailed in Sections 1.2 and 1.3. Chapter 5 presents the **analysis** of the course based on learning transfer processes. Chapter 6 presents our first published **exploratory** study. Chapter 7 details the methodology used for the **creation** of our own subgoals. Chapter 8 presents the second iteration on the **characterisation** of tutors' use of two instructional strategies: explicit tracing and subgoal learning. Chapter 9 presents the last study on the full **integration** of subgoal learning in the CS1 course.

Part III concludes the dissertation. Chapter 10 summarises the overall results of the three iterations, answers our main research questions and provides actionable advice to practitioners willing to integrate explicit instructional strategies in their own course setup. Chapter 11 discusses future directions this work could take in the form of another possible target audience (i.e. school teachers) to train with such strategies and the possible challenges and differences we envision. This last chapter also concludes the dissertation.

Part I

Related Work

Computing Education Principles

2

This chapter starts with a broad presentation of computing education research on introductory programming (2.1). We then present the education principles that served as a theoretical justification for the selection of evidence-based strategies we trained our tutors with. The theoretical foundations of this work are the cognitive processes that lead to learning and to reusing learned knowledge in new situations. First, we present cognitive load theory (2.2) which informs us on the limitations of the working memory and proposes instructional design adaptations to avoid overloading students when learning. Then, we present learning transfer (2.3) that describes how learning is acquired and can be reused in similar tasks once available in our memory. We present the problem-based methodology (2.4) of the studied course. Next, we present the instructional strategies (2.5) selected based on the theory to lower cognitive load and foster learning transfer. Finally, we present background and related work on the use of undergraduate teaching assistants (2.6) in computer science courses.

2.1 Introductory Programming Education

Many disciplines have a long tradition of researching and understanding the challenges and possible improvements to teaching and learning their own subject. While introducing the whole field of computing education research seems overly ambitious and out of the scope of this work, we present in this section a brief overview of the field of introductory programming education research.

The field explored different areas of programming education such as: assessment [Kal17], introductory programming in schools [HMF16; Sza+19], teacher education [Men15], novices' difficulties [QL17] or automated feedback [KJH18]. More details can be found in literature reviews of the field [Pea+07; Lux+18] that identify and comment on research on curricula, teaching and pedagogy, language choice, assessment and tools for teaching in introductory programming.

Among interesting findings in programming education literature, we can name the concept of *notional machine* introduced by du Boulay [duB86]

and defined as a: *"pedagogic device to assist the understanding of some aspect of programs or programming"* [Fin+20] by a recent ITiCSE working group on the topic. When teaching, it is important to match that notional machine with the actual mental model of the student.

Regarding teaching approaches, since early research in computing and programming education, the field reported a lot on teaching with active student-centered practice. Papert contributed the *constructionism* [Pap80] learning theory, Guzdial advocated for student-centered teaching [Guz16] and contributed the contextualised computing pedagogy known as *media computation* [Guz03]. A review on different teaching approaches for introductory programming courses shows that the 60 studied interventions improve on average students' pass rates by a third from the pre-intervention setup [VAW14]. The most reported-on interventions in this review are: contextualisation, collaboration, preliminary courses, content change and peer support.

CEd literature also documented a long list of difficulties and misconceptions [QL17; Chi+21] among which Pea's infamous "superbug" that novices think a computer is somehow intelligent and will for example infer the programmer's intention [Pea86].

2.2 Cognitive Load Theory

Cognitive load theory (CLT) [Swe88] is a cognitive framework investigating links between human cognitive architecture and learning. CLT informs a lot of instructional design based on the fact that our working memory is limited and can be overloaded by too many information elements when learning. Part of the cognitive load is intrinsic to the complexity of the learning concept, but more load can be attributed to instructional design or teaching strategies. Too much context or unnecessary information when teaching increases the cognitive load on the learner and becomes an obstacle to learning. Many effects have been documented based on CLT [MS05; SvP19] such as: the worked example effect (cf. Section 2.5.2) which states that conventional problems can be replaced by the careful studying of worked-out examples; the completion problem effect (cf. Section 2.5.4) which states that conventional problems can be replaced with providing a partial solution the learner has to complete; the guidance-fading effect which states that scaffolding and guidance should be faded out to avoid being redundant and add more cognitive load. To reduce the load on novices, instructional design recommendations coming from documented cognitive effects have been highlighted [PRS03; MS05; SvP19] related to minimising cognitive load and helping to apply generic skills [SAK11]. This can be achieved through the use of instructional strategies such as worked examples [vPS10], automating

rules, faded guidance [SvP19] or using external representations [Kir02].

As detailed in the subsections of Section 2.5, most of the selected instructional strategies find some justifications in the scope of CLT. CLT is often mentioned to justify interventions in computing education[1] [DZS22].

2.3 Learning Transfer

Learning transfer is the process of reusing previously learned knowledge in a new situation. Transfer is a process studied in psychology and educational science since the work of Thorndike and Woodworth [TW01]. Transfer is a complex cognitive process [Tar99] deemed hard to achieve and dependent on a lot of parameters like: the context of the initial task or the domain-specific knowledge needed to learn it, the similarity between both the initial and the new tasks, the capacity of the learner to retrieve useful knowledge to work on the new situation or their capacity to even recognise that this new situation may benefit of previously acquired knowledge, etc. But it is also an active process that can be trained and acted upon by giving tools to the learners such as recall strategies or meta-cognitive strategies like analogical reasoning [Nat00]. Specifically in programming, transfer is also studied [IM21; Tsh+21] and considered hard to achieve. However, CS1 courses offer many opportunities to reuse conceptual and procedural knowledge from one exercise to another or from a concept like list traversal to parsing files or strings. Researchers argue that intervention on the design of instruction [GSM11] and the use of specific teaching strategies like worked-out examples or subgoal learning can help with learning transfer [MGC12]. A more elaborate description of learning transfer models is provided in Section 5.2.1 when describing the conceptual model used in our research.

2.4 Problem-Based Learning

Problem-based learning (PBL) is a student-centered instructional methodology where students work in small groups on real-life problems introducing new learning material [Bar96]. The learning process is mostly self-directed: by solving problems and answering questions, they discover the theory by reading and studying by themselves. PBL has sometimes been criticised as an active teaching approach, because the minimal guidance it provides is in contradiction with CLT principles [KSC06]. Other works nuance this cri-

[1] We are aware and deliberately decided not to discuss germane cognitive load mentioned in this review [DZS22] and by Nelson and Ko [NK18] since it does not apply to this work

tique by asserting that PBL does include scaffolding and guidance provided among others by tutors [Sch+07; HDC07].

2.5 Explicit Instructional Programming Strategies

We introduce here the term of "*explicit instructional programming strategy*", inspired by LaToza et al. [LaT+20] who define an *explicit programming strategy* as: a *"human-executable procedure for accomplishing a programming task."* Since here we are focusing on instructional strategies that will be used by tutors and not just explicit strategies to be used by programmers, we use "explicit instructional programming strategy".

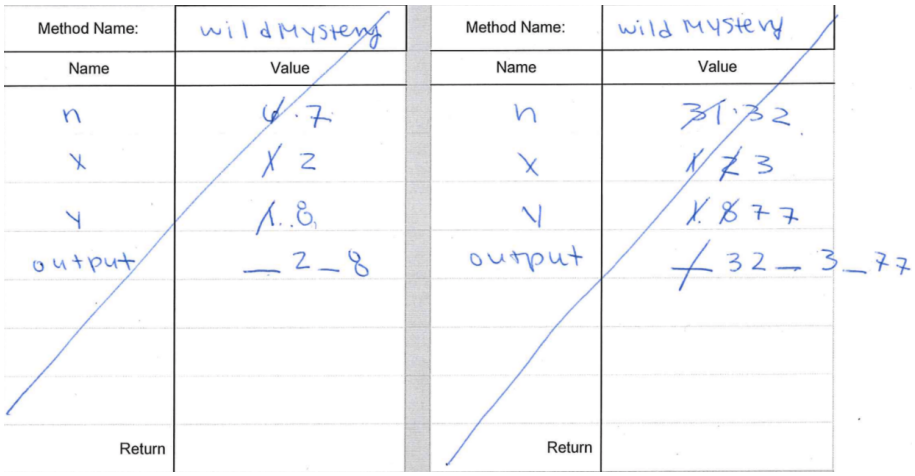
Cognitive load theory informs instructional design and states that instructional strategies that explicitly teach problem solving steps are beneficial to novice learners [KSC06; SvP19]. In computing education in particular, explicit instructional strategies are often mentioned in recent literature: e.g., by explicitly teaching recurring patterns [RTW07], goals and plans [HWC13] or using explicit strategies for tracing [XNK18], debugging [LaT+20], code reuse [Ko+19] or problem solving [Lok+16]. The use of worked examples and subgoal-labeled material as techniques aiming to reduce cognitive load when learning has also been explored in numerous recent studies [MCG13; MMG15; MMD19]. Xie et al. state that: *"The implications for teaching are simple: help students practice an explicit strategy"* [XNK18]. Those strategies can be either specific list of steps to reproduce, or metacognitive hints to help recall some specific technique.

The selected strategies for this thesis are aligned with CLT and promote learning transfer. In particular the subset of learning transfer processes of knowledge acquisition and organisation in long term memory and how to remember that knowledge properly when needed, i.e. being able to recognise target situations where previously acquired knowledge can be mobilised, used, adapted, modified and lead to new learning.

It is important to note that explicit instruction is not incompatible with PBL. It has been shown that explicit instruction can be used first, followed by PBL [AKS20] and other researchers advocate for using both [Sor12; deJ+22].

In the following subsections, we present the four strategies we have decided to work with in this thesis. We discuss the motivation and literature for each of them. We then provide descriptions of the actual implementations of these strategies as observed in the classroom and, when useful, provide information on the "baseline" – the way tutors usually teach prior to our interventions on tutors' teaching practices.

Figure 2.1: Two memory table examples with the method name at the top, variable names in the Name column, and variable values in the Value column (caption and figure extracted from Xie et al. [XNK18]).



2.5.1 Explicit Tracing

Tracing code is the process of executing a program in one's head or by hand, it is the equivalent of reading for code. Research shows that students who trace code better can explain and write code better [Lis+04; LFT09]. However, novice students learning to program often struggle with tracing [Lis+04]. This struggle can be attributed to cognitive overload. Indeed, their limited working memory cannot handle tracking the value updates of the different variables of a program simultaneously, which leads to tracing errors [VS07]. It has been proposed to teach explicitly to students a proper tracing strategy using an external representation of the memory to help reduce their cognitive load [VS07; Cun+17; XNK18].

In particular, Xie et al. [XNK18] proposed a strategy they introduced to students before tracing six problems. The strategy helped students trace programs while updating a memory table. This strategy proposes step-by-step instructions to systematically trace the program. Each variable encountered in the code has a line in the memory table and their values are updated when executing line by line the instructions of a given program. Figure 2.1 presents an example of such a memory table.

Research shows that combining the explicit teaching of a tracing strategy combining both the use of an external memory representation and a systematic line-by-line tracing procedure may improve tracing performance [HJ13; XNK18].

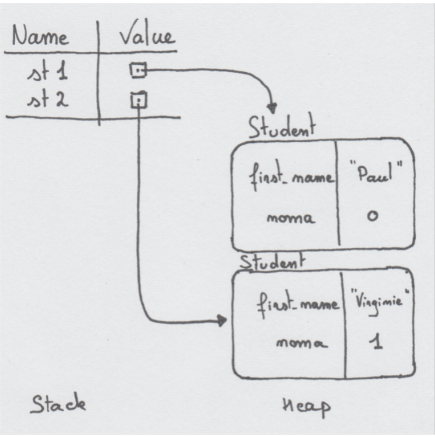
Actual Implementation

While in the first iteration of our design research approach (cf. Chapter 6), we mainly proposed to tutors to use the tracing strategy as described by Xie et al., in the second iteration (cf. Chapter 8), to support a more systematic representation of the external memory, the explicit tracing strategy was supplemented with a standardised memory representation for more complex structures from Dragon and Dickson [DD16; DD21]. Figure 2.2 presents examples of our own training material and usage of the explicit strategy.

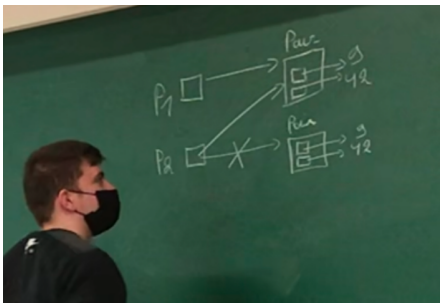
It is important to note for this strategy that some efforts were already being made in the course material to ask students to make some drawings and sketches of data structures such as lists, objects or linked lists (cf Chapter 4). This means that in the "baseline" i.e. the course's design before we introduced the explicit tracing strategy, we could already expect students to use some sort of memory representation. However, it was not explicitly asked from the tutors that they reuse such memory representation outside of the prompted exercises. It was also not the case that tutors used a systematic line-by-line approach. Neither was it the case that they used a standardised memory representation. The introduction of this strategy was even beneficial to teachers to harmonise their own sketches in their slide-decks and throughout the course material.

Figure 2.2: Examples of training material and usage of the explicit tracing strategy.

(a) Object representation inspired by Dragon and Dickson [DD16; DD21] from tutors' training documents.



(b) A student using explicit tracing during a practical lab session.



2.5.2 Worked Examples

Worked examples is a type of exercise aligned with the worked example effect [SvP19] and encouraged by CLT informed instruction design [vPS10]. Among the effects predicted by cognitive load theory, the worked example effect shows that exposing students to detailed solutions to problems improves their learning [WS90]. Worked examples have been successfully use in problem-based learning setups, research suggesting they provide novices more quickly with information need to generalise knowledge [Sal+10]. We introduce worked examples here as they support two of our selected strategies: subgoal learning and Parsons problems. For more information about the use of worked examples in CEd, see the review by Skudder and Luxton-Reilly [SL14]. An known issue with worked examples is that students have to engage actively with them in order to benefit from the learning, but learners do not spontaneously engage in self-explanation [Ren+98]. However, self-explanation can be elicited through instruction or Parsons' problems.

2.5.3 Subgoal Learning

Subgoal learning was introduced by Catrambone [Cat98] with the idea to foster learning transfer [CH89] by highlighting the generic potential in solutions. It combines the worked-example effect and the annotation of steps or subgoals with labels, in order to highlight the generic structure of a problem solving procedure. Worked examples have been studied and shown to help students learn programming [MMG15; EMR17]. Highlighting and explicitly teaching the steps of recurring patterns in the resolution of similar problems is not new in programming education [Sol86; JS87; RTW07].

Margulieux in her thesis under the supervision of Catrambone combined the idea of worked examples with labeling the steps behind common code structures, proposing subgoal learning (SL) with the help of subgoal labeled worked examples (SLWEs) in computing education [MMD19]. Figure 2.3 presents examples subgoals in Java from the work of Margulieux et al. [MMD19].

Figure 2.3: Subgoals for evaluating (left) and writing (right) expression (assignment) statements (caption and figure adapted from Margulieux et al.[MMD19]).

Subgoals for evaluating and writing expression (assignment) statements	
A. Evaluate expression statement	B. Write expression statement
1. Determine whether data type of expression is compatible with data type of variable 2. Update variable for pre based on side effect 3. Solve arithmetic equation 4. Check data type of copied value against data type of variable 5. Update variable for post based on side effect	1. Determine expression that will yield variable 2. Determine data type and name of variable and data type of expression 3. Determine arithmetic equation with operators 4. Determine expression components 5. Operators and operands must be compatible

Margulieux et al. showed that subgoal learning helps *"students learn more effectively without increasing the amount of time students take to learn"* [MCG13]. They argued that subgoal learning benefits come from the combined effect of worked examples and reduced load on the learners and that it aids the development of mental models which supports learning transfer. They also showed that students learning subgoals had lower variance in their exam results and had lower drop rates [MMD20]. Atkinson et al. also found that *"sequentially-presented examples with clearly isolated subgoals produce better conceptual performance"* [AD13]. Subgoal learning has also already been used to teach algorithmic [Cho+22] using high-quality subgoals collected from learners.

Actual Implementation

In our first iteration (cf. Chapter 6), we mainly proposed to tutors to use the SL strategy by presenting and using the subgoals as described by Margulieux et al. [MMD19]. We later decided to adapt the strategy to the concepts and programming language of the course (cf. Chapter 7), and to provide a more complete training document (cf. Chapter 8). Adapting the strategy to the course allowed us to later further integrate it in the course (cf. Chapter 9). In the second iteration, we asked tutors to present the full SLWE to students. This part was dropped in the third iteration since subgoals and labels had been integrated in the course material. In the third integration we introduce the distinction between *Step-By-Step* Subgoal labeled worked examples – SBS – and *Subgoal Labeled Final Solutions* – SLFS (cf. section 9.1.1. An SBS is a worked-out example where the solution is detailed by going through each subgoal step-by-step, developing and enriching the solution by applying each of the subgoal. This was typically used when teachers introduced concepts with SBS integrated in their slides or when a tutor would systematically go through all the subgoals of a concept to solve an exercise. An SLFS is an annotated final solution to a problem and is typically used either as a reminder of the labels for a specific concept or is the result of a tutor annotating with labels a student's solution on the blackboard. For both iterations, we asked tutors to mention the labels when needed to help struggling students or to remind them of the proper solving strategy when using a concept. We also asked them to label code segments on the blackboard to highlight subgoals in student's code. Figure 2.4 presents examples of the training material and usage of the subgoal learning strategy.

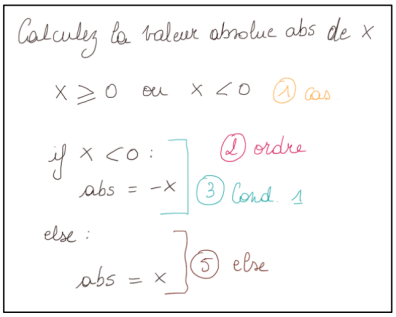
In the "baseline", before the introduction of subgoal learning, it was of course common for tutors to sometimes give theoretical reminders on important new concepts. However, when doing so, they did not use any kind of systematic labeled solving steps for explaining the concepts. Particular

Figure 2.4: Examples (in French) of training material and usage of the subgoal learning strategy.

(a) A list of subgoals to write a conditional statement on the left and a subgoal labeled worked example on the right, from tutors' training document.

Condition

1. Définir tous les **cas** mutuellement exclusifs nécessaires ;
2. Les **ordonner** pour faciliter la lisibilité des conditions ;
3. Écrire if, la **première condition**, « : » et les instructions du cas `True` indentées ensuite ;
4. Si vous avez plusieurs conditions, faites de même pour chacune avec **`elif`** ;
5. Finir si nécessaire par **`else`**, « : » et les instructions correspondantes.



(b) A tutor using the subgoal learning strategy during a lab session.

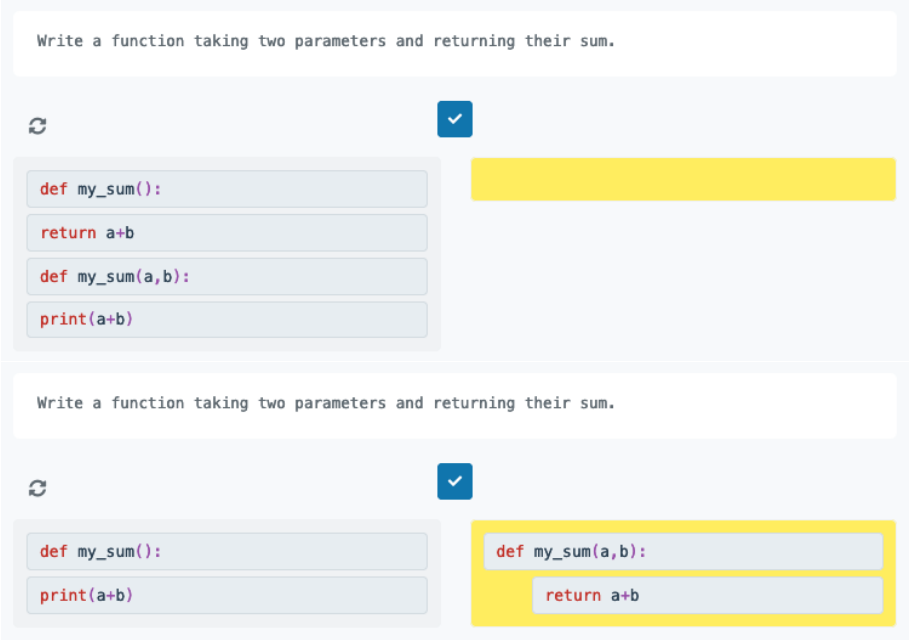
attention went into the observations of this strategy for the proper vocabulary of the subgoals and labels to distinguish when possible a more classic explanation than an actual use of this strategy. This is reflected in the way our observation criteria were developed in the second (cf. Chapter 8) and third (cf. Chapter 9) iterations.

2.5.4 Parsons Problems

A *Parsons problem*[2] [PH06] is a type of programming puzzle where lines of a solution code are cut into multiple blocks and provided out of order. The student has to solve the initial exercise by reordering all the pieces of

[2]Multiple notations appear in literature. Even Parsons himself referred to their problems as "Parson's problems" [PH06]. For the rest of the dissertation, we will use "Parsons problems"

Figure 2.5: A simple 2D Parsons problem with two distractors, unsolved on top and solved on the bottom. The task is available at https://www.codepuzzle.io/PEQCLB.



the puzzle to recreate a correct solution. Figure 2.5 shows an example of a Parsons problem. The blocks could represent code fragments but also algorithm steps, math proofs, execution traces, or any related components. The idea of reordering lines of a code example instead of writing them is heavily inspired by CLT and diminishes cognitive load. This type of problem could be situated on a spectrum between worked examples and a writing problem and is called a *completion problem* [VKK03]. It is also a way to practice the recognition of patterns shown in worked examples (cf. Section 2.5.2). Ericson et al.'s review [Eri+22] establishes that Parsons problems draw on CLT, worked examples, self-efficacy and metacognition and self-regulation. An earlier review on Parsons problems [DLD20] mentions as motivations to use them: identifying student difficulties, providing immediate feedback, improving student engagement and reducing cognitive load.

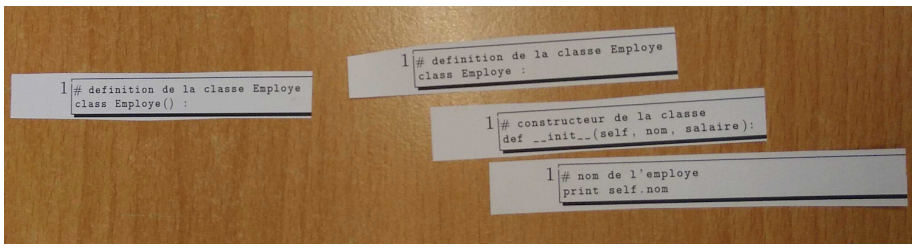
Parsons problems have been used on computers as well as in paper form [DLS08]. Parsons problems are engaging for students [EMR17]. The difficulty can be adjusted by adding *distractors*, unnecessary lines that have to be left out. Distractors often reflect common mistakes that novices can make. A distractor can be *paired* or not with its corresponding correct line as visual cue, for example using the same label or by visually linking

them. Another variant are two-dimensional [IK11] Parsons problems where the indentation of the code is also to be taken into account by the student. Research showed that using two-dimensional Parsons problems with paired distractors lead to the same amount of learning as fixing or writing the same code while taking less time [EMR17]. Parsons problems' difficulty can also be reduced by being *adaptive* [EFR18] and grouping puzzle blocks together if the student is stuck after multiple unsuccessful trials.

Figure 2.6: Examples of Parsons problems used on paper during the first iteration.

(a) Students solving a paper Parsons problem.

(b) Details of a paper Parsons problem with distractors.



Actual Implementation

Parsons problems were used in our first iteration (cf. Chapter 6). They were mainly used on paper, with strips of paper representing labeled lines of code. Tutors used exercises from the course and provided students with typical distractors. Figure 2.6 presents examples of paper Parsons problems.

2.5.5 Explicit Problem Solving

Literature says that transfer possibilities between introductory programming instruction and problem solving skills are low, but still concludes that *"designing the treatment to facilitate problem-solving strategies and allowing students adequate time to build the knowledge base in the programming language and to use that knowledge in strategically different ways are crucial if any type of problem-solving transfer is to occur"* [Pal90].

Explicit problem solving strategies are about metacognition in problem solving. By explicitly giving students a list of steps to follow and prompting them with associated reflection questions, we aim to help them self-regulate the process of solving a problem. It is inspired by a study by Loksa et al. [Lok+16] in which the authors identified the following six steps to follow when problem-solving:

1. Reinterpret problem statement
2. Search for analogous problems
3. Search for solutions
4. Evaluate a potential solution
5. Implement a solution
6. Evaluate implemented solution

Those steps are similar to those proposed in learning transfer models (see Section 5) since both draw on problem solving strategies. Explicit problem solving consists of explaining the six steps to the students, giving them a handout as a reminder and asking them in which phase they are when they ask for help in a lab session. This strategy has as main goal to automate the self-regulation that will help a learner take advantage of metacognition. It is a way to diminish the difficulty of using higher-level cognitive strategies in an unfamiliar context as suggested by CLT.

Actual Implementation

Explicit problem solving was used during our first iteration (cf. Chapter 6). We asked tutors to present and hand out the problem-solving steps mentioned above to students and to encourage them to go through these steps before asking questions when solving exercises. Figure 2.7 presents the problem-solving handout distributed to students.

2.6 Undergraduate Teaching Assistants in Computer Science

This thesis aims to bring research to practice by bringing evidence-based instructional strategies to Undergraduate Teaching Assistants (UTAs)' tutoring practice. UTAs are heavily used in computer science courses [Mir+19] because they allow to break down large students cohorts into more efficient classrooms settings, allowing highly demanded courses to scale up [For+17]. It is important to note that different course setups use UTAs in different ways, some use them in pairs [Pat13], some use them for teaching and grading [RLR95]. While we provide detailed information on our own course setup and UTA duties in Chapter 4 it is important that the reader assesses and understands the differences between our specific context and their own. However, tutors are also used for recitation sessions and labs in CS1 courses in similar setups at other universities [DDL17; Dan+17]. The practice of using senior students as UTAs is typical of, but not limited to, the problem-based learning inspired methodology [Fre+07] used in the studied course.

In their literature review, Mirza et al. mention that using TAs benefits to UTAs and to students [Mir+19]. Research on benefits for UTAs mention long-term impacts like improved self-confidence, self-regulation and a sense of community [MGG12; FHE22]. However, being an UTA does not come without challenges: Riese et al. [Rie+21] mention student-focused challenges like teaching students with different profiles, but also what best practices to use to teach CS content properly and threats like time constraints. Research shows that UTAs need classroom management skills [LBG00], are affected by their environment [Pat12], and are more effective when trained and motivated [Moo+13; Rod+14].

Some universities offer dedicated training to provide UTAs with a.o. pedagogical knowledge on learning theories, assessment and classroom management [Dan+17; RK22; MS23]. Sometimes these trainings require (U)TAs to give mini-lessons [ET17]. While the UTAs we studied also follow such a (non CS-specific) course [SDW17], the aim of this research is to provide them with pedagogical content knowledge [Shu86] specific to a CS1 course [Hub18] to foster learning transfer for students. Few studies actually focus on training UTA with pedagogical content knowledge and instruc-

tional practices in CEd and we hope our contributions will be beneficial to the field.

2.7 Summary

In this chapter, we first gave an overview of the computing education research around introductory programming education. We presented the two theories of cognitive load and of learning transfer. These two theories are the lens through which we analyse a course in Chapter 5 and select the instructional strategies used in our research. We presented the problem-based learning methodology used in the course we present in Chapter 4. We presented the four explicit instructional programming strategies we selected in Section 2.5. All four have been used in Chapter 6, then narrowed down to explicit tracing and subgoal labeling only in Chapter 8 and finally we focus on subgoal labeling integration in Chapter 9. We presented the usage of UTAs in computing education and will develop their precise involvement in the studied course in Chapter 4.

Figure 2.7: The problem-solving steps handout (in French) distributed to students during the first iteration.

Methods and Tools 3

Having presented the computing education principles that support the work of this thesis, this chapter presents the research paradigm, tools and methods used for the different studies of this thesis. First, we provide more personal insights on the research journey to reflect on the author's evolution and personality as a computing education researcher coming from a computer science background. This analysis may be relevant for other researchers in the computing education research field since many share this kind of mixed background. We hence develop on the epistemological foundation of our research approach i.e. *pragmatism* (3.1) which led our research to be based on both quantitative and qualitative research methods – or mixed-methods. We then introduce the design research method (3.2) adopted to guide our iterations on the instructional design interventions of the course. We then present the concepts of adaptation and fidelity of implementation (3.3) which we will use to analyse more deductively our observations in Chapters 8 and 9. Finally, we present the qualitative research methods (3.4) we used in our research to collect data and analyse it more inductively.

3.1 Pragmatism as a Research Paradigm

Before presenting in more detail the research methods and tools we will use, this section will try to bring some nuance between two different mainstream philosophical paradigms and their respective approaches to reality, truth and research methods. These two paradigms are very briefly introduced based on Hennink et al. [HHB20] and Alharahsheh and Pius [AP20].

The *positivist* paradigm considers reality is made of facts and can be *objectively* observed and measured without researchers' subjectivity influencing data. For positivist researchers, data has to be quantified and analysed using statistical methods.

The *interpretivist* paradigm considers reality is *subjective* and different depending on the context and the individuals. Interpretivism considers researchers do influence reality and thus have to reflect on this influence and on their positionality. For interpretivist researchers, data are words collected through interviews or observations, and analysed through qualitative

methods to develop a contextualised understanding of a phenomenon.

Since this section is more a personal reflection introducing my reality perception and the methodological choices that it implies, I will switch to the first person. Whereas these two paradigms are often presented in opposition, I will instead try to argue in this section for blending them together. First I will introduce my own journey between those two approaches and in a second time I will elaborate more on *pragmatism* as its own paradigm and its impact on the chosen research methods.

3.1.1 From CS to CEd

As briefly introduced in my positionality statement (cf. Preamble, Page 1), I was lucky to have been a CS student, a math and natural sciences teacher and am now a researcher in computing education which is peculiar for a PhD student in an engineering faculty. Since this multi-faceted identity has had some impact on the choices made in this research, it is important to discuss its impact on my research journey. This section reflects upon this positionality as a CEd researcher.

It is often difficult to self-reflect without a specific method, so here is my modest tentative to reflect on the evolution of my own positionality throughout my PhD journey. Following a suggestion by Inge Hutter[1] at a summer school in computing education research in which I recently participated, I tried to approach this reflective exercise by loosely following the principles of *autoethnography* as outlined by Adams et al. [AEJ17]. In their chapter on autoethnography, Adams et al. give as a purpose: *"As such, the third purpose of autoethnography is to show how researchers are implicated by their observations and conclusions . . . "*.

My background as an MSc in Computer Science in an engineering faculty made me heavily biased towards a positivist stance on the world and knowledge in general. Early in my life though, I learned that representations and views on societal topics were personal by essence, when confronted to friends' or colleagues' political opinions, or to different perspectives on historical events. These representations of reality were influenced by so many elements such as family, cultural background, socio-economic situation, beliefs, etc. While studying to become a teacher and while teaching in lower secondary education for a year, it became only more obvious to me that each student held different opinions on the natural sciences and mathematics topics I was teaching them. Training in-service teachers for three years

[1] Inge Hutter is an anthropologist and demographer, and a professor in qualitative research methods. Her own qualitative research method is inspired by her positivist demographer background. For more information, see her webpage: https://www.eur.nl/en/people/inge-hutter

then opened my eyes on what words such as "computing", "informatics" or "digital" even meant or the reactions they provoked on sheer mention. Taking my first qualitative research class in preparation for my PhD journey allowed me to put words on these different perspectives and to formalise what ontology and epistemology meant.

Since the beginning of my work, my main aim was to better understand both sides of the learning theories spectrum — with more or less guidance — and blend both sides together in order to achieve the best results. Working towards the integration of explicit strategies in a course using a problem-based learning methodology is in itself a proof of the trade-off I was looking for. Far from radical views of some of the researchers (cognitivists and constructivists) that sometimes inspired me. This is what inspired my mixed-methodology approach. I did a lot of literature-informed decisions, research-based choices, deductive analysis as well as using open-ended analysis tools, qualitative methods and inductive research.

Being a lone CEd researcher in a more CS department has sometimes been challenging. Not a lot of social sciences perspective is available in a CS department (despite my skills as a former CS master student myself). It definitely felt sometimes like my work was disregarded by some senior professors. Thankfully, having co-supervisors as well as members from my supervising committee in both worlds of CS and (C)Ed helped a lot to raise the proper questions and issues with my research approach. Specifically, being able to share with my co-supervisor Felienne's researchers and network.

Due to the interpretive nature of qualitative work, I thrived for methodological rigor in my work. I cannot say the epistemology of my work is positivist even tough my background is in natural sciences. When interpreting observations, coding recordings and interview transcripts, there is room for interpretation and bias. Qualitative researchers need to self-reflect on their own bias and be as transparent as possible on the lens they used for interpretation. This balance between personal interpretation, bias mitigation when possible and being honest throughout the analysis is what makes qualitative research results in CEd interesting in my opinion. It is undeniable that intervening in a course design is by essence interfering with the measured. I also have trained and interviewed UTAs for the course myself. Recording UTAs must have had an impact on UTAs' practice too. I have to acknowledge the *"co-constructive nature of data collection with human beings"* [HHB20]. I like to believe that my qualitative methods were influenced by a positivist stance when looking for literature-informed strategies, when establishing fidelity of implementation criteria that served as deductive codes in my analysis and more generally when producing quantitative results, like correlations. I also like to believe that what I called "strong" strategy use, or the criteria for the strategies I extracted from interviews, or

even the choices on which strategies to use, have been guided by both rational criteria and a good amount of interpretation in my analysis as well as some subjectivity on my side. I do think there is a sweet spot between these epistemologies, and that by using a wide range of methods a researcher can pick the right tool depending on the information he tries to get and thus shed appropriate light on the phenomenon they study.

A notable moment in my research journey was a video call I had with Lauren Margulieux (who was kind enough to discuss with me on my use of subgoals in my research). I was sharing with her that I was a bit uncomfortable with results of my research being published and especially with the subgoals created through the task analysis. I asked her something like: *"How can I be sure that I discovered the truth?"* and she answered[2]: *"At least it's 'a' truth."*. This was a key moment that led to me accepting that truth can be local and different, depending on perspective and context.

In a sense both paradigms strengthen each other. On one side, qualitative research allows to better understand the why and the how of a specific phenomenon, in my case, the adoption of a specific strategy by the tutors. On the other side, a quantitative approach allows to draw on existing theories, use deductive elements in the analysis work and to extract statistical significance from data. Prior readings and review of the literature allow the researcher to be informed on the existing assumptions and theories in the context of their research subject. They will use this knowledge to analyse a situation or a course. By using such a conceptual model from literature on a real-life teaching setup, some discrepancies necessarily emerge. In my PhD, the analysis of the CS1 course under the lens of learning transfer models led me to the idea of introducing explicit strategies to the UTAs. A first intervention was prepared, informed again by strategies from the literature. Then interviews were made and this led to some inductive analysis and data that formed a rough model and gave a better understanding of the factors at hand, etc.

All of this makes me some kind of a pragmatic, not shy on mixing methods depending on the research question. My take-away message for other CEd researchers reflecting on the different learning theories and research methods is: pick those methods that will allow you to answer your question, do not be shy of mixing them, even if it does not fit what others have usually done before. As Sorva stated in Part II of his excellent dissertation while discussing different conflicting stances on learning [Sor12]: *"Can we all just get along? . . . Multiple theories give complementary perspectives"*.

[2] I am quoting from memory here.

3.1.2 On Pragmatism

Breaking away from the injunction that each specific paradigm is to be used with a predefined set of research methods, pragmatism as a paradigm advocates for appropriate approaches through mixed-methods research. Pragmatism tries to put more emphasis on the *how to* aspects of research on top of the *why to* [Mor14]. In his article on pragmatism [Mor14], Morgan argues for an intermediate point of view considering both post-positivism and interpretivism's claims as equally important. Morgan is inspired by Dewey' standpoint on philosophy and his model of experience and inquiry which states that human's actions are influenced by interpreting their beliefs and that in turn beliefs are constructed by interpreting their actions. For Dewey, all our beliefs and actions are social in nature. Morgan states that a researcher has to not only choose their research topic but also their research method and consider what difference it would make to explore that research topic one way or the other.

According to Morgan, this paradigm replaces arguments on the nature of truth and reality by recognising the importance and value of different approaches to research and their respective research communities. Pragmatism puts the emphasis on experience questioning both the kind of knowledge produced but also the way knowledge is produced. This is done in an active process of inquiry that leads to continual interaction between action and beliefs.

In the rest of this dissertation, this approach led the choices of my research methods. I pondered for each research topic what kind of knowledge was needed to better understand it and contribute to the field, and which method would be most appropriate to research it, mixing when appropriate methods traditionally attributed to both post-positivism and interpretivism.

3.2 Design Research in Education

The goal of *design research* is to produce actionable knowledge to achieve an educational goal through the design of instruction [Bak18]. It is used to guide, test and refine educational practices, to fill a research gap in the practical design of instruction in a real-life classroom setting. This process often goes through multiple iterations and is guided by design principles. Typically, the output of a design research will be a list of design principles on characteristics and methodological aspects to include in the instruction supported by both theoretical and empirical arguments [Bak18].

Design research has already been used to conduct interventions in computing education research, for example for the design of teacher ebooks integrating worked examples [Eri+16], for introducing programming in pri-

mary schools [GSH19] or for integrating computing activities in pre-service teacher programs [Mar+22]. To use this approach, we were also inspired by Margulieux et al.: *"A key idea in this paper is that instructional design matters. The two groups did not differ in the content of the instruction but in the design of that material (e.g., whether subgoals were made explicit)."* [MGC12]. Another key element for choosing this approach was research showing successful interventions in CS1 courses already existed, showing that the integration of evidence-based instructional strategies in a CS1 course had a positive effect on students retention [PS13].

3.3 Adaptation and Fidelity of Implementation

When trying to change instructional practice through professional development, a teacher will ideally use the strategy and integrate it in their own practice. Research looks at their appropriation through two constructs: the *adaptation* that teachers will inevitably make and the *fidelity of implementation* of that strategy defined as the degree to which procedures are implemented as planned [ODo08]. *"Adaptation and fidelity of implementation are different constructs and should be separately measured and related to outcomes"* [ODo08]. Fidelity of implementation is measured based on fidelity criteria. Mobray et al. [Mow+03] propose to develop fidelity criteria by identifying *"critical components . . . based on an expert consensus process or the existence of a proven model which has been explicitly described"* [Mow+03]. They suggest to use a model with proven efficacy, effectiveness or acceptance, and expert's opinions (surveys of experts or literature reviews).

Research on teachers' adaptations and implementation of instructional strategies distinguishes *efficacy studies* from *effectiveness studies* [ODo08]. Efficacy studies are lab studies focusing on the development stage and aiming to show that an innovation performed by experts in perfect conditions is indeed linked with better outcomes. Effectiveness studies look at natural classroom setups where adaptation is inevitable and where we try to understand when and how a practitioner adapts the innovation. We will use the fidelity of implementation in effectiveness studies since we will consider the efficacy of the strategies as demonstrated. We will observe how close were the tutors' practice to the prescribed strategies and what adaptations they made to the strategies. While some studies measure fidelity through participant responsiveness, we will measure directly the tutor's quality of delivery.

It is recommended to compare adaptations and fidelity to the desired outcomes, to be able to discriminate whether a failure to achieve these outcomes is due to an ineffective instructional strategy or too much diver-

gence from a prescribed strategy. However, our main goal is to describe and understand *how* and *when* tutors use and adapt the strategies. There is a dire need in literature to more accurately document instructional strategies adaptation and fidelity of implementation, as confirmed by Borego et al. [Bor+13] who state that *"it is important to have complete and accurate descriptions and measures of how and when an instructional strategy is being implemented"*. This is an important goal of this dissertation.

3.4 Qualitative Research Tools

Doing qualitative research is by essence a *"creative, reflexive and subjective"* process [BC19]. It is a way for us researchers to uncover a truth found in the data. It has already been stated that the field of computing education research can benefit from more qualitative research [Haz+06]. The kind of questions qualitative research answers are about understanding phenomena. This is not especially about big cohorts of studied individuals but more about understanding the "Why?" of a certain practice. When and why did we observe this behavior?

Tools coming from qualitative research have been used throughout the research done in the scope of this thesis, notably, for data collection: *surveys*, *interviews* and *observations*. The resulting data has been systematically analysed qualitatively using thematic analysis, coding and statistics when pertinent.

3.4.1 Data Collection

Interviews In the two first iteration studies presented in Chapter 6 and 8, we used *semi-structured interviews*. This means that an interview guide was prepared with main themes to discuss, a list of questions on these themes in order not to forget any important point and some follow-up questions to dig in deeper when the discussion asked for it. This method is inspired by the methodology described by Kaufmann [Kau16]. The goal of interviews was really to gain in-depth information on phenomena and identify personal experiences and perspective, in our case, on their usage of the strategies. During the interview, consent for recording and data usage is collected from the interviewee, a recording of the interview is made, the broad context of the research is provided even though the main research questions are never directly asked "as is". Interviews are then transcribed and coded (cf. Section 3.4.2).

Observations Recorded *observations* of tutors' practice have been made in the real-life setup of the tutors' classrooms. Observation allows for the

gathering of firsthand information on a phenomenon. It allows to observe what people actually do in contrast to what they say they do as with previous data collection methods. When doing observations, we choose the role of *nonparticipant observer* [Cre12b] and our goal as researchers and observers is to be as transparent to the classroom as possible. In order to allow this, we each time let the students know in advance that observations of their UTAs might happen. We made it clear to the students and the tutors themselves the observations were in no way a grading or a judgment of their practice but were only done for research purposes. We also foresaw "practice" observations were we would come into the classroom to let the tutors adapt to our presence but without taking those recordings into account for our research. Observations of the tutored sessions for the second (cf. Chapter 8) and third (cf. Chapter 9) iterations have been recorded. During the observations presented in Chapter 8, we combined the filming of the tutor with observer *fieldnotes* taken during the tutored session not only for redundancy (i.e., in order to have backup data in case of data loss) but also for early impressions, context, feelings and possible early coding when applicable. The process was more inductive in the second iteration. The recordings were then watched again in order to write down notes on the contexts, timestamps, quotes and to code the segments. For the third iteration, since we wanted to scale up observations due to our intervention being a full strategy integration into the course, we hired external observes to whom we provided deductive codes to analyse the recordings with. More details are given in Section 9.1.4.

Open Questions in Surveys While *surveys* are mostly used in quantitative research [Cre12c], we used them also for feedback collection through open questions. Some quantitative data was collected through likert-scaled items on tutor's familiarity and frequency of the subgoal learning strategy use in Chapter 9 and on students' perception. Tutors participated in a survey twice at different moments so that we could obtain a *longitudinal* view of the evolution of their usage while we only conducted one survey on students.

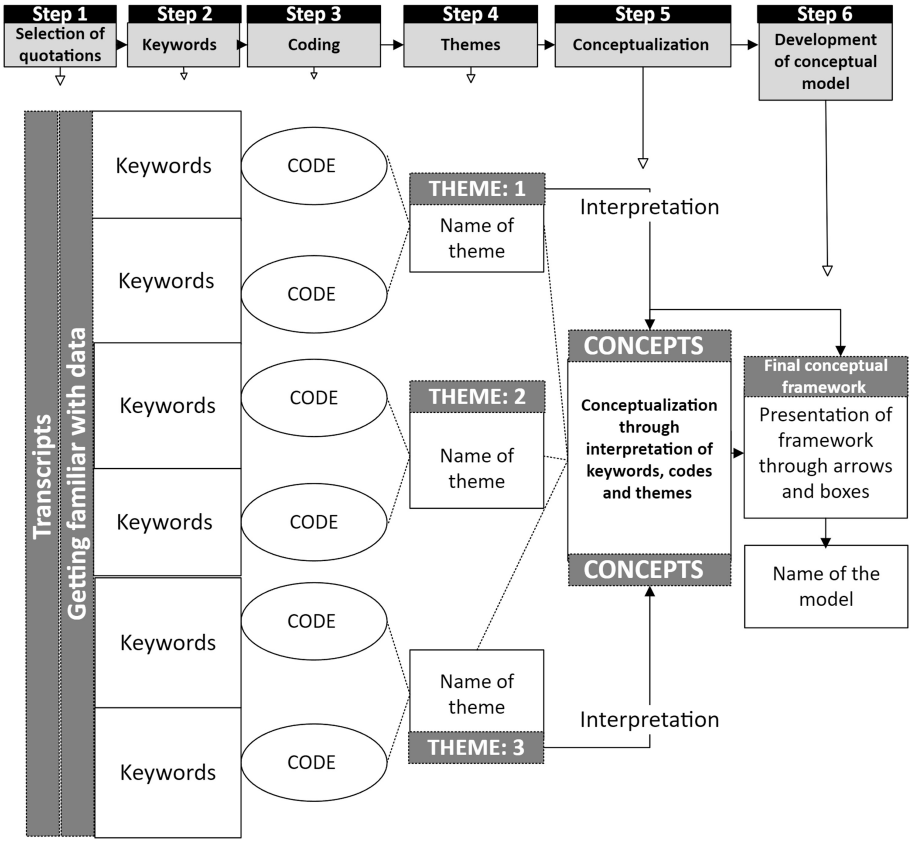
For more understanding on tutors' practice and in order to collect their feedback and suggestions, open questions were used. Tutors' answers to open questions were analysed in the same way as interview transcripts.

3.4.2 Coding

Coding data is the process of tagging segments of the data (quotes, part of a recording, etc.) with specific labels in order to organise and group them into categories and larger themes [Cre12a]. The process is iterative in

the sense that the researcher will typically apply coding to multiple documents (multiple interview transcripts, recordings, etc.). For each document, codes are listed and compiled in a code-book. If new codes emerge from subsequent document, a new pass has to be made on previously coded documents. Codes can be merged or split when needed, codes will also be grouped into categories and themes. Typically, where possible, multiple different knowledgeable coders apply this same technique on part of the data to increase the reliability of the coding. This process is repeated until *saturation* is reached which means no new codes appear when coding more documents or when all data has been consumed but then with the limitation that some more data could have yielded more codes. The code-book is then completed and the *description* of what had been observed in the data can be rendered in the results. Typically, this description will be illustrated with good representative example segments to give a sense to the reader on the thought process of the research team.

Figure 3.1: A schematic representation of a systematic thematic analysis (extracted from Naeem et al. [Nae+23]).



In practice, when we coded material (answers to open questions, interview transcripts and recording segments) during our research, what we mean is that we used a working document for highlighting and annotating text segments and then (or directly) a code-book spreadsheet to associate data segments with a specific label or tag – a code – depending on the meaning, the category or the theme of said segments. We used both *inductive* and *deductive* coding depending on the research question and the availability of pre-existing codes (like a conceptual model, known themes for the studied phenomenon or fidelity criteria). Inductive coding is more often used when doing exploratory work, when the researcher wants to be guided by the data itself. For this, we mainly used thematic analysis [BC19; Nae+23] to identify themes from the coded segments. The process of thematic analysis can be represented as in Figure 3.1.

A good example of the results of such a thematic analyses are the adaptations we will report on in Section 8.2.3. We also used deductive coding with codes coming from literature, pre-established criteria, chosen themes for different parts of interviews or guided by our own research. We used deductive analysis mainly when looking for fidelity criteria in recording segments for fidelity of implementation coding like in Section 8.2.2 and 9.2.1 We did this with multiple iterative passes through the data from a more open first read to *"get a sense of the whole"* to the emergence of codes and then grouping of those codes in categories when needed following the method proposed by Creswell [Cre12a].

3.5 Summary

In this chapter, we presented our pragmatic approach to research. We introduced the overall design research method that we used to iterate on our design interventions that will be presented in Part II. We introduced the concept of fidelity of implementation used in Chapters 8 and 9, as well as the different research methods that have been used in our different research. Table 1.1 provides a summary of which methods are used in which chapter.

Use Case: a CS1 Course

4

This chapter presents the organisational details and setup of the introductory programming course that served as use case for the different interventions and studies carried out in this dissertation.

4.1 Introduction

The different studies carried out in this dissertation were conducted on the use case of an introductory programming course taught at UCLouvain, a full Belgian research-level university, in the French-speaking region of the country. At the turn of the millennium, EPLouvain (Louvain school of engineering), then still known as FSA (Faculty of Applied Sciences), decided to review the pedagogical approach to the first two years of its engineering curriculum. Under the reform known as 'candis 2000', the faculty moved towards a newer, more active learning method centered on problem-based and project-based learning [Fre+07].

Apart from breaking down barriers between disciplines through integrated, cross-curricular projects, the reform impacted the way each discipline is taught. In particular, it is on these principles that the first-year bachelor introductory programming course has also been adapted to adopt a problem-based learning approach. Elie Milgrom, co-author of some of the articles referenced in this chapter [Rau04; Fre+07], was at the helm at the time of the reform for the computer science part of the training of future engineers.

The CS1 course follows a methodology inspired by Problem-Based Learning (cf. Section 2.4). This methodology was introduced in the bachelor prgoram after the reform. The methodological principles guiding this reform can be summarised as follow (free translation from Raucent et al. [Rau+04]):

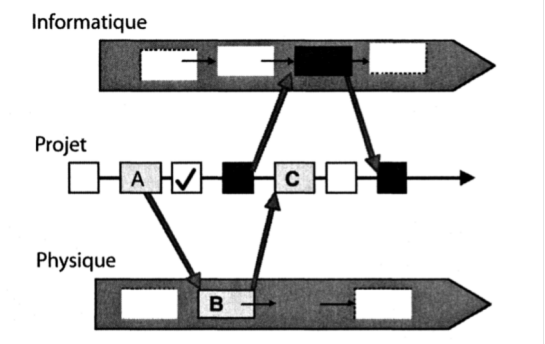
"The choices made in developing the program are based on the articulation of three key principles:"

1. *"contextualisation of learning: students learn from problem situations arising from professional contexts;*

2. *cooperative learning: students approach most of the activities they encounter in stable groups;*

3. *tutoring: the active approach to learning and the use of small groups lead to a modification of the roles of the various players. Student supervision has been revised accordingly."*

Figure 4.1: Interaction of the introductory programming course with the cross-curricular engineering project course after the reform (extract from [Rau+04]).

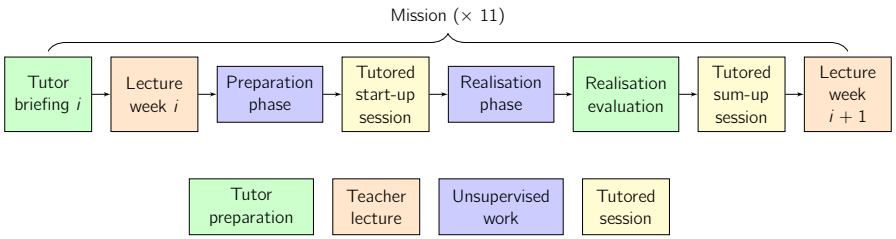


The reform applied to the organisation of the entire bachelor curriculum in engineering. Figure 4.1 shows how it affected the computer science course [Rau04]. In particular, as part of the reform, the first bachelor engineering project required java programming skills. These skills had to be covered in the introductory programming course and used again later in the context of the project.

4.2 Course Setup

The introductory programming (CS1) course lasts thirteen weeks. As illustrated in Figure 4.2, the CS1 course is organised around small weekly projects called *missions*. The course comprises 11 missions spread over 13 weeks (there are no mission on the first and last weeks). Each mission introduces some new programming concepts and covers a theme. For example, the "strings and lists" theme is about DNA sequences and students have to develop helper methods for calculating a Hamming distance between sequences. The course currently uses Python, covers imperative programming and introduces programming with objects. All course material is provided online to the students.

Figure 4.2: The different moments that structure a mission of the CS1 course.



4.2.1 Stakeholders

Teachers. Three professors are in charge of the different modules of the course. They are in charge of the course content in the theory syllabus and give lectures in auditoriums to the students. They are also present during the tutor briefings.

Undergraduate Teaching Assistants. More than twenty five senior students are recruited as undergraduate teaching assistants (UTAs) also called *tutors* throughout this dissertation. They are in charge of the lab sessions twice a week in classrooms of about 24 students. These tutored sessions are detailed in Section 4.2.2 below and do not include programming exercises on computer. UTAs tutoring the course for the first time are juniors, while from their second time on they are seniors. Junior UTAs also need to follow a pedagogical training (3 ECTS) organised by the university [SDW17].

Teaching Assistants. Half-a-dozen teaching assistants (doctoral students) complete the teaching staff. They help with the organisation of the course, handle technical issues and edit online exercises when needed. They answer student questions online, manage UTAs, organise an open optional one-hour session once a week and help with the organisation of the test and exams.

Students. The course is mandatory for first year undergraduate (bachelor) computer science and engineering students with no assumed prior programming experience. This course currently targets over 600 students. They attend the lectures in auditoriums and the labs session by classrooms of around 24.

4.2.2 Course Organisation

Each mission i is organised around different moments as presented in Figure 4.2. This pattern is repeated 11 times for each mission and is detailed below:

Tutor Briefing. A meeting with all teaching staff takes place at the beginning of a new mission. The unfolding of the previous mission is discussed. Pedagogical approaches, common difficulties and classroom management issues are shared by tutors and solutions discussed. Then, concepts foreseen for the starting mission are reminded to the UTAs. The subtleties and pitfalls as well as typical students misconceptions on these concepts are presented. Exercises for both upcoming tutored sessions are reviewed. Typically this meeting takes place on a Friday at noon, before the start of the next week.

Lecture i. A one-hour lecture is given in an auditorium by the professor to the students. It is the opportunity to structure the concepts of the previous week during the first half of the lecture and to introduce new ones. For week i, this last half hour is considered the **introductory lecture**.

Preparation Phase. A preparation phase during the week-end is meant for individual student work. Students have to read the theory syllabus and answer multiple-choice and open questions provided in the exercise syllabus before the first tutored session. Part of the open questions are implemented in an auto-grader. Each week a new fictional context is introduced and the work of that week will be situated in this context.

Start-up Session. During tutored sessions, students work in their usual group of six students and each tutor is assigned to a room of four groups. During this first one-hour tutored session, the prepared answers to the exercises are shared and discussed. The tutor helps organising the discussions, sending students to the blackboard. UTAs are not there to just give answers but to clarify theory concepts when needed and to ensure a correct solution is found by the groups.

Realisation Phase. Each group of students is split in pairs who then have two days to submit their solution to a small weekly project. The project statement is in the exercise syllabus and will typically be situated in the weekly context. Some code is provided and students need to develop a program using the concepts seen up to this moment in the course.

Realisation Evaluation. Tutors have to evaluate their classroom's submissions. A correction guide is provided with indications on the evaluation rubrics to pay attention to. Tutors have to prepare some global feedback and print it out for the sum-up session.

Sum-up Session. The sum-up session is the second and last tutored session of the week. During this one-hour session, the tutor provides feedback on the submissions of the weekly realisation project, discussing common mistakes and pitfalls the students have ran into. A summary exercise covering the concepts of the week is then solved by the students.

Lecture $i+1$. The mission ends during the first half of the lecture at the end of the week. The cycle starts all over again with a new mission focusing on new concepts and a new context. For week i, the first half hour of this lecture is considered the **restructuring lecture**.

4.2.3 Resources

This section presents the main resources for the course.

Pedagogical Approach The pedagogical approach has already been detailed in Section 4.1.

Tutoring This resource is an integral part of the course methodology. The tutor, between friend and teacher, can intervene where the teacher cannot in front of an audience, in a personalised way by supervising a more limited number of students. Often a former student of the course, and in any case a more advanced student, he or she can support the students they supervise, both in terms of subject matter and course methodology. In our case, tutors are responsible for a single classroom two hours a week. While they will need to evaluate and give feedback on the realisation phase, they are not involved in grading.

Documentary resources Lecture slides, theory syllabus[1], exercises syllabus[2], auto-grader[3], the resources of the course have been designed to follow the structure of a week, providing information as and when it is needed. A digital workspace is provided on the Moodle platform of the university. In particular, it enables the staff to discuss with students and

[1] https://syllabus-interactif.info.ucl.ac.be/index/info1-theory
[2] https://syllabus-interactif.info.ucl.ac.be/syllabus/info1-exercises
[3] https://inginious.info.ucl.ac.be/courselist

give them access to the various course resources. Its forum is especially useful for exchanges and questions/answers between students and with course assistants. The online exerciser, Inginious[4], is an online exercise platform with auto-grading [Der+15] that students can use during their preparation phase. After the first tutored session, additional exercises with feedback are offered via the platform, as well as after the sum-up session.

[4] https://inginious.org

Part II

Design Research Iterations

Course Analysis

5

This chapter is largely based on my final assignment of the course LFOPA2115B: Pedagogical practices and training setups - (part of UCLouvain's Master of Education) followed in anticipation of this thesis (and which received the highest grade):
O. Goletti. *En quoi le dispositif mis en œuvre dans le cours d'introduction à l'informatique en BAC1 ingénieur civil basé sur l'apprentissage par problèmes soutient les processus du transfert des apprentissages : l'encodage et l'accessibilité aux connaissances ?* Tech. rep. `http://hdl.handle.net/2078.1/245579`. UCLouvain, 2019.
The title of that work can be translated as "How does the problem-based learning approach implemented in the engineering CS1 course support the processes of learning transfer: knowledge encoding and knowledge accessibility?".

This chapter presents our analysis of the introductory programming course. A presentation of the CS1 course was already provided in Chapter 4 and its reading necessary to understand this chapter. Our analysis is based on two processes of learning transfer. We first present in more detail the learning transfer processes of knowledge encoding and knowledge accessibility (5.2.1). We then propose some criteria (5.2.2) inspired by literature on learning transfer to effectively analyse (5.3) the material of a specific week of the course. From this analysis, we derive three proposals (5.4) to guide instructional design intervention in the course in order to foster learning transfer.

5.1 Introduction

Having been a student of this course during my first year of a bachelor's degree in civil engineering, then a tutor during my years of a master's degree in computer science and now a teaching assistant in the department, I know the course particularly well. It should be noted that since the academic year 2018-2019, a change in content has been implemented as the course

uses from now on the Python programming language rather than the Java language used previously. The analysis presented here is based on previous editions of the course when it was still based on the Java language. However, the methodology and concepts of the course are essentially the same, which allows the results of the analysis to be extended to the current course setup.

This is the research question we answer in this chapter:

RQ5.1 How does the CS1 course align with the learning transfer processes of knowledge encoding and knowledge accessibility?

5.2 Study Design and Methodology

To analyse this course, we performed a document analysis based on the different resources for the course as presented in Section 4.2.3. We use a conceptual model of learning transfer by Tardif [Tar99]. This model is based on Bracke's model [Bra04]. Tardif's model is close to the steps of problem-solving strategies. Typical problem-solving steps include: problem identification, the formulation of a strategy based on similar problems, and evaluating this solution. This is well-adapted to the way CS is generally taught and these steps are often reused and adapted in the context of CS problem-solving [LK16].

5.2.1 Conceptual Model

For Tardif [Tar99], the transfer mechanism is characterised by a new situation (target task) which has the characteristics of a problem that has to be solved. In order to solve this target task, knowledge acquired previously (in a source task) will have to be reused. However, a difficulty must be present in the target task. This will therefore require a certain adaptation of knowledge and will lead to learning.

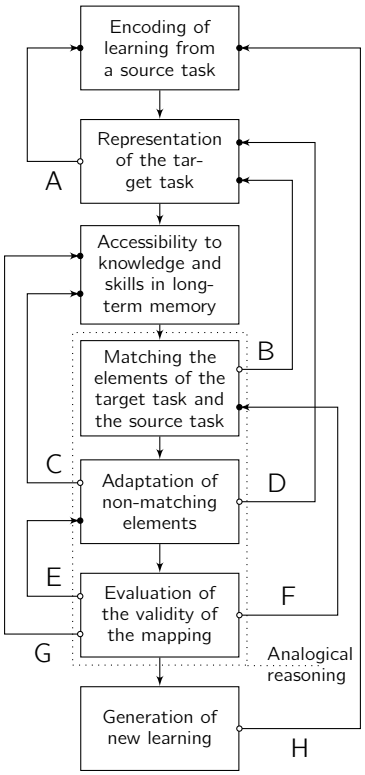
Tardif [Tar99] proposes a model with seven processes. The interactions between the different processes of this model are shown in Figure 5.1.

1. *Encoding of learning from a source task*: from the beginning of the construction of new knowledge, the individual must project himself in possible contexts of reuse.

2. *Representation of the target task*: construction of a provisional and evolving mental model of the problem to be solved.

3. *Accessibility to knowledge and skills in long-term memory (LTM)*: provide access to / awareness of knowledge and skills (= cognitive tools) directly related to the mental model and create the conditions for them to be reusable in the particular context of the problem.

Figure 5.1: Tardif's model of learning transfer (adapted from [Tar99]).



Ideally, the person makes an exhaustive inventory of the cognitive tools that they have mastered in order to avoid placing themselves in an initial learning situation.

4. *Matching the elements of the target task and the source task* (beginning of reasoning by analogy): identify similar elements and determine the power of these similarities in solving the problem leading to a conclusion.

5. *Adaptation of non-matching elements*: assess the importance of similarities and differences. If the differences are significant, adaptation is necessary in order to reduce them or even eliminate them.

6. *Evaluation of the validity of the mapping* (termination and validation of reasoning by analogy): The conclusion of this step is either to solve the problem, or to return to a previous process because something has been forgotten (links E, F, G in Figure 5.1), or to stop for lack of transferable knowledge.

7. *Generation of new learning*: supporting decontextualisation, identification of new structures or adaptations of existing structures, links with learning already encoded in relation to the source task.

Bracke explains in detail the cognitive processes involved during the knowledge transfer [Bra04] and how the different knowledge elements are stored in long term memory and retrieved through processes such as the third sub-process in Tardif's model. All those processes are there to compensate the very limited Short Term Memory (STM) of the learner:

1. Mental model representation of the problem in Short Term Memory (STM): this phase corresponds to the phase 2 of Tardif's model.
2. Access to previous useful knowledge through External Memory and Long Term Memory. This matches the third phase of Tardif's model by emphasising information and knowledge structures.
3. Processing using analogous reasoning in the STM. This step is akin to Tardif's phases 4 to 6 with some steps out of order, but this is not discussed here.
4. The conclusion corresponds to the last phase in Tardif's model.

The first difference between both models is that there is no knowledge encoding process in Bracke's model. In fact, she explicitly states that it is *"difficult to specify which of the multiple previous encodings should be retrospectively retained as phase 0 of the current process"* [Bra04]. Bracke also sets out to define the different knowledge structures specific to the different memories involved in transfer. Without going into too much detail, she nevertheless emphasises the following concepts:

Affordances are structures associated with external memory (EM) and the capacity for action offered by the learner's environment.

Categories are structures associated with long-term memory (LTM), which enable knowledge to be stored, prioritised and associated with more or less distant examples of a prototype. These LTM categories are also linked together by a series of relations.

Mental models are structures associated with short-term memory (STM), which are temporary and "rough" constructions due to the very nature of STM and its limitations.

What these structures have in common is that they are functional, i.e. associated with the functions inherent to the different information and

knowledge elements to which they relate. This is important in her model, as access to relevant knowledge is a specific process, distinct from processing based on similarities between these different structures. This justifies the attention paid in this work to the support offered by the instruction design to the representation and relationships proposed between these structures.

5.2.2 Analysis Criteria

To analyse our CS1 course, we selected only the learning transfer processes that were most appropriate to its methodology and organisation, and most likely to yield interesting results. Our analysis of the course material focused on the first and third steps of this model: the encoding of previous knowledge and the access to this knowledge in a situation of transfer.

Based on those steps, our analysis was done on specific extracted criteria to see whether the course was in accordance with them or not. The

Table 5.1: Analysis criteria of learning transfer sub-processes (inspired from Brouillette et Presseau [BP04]).

Sub-process	Category	Criteria
Encoding of learning from the source task	**Viability of learning**	- The source task is presented in a meaningful context. - Many contextualised examples are offered. - Identification of potential transfer contexts. - Identification of necessary and sufficient conditions for reuse.
	Organisation of learning	- Presence of preparatory stages. - Conditional indexing according to both affordances and categories. - Proposal for an organisational / summary diagram. - Proposal of relationships with other knowledge. - Validation by peers, tutor, teacher.
Accessibility to knowledge and skills in long-term memory (LTM)	-	- Prior knowledge exploration strategies. - Presence of recall strategies. - Taking into account the External Memory (EM).

criteria we used are inspired in particular by a grid proposed by Brouillette and Presseau [BP04]. Our criteria are presented in Table 5.1. The first two categories in this table correspond to the first process of encoding of learning. They are taken from Tardif, who talks about the *viability of learning*, which is making sure that the learner perceives the usefulness of his learning process and the phenomena that they allow to understand. The associated criteria are:

- *The source task is presented in a meaningful context*, with reference to the importance of the learning context as emphasised above. Beyond a context that makes sense, we also look from the perspective of situated learning whether this context refers to the future profession (software engineer in this case) of the learners, so that the learning makes sense for the learner.
- *Many contextualised examples are offered*. Here the idea is to observe what are the areas of application of learning and to multiply the possibilities of links with future areas of transfer.
- *Identification of potential transfer contexts*. Beyond the examples worked on, does the course explicitly support this identification?
- *Identification of necessary and sufficient conditions for reuse*.

The second category deals with the *organisation of learning*. The associated criteria are linked to the identification of links and strategies for recalling this new learning:

- *Presence of preparatory stages*, in order to bring out concepts specific to the learning of the source task in order to be able to *"classify and synthesise them"*.
- *Conditional indexing according to both affordances and categories*: explicit implementation of a mechanism for recalling knowledge linked to the context.
- *Proposal for an organisational / summary diagram*. As pointed out by Bracke, knowledge is organised in LTM in the form of hierarchical categories. It makes sense to support the encoding process and to ease the recall of knowledge. Does the course explicitly propose this kind of organisation of new learning?
- *Proposal of relationships with other knowledge*, in connection with the representation of knowledge structures in LTM. The previous criterion was more about hierarchical organisation. This one is more about the links of similarity with other knowledge.

- *Validation by peers, tutor, teacher*. Explicit restructuring with exchanges between different actors and learners to provide feedback on the organisation of learning. This is also linked to the cognitive companionship recommended by contextualists.

The second process analysed is Bracke's second and Tardif's third, namely access to LTM knowledge. The criteria used are:

- *Prior knowledge exploration strategies*. Are these strategies explicitly proposed to seek out prior knowledge in a systematic way ?
- *Presence of recall strategies*. We are not yet in the comparison but rather on the exploitation of task similarities to recall structures in STM.
- *Taking into account the external memory*: Presence of external supports and indices: learners are encouraged to use relevant documents such as summaries or previous diagrams of organisation of knowledge. In particular to overcome the limitations of STM in the recall strategies of the previous criterion.

5.3 Analysis of the CS1 Course Material

This section summarises the salient features of the CS1 course setup matching our analysis criteria from the grid presented in Table 5.1. A filled-in grid resulted of this analysis and the results are provided here inline and in a better organised way. The grid was filled according to the different moments of a course's mission, focusing mainly on the fourth week of the course. However, some elements were taken from other parts of the course when week 4 alone was not sufficient to illustrate certain criteria. The main concepts seen in week 4 are strings manipulations and functions, the context is around DNA sequences.

5.3.1 Viability of Learning

This category specifically covers the criteria which ensure that the learning involved in the source task is meaningful for the learner, in particular by linking the learning to a context which is meaningful for them, but also by presenting opportunities for re-use.

5.3.1.1 The Source Task is Presented in a Meaningful Context

It is mainly in the realisation phase (see Figure 4.2) on the processing of DNA strings, the sum-up session and the restructuring lecture that we find

this notion of context. In the restructuring lecture, we talk in particular about the encoding of characters as used in today's information systems. Moreover, the exercises regularly ask for implementation and offer explanation on the working of functions that the student will be asked to use again later. For example, comparing two strings of characters.

5.3.1.2 Many Contextualised Examples are Offered

There are different examples of exercises, in particular this week the exercises that work on string traversal. These exercises are proposed in different contexts. Gradually, from the preparation phase onward, they are included in the realisation phase and in the sum-up session at increasing levels of difficulty, and this technique is used again in the restructuring lecture. It should be noted, however, that in "drill" exercises, a meaningful context is not systematically provided.

5.3.1.3 Identification of Potential Transfer Contexts

Although most of what is learned about string traversal is reusable, this is not really presented *explicitly* in the course. This issue will be discussed more generally in Section 5.4.

5.3.1.4 Identification of Necessary and Sufficient Conditions for Reuse

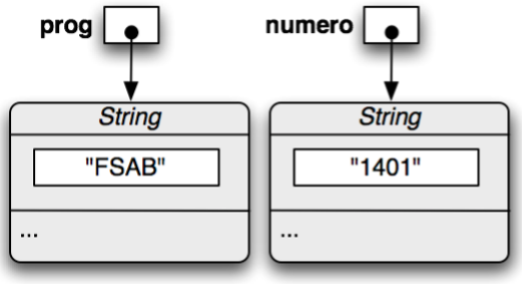
A whole more schematic part linked to the graphical representation of objects will clearly be useful later on and is introduced here and reworked in the following week with arrays. However, once again, the conditions of reuse for the concept of string traversals are not explicitly highlighted in the course.

5.3.2 Organisation of Learning

5.3.2.1 Presence of Preparatory Stages

Clearly, the course methodology makes it possible to really work on this criterion. Giving only certain notions about the new concept at first, then providing resources to read during the preparation phase and then getting the learners to work individually on the new learning really fits with what the theory was proposing in order to bring out the characteristics of the new learning and thus be better able to relate it to existing knowledge. Some of these steps are specific to computer science, such as that the learner is also asked to read the API, the official documentation for the Java language.

Figure 5.2: Graphical representation of variables containing references to `String` objects.



5.3.2.2 Conditional Indexing According to Both Affordances and Categories

Again for this criterion, even if it is not *explicitly* mentioned to the learner, there are two striking examples in the course of highlighting similarities which will later be useful in learning. Firstly, the representation of objects in the form of a drawn reference[1] as shown on Figure 5.2. This graphical concept will help learners to imagine that variables can contain references to objects, which is an important mental model and used in many future learning activities in the course (in particular object comparisons, tables or the manipulation of linked data structures, and even in later courses in their curriculum). This notation is also used in the reference book.

The second element is the steps involved in traversing a character string. Although there is never a description of a step-by-step procedure for this type of exercise, this concept is covered several times during the week and will be reused the week that follows in the array traversals. The graphical representation, particularly in the preparation of such exercises when deciphering a statement of this type, can be used as a trigger to remind students of the framework for solving such exercises.

5.3.2.3 Proposal for an Organisational / Summary Diagram

While this criterion could help students better organise their knowledge, the analysis of the course did not identify any elements for this criterion.

5.3.2.4 Proposal of Relationships with Other Knowledge

Systematically, the construction of new learning is based on concepts already covered, and the course does not fail to emphasise this. For example, by

[1] While strings are not objects in Python, the notation is still used in previous mission when using the `turtle` module for example.

showing in the restructuring lecture that characters are treated as integers, the learner will be encouraged to transfer what they know about integers for this type of data. We can also appreciate that even for a concept that has not yet been covered in detail in week 4, the concept of objects, strings are nevertheless presented as specific objects and as such already get a special place in the concepts handled. This will make a lot of sense in the weeks to come.

5.3.2.5 Validation by Peers, Tutor, Teacher

The aim of this criterion is to discuss as much as possible, with help when necessary, to bring out as many links as possible with existing knowledge. The course methodology specifically supports these approaches by encouraging and alternating phases of individual work followed by tutored sessions, group work and restructuring in lectures. In particular, the tutored sessions provide a safe framework for all misunderstandings and errors. Because of the structure of the course, learners know that they are likely to arrive at the start-up session with misconceptions. It is then by going over the various start-up questions with the tutor that students hopefully find themselves spotting these errors, correcting them and making links with what they had understood up to that point.

5.3.3 Accessibility to Knowledge and Skills in Long-Term Memory (LTM)

5.3.3.1 Prior Knowledge Exploration Strategies

Although some tutors or teachers may suggest such strategies from time to time, they were not found *explicitly* in the course resources.

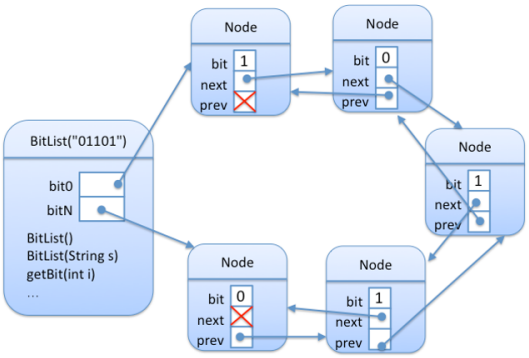
5.3.3.2 Presence of Recall Strategies

Again, for this criterion, nothing found explicitly in the course resources.

5.3.3.3 Taking into Account the External Memory (EM)

Of the three criteria, this is where we find the most elements in the course material. In particular, because many strategies are suggested in the course for working graphically, on paper, even before starting to think about a solution in the form of a program. Learners are therefore constantly encouraged to use drawings to represent objects, and those who have acquired this reflex will use it again later to solve reference problems such as adding a node to a linked list data structure. As these notations are also used by teachers and in the book, they undoubtedly help learners to remember the associated

Figure 5.3: A schematic instance diagram of a linked list data structure.



knowledge. A good example of that is Figure 5.3, which shows a chained structure from the January 2018 exam. It clearly shows the references as worked on in the course in week 4.

During the exam, students also have access to the entire syllabus and official Java documentation. So clearly, an external memory representation is present and its use encouraged.

5.4 Results

In terms of the course itself, the analysis shows that despite the fact that the course implements a PBL methodology and that learning transfer processes are inspired by problem solving, some of the elements suggested by the analysis grid were missing. This does not come as a total surprise, given that the chosen focus is on certain processes of learning transfer which are not necessarily at the centre of problem-based learning, which is itself no more than a methodological resource for the course.

In the chosen conceptual framework, much has been said about similarity, and it is clear that several of the criteria mentioned refer to it. However, the disadvantages of similarity for learning transfer were not really explored. Such an analysis could be done using Bracke's model which shows that the learner relies mainly on surface similarities which can lead to misconceptions and negative transfers. Such an analysis would allow to identify these similarities and try to avoid misconceptions as soon as students start learning the associated concepts.

In the following subsections, we propose three ways to foster learning transfer based on this chapter analysis.

5.4.1 Explain how Knowledge is Organised

Effective encoding of knowledge in the learner's memory in order to foster learning transfer requires links between knowledge elements to be made explicit. While this was noted as being somewhat present, it is perhaps due to our expert point of view. Quite a few interconnected concepts are not explicitly identified as such in the course, which can give a certain impression of disorganisation. In short, even if the different subjects are sequenced, the order in which they are taught might only make sense to the expert. In the light of the analysis carried out as part of this work, it would seem that from a transfer perspective, it would be beneficial to make these different links and choices more explicit in the course. In particular, as suggested by the criteria, by organising the learning more systematically, in particular by means of summaries and organisational diagrams.

5.4.2 Emphasise Transfer Opportunities a Priori

Another shortcoming identified was the lack of strategies and elements for identifying potential transfer contexts. This is particularly true of the schematic representation of objects introduced in this week, or string traversal, which turns out to be similar to the topic of array traversal covered the following week.

If the potential transfer context is not explicitly highlighted, there is a risk that the learner will not realise that this is a method to be reused by adapting it. Even if the method is repeated the following week, the chances of transfer are increased by mentioning it in the source task, thus giving meaning to the learning and enabling the learner to be 'on the lookout' for a transfer situation. The specific format of the restructuring lecture partly compensates for this shortcoming, as it tackles the new subject just after the previous week's restructuring lecture, thus facilitating the transfer.

One recommendation at this level could be to teach the questions to ask in order to identify the situations in which certain knowledge can be reused. In this way, the learner will be able to recognise a problem that calls on the strategies to do a data structure traversal, for example, and will therefore be able to make the link with what he already knows. In short, we would be working on structuring and making explicit the abstractions to be made in the various learning processes.

5.4.3 Use Explicit Recall Strategies

The grid is also quite empty when it comes to recall strategies. It would seem that, by following the previous recommendations for clarifying the conditions for reusing knowledge, we could equip the learner with a series

of questions and criteria to use when tackling a new problem. This could then feed into a more general strategy of exploring existing knowledge and enrich recall strategies.

One tool from explicit teaching is to think aloud or 'put a loudspeaker on your thoughts'. If this technique is used by the tutors during the session, it will make it possible to identify what strategy the learners need to use to identify the knowledge to be mobilised depending on the type of problem. In this case, the example solution and clarification of criteria would no doubt provide learners with more specific tools, enabling them to remember how to approach a similar problem in the future and mobilise the appropriate knowledge.

5.4.4 Threats to Validity

It is worth pointing out that for this work, we only analysed selected parts of a larger course. While the overall structure of this course repeats this organisation, this limited analysis still might constitute a limitation. However, thanks to our knowledge of the course and the links between its different concepts we did not entirely restrict the analysis to week 4 anyway in order to not miss important links relevant to learning transfer. In other words, even though week 4 was used to analyse certain aspects of the course in detail, we still took elements from other moments in the course to illustrate all aspects, in particular the evaluation or aspects of content taught later in the course.

As mentioned in the introduction, another possible limitation of our analysis was that it was performed on the material of an earlier course setup when the programming language of the course still was Java and not Python. Nevertheless, the course has always had as main learning objective to teach introductory programming in a way that is not tied too closely to a particular programming language. It is regarded by the teachers as a conceptual introduction to programming, where the particular programming language chosen is only a means to an end. In fact, when the course switched to Python, the structure, missions and basic concepts have mainly remained the same. While some adaptations necessarily have been made for Python, we still think the results of the analysis can be extended straightforwardly to the current setup of the course.

5.5 Conclusion

In this chapter, in order to answer **RQ5.1**, we analysed the CS1 course based on the following processes of learning transfer models [Tar99; Bra04]: knowledge encoding (knowledge viability and knowledge organisation) and

knowledge retrieval. Based on this thorough inspection of the course material, we suggested that explicit programming strategies can help foster learning transfer in this context. In particular, the analysis yielded three proposals to improve the course:

P1. The organisation of the learning objectives should be made more apparent throughout the entire course. P1 is related to literature on explicit direct instruction and the way Hollingsworth et al. suggest to organise lessons [HY17].

P2. Transfer opportunities should be pointed at beforehand in order to prepare the learner to recognise future situations in which they could recall previous knowledge.

P3. More explicit recall strategies should be proposed in the course to help retrieve known bits of knowledge in order to apply them to solve new similar situations.

P2 and P3 are in accordance with what CLT prescribes about instructional design. It promotes explicit strategies in the sense that we must automate content-related knowledge and so, familiarise learners to new content and enable them to transfer knowledge.

In the next chapter, these proposals serve as a basis to select the explicit instructional programming strategies we will use throughout our research. The four strategies we selected are aligned with P2, P3 and the instructional recommendations aligned with CLT.

First iteration: Exploration

6

This chapter is largely based on the paper:
O. Goletti, K. Mens, and F. Hermans. "Tutors' Experiences in Using Explicit Strategies in a Problem-Based Learning Introductory Programming Course". In: *ITiCSE '21*. Virtual Event, Germany: ACM Press, June 2021. DOI: 10.1145/3430665.3456348.

This first exploratory study aims to explore how tutors could benefit in their tutoring from explicit instructional strategies. Prior work has shown that the strategies we selected in this study are effective [XNK18; MMD19; EMR17; Lok+16] (cf. Section 2.5). Therefore, we focus our study on tutors to assess how comfortable they are with adopting new instructional strategies. These are the research questions we answer in this chapter:

RQ6.1 How did tutors apply and adapt the strategies?

RQ6.2 What strategies did tutors prefer and why?

RQ6.3 What do the tutors think about using explicit programming strategies?

6.1 Selected Strategies

Our study focuses on four strategies from research literature. Each of these strategies was already presented in Chapter 2. To select these strategies, we first identified criteria based on the analysis' results presented in the previous chapter and on cognitive load theory. We then searched the CSEd literature for evidence-based instructional strategies reported on in the context of introductory programming courses and picked those strategies that correspond to our identified criteria and that match cognitive load theory (CLT) instructional recommendations and the three learning transfer inspired proposals (P1, P2 and P3, cf. Section 5.5) from our course analysis:

Explicit tracing, cf. Section 2.5.1. Explicit tracing is in alignment with CLT because it automates the process of executing code and uses external representations to lower cognitive load. The use of a written representation of the state of the program is also aligned with the learning transfer model of Bracke as discussed in the previous chapter (5.2.1). This helps with unloading the working memory, fostering accessibility of knowledge and learning transfer. It addresses P3 because it is an explicit recall strategy that will remind students about the semantics of a program and how to execute code properly.

Subgoal learning (SL), cf. Section 2.5.3. SL is linked to proposal P2 and the promotion of transfer opportunities. This is achieved by naming the concepts and labeling the steps needed to use these concepts when solving a related programming exercise. It also aims to automate the recognition of patterns as proposed in P3. Worked examples are also a strategy identified by CLT proponents to lower cognitive load for learners. By showing and explaining in detail the steps of a solved example, the learner is able to focus on the solving procedure better than when they need to solve an exercise at the same time, allowing for better learning.

Parsons problems, cf. Section 2.5.4. The idea of putting lines of code in the right order instead of writing them is heavily inspired by CLT and diminishes cognitive load. Novice programmers do not need to focus on remembering the syntax and how to write code properly. They can focus on understanding the proposed pieces of code and on the logic needed to rearrange them in a working solution. It is also a way to practice the recognition of patterns shown in worked examples, and is therefore linked to proposal P3.

Explicit problem solving, cf. Section 2.5.5. This strategy has as main goal to automate the self-regulation that will help a learner take advantage of metacognition. It is a way to diminish the difficulty of using higher level cognitive strategies in an unfamiliar context as suggested by CLT. Naming the different steps involved in the higher level process of solving problems and explicitly prompting students to think about previously seen problems and how to adapt their solutions fosters the learning transfer by making it visible. Doing this in an explicit way is aligned with proposal P3.

6.2 Study Design and Methodology

The goal of this study was *to understand how tutors could incorporate explicit strategies in their tutoring* and what were their thoughts on the use of explicit strategies. Therefore we had to find those tutors, explain the strategies to them and interview them. This section details our methodology.

To the twenty-five tutors of our CS1 course, we distributed a survey with questions on how they saw their role as tutors for the course. Twelve responded and four among them accepted to test our explicit programming strategies. Since tutors had little pedagogical experience, we introduced the strategies to the tutors gradually. This was done to not overwhelm them with too much information at once. We also thought it would give them more time to fully focus on each strategy at a time. We met those four tutors once a week nearly each week of the semester. During these weekly twenty-minute meetings, we would discuss the previously seen strategies. We discussed feedback, adaptation, best practices, interrogations, etc. Every two other weeks, we proposed a new strategy, gave the paper it was taken from to the tutors, presented its motivations and objectives, explained how to apply it, gave a usage example and answered tutors questions.

Tutors were encouraged to use these strategies with their students and to modify and adapt them as needed. Regular feedback through the weekly meetings allowed us to know what tutors tested and how they adapted the initial proposed strategy. This approach was preferred instead of a more rigid "follow these steps" approach because we trust them in the end to *"weave it all together into something that works in the classroom"* [Lis16].

At the end of the semester, each of the tutors was interviewed for about one hour in a semi-structured way. Questions were prepared to not miss any specific point during the interview, as presented in Section 3.4.1. In order to answer **RQ6.1** on their use and adaptation of the strategies, we asked more detailed questions on what they recalled of each of the four strategies, how and when they applied it and its perceived pros and cons. To answer **RQ6.2** on the tutors' preference, we asked them to compare the strategies and rate them according to some criteria such as ease of application or effectiveness. The interview also included a few questions on their experience as a tutor, what changed in their practice throughout the semester and their thoughts on explicit methodologies. These questions aimed at answering **RQ6.3** on their views on explicit strategies. The weekly meetings were recorded and were used along with the interviews of the tutors as a source for the qualitative aspect of this study.

The transcripts of the interviews were coded following the method described in Section 3.4.2. We used codes emerging from the interviews, as

well as codes induced by the themes of the different parts of the interview. The first interview was coded by two researchers and then the author of this dissertation coded the three other interviews. In total, we coded 431 quotes from the interviews and the weekly meetings. The codes were then regrouped in categories that we used to answer our three research questions.

6.3 Results

This section first presents overall information derived from the coding of the interviews, then we answer the three research questions one by one. In this section, some quotes from tutors' interviews are provided and tutors are anonymised by using "TX", with X between 1 and 4, which refers to tutor X.

6.3.1 Tutor's Use of the Four Strategies

In this section we analyse what the tutors reported on their use of each of the four strategies, by order in which they were introduced to the tutors. For each strategy, the categories are treated in order of number of coded quotes given in brackets. When no quote was coded in a category, it is left out. The final coded categories are presented in Table 6.1.

Table 6.1: Final coded categories representing tutors' comments on the strategies during their interviews.

Category	Description from tutors' point of view
Pros for tutors	seen as a benefit of using a strategy
Pros for students	seen as helping the students
Limitations	what hindered students when using a strategy
Application difficulties	what hindered tutors
Suggestions	improvement suggested to a strategy

6.3.1.1 Explicit Tracing

This strategy was the favorite of all four tutors. For tutors, it is easy to understand and simple to apply. It is useful for all students, simple to use, and helps code comprehension, testing and debugging code. Tutors considered explicit tracing as reassuring for the students. T4 said tracing would help later in their curriculum. Tutors also stated it reduces effectively the mental load for students. Tutors regarded the strategy as unfit for longer executions. They made some suggestions to improve this strategy.

Pros for students (38) Tracing was mostly used to help students identify where they misunderstood a statement or made an incorrect assumption. T3 said: *"Students can figure out by executing step by step that what they think is different from the result of the execution. By forcing oneself to write and trace, a student can arrive by himself to a conflict and then to the correction of the code."*

Tutors agreed it really helped students. For example T2 said: *"The idea is to write it instead of keeping it all in their head"*.

Pros for tutors (25) Tracing was compared with debugging and thus felt familiar to tutors. T4 was shocked that it wasn't explicitly taught to students: *"I thought tracing code was already taught in [this course]... I think it is very useful to trace at the beginning."*

This strategy was the easiest to apply with the best results. For example, T1 described how the strategy helps students: *"It's the strategy I used the most [...] At first, one doesn't have the proper methodology to trace code. We just try to remember all variables and we just go too fast. But with the table we take our time, we update it after each statement, we calculate each expression separately, I think it helped them. I just did a reminder on this at Tuesday's lab because they asked me [...] It's just that, it is clear and it works well [...] They seemed to say they would use it during the exam."*

Limitations (12) The four tutors were unanimous to say that tracing takes a lot of time, especially for longer code.

Suggestions (7) Tutors suggested how to improve explicit tracing. E.g., one proposed to use several students *"chained"* to execute separate parts of the code or nested function calls. Another adaptation was to trace only chosen variables of interest in a program.

6.3.1.2 Subgoal Learning

Tutors regarded subgoal learning (SL) as more complicated to apply than the previous strategy. They had a hard time applying this strategy properly. They said they understood the underlying ideas but that it needed preparation and memorising of the proper labels.

Application difficulties (20) Although tutors saw that identifying the subgoals for the students was helpful, they said it was difficult to articulate the difference between context steps and generic steps. Tutors could not relate to a similar strategy emphasising the generality of the steps to write

a loop. Identifying these steps felt implicit for the tutors so the strategy seemed *"overly theoretical"* (T3).

Tutors fell into a live attempt to find the proper steps to use a specific construct if they did not stick to the provided labels. Such a live exercise is difficult and is a reason why the strategy was designed [MMD19]. T1 said on this topic: *"Well, if it's not explicit for us, it's difficult to explain it for the students... We know all that implicitly. But, it's never easy to explain like that if we haven't taken the time to sit down and say when I do a loop, step 1 is that, then that, etc."*

Because SL needs preparation and memorising, it demanded time from the tutors. This was a major hindrance in their first attempt at applying the strategy. Another difficulty was that subgoals were provided for many different constructs and are different when reading or writing. This led tutors to adapt the strategy by doing it orally and not as explicitly as proposed in the paper.

Limitations (7) A limitation mentioned by two tutors is that students would expect the solution to be given to them if the tutor often wrote it on the board to highlight the different subgoals.

Pros for students (7) Three tutors used the strategy and found that it was particularly useful for students who had more difficulties starting from a blank page. The idea of showing an example was fairly well adopted, especially during the introduction of a new concept. Tutors saw it as presenting a plan that students could refer to later on but stressed that one example is not enough and that balance should be found to avoid students expecting answers to be given. For example, T2 said: *"It's good to do this for the first time when a concept is discussed to show them how to solve a new type of exercise... It saves time rather than floundering... It provides them a well-solved reference exercise that shows the steps."*

6.3.1.3 Parsons Problems

This was the second-favourite strategy of the tutors. They liked it because it was engaging for students of all levels. Another main advantage was that students could see more examples in less time. Nevertheless, tutors said it required time to prepare and it was not always clear when to use it.

Pros for students (28) The importance of good paired distractors was stressed since it forced students to justify their choice. One tutor tried unpaired distractors, but found that this was too hard for the students since they would try to fit all the lines of code in one solution. T3 describes

its use: *"Line by line, without indentation, with distractors. To stimulate discussions in certain structures. For example to see if an* `else` *is mandatory or not."*

Tutors highlighted the benefit of Parsons problems to enable students to see more examples of solved programs in a short time, as reported in the original study [EMR17]. E.g., T1 said *"[students are] really just thinking about the meaning. That way, they could see more different examples since they don't have to waste time writing. And learn faster, well that's the objective of the method."*

It was motivating for student not having to write all exercises by themselves. It mitigated the blank page syndrome. Tutors reported that students were more involved, more active and enjoyed the strategy; even those students who had more difficulties.

Limitations (10) Nevertheless all tutors agreed that Parsons problems, when done on paper, required time and preparation.

Application difficulties (10) The main difficulty for tutors was to identify exercises that would benefit from Parsons problems. Two tutors said they did not know when to use it and would have preferred to experience it by themselves first, to have more practice.

Suggestions (8) Some suggestions were made for Parsons problems. T3 suggested it could be used as a way of assessing student knowledge or diagnosing if a mistake is systematic, by giving them distractors on a specific misconception. T3 said that *"[with this strategy] we could see if we used a variable before assigning it or if we swapped two lines of code if it is systematic or a distraction error."*

Tutors also said that an automatic online solution might be more usable, but then they would lose the opportunity of discussion with and between the students in the classroom.

Pros for tutors (6) In order to properly invent a new Parsons exercise, T2 said tutors need to *"know how to trap students. To make good distractors, you have to know the corner cases that make a program not work"* and that he liked that. T3 said that it allowed him to put more or less difficulty in an exercise.

6.3.1.4 Explicit Problem Solving

This strategy was more controversial. While two tutors rated it as difficult to apply and understand, the other two used it successfully. It was straight-

forward to use for the students but difficult to understand because it was meta and at the same time described obvious steps of problem solving.

Pros for students (19) As with previous strategies, tutors noted that this strategy helped students who did not know where to begin. T3 said: *"some students need a course of action or they don't know what to do."*

The strategy was helpful to students. It needed to be applied systematically and if students didn't stick to it, they would burn steps and make mistakes. Self-regulation is difficult for students [LK16]. We can see the meta-cognitive impact when T2 said: *"They can see that it works when I ask them questions but cannot ask the question themselves."* or when T1 said: *"It's a bit like trying to work as if you are working in a group but alone."*

Tutors found this strategy complementary to the second strategy as they regarded that one as about translating a natural language resolution to code and this one about the whole process, starting from reinterpreting the statement in natural language.

This recall strategy especially matches a need identified in our analysis and was seen as such by T1: *"If you remember something that worked, it's very easy to reapply it. And taking the time to explicitly ask 'OK have I already done something that looks like that?' Sometimes it's quite silly but if you think about it for two or three minutes you find that it is the case."*

Limitations (11) Some difficulties were also mentioned. Tutors found it difficult for students to identify similar problems, especially when seeing so much new material in the course. But even tutors often only saw similar problems in a very narrow sense.

Application difficulties (9) Even though tutors saw the need to make the steps of problem solving explicit, they found it sometimes too obvious and so did students. Some tutors reported that they were not sure if it would help. They had to be convinced of the usefulness of a strategy before using it, as T4 stated: *"I didn't test it for lack of time and because I didn't really see the point."*

6.3.2 RQ6.1: Use and adaptation of the strategies

The overall impression is that tutors found the strategies useful and effective. Tutors used a strategy more easily if they could understand the motivation behind it. They would reuse a strategy when it helped the students, didn't take too much time and when it increased the students' motivation. The tutors did not hesitate to test the strategies. They adapted it to their own

practice or to what they understood. Still, they sometimes reported not being sure on when to use a strategy and a need for more practice before using it with the students. T4 even said that for one strategy they felt lost: *"Do not just leave the tutors facing the paper because there is quickly a way to get lost and make mistakes."*

Tutors really need to understand the goal of a strategy to be convinced it has value and to use it properly. Otherwise, they would have a tendency to focus on the context and not on the general principle a strategy emphasises. Especially with more abstract strategies like subgoal learning, they needed to understand them well. Otherwise, they would struggle to find the proper labels on the spot, which is kind of the main reason for using that strategy.

A difficulty encountered was that lack of time and preparation were a brake on instructional changes. It is known that TA's and by extension also tutors suffer from class management issues [LBG00]. It is well possible that in this case, since they had little pedagogical background and since three of them were tutoring this course for the first time, it hampered their pedagogical confidence and hence the degree to which they could adapt instructional experiments.

6.3.3 RQ6.2: Preferred Strategies

When comparing strategies, tutors found that explicit tracing was the easiest to understand and apply as well as the most effective one. Using Parsons problems was considered easy and effective even though it took more time to prepare. The third most effective strategy was explicit problem solving according to tutors and it was also third easiest to understand. Finally, subgoal learning was the most difficult to understand and least effective according to the tutors.

The strategies seen as more effective were not necessarily easier to apply. For example Parsons' problems was difficult to apply according to tutors. They liked to see that a strategy was useful. Even though some strategies were seen as broadly applicable, tutors mainly saw them as useful for students in difficulty. This was a major motivation for tutors to try the strategies.

Overall, based on the follow-up of four different tutors and our analysis of the interviews, we can say that when adopting strategies tutors preferred those matching these four criteria: easy to understand, straightforward to apply, useful for students on the long term and supported by literature.

6.3.4 RQ6.3: On Explicit Programming Strategies

Tutors found the explicit nature of the strategies effective and *"less demanding for the students"* (T1). To counterbalance explicit strategies, they said

that students still need occasions to explore strategies by themselves and figure out what works best for them. They said that explicit strategies should be shown especially when introducing new material but that students should then have some less guided time to try them out and adopt or adapt them. T2 said: *"I think it is good to do things that are very explicit for a student, to show him how to do an exercise properly and that he can adapt it for himself."*

Two tutors also feared that students would not search by themselves anymore because of explicit strategies. T1 said: *"The disadvantage is that they may not search by themselves, it will be much less personal. It will be like that. Because we said it works. But not because they have tested it and . . . Maybe there are other methods that work well and they may never find out on their own."*

The tutors seem to agree that more open exercises are also beneficial for students and that less explicit methods would force students to learn how to "figure it out". They clearly associate implicit with letting the students work it out by themselves, like T2: *"Less explicit, it is rather when they have more freedom, when we give them the statement . . . and we do not give a course of action [. . .] It is good from time to time to leave the students a little more on their own."*

To summarise, tutors view explicit strategies as an effective way to teach and automate good practices for students. But they have a more balanced view than expected on this topic. Indeed, for most of the tutors, an inquiry-based, less guided methodology still has its place in the course. This might be due to their own experience of the course and also by their feeling that students need to learn to search by themselves.

6.3.5 Threats to Validity

One main perceived limitation of this first study is the small number of participants. Nevertheless, we think the qualitative nature of this exploratory work allowed us to have an in-depth view of tutors' perspective on strategy usage. Also, for this exploratory study, we preferred motivated tutors genuinely interested in learning new pedagogic tools, rather than imposing new strategies on all tutors yet. Since tutors are used in similar setups in other institutions' CS courses, we also hypothesise our criteria can be generalisable. However, our main goal was to take these criteria as a starting point to guide the design of our next iteration, in which we intend to involve more tutors.

6.4 Conclusion

In this exploratory study, we studied how four tutors used and experienced the use of four explicit strategies in a CS1 university course. The strategies were chosen because they were shown to be effective and explicit, and in accordance with the three proposals that were drawn from our earlier analysis of the course and following instructional recommendations made by cognitive load theory.

The four strategies were gradually presented to and tested by the tutors in the course during which regular meetings were held. At the end of the semester interviews were conducted, transcribed and analysed to answer three research questions. We observed that the tutors liked these new instructional strategies even though it was sometimes difficult for them to use them properly. They neither had sufficient time to prepare their lab sessions with these strategies nor the pedagogical experience to be confident enough to try more complicated strategies.

Based upon our interview analysis, we propose four criteria for a strategy to be more easily adopted by tutors with little pedagogical background. A strategy has to be easy to understand, straightforward to apply, useful on the long term and supported by literature. Tutors consider explicit strategies effective and useful. In particular, tutors preferred explicit tracing according to these criteria. Nevertheless, they believe a trade-off is to be found regarding more inquiry-based strategies so that students are left the opportunity to find out by themselves the best strategies for them to use.

We believe our results can be generalised to other introductory programming courses with a similar setup. In particular, since problem-based learning and tutoring are widely adopted in CS courses, our work could support them using more explicit programming strategies. Our research points in the direction of more actionable materials that could be given to tutors. A short training with the highlights of a strategy and an example of how and when to use it properly could help them a lot.

This exploratory study still left a lot of research questions unanswered. In further chapters, we will elaborate on some of them like how do tutors do actually put these strategies into practice? How do tutors adapt them? How often do tutors do use the strategies? And how does it all compare to what they said and reported in this chapter?

Subgoals Creation 7

This chapter is largely based on the paper:
O. Goletti, F. De Pierpont, and K. Mens. "Création d'exemples résolus avec objectifs étiquetés pour l'apprentissage de la programmation avec Python". In: *Didapro 9–DidaSTIC*. 2022.
The title of this publication can be translated as "Creation of subgoal labeled worked examples for teaching programming with Python".

To continue with the subgoal learning strategy, and with the aim of creating a training document to encourage its use by tutors, in this chapter, we present the work that went into creating the subgoal labeled worked examples (SLWEs) adapted to the concepts of the course and to Python, the programming language of the course.

7.1 Introduction

To create the SLWEs, we analyzed exercise resolutions made by experts. The methodology for this analysis is Task Analysis, an effective method for extracting implicit procedures that domain experts use implicitly to solve complex tasks. In particular, we have adapted Catrambone [Cat11]'s Task analysis by problem solving (TAPS) methodology, used and recommended by Margulieux et al. [MMD19] when creating SLWEs for Java concepts.

This chapter therefore presents our adaptation of the TAPS methodology for creating SLWEs, as well as the SLWEs we have created.

This chapter answers the following research questions:

RQ7.1 How to create appropriate subgoals for the concepts of a CS1 course?

RQ7.2 How to present subgoals for tutors' training and usage?

7.2 Study Design and Methodology

7.2.1 Task analysis by problem solving (TAPS)

The aim of task analysis is to make a domain expert extract his implicit automated knowledge and his way of solving a problem [Cla+08]. Several techniques exist for this purpose, based in particular on observation of problem-solving by an expert, note-taking, writing up a procedure, and so on. Experts solve problems differently from novices, and their explanations are typically incomplete and insufficient for a novice. Even teachers who try have this expert blind spot and don't especially succeed [Cat98].

Catrambone's Task Analysis by Problem Solving (TAPS) [Cat11] is the methodology used and recommended by Margulieux et al. In TAPS, the subject expert is called SME (Subject-Matter Expert), and will be referred to as "the expert" in the remainder of this chapter. As for the Knowedlge Extractor Expert (KEE), we'll call him "the analyst". The expert must identify tasks related to the concepts to be processed and solve them in front of the analyst. The analyst is preferably a novice in the subject to be analyzed, and should take notes on the expert(s)' solving technique.

The steps recommended by TAPS are:

1. The expert identifies the exercises that a learner should be able to solve if they mastered the material.
2. The expert solves one of these exercises.
3. The analyst takes detailed notes on the reason for each step in the solution and asks the expert to justify each of these steps, which the expert sometimes finds difficult to verbalise.
4. The analyst goes back to their notes, reorganises them, extracts the procedures and justifications.
5. The expert solves a new problem.
6. The analyst completes their notes to fill in gaps and resolve inconsistencies.
7. The analyst attempts to solve a similar problem based on their notes.
8. The analyst re-interviews the expert until he can solve the exercises themselves.

Finally, exchanges between the experts and the analyst lead to the drafting of a document. For each concept addressed, this document describes the resolution procedure, labeled objectives for structuring generic elements, and annotated solved examples illustrating the resolution procedure.

7.2.2 Adapted TAPS

The methodology used in our work is very similar to TAPS. The experts were four: a professor of computer science for over twenty years and co-teacher of the introductory programming course for the past four years[1]; the author of this dissertation, an assistant for the introductory programming course for the past six years and a doctoral student in computer science education; and two doctoral students in computer science, also TAs for the course. As for the analyst, he was a master's student in computer science, so he was not a novice in programming as recommended by TAPS, but he had never taken a course in Python.

Each time, the experts identified the exercises to be mastered for the various concepts covered in the course (cf. Table 7.1). The analyst then invited one or more experts, through videoconferencing if necessary, to record their resolution of the selected exercises. The experts worked in Thonny[2] (a simple Python IDE) with a shared screen. The analyst asked the experts questions to justify the various steps. The analyst took notes and could consult the recording afterwards. If necessary, however, he could call in another expert for the same concept to fill in gaps or resolve inconsistencies in his notes.

As the analyst was a master's student in computer science, we decided not to do the last two steps of TAPS, which expect the KEE to attempt to solve problems on his own, based on his notes.

7.3 Results

We have created SLWEs for the concepts given in Table 7.1. The entire training document is available online[3] (in French).

To illustrate the training document and the created labels, Figure 7.1 presents the subgoal labels for writing a program that reads a file in Python with the labels in bold and the associated SLWE in Figure 7.2.

The training document containing all SLWEs and a description of their use was used to train tutors in the first semester of the academic year 2021-2022 and again in 2023-2024. The way in which tutors used this strategy during their tutoring session will be the subject of the next chapters.

[1] The dates mentioned in this paragraph have not been updated and need to be understood as at the time of submission of the corresponding paper, ie. 2022

[2] https://thonny.org/

[3] `SLWE - v13.pdf` at http://hdl.handle.net/2078.1/263637.

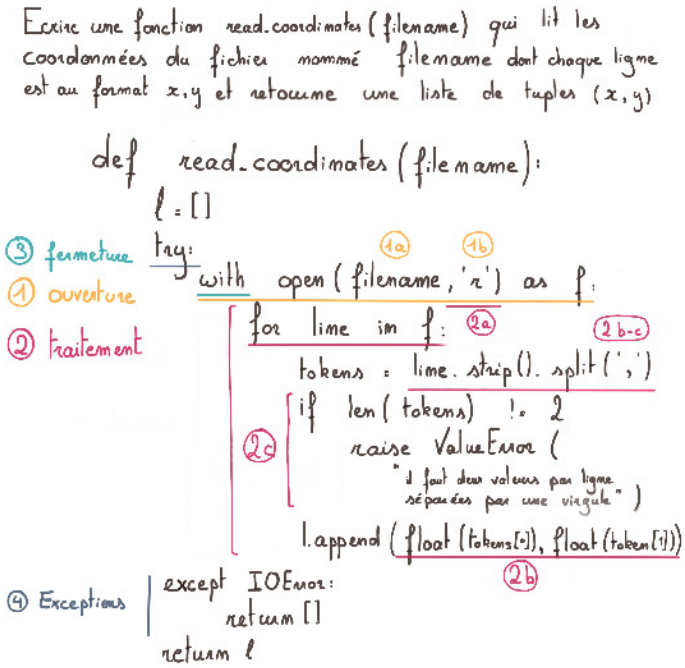
Table 7.1: Concepts for which SLWEs have been created for Python

Concept	**code reading**	**code writing**	**Python adaption from [MMD19]**
assignment	x	x	x
conditional	x	x	x
while loop	x	x	x
function	x	x	x
for loop/ sequence traversal		x	
file read		x	
file write		x	
dictionary creation or update		x	
class creation		x	
add a node in a linked list		x	
remove a node in a linked list		x	

Figure 7.1: Subgoal labels for writing a program that reads a file in Python.

1. **Open** file
   (a) Identify the file name and path (usually as part of `filename`)
   (b) Choose appropriate mode (typically: `"r"`)
   (c) Open the file by choosing either:
      - `with open(filename, mode) as f:`
      - `f = open(filename, mode)`
2. File **processing** depending on its format
   (a) File traversal
      - Line by line with: `f.readline()`
      - Iterating over each line with: `for line in f:`
   (b) Line processing
      - Getting rid of leading and trailing white spaces: `line.strip()`
      - Splitting in tokens according to line format: `line.split()`
      - Converting tokens depending on expected type
   (c) Handling formatting errors (ignore line, `raise ValueError`)
3. **Close** file
   - If you used a `with` statement, skip this step
   - Else, with: `f.close()`
4. Handle **exceptions** that might occur during file processing (typically `IOError`)
   (a) Surround your file processing code by: `try: ... except:`
   (b) Handle specific exceptions with: `except error_type: ...`

Figure 7.2: Worked example of a file reading exercise in Python labeled with our created subgoals for that concept: (1) Open; (2) Processing; (3) Close; (4) Exceptions.



Threats to Validity

As explained in Section 7.2.2, as opposed to the original TAPS protocol, the KEE was not a novice. While he had never followed a course on Python, we are aware that his programming expertise in other languages may introduce a bias in the SLWEs. However, we felt that the richness of the different profiles of the research team (researcher in CS education and assistant, teacher, student) enriched the view and attention paid to the choices of SLWEs and constituted an advantage for this work.

7.4 Conclusion

The aim of this chapter was to contribute to computing education by proposing subgoal labeled worked examples (SLWEs) for teaching programming concepts with Python. The use of these SLWEs is aligned with the instructional recommendations of cognitive load and learning transfer theories. The creation of SLWEs was necessary because none existed yet for the concepts studied in Python. We answered **RQ7.1** by showing it was possible to produce SLWEs for the selected concepts using a task analysis methodology documented in literature. The TAPS task analysis was adapted to extract the knowledge of a programming expert. A list of concepts seen in a first university-level introductory programming course was selected, and the recordings of exercise resolutions for each concept by several experts were analysed by our analyst, the KEE. He derived a problem-solving procedure for each concept and identified labeled subgoals for each procedure. These labeled procedures and subgoals were discussed and validated by the authors. A training document was then written to train the course tutors involved in the experiment to answer **RQ7.2**. In this document, the procedures for each concept are detailed, labeled subgoals are provided and one or more subgoal labeled worked examples are provided. This document was used to train seven tutors in the use of subgoal learning in their tutoring of an introductory programming course in Python. This document is publicly available (in French) and reusable by other members of the computing education community.

Second Iteration: Characterisation

8

This chapter is largely based on the paper:
O. Goletti, K. Mens, and F. Hermans. "An Analysis of Tutors' Adoption of Explicit Instructional Strategies in an Introductory Programming Course". In: *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research*. 2022, pp. 1–12. DOI: 10.1145/3564721.3565951.

Previous research showed that tutors struggle to identify and use best practices [Rie+21]. Explicit instructional strategies are gaining traction lately and the effort to bring research results into tutors' practice is worth exploring as advocated by Sue Sentance during the keynote she presented at Koli Calling 2021 [Sen21].

Based on the opinion of the tutors in their interviews during the first iteration (see Chapter 6), we decided to support their adoption of explicit strategies by providing dedicated training material, by limiting to only two the amount of presented strategies (explicit tracing and subgoal learning), by introducing both strategies at the beginning of the semester in dedicated training sessions and with a personalised follow-up during the course semester. For the subgoal learning strategy, we used subgoals and subgoal labeled worked examples created specifically for the CS1 course in Python as presented in the previous Chapter 7. This extra training and support aimed at ensuring both strategies were easy to use and straightforward to apply.

Based on research on educational change, *fidelity of implementation* and *adaptation* of research-based instructional strategies [ODo08; Bor+13], we used in this study a mixed methodology to analyse recordings of tutor interventions during their lab sessions. Our aim was to understand how tutors adopt and adapt explicit instructional strategies in their classroom setting.

Based on Chapter 6), we already have a good grasp of what tutors' views on using such explicit strategies are. Our current chapter gets a better insight on what tutors actually *do* in the classroom. The research

questions we answer in this chapter:

RQ8.1 What triggers the use of a specific instructional strategy by tutors?

RQ8.2 What is the fidelity of implementation of the instructional strategies by tutors?

RQ8.3 What adaptations of the strategies are made by tutors during instruction?

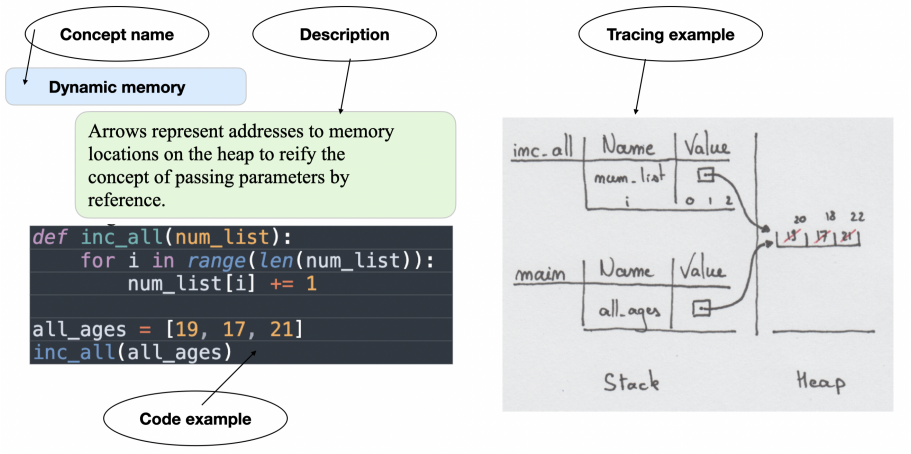
8.1 Study Design and Methodology

8.1.1 Training Material

Our prior experiences with training tutors with instructional strategies advised to put more effort in training them. Moreover, it was observed that adoption is facilitated when tutors understand the underlying hypotheses and claims of instructional strategies. In that study, tutors themselves also asked for more guidance on how and when to apply the strategies. In response to that request, for this study two documents were prepared for training the tutors. Each of these documents explain the promises of the strategies, provide instructions on how to apply them as well as quotes from previous adopters and exercises to verify proper understanding of the strategies[1] [GDM22]. In Chapter 6, four strategies were progressively introduced to tutors throughout the semester of the course. In the current experiment, we decided to present our strategies at the start of the semester instead and limit ourselves to two strategies, in order to reach more depth. The two strategies were selected among the four strategies explored previously as follows. Explicit tracing was chosen because it seemed to have been the most effective and preferred strategy according to the tutors. And even though it seemed less effective to tutors, subgoal learning was also selected because we believe that with better training it could also be effective as supported by the literature presented in Section 2.5.3. Tutors in our prior study explicitly mentioned lack of training and lack of time as main obstacle for proper strategy use. We addressed that shortcoming in this chapter by providing more training and guidance through dedicated training, dedicated training documents and follow-up during the semester. Also, both selected strategies have the advantage of being easily observable since they require tutors to make use of the blackboard.

[1] `SLWE - v13.pdf` and `Explicit Tracing - v5.pdf` at http://hdl.handle.net/2078.1/263637.

Figure 8.1: External representation of a list (inspired by Dickson et al. [DD21]) as handed out to our tutors. Translation, ellipses, arrows and highlighted parts have been provided for readability.



8.1.1.1 Explicit Tracing

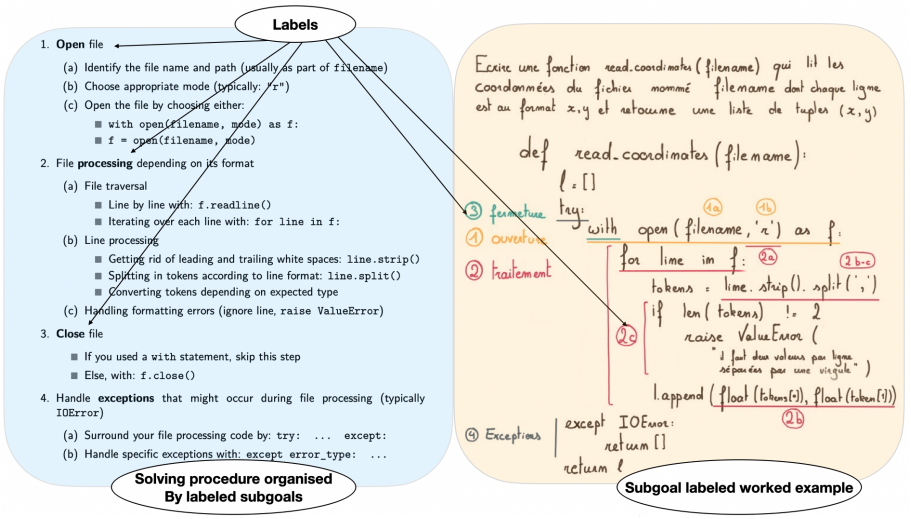
The document presenting the explicit tracing strategy is written in French and available online[2] [GM22]. Figure 8.1 is an example of memory representation of a concept as given in the document. Annotations have been added to highlight the important parts and we translated the example to English for the sake of readability.

Our document introduces the explicit tracing strategy, gives step by step instructions on how to apply it as presented in Section 2.5.1, sums up its advantages and claims, gives advice on when to use it and introduces and exemplifies each memory representation of the different programming constructs introduced in the course.

The explicit tracing strategy we use has been presented in Section 2.5.1. The main steps are reused and explained in the handout document we prepared. However, since using an external representation when tracing code can take many forms [Cun+17], we decided to use a normalised memory representation based on the work of Dragon and Dickson [DD16; DD21]. The examples and exercises proposed in our document are adapted to the Python programming language and specifically limited to the programming concepts seen in the CS1 course.

[2] `Explicit Tracing - v5.pdf` at http://hdl.handle.net/2078.1/263637.

Figure 8.2: SLWE for file writing in Python. (The presentation on the left is inspired by Margulieux et al. [MMD19].) Subgoals' translation, ellipses, arrows and highlighted parts have been provided for readability.



8.1.1.2 Subgoal learning

The document presenting the subgoal learning strategy is written in French and also available online[3] [GDM22]. Figure 8.2 is an example of an SLWE as given in the document. Annotations were added to highlight the important parts and we translated part of the example to English for the sake of readability.

The strategy is for the tutors to use subgoal learning (SL) in their teaching by presenting the subgoal labeled worked examples (SLWEs) to the students to illustrate how to solve problems involving new programming concepts. Then tutors need to reuse the labels of the subgoals to annotate code when solving similar problems and go through the labels to help construct such a solution.

The worked examples presented in the training document are extracted from the CS1 course to illustrate the different concepts. They are annotated in color with the corresponding subgoal labels. The subgoal labels for the four first concepts (assignment, conditionals, loops and functions) are adaptations to the Python programming language of those proposed by Margulieux et al. in their article [MMD19]. The others (list traversal, file reading and writing, dictionary creation and update, class creation, node insertion and removal in a linked list) have been created by the authors.

[3] `SLWE - v13.pdf` at http://hdl.handle.net/2078.1/263637.

For more details on the procedure we used to extract these subgoal labels, see Chapter 7.

8.1.2 Participants

Twenty eight tutors were recruited for this iteration of the course, in 2021.

Before the start of the semester, our research project was exposed to all tutors. Tutors were offered the possibility to be trained and to test the selected explicit strategies. There was a small financial incentive to participate in the study to compensate for the hours invested in preparation, training, focus group meetings and interviews. Among the twenty eight CS1 tutors, seven tutors volunteered and participated in the study. Five were tutors for the first time, the two others had prior experience tutoring courses and only one of them had experience tutoring this CS1 course and was tutoring it for the third time.

In the scope of this study, a training session took place at the start of the semester. The seven tutors were introduced in detail to the two strategies, the theoretical bases on which they are built and their main objectives. The two aforementioned documents were handed out to the tutors. The strategies were exemplified by the author of this dissertation and then practised by the seven tutors to ensure proper understanding. The training session lasted two hours.

During the course semester, the author of this dissertation met weekly with the seven participating tutors for one hour. The goal of those focus groups was to discuss, share and troubleshoot the use of the strategies. Each tutor would typically share how he or she used the strategies during the past week. During the first weeks of the semester, class observations were made by the author of this dissertation to give feedback on how the strategies were used. At the end of the semester, tutors were also interviewed in a semi-directed way to collect their impressions over the course and their practice of using the strategies. The training session, focus groups and interviews were recorded.

8.1.3 Fidelity criteria

In order to assess the fidelity of implementation (cf. Section 3.3) of the strategies by the tutors, criteria had to be developed. To develop our fidelity criteria (see Table 8.1), we used that method. Similarly to Borrego et al.'s study on research-based instructional strategies' fidelity of implementation [Bor+13], which also used this method, we based ourselves on the literature describing each of the strategies and on our own expertise in using and applying the strategies with tutors from our previous iteration presented in Chapter 6.

8.1.3.1 Explicit Tracing

Two critical aspects come from the description of the explicit tracing strategy in Xie et al. [XNK18] who affirm the strategy is effective for ameliorating tracing performance. We also find these aspects in other similar work on tracing [HJ13]. They ground their strategy in two critical aspects:

1. source code needs to be traced **line-by-line** "like a computer does" to enforce a systematic tracing strategy for students and avoid errors. This aspect is supported by the importance of tracing [Lis+04; HJ13] and not letting students invent their own strategy [Per+86];

2. an **external representation of memory** needs to be used to reduce the working memory load of remembering the values of variables. This aspect is grounded in CLT [SvP19] and in the efficiency of sketching in computer science [Cun+17; DD16].

These two aspects combine three elements: **line-by-line** tracing, using an **external representation**, but also the mapping process between the execution of a line and the **update of the state** of the program represented in the memory table. We translated these in the three explicit tracing fidelity criteria **ET1**, **ET2** and **ET3** listed in Table 8.1.
Since the decision was made from the beginning that we wanted to focus on adaptations and not to enforce only exact use of the strategy, we decided to keep the external representation criterion unconstrained to a specific representation such as the one presented in Dragon et al. [DD16]. The strategy's efficiency seems not to be based on that representation and the

Table 8.1: Fidelity criteria derived from literature for the considered explicit instructional strategies.

1. Explicit Tracing	
ET1.	The source code is traced line-by-line;
ET2.	An external representation of the memory is used;
ET3.	The values of the traced variables are explicitly updated.
2. Subgoal Learning	
SL1.	An SLWE from the document is explicitly introduced to the students;
SL2.	The tutor solves a similar problem in a worked example;
SL3.	The tutor reminds the students about the subgoal labels of a SLWE;
SL4.	The solving procedure of a SLWE is used to solve a practice exercise with the students.

validation of the strategy for more advanced memory representation such as objects and data structures had not been established by Xie et al. [XNK18].

8.1.3.2 Subgoal learning

Looking at the different studies using subgoal learning (SL) through the use of worked examples [MGC12; MMG15; MMD19] the main aspect of this strategy is to interleave subgoal labeled worked examples (SLWEs) with practical exercises. This element is grounded in the fading guidance effect [SvP19] and the interleaving of theory and exercises [TR93].
As explained in Section 2.5.3, SLWEs are built on the effectiveness of showing worked examples. Therefore, the labels are first shown in a **given SLWE** provided in the document. During instruction a **worked example** needs to be solved by the tutor and the steps of the solution procedure need to be shown and addressed. Subgoal learning through the use of SLWEs adds to this the **use of labels** to highlight the generic structure of the solution procedure. Finally the last critical component is to reuse the solving procedure in a similar problem with the students in a **practice exercise**. We translated these elements in the four SL fidelity criteria **SL1**, **SL2**, **SL3** and **SL4** listed in Table 8.1.

8.1.4 Methodology

We used a mixed methodology in this study. We observed the tutors during their lab sessions and recorded those sessions for further analysis. During the semester, we observed tutors' lab practices during three different weeks: week 5 on tuples and binary search, week 8 on classes and objects, and week 11 on linked lists. Due to schedule overlap, it was not physically possible to observe all 42 tutors' interventions (7 tutors with 2 sessions per week on 3 different weeks), hence only 24 observations were made. During each observation, the author of this dissertation sat in the classroom on the side and did not intervene. He took notes of tutor's usage of the strategies and recorded the lab session. In total, more than 26 hours of videos were analysed. In the rest of our analysis, we refer to tutors by the notations T1 to T7. Since the different observations were made of different sessions and by different tutors, we decided to analyse *all* the recordings. The final codes and categories presented in the results below are the result of the whole data, we did not stop for saturation in order not to miss adaptations.

During the analysis, we coded segments of all observed sessions using both deductive and inductive coding. Deductive codes come from the strategies' fidelity criteria introduced in Section 8.1.3. They were used for determining when a strategy was used. The coded segments corresponding to fidelity criteria occurrences were counted and used to answer the second

research question on the fidelity of implementation of the strategies by tutors (**RQ8.2**) in a quantitative manner. Those segments were also part of the answer to the third research question on adaptation of the strategies by tutors (**RQ8.3**). For this third research question, a more inductive coding emerged from the recordings and we categorised those codes using a thematic analysis [BC19]. For each strategy use, we also coded and categorised the trigger event in order to answer the first research question on what triggers strategy usage by tutors (**RQ8.1**). In order to code the tutor's usage of the strategies, we took note of the timestamp, the context, the trigger, the fidelity criteria observed and the adaptations when appropriate. We also transcribed quotes of tutors speaking about the strategy and kept a screenshot of visible usage or adaptation of the strategies. This allowed us to easily come back to previous coded segments when the codes and categories had saturated. The coding of the different video segments was mainly done by the author of this dissertation with the help of Prof. Kim Mens to confirm that deductive coding was done in a consistent manner and that the inductive coding covered all observed triggers and adaptations. Two full lab sessions were coded by both coders and conflicts were discussed and resolved. Once all the material had been coded, themes emerged for the first and third research questions. Codes were regrouped in those themes through discussions between the coders.

8.2 Results

Based on the analysis of the recordings of the tutors' lab sessions, we answer our three research questions.

8.2.1 Triggers for Strategy Usage

Since we wanted to understand what triggers strategy usage (**RQ8.1**), during our analysis of the recorded observations we documented for each strategy use the event that triggered the use of the strategy. For this part of the analysis, the coding was induced by the observations and the source categories emerged from the data (cf. Section 3.4.2). The following subsections detail the triggers for each strategy and Table 8.2 summarises the number of occurrences of each source.

8.2.1.1 Explicit Tracing

Among the different events that triggered the use of explicit tracing, we identified three sources depending on the origin of the strategy use:

Students. Students are the principal triggers (total: 33 times). Tutors would use the strategy when a student would ask a question (16 times), when students would be confused about the expected output of a program (7 times), when they observed a bug in a student's answer on the blackboard (5 times), when they would see a lack of understanding of an important concept (4 times) or when a conflict would arise between two students' predictions on the output of a program (1 time).

Tutors. Tutors' decisions are the second most occurring trigger (total: 27 times). A tutor would take the initiative to trace a program because they prepared an example for a topic they would consider difficult or to emphasise an important concept (16 times). Tutors also asked students to trace code to verify their proper understanding (6 times). Tutors would also be triggered to use explicit tracing when they wanted to show that two alternative solutions were equivalent or not (4 times). On one occasion, a tutor traced a code snippet because he made an error of prediction himself and decided to take advantage of the situation to model the strategy.

Exercises. Exercises themselves are the third trigger. Some exercises explicitly required the students to make sketches or to draw the state of the memory (13 times).

Table 8.2: Sources of the trigger event for each strategy use.

Sources	Trigger events	Explicit Tracing	Subgoal Learning
Students	Question from a student	16	1
	Expected output unclear	7	-
	Bug in student's code	5	-
	Important concept misunderstood	4	3
	Conflict in students' predictions	1	-
Tutors	Important concept	16	16
	Verify proper understanding	6	-
	Show alternative solution	4	-
	Correct own mistake	1	-
	Verification check list	-	1
Exercises	Exercise statement	13	-
Total		**73**	**21**

8.2.1.2 Subgoal learning

For subgoal learning, we only observed triggers from two of the three sources mentioned above.

Tutors. Tutors would mainly decide to read an SLWE or use it to solve a similar problem, for introducing new concepts, or to summarise a concept at the end of a week (16 times). One tutor would also use the labels as a check list to verify that a proposed solution contains all the needed steps of the solution (1 time).

Students. Students were a second encountered trigger. Tutors would use the labels or revisit an SLWE after having observed multiple instances of misunderstanding (3 times) or similar questions by different students on a corresponding concept (1 time).

8.2.2 Fidelity of Tutors' Strategy Implementation

During our analysis of the video recordings of our observations, systematic counting of the fidelity criteria was done to understand the fidelity of implementation of the strategies. While our goal was not to force tutors to respect the strategies to the letter, we wanted a measure of how often they did apply the strategy as presented. This data will help us answer the second research question on the fidelity of implementation of the strategies by tutors (**RQ8.2**). Since the criteria were deduced from research literature on the strategies, the codes for this part of the analysis were a priori codes and lead to deductive coding (cf. Section 3.4.2). This part of our analysis was thus of more quantitative nature.

We used the following metrics. We counted how many times each strategy was used by each tutor per session. Then we calculate the percentage of number of criteria used for these strategy uses. We counted a strategy as used as soon as at least one of the criteria for the corresponding strategy was observed in the instructional practice of the tutor. One might argue that if not all criteria are present, the strategy is not properly used but that will be the whole point of looking at the adaptations made by tutors in Section 8.2.3 (RQ8.3).

8.2.2.1 Explicit Tracing

Our measures for the usage of the explicit tracing strategy are summarised in Table 8.3.

From the 24 observed lab sessions, we can see that overall tutors used explicit tracing 73 times. The duration of a tracing intervention ranged from 1 minute to 16 minutes with a mean of 3 minutes 27 seconds. Slightly less

than half of the observations (42%) can be considered as expected strategy uses: the tutor would trace a program line by line, typically with the help of the students, and update variable values in a memory table. In nearly all observed strategy uses (99%), an external representation of memory was used (ET2). In fact, it was not observed in one case only. In that specific case, T4 traced the program line by line (ET1) but orally and without external representation. We did observe other instances of oral tracing, but those were quickly followed by the use of a memory table and were counted as using one. The different usage variations of this strategy are discussed in Section 8.2.3.1. When only an external representation was used (i.e., ET2 without ET1 nor ET3), it was to illustrate a specific state of the memory at a specific moment of the execution. It was often used to illustrate the outcome of an execution or the starting state before the execution. Often, such sketches would serve as a basis for more explanations by the tutors but without actual tracing or value updates. This has been observed 21 times. In all 51 occurrences of value updates (ET3), an external representation was used (ET2), but sometimes without a systematic line by line tracing (ET1). This occurred typically when a tutor would draw a sketch of the execution and update variables while making shortcuts in his or her explanations or when they would explain the steps of the program but without referring to the code.

8.2.2.2 Subgoal learning

Our measures for the usage of the SL strategy are summarised in Table 8.4.

The first observation from the data is that the subgoal learning strategy has been used much less than the explicit tracing strategy. Most of the 21 observed uses were pretty shallow with nearly half matching only one criteria (11 times) which often was only mentioning the labels (SL3: 7 times) or providing a detailed worked example solution (SL2: 3 times) on the board

Table 8.3: Measures of fidelity of implementation for tutors' usage of the Explicit Tracing Strategy.

Tutors	Uses	Number of fidelity criteria observed		
		3	2	1
7	73	31 (42%)	20 (27%)	22 (30%)

Which fidelity criteria observed		
ET1	ET2	ET3
32 (44%)	72 (99%)	51 (70%)

Table 8.4: Scores of fidelity of implementation for tutors' usage of the subgoal learning strategy

Tutors	Uses	Number of fidelity criteria observed			
		4	3	2	1
7	21	3 (12%)	1 (4%)	10 (40%)	11 (44%)

Which fidelity criteria observed			
SL1	SL2	SL3	SL4
6 (24%)	8 (32%)	20 (80%)	12 (48%)

for the students or discussing/reminding the students of specific labels not to forget before solving an exercise. In 6 occurrences, mentioning labels (SL3) was done while solving a practice example with the students (SL4). These cases represent the majority of occurrences of two criteria for the strategy. This is a typical use of the strategy that we expected when a tutor would solve with the students a similar problem as one illustrated in the training document using a similar solving procedure. In two occurrences though, the labels were discussed by the tutor (SL3) and a worked example to illustrate them (SL2) was provided. In 3 occurrences, we observed the four criteria, which means that the tutor would use the document (SL1) with the students, provide an equivalent worked example (SL2) or go through the worked example of the document while discussing the labels (SL3) of the corresponding solving steps and then apply the solving procedure to solve a similar exercise (SL4).

Globally, it seems that it was more difficult for tutors to apply the subgoal learning strategy. This might be explained by the need to present the labels to the students. Tutors might not have been prepared enough. Tutors mentioned also in the first iteration they were less familiar with this more abstract strategy than with explicit tracing.

8.2.3 Tutors' Adaptation of Strategies

Finally, we also performed a qualitative analysis to understand and observe which adaptations were made by the tutors to the strategies they used and how far different tutors' instruction diverged from the proposed instructional strategies (**RQ8.3**). For that, we analysed expected and partial uses of the strategies. Figure 8.3 illustrates expected uses and adaptations of both strategies. The coding of the segments was done with the fidelity criteria and also directly induced by the observations. Our acceptation of what constituted an adaptation was broad and is detailed in the following. We present the observed adaptations of strategies by tutors and, as a result of

a thematic analysis (cf. Section 3.4.2), we categorise them in three kind of adaptations: additions, effects and misuses.

Additions. These additions are small adaptations made by the tutors to the proposed strategy. We hypothesise those adaptations will mainly have a neutral or small positive effect on the outcomes of the strategy.

Effects. These effects are observed changes to tutors' instructional practice. They are not modifications to the strategy itself but observations of recurring instructional change in tutors' practice.

Misuses. These are adaptations made by tutors that we hypothesise might lessen the outcomes of the strategy.

8.2.3.1 Explicit Tracing

Additions

Inverse tracing. In this interesting adaptation of the explicit tracing strategy, we observed multiple cases of inverse usage of an external representation. By this we mean, from a misunderstanding of a statement, a tutor would advise to use or use himself an external representation of the expected output or even a sequence of dynamic manipulation on a drawing and this would then serve as a basis to write the solution program to the statement. It is interesting to notice that in several instances, this practice was quite dynamic in the sense that the external representation would be updated while discussing with the students the steps needed to achieve the expected output. This addition was observed 10 times (in 7 different sessions by 4 different tutors).

Progressive tracing. Another typical use of explicit tracing was a progression in the details added to the sketch. A tutor would typically explain the solution orally, then do a first drawing with very few details, sometimes with generic elements. And when students would still require explanations, they would fall back to a full line by line systematic use of the strategy and update the drawing systematically. This addition was observed 9 times (in 6 different sessions by 4 different tutors).

Tracing of program variations. Using an external representation was sometimes triggered by a difficulty identified by the tutor, through a misunderstanding or in response to a question from a student. In such cases, after tracing the program in question, another adaptation we observed was that tutors would ask students to evaluate expressions

based on slight variations of the code and check whether they understood it properly with the help of an external representation. This addition was observed 8 times (in 6 different sessions by 4 different tutors, one did it 4 different times).

Coloring updates. Tutors would sometimes use different colors to note updates in the traces to make them more visible. This addition was observed 6 times (in 6 differents sessions by 5 different tutors).

Effects

Representation-based explanation. An interesting positive effect observed nearly every time an external representation was used, is that this representation served as a basis for further explanation. Tutors would often gesture towards the memory representation, replaying with their hand the evolution that brought the memory in its actual state. This effect was observed 47 times (in 19 different sessions by all tutors. One used it 24 times over 4 sessions and the others between 1 and 7 times). An example is given in Figure 8.3a.

Collaborative tracing. Instead of just verbalising aloud their thought processes while tracing, tutors often involved their students in the evaluation of the expressions and updating the memory table. Tutors would often ask the class questions such as: *"Do I enter this while loop?"* (T5), *"What is the new value of this variable? What do I write here?"* (T4) while pointing in the table towards a value to update. This effect was observed 29 times (in 14 different sessions by all tutors. Tutors used it between 1 and 9 times).

Encouraging discussions. In multiple instances, tracing code stimulated relevant questions from the students, explanations from the tutors, etc. Students would ask about small variations and exchanges between students and the tutor would occur naturally. This effect was observed 9 times (in 8 different sessions by 5 different tutors).

Code annotation. Tutors would sometimes write down intermediate values directly in the code. This often happened for compound expressions that did not appear directly in the table. This was sometimes used in conjunction with a memory table, but sometimes not. This effect was observed 6 times (in 5 different sessions by 5 different tutors).

Program explanation. While tracing, tutors would sometimes give more natural language explanations about the code. As they built the memory table on the board, copying variable names and initial values or

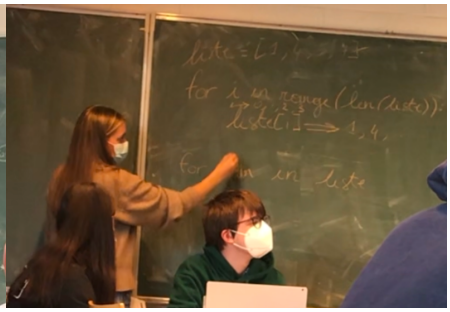
Figure 8.3: Some examples of both strategies used by tutors. (a) and (c) are expected uses while (b) and (d) are observed adaptations. Permissions to publish these pictures have been granted by the tutors and students.

(a) Expected use of the Explicit Tracing with Representation-based explanation effect.

(b) Oversimplified memory representation (not all variables are sketched).

(c) Expected use of the Subgoal Learning through (colored) code annotations.

(d) Unlabeled subgoals during an explanation of alternative solutions for list traversals.



when they executed specific lines of a program, tutors would naturally explain the role and reasons for those variable names or the executed statements. Tutors seemed naturally driven to give more comments about the names and constructs used in traced code. This effect was observed 6 times (in 4 different sessions by 3 different tutors. One used it 4 times over two sessions).

Misuses

Oversimplified memory representation. Often, tutors would not use the proposed memory representations and would simplify a lot their drawings. Sometimes, this translated in incomplete drawings, oversimplified notations (e.g., object instances without attribute names), or very partial sketches. Those less detailed drawings sometimes evolved in more refined traces as with progressive tracing. This misuse has been observed 32 times (in 14 different sessions by all tutors. One used it 12 times over 4 sessions, it was also the tutor who used explicit tracing the most overall). An example is given in Figure 8.3b.

Tracing aloud. Tutors would sometimes do the tracing out loud, without the code available to the students or at least without them following. The tutor executes the code in their head and just give the different values aloud as if it were evident. This often triggered a progressive tracing effect. This misuse was observed 6 times (in 6 different sessions by 5 different tutors).

8.2.3.2 Subgoal learning

Additions No addition was observed.

Effects

Shared vocabulary. A positive effect of the SL strategy is that since the explanations of the solution steps use a specific vocabulary, tutors would use themselves but also encourage students to use a more precise vocabulary when talking about their code, so that they could associate it more easily with the corresponding SLWE. A tutor, when distinguishing arguments from parameters for the function concept would say *"It is important to use the proper vocabulary"* (T5). This effect was difficult to count exactly but was observed in a lot of recordings.

Collaborative code building. On a few occasions, a tutor used the strategy to solve a new problem with the students based on the labels (SL4). But they did it cooperatively, with a high degree of interaction and students helping to build a correct solution on the board. The tutors sometimes built correct code only by themselves too. This notable effect was observed 6 times (in 5 different sessions by 3 different tutors).

Explanation of alternative solutions. Since some of the subgoal labels we used correspond to steps only to follow as the result of an alternative, when discussing those with the students, tutors had to explain

them and give explanations on the different alternatives. Using the strategy thus favored discussion and understanding on those specific points. *"If we use a* `for` *loop with a range, the* `i` *variable will successively have the value from 0 to the end of the list.* `l[i]` *will have exactly the same value as the element of the list and this can also be achieved by iterating with the form* `for e in l`*. [...] Who can tell me when we should prefer one or the other form?"* (T5). This effect was observed 6 times (in 4 different sessions by 4 different tutors). An example of this effect is shown in Figure 8.3d.

Misuses

Unlabeled subgoals. The labels provide a shared vocabulary. But in some observations, tutors would not use the labels, or only just a few. In those cases, tutors would discuss about subgoals linked to a concept but without mentioning the corresponding name of the label or without referring to the generic problem solving procedure. Some examples were tutors discussing list traversal with the students without mentioning the provided labels in the document, or a tutor discussing the importance of naming for classes and attributes without referring to the corresponding SLWE. This misuse was observed 12 times (in 8 different sessions by 4 different tutors). An example is shown in Figure 8.3d.

Tutor-defined labels. On a few occasions, we also observed tutors using other labels than those provided in the SL document. It was because they did their own attempt at labeling the important solving steps for a given concept or because they misunderstood the strategy as a "divide and conquer" strategy and would develop subgoals for solving a problem, label them, and then annotate the solution code with those different subgoals. In those cases, subgoals and labels would not be complete or generic. This misuse was observed two times (in 2 different sessions by 2 different tutors).

Oral annotation. One tutor tried to apply the strategy by reading and mentioning the labels and subgoals while students were writing their solution on the board. This caused confusion since students in the classroom were trying to read the code first. Explicitly annotating the code as exemplified in the document was more effective.

8.2.3.3 Combined strategies

Subgoal learning with external representation. Observed on 4 different occasions during 3 different sessions given by 3 different tutors, tutors

would reuse the labels of a specific concept and illustrate the different steps and cases covered by the labels with an external representation. On one of those occurrences, for writing the code solution for inserting a node in a linked list, the adaptation was combined with the "Inverse tracing" variation of the explicit tracing strategy and the code of the solution was built based on the sketches.

8.2.4 Threats to Validity

Again, as in Section 6.3.5, the small number of 7 participants and the fact that they volunteered for this study poses obvious threats to generalizations beyond the context of our study. However, we rigorously based our analysis on 24 observations. We reported for each trigger and adaptation the number of times they had been observed so that the reader can consider their prevalence. Our main goal was to characterise tutors' usage of the proposed strategies in this second iteration in order to refine the design of our next design intervention.

The high number of adaptations and misuses (nearly half of the strategy uses for explicit tracing) show that it is not trivial for tutors nor necessarily advisable for tutors to apply instructional strategies as is. A lack of rigor in the use of a standardised memory representation was at the root of many misuses. It is difficult to assess the origin of these partial implementations of the strategies. Tutors mentioned in interviews that they were doing that deliberately, mostly for time reasons and for *"fast forwarding"* to the student's actual mistake. It could also be that tutors could benefit from more pedagogical training and follow up to remedy their misuses.

It is important to note that some prompts were already present in the course material for making drawings and memory representations. These assignments however were not explicitly linked with the systematic line-by-line explicit tracing strategy. This might have biased the number of observed uses of this strategy by raising it. However, Table 8.2 shows that only roughly one in five of the explicit tracing uses were triggered by exercise statements. Also, the fact that exercises statements can be used for triggering strategy usage will lead to a more systematic use of prompts for the subgoal learning strategy integration in the next chapter.

8.3 Conclusion

In this study, we analysed how seven tutors adopted two explicit instructional strategies in a CS1 course. The two chosen strategies were selected because they are research-based, recognised as effective instructional strategies and designed to reduce the cognitive load of the learners. We based

ourselves on the education change literature to analyse the fidelity of implementation and the adaptations that tutors made to the strategies. Based on that, we derived fidelity criteria for both strategies based on the corresponding literature. We used a mixed methodology to treat three research questions by coding tutors' session recordings: we did a qualitative analysis of the observations to understand what triggers strategy usage by the tutors; we quantified based on our criteria the fidelity of implementation of both strategies; we further qualitatively analysed the different adaptations made by the tutors to the strategies.

The first result we can draw from the analysis of our observations is that Explicit Tracing was more used than Subgoal Learning. We identified three main categories that could trigger either strategy. Events that made tutors use a strategy originated either from: the **students** (their questions, their mistakes and their misunderstandings); the **tutors** (their preparation, the importance of concepts and their modeling of solutions and alternatives); the **exercises** (when prompting to trace). Both strategies were adopted by tutors with adaptations, simplifications and even by combining them. The fidelity of implementation of the explicit tracing strategy was higher than for the subgoal learning strategy. Tutors used overall a lot of external representations with a variety of adaptations. In their main use, they progressively transitioned from an oral explanation to a detailed line-by-line tracing depending on students' needs and understanding. The subgoal learning strategy took more time to apply and was often applied partially. A significant number of adaptations for both strategies have been documented. These adaptation were categorised as **additions**, **effects** and **misuses**. We hypothesise most of these additions and side-effects marginally favor the expected outcomes of the strategy. The main obstacle to the use of both strategies seems to have been a lack of time since we can see that a full use of the subgoal learning strategy could take up to half an hour. While many adaptations were made, some seemed to be an incomplete use of the strategies or a misunderstanding by the tutor of the strategy. Based on our analysis, we proposed some pieces of advice to teachers responsible for introductory programming courses.

Tutors' feedback again pointed to the time constraints of introducing students to the SLWEs. It is not uncommon for TAs to mention time constraints as a threat to best practice [Rie+21]. Integrating them in the course material and observing the impact on strategy use in classroom would be an interesting lead to explore. We hypothesize that in such a setup, subgoal learning would be used more by the tutors. The next chapter will present the integration of the subgoal learning strategy throughout the course, taking into account the advice to practitioners proposed in this chapter. The Koli publication this chapter is based on already proposed in

it's future work section to use trained observers that could help in such an experimental setup with an observation grid to fill out immediately during class observation.

Third Iteration: Integration

9

This chapter is largely based on the paper:
O. Goletti, K. Mens, and F. Hermans. "An Observational Study of Undergraduate Teaching Assistants' Use of Subgoal Learning Integrated in an Introductory Programming Course". In: *Proceedings of the 2024 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E '24)*. Pasadena, CA, USA: ACM Press, Oct. 2024. DOI: 10.1145/3689493.3689986.

Extending the research presented in Chapters 6 and 8 on the integration of explicit strategies and following design research principles in education [Bak18], this chapter presents the integration of subgoal learning (SL) throughout the studied course. We followed the advice to practitioners proposed in the previous chapter. Since prompts to the strategy were a big trigger (cf Table 8.2) for strategy use and since tutors mentioned lacking time to properly introduce subgoals to students during their lab sessions, in this study, we integrated SL throughout the course by integrating these labels in the different resources of the course: course's slides, exercises' statement, UTAs training and material. Our goal is to bring research to practice [Sen21] and to observe and analyse how 21 UTAs make actual use of subgoal learning as explicit instructional strategy in their teaching practice. We extract empirical evidence for instructional design advice on the integration of subgoal learning throughout a CS1 course. This chapter answers the following research questions:

RQ9.1 What are the effects of subgoal learning integration on UTAs' uses of the strategy?

RQ9.2 How do UTAs use the subgoal learning strategy according to our fidelity of implementation criteria?

RQ9.3 Can we classify UTAs' usage of the strategy in categories according to their observed usage and self-reported opinion on their usage of subgoal learning?

We will also discuss briefly what is the level of students' awareness of the strategy.

9.1 Study Design and Methodology

This section describes the methodology followed to integrate subgoal learning in the course and to collect and analyse data during the study. Figure 9.1 presents a comprehensive timeline of the different moments in the semester when training and data collection were done.

We used a mixed-method approach in our study. We used qualitative analysis throughout both the coding of observations and surveys. To answer **RQ9.3** we wanted to be able to cross the results of the observations of the 21 UTAs made by our 6 observers, the UTAs' self-reported perception on their appropriation of the strategy and the students' perspective.

In Figure 9.1, for each observed week, an id of the type "wXmY" is used. It stands for "week number X of the semester and mission number Y of the course"[1]. Apart from the training week (w3m2), three other weeks were observed, each of which with its associated course topics and concepts:

- **w5m4** on list traversals and functions;
- **w7m6** on file manipulation and exceptions;
- **w12-13m11** on the linked list data structure.

9.1.1 Subgoal Learning Integration

In our current study, we identified key resources of the course that could be impacted by the integration of Subgoal Learning (SL) and designed a fading approach for the integration of subgoals and labels in the course, in accordance with CLT inspired instructional principles. The present study makes use of our own subgoals and labels developed and adapted to the concepts and language (Python) of our own CS1 course (cf. Chapter 7). Table 9.1 presents for each course moment as presented in Figure 4.2, what staff is involved, what resources mobilised and how SL was integrated.

The subgoals used in our study are specific to the concepts seen in the course and adapted to Python. They were written following a task analysis procedure as describe in Chapter 7. The concepts covered are: assignment, conditionals, while loop, for loop and sequence traversal, functions, file read, file write, dictionary update, adding a node to a linked list and removing a node from a linked list.

[1] Typically, Y = X - 1, except for the last mission wich was spread over two semester weeks.

Figure 9.1: Timeline of the semester with observers' training weeks in green and actual observations in purple

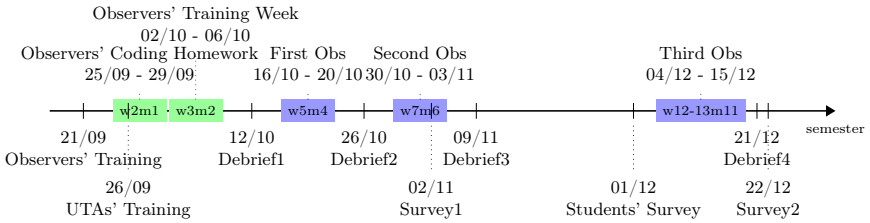


Table 9.1: Resources used in the course per moment and how they were adapted to integrate subgoal learning. (T = Teachers; TA = Teaching Assistants; UTA = Undergraduate Teaching Assistants; S = Students-; SL = Subgoal Learning; SBS = Step-By-Step SLWEs; SLFS = Subgoal Labeled Final Solution; RQ = SL integration done by tutors we are researching in this chapter).

moment	staff	audience	resource	SL
Tutor briefing i	T + TA	UTA		oral
Lecture week i	T	S	introduction slides	SBS
Preparation phase	-	S	exercises syllabus	prompts
Tutored start-up session	UTA	S	exercises syllabus	RQ
Realisation phase	-	S	exercises syllabus	-
Realisation evaluation	UTA		correction guide	SLFS
Tutored sum-up session	UTA	S	correction guide	RQ
Lecture week $i+1$	T	S	restructuring slides	SLFS

We distinguish two different types of subgoal labeling utilisation. (1) *Step-By-Step Subgoal labeled worked examples* (SBS) are worked examples that use each label one by one to illustrate at each step how the generic corresponding subgoal for that concept is used in a concrete example. The integration of such SBSs is mainly done by teachers during their lectures, but a UTA using the labels and solving procedure to provide students with an exercise solution could also use SBSs. In order to integrate SBSs in the

course, we had to modify teachers' slide decks and this took some work back and forth with teachers. (2) *Subgoal Labeled Final Solutions* (SLFS) would be used after the labels had already been introduced once. The aim is to show final solution or solved examples without necessarily going through the whole solving process but by annotating what code pertains to what label. Typically, those would be shown in the teachers restructuring slides to remind students of the labels or the result of an UTA annotating code on the blackboard during a lab session.

The integration of SL in the course consisted in the modification of the slides with SBSs and SLFSs, as shown in Figure 9.2. This allowed for the introduction and repetition of the labels for the students in a passive way. Since it was also new for the teachers, some details on the extended subgoals, not just the labels, were also added in their presenter notes. Since we knew that prompts in students' exercises statements were also triggers for strategy use, we also added the labels in the first exercises making use of the targeted concepts. The labels are linked each time to a complete SLFS listing, called the catalogue, where the subgoals, labels and annotated examples (SLWEs) are gathered per concept. This full catalogue is available to the students as an appendix of the exercises syllabus[2]. Prompts are a list of the labels or simplified subgoals with a link to this catalogue.

The last integration step relates to the UTAs. Next to the training of UTAs, we add reminders in the correction guide for UTAs that they will use for the "Realisation evaluation" [3]. This helps them to know where and when new concepts are used and which labels to use. This information is also discussed during the weekly "Tutor briefing" meetings with the UTAs.

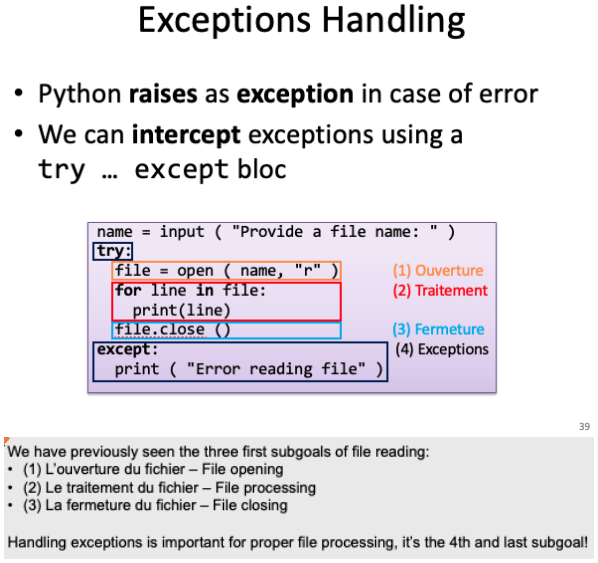
9.1.2 UTAs' Training and Follow up

In our new course design, the subgoal learning integration is systemic throughout the whole course, its material as well as its teaching methodology. All UTAs followed a mandatory training before the beginning of the course, consisting of theoretical background on the SL strategy, a description and demonstration of when and how to use the strategy, recording extracts from previous observations of good use and misuse and previous results, like the main triggers for strategy use (cf. Section 8.2.1). Follow up sessions for the UTAs occurred during the weekly "Tutor briefing" meetings. Questions were asked weekly on UTAs' strategy uses, feedback and rich discussions occurred between the UTAs and with the staff. Subgoals and labels corresponding to the weekly concepts were discussed during those meetings and

[2] https://syllabus-interactif.info.ucl.ac.be/syllabus/info1-exercises/EXTRA

[3] I.e., the evaluation of and feedback by UTAs on a weekly assignment made by the students on the course topic of that week (cf. Chapter 4).

Figure 9.2: Modified slide with the code annotated with (French) subgoals and the presenter notes for the teacher. The presenter notes have been translated from French for readability.



strategy use was encouraged.

9.1.3 Fidelity of Implementation and Deductive Codes

The criteria used to measure fidelity of implementation were adapted from Chapter 8 to take into account the integration of the presentation of the different subgoals in the course. Subgoals and labels are now introduced directly during the lectures by the teachers, so that tutors don't have to introduce them during their one hour lab sessions. The criteria, derived from literature, adapted and refined for this study are:

SL1 The strategy or the catalogue is mentioned;

SL2 The tutor solves a similar problem in a worked example;

SL3a The tutor reminds the students about one of the subgoals of a concept;

SL3al The tutor reminds the students about one of the subgoals of a concept with mention of the label;

SL3b The tutor reminds the students about *all* the subgoals of a concept;

SL3bl The tutor reminds the students about *all* the subgoals of a concept with mention of the labels;

SL4 The solving procedure of a concept is used to solve a practice exercise with the students.

During the coding, we noticed some observers made a difference in coding when UTAs mentioned one or all the subgoals of a concept and with or without the provided labels. Since this was an interesting distinction that brought nuance to strategy usage, it led us to separate SL3 from Section 8.1.3 across these two dimensions. SL3a and SL3b are with one or multiple subgoals but without mention of the labels, whereas in SL3al and SL3bl the labels are explicitly mentioned. So in the end we split SL3 into four sub-criteria that are mutually exclusive. This mention of the subgoals can either be done orally or by code annotation.

Based on this distinction, we will also introduce the notion of "stronger" use of the strategy when either two or more criteria are being used at the same time, or when SL3 is used with the labels being mentioned explicitly (ie. SL3al or SL3bl).

9.1.4 Observations

Since the intervention of this study involved a full integration of an instructional strategy at the scale of a whole CS1 course (∼600 students), it involved 21 UTAs and 22 different lab sessions. The 11 missions of the course (2 lab sessions per week) are organised following five different series, dependent on the students' program, each with their own schedule. Two series for a total of 358 engineering students, two series for a total of 169 computer science students, and one series for 69 mathematicians and students with other backgrounds. Together, the series amounted to 27 different classrooms of ∼24 students, each with their own (U)TA or in total 594 sessions to be observe. Among these 27 classrooms, 25 are tutored by UTAs (the other two by doctoral students, TAs), but only 21 of them agreed to participate in the study (ie. being observed and recorded during the whole semester).

In order to observe enough different UTAs on different sessions, 6 master students were recruited as *observers*. Five of them were paid as part of a student's job and the sixth one participated in the scope of his master thesis.

The observers were trained in the SL strategy with the same material used for tutors but also on what to observe and how to report it. They were tasked with the recording of the sessions and (pre)coding of those videos. Since the observers are CS master students and not qualitative researchers, they needed a training for their observation and coding tasks. Also, they were asked to do deductive coding based on the fidelity of implementation criteria of Section 9.1.3. We assigned them to code three recording extracts from past observations. Their coding was then discussed

and corrected. Moreover, a first round of observation in classrooms was dedicated as a training week, so they could test their recording setup (their smartphone), and the time it took them to code. Most of them took notes during the lab sessions, then later watched again the recording to refine their coding. Both the recording and coding of each session was then uploaded to a shared repository according to a simple naming convention `obs/YYYYMMDD-HHHH-room`. The first author was able to download, watch and control the observation process.

Each observed week, observers were randomly assigned to three to four sessions. For example, for w7m6, one observer was assigned two "Tutored start-up sessions" (noted session w7m6-1): those of tutors T15 and T18 and two "Tutored sum-up sessions" (noted session w7m6-2): those of T14 and T13.

Each observer did between 8 and 11 observations spread over the three selected weeks as seen in Figure 9.1. For each session they observed, they had to record the session and code the observed strategy uses by the UTA. A line in the coding file consists of a timestamp, some notes on the context, an eventual quote and the code itself. For each strategy use, they were asked to also note the trigger. Their coding was then reviewed by the first author who coded entirely three sessions of each observer. The SL3 codes had to be corrected and some strategy uses that were not generic enough had to be removed, some repetitions of codes for a same strategy use also had to be removed. Nevertheless, globally the observers coded accurately and thanks to their notes and quote fields, all corrections could be applied to the entire coding.

9.1.5 Surveys

UTAs' Surveys. To answer **RQ9.3**, we crossed the observations data with self-reported data from UTAs through surveys. At two points in the semester, UTAs were asked to fill a survey on their usage of the strategy. They were asked Likert-scaled questions on familiarity, frequency of use and opinion on the pedagogical value of the strategy. They were also provided with more open questions on their strategy usage. For the first survey, we asked for their agreement level on the following items:

S1Q1 I feel confident about giving my lab sessions;

S1Q2 I can see how to use the subgoals in practice in my lab sessions;

S1Q3 I had time to prepare my sessions this week;

S1Q4 I was influenced by the subgoals to prepare my sessions this week;

S1Q5 I used the subgoals during my sessions this week;

S1Q6 I thought about using the subgoals during my sessions this week but didn't actually do it.

For the second survey, the same items as the five last ones of the first survey were proposed but with a more global timeline (i.e. "my sessions this week" was replaced by "my sessions"). Moreover, seven new opinion items were added in the form of "I think the subgoals are useful":

S2Q7 for all students;

S2Q8 for stronger students;

S2Q9 for weaker students;

S2Q10 for all concepts;

S2Q11 for some concepts;

S2Q12 more for introducing concepts;

S2Q13 more for restructuring concepts.

Students' Survey. One survey was also passed to the students of the course. It mainly consisted of Likert-scaled questions on their awareness, familiarity and perception of the subgoal learning strategy and its integration in the course.

9.2 Results

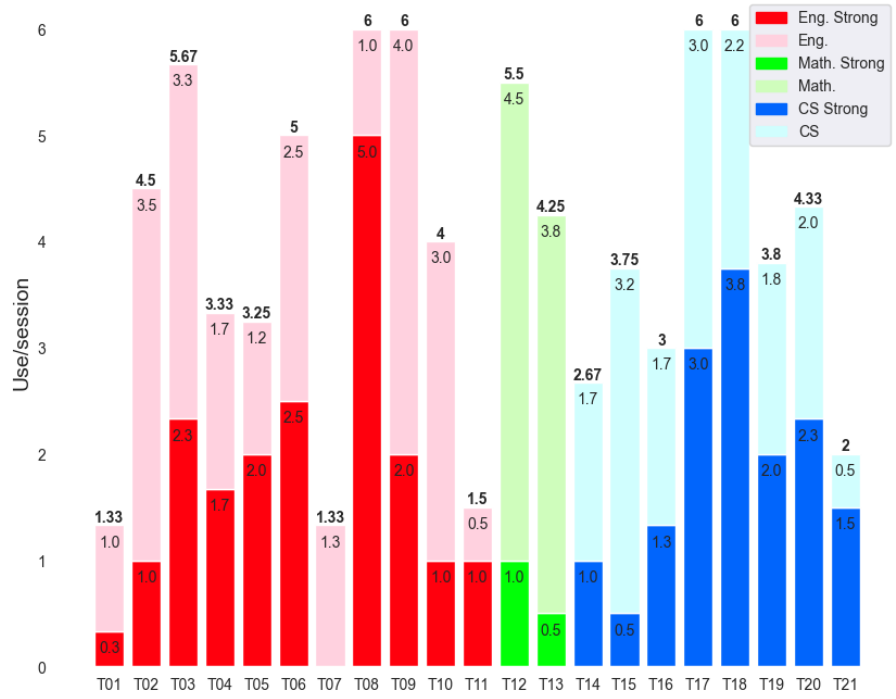
9.2.1 Observations

In the end, 74 observations were made, 12 of which during w2m1 for training purposes. Each lab session is supposed to last one hour but the UTAs have sometimes room for more, so we ended up coding more than 62 hours of video.

Strategy Uses. Figure 9.3 presents the normalised number of uses and "stronger" uses per session of the strategy for each UTA[4]. With a mean of 3.92 observed uses of the strategy per session (see Table. 9.5), we can answer **RQ9.1**: the integration of subgoal learning throughout the course pushed UTAs to effectively use the strategy. We can also see that the vast majority (16 out of the 21 studied UTAs) use on average 3 times or more

[4]UTAs are referred to as TXX (T from tutor and XX is an id between 01 and 21) throughout the Results Section.

Figure 9.3: Normalised number of use of the SL strategy per session for each UTA. The darker bottom part of each bar corresponds to "stronger" uses. In red: the engineering program, in green: the math program, in blue: the CS program.



the strategy per session. The rest of this section will answer **RQ9.2** on the observed fidelity criteria.

From Table 9.2 we can see that the most used criterion is SL3 (214 times in total), i.e. mentioning one or more subgoals with or without especially mentioning the labels. This occurs naturally in a UTAs' mentoring when reexplaining a concept. Only generic subgoals explanations were coded as such, not explanations linked to the specific context of an exercise.

From Table 9.2 we also see that SL3 was mostly used alone. Drilling down in detail in Table 9.3[5], it was flagged as criterion SL3a (53×), SL3al (38×), SL3bl (26×) and finally SL3b (21×) or in total 138 times or more than half of the total 243 strategy uses. The strategy was mostly used to explain a single subgoal at a time (53+38=91 with only one subgoal vs. 26+21=47 with multiple subgoals) and the labels are explicitly mentioned 38+26=64 times or nearly for half of the SL3 single uses.

[5]For conciseness and readability these details were dropped from Table 9.2 in the original publication and added as Table 9.3 in the dissertation.

Table 9.2: Number of occurrences of each observed combination of criteria (with SL3 sub-criteria summed for readability).

criteria	SL1	SL2	SL3	SL4	n_occ
			×		138
		×	×		44
		×			20
			×	×	13
	×		×		9
	×				8
	×		×	×	8
		×		×	1
		×	×	×	1
	×	×	×		1
total	26	67	214	23	243
%	11	28	88	9	100

Illustrated further by Table 9.2, when SL3 is used with other criteria, it is mostly (44×) with SL2 for detailing the steps in a worked example. This corresponds to an UTA demonstrating how to use a specific concept while mentioning the subgoals they want to highlight. This is done either orally or by annotating code on the blackboard. An example of SL3bl with blackboard annotations is provided in Figure 9.4a. For oral SL3 strategy usage, the triggers were mostly the exercise prompts or the correction of an exercise. SL3bl was observed 17× with SL4 (among which 8× also with SL1), those are the "strongest" uses of the strategy we can hope for. Again, adaptations of the strategy usage were also observed, for example six UTAs would make the students recite the subgoals for a concept (like writing a function or opening a file). One UTA also proposed a skeleton of a solution code based on the subgoals for helping the students to visualise this generic template (see Figure 9.4c).

The second most used criteria is SL2, i.e. providing students with a worked example. This was mainly used for demonstrating the use of a concept, at the start of a session or to illustrate an alternative solution. SL2 was used 20× alone as a criterion, and 44× with SL3 as mentioned above. An example of SL2 and SL3 is provided in Figure 9.4d.

SL4 was only observed together with other criteria, and mainly with SL3bl which makes total sense since it represents a reuse of the subgoals to solve another similar problem with the tutors. SL1, mentioning the catalogue, was observed 26× in total and only 8× alone. This low use of the catalogue might be an indication of a misunderstanding or a lack of

Table 9.3: Number of occurrences of each observed combination of criteria.

criteria	SL1	SL2	SL3a	SL3al	SL3b	SL3bl	SL4	
			×					53
				×				38
						×		26
					×			21
		×						20
		×	×					17
		×			×			10
		×				×		10
						×	×	9
	×							8
	×					×	×	8
		×		×				7
	×					×		7
					×		×	2
			×				×	1
		×					×	1
				×			×	1
		×	×				×	1
	×				×			1
	×			×				1
	×	×		×				1
total	26	67	72	48	34	60	23	243
%	11	28	30	20	14	25	9	100

training and/or preparation of the UTAs. An example of SL4 used with SL3bl is an UTA annotating or mentioning the subgoals labels and using them to solve a new example, what we called earlier an SBS for step-by-step use of the strategy. An example is provided in Figure 9.4b.

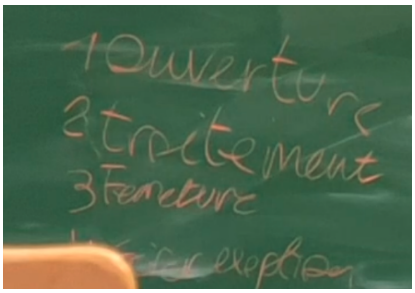
Apart from these criteria uses, we can see in Table 9.5 that most strategy uses were accounted for in w7m6. That week concept was file manipulation. In the upper part of the table, both w7m6-1 and w12-13m11-1 have the most strategy uses per session. We can see that the start-up sessions gathered significantly more strategy uses than sum-up sessions, meaning that UTAs use the strategy more often when introducing concepts.

There seemed to be little or no impact of the student audience on strategy usage (i.e., either engineering students or computer science students[6]).

[6]Since the Math. audience was only observed six times it is hard to draw conclusions

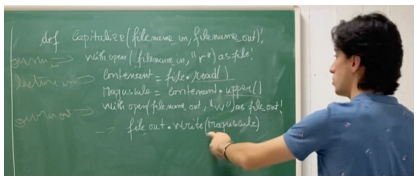
Figure 9.4: Some strategy use examples from the recordings.

(a) SL3bl example.



(b) SL3bl (oral) with SL4 example with code skeleton.

(c) SL3bl annotations during students' code correction.



(d) SL3bl with SL2 example.

Table 9.4: Number of observed triggers by source.

source	trigger	code	n
UTA	before an exercise	t1	40
	as a correction check-list	t2	39
	before a session	t3	21
	to show a better solution	t4	13
	as an exercise for students	t5	8
	sub-total		**121**
Exercise	prompts	e1	63
	sub-total		**63**
Student	question from a student	s1	29
	to help student solve exercise	s2	18
	misunderstanding	s3	7
	bug in a student's code	s4	5
	sub-total		**59**
Total			**243**

from that.

Table 9.5: Number of times each fidelity of implementation criterion was observed per mission, session or program. * means all different values are considered, which makes the last line an aggregated total of all observed uses. The "SL3 (Σ)" column adds up the four SL3 sub-criteria. The last column gives a normalised strategy uses per session value.

mission	session	program	SL1	SL2	SL3a	SL3al	SL3b	SL3bl	SL3 (Σ)	SL4	uses	strong uses	n	uses/ session
w5m4	**1**	*	5	9	14	5	4	5	28	4	35	11	9	3.89
	2		2	16	10	5	6	7	28	3	34	15	11	3.09
	*		7	25	24	10	10	12	56	7	69	26	20	3.45
w7m6	**1**	*	4	10	16	13	6	24	49	6	63	31	10	6.30
	2		2	10	12	14	2	5	33	1	35	11	12	2.92
	*		6	20	28	27	8	29	92	7	98	42	22	4.45
w12-13m11	**1**	*	9	13	12	4	9	14	39	9	45	23	9	5.00
	2		4	9	8	7	7	5	19	0	31	11	11	2.82
	*		13	22	20	11	16	19	66	9	76	34	20	3.80
*	**1**	*	18	32	42	22	19	43	128	19	143	65	28	5.11
	2		8	35	30	26	15	17	88	4	100	37	34	2.94
*	*	**Eng.**	14	39	29	20	15	30	94	13	111	48	28	3.96
		CS	11	26	28	19	16	29	92	10	104	50	26	4.00
		Math.	1	2	15	9	3	1	28	0	28	4	6	4.67
*	*	*	26	67	72	48	34	60	214	23	243	102	62	3.92

Triggers. Table 9.4 summarises what triggered UTAs to use the strategy. First, we notice that UTAs remain the main trigger for strategy use and mainly for reintroducing a concept at the beginning of a session of an exercise, or as a check-list to restructure the concept after an answer has been given. Then, we can see that the exercise prompts we added accounted for one out of four strategy. When students are the triggers, it is mainly when they ask a question or when they need to be guided to solve an exercise, that UTAs will use the strategy. The effect of exercise prompts weight also in favor of SL integration.

9.2.2 UTAs' Surveys

Of the overall 25 UTAs of the course, 21 agreed to take part in the research meaning being observed, recorded and letting us use their data for this research by answering surveys. Among those, 12 answered the first survey and 11 the second one. The first survey was sent to the UTAs in the middle of the semester after all concepts had not yet been seen in the course (see Figure 9.1 for details). The second survey took place after the end of the course. UTAs' answers will help us answer **RQ9.3** on classification of UTAs' usage of the strategy.

Familiarity and Frequency

The results of the Likert-scaled questions on familiarity and frequency of use for the different concepts are shown in Figure 9.5a, 9.5b, 9.5c and 9.5d.

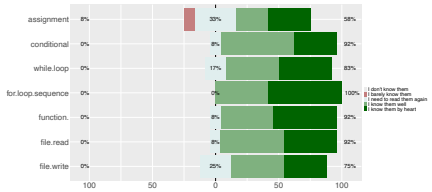
The data is only partial since not all UTAs answered the surveys. Overall, we can say that the familiarity of the UTAs with the concepts is reported as high. There's more variation in understanding of file operations and advanced topics. Between the two surveys, most UTAs maintained or improved their familiarity with the different concepts. T02, T05 and T20 have a lower aggregated score but due to the newest, more advanced concepts.

Regarding the frequency, there's a general trend towards increased frequency of use for most concepts among the UTAs who participated in both surveys. The best scoring concepts are loops, file operations and linked list operations.

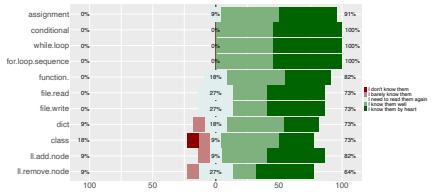
We found that UTAs' self-reported familiarity correlates positively with their self-reported frequency (Pearson's correlation $r = 0.72, p = 0.00$) as presented in Figure 9.6.

Figure 9.5: UTAs' aggregated answers from the two surveys to Likert-scaled questions on their familiarity, frequency and opinion on their use of subgoal learning.

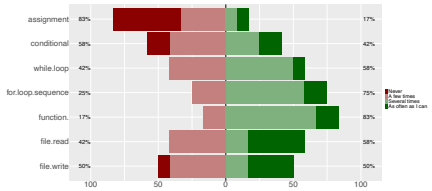
(a) Familiarity of use of concepts reported in first survey



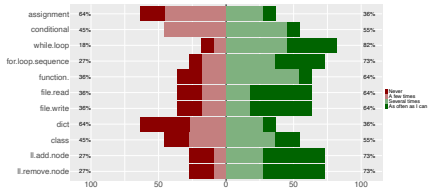
(b) Familiarity of use of concepts reported in second survey



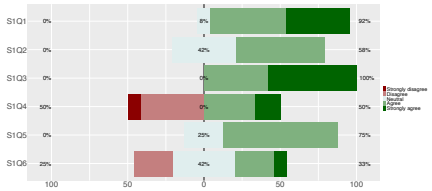
(c) Frequency of use of concepts reported in first survey



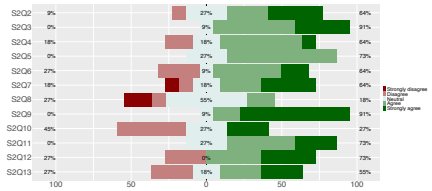
(d) Frequency of use of concepts reported in second survey



(e) Opinion on the strategy as reported in first survey



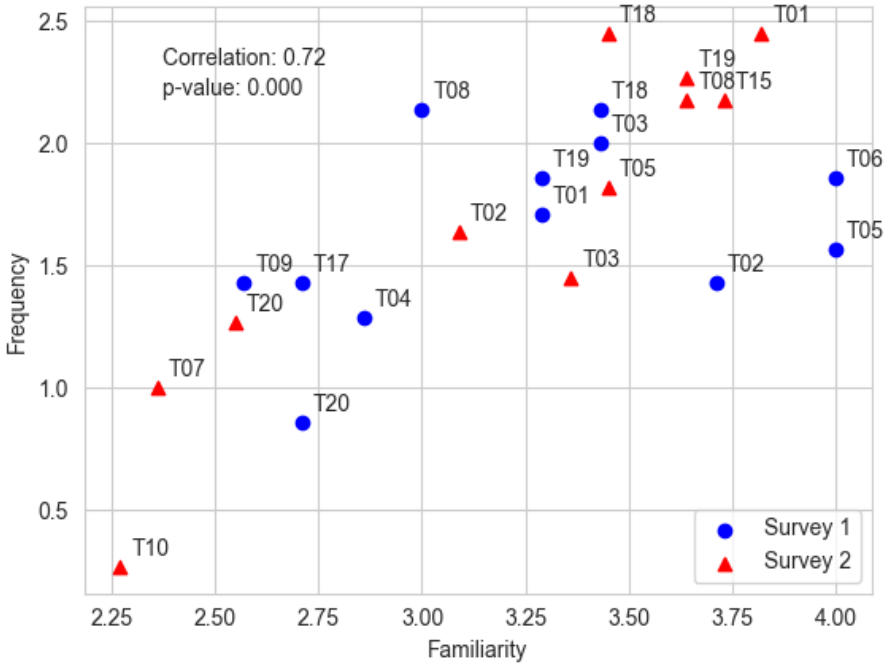
(f) Opinion on the strategy as reported in second survey



UTAs' Opinion of the Strategy

From the survey results presented in Figure 9.5e and 9.5f, we can see that Q2 on UTAs vision on how to use subgoals got overall better results in the second survey, which is good news meaning that with time, UTAs felt they got better at applying the strategy. Q3's result on whether they had enough time to prepare their sessions are a bit worse but overall UTAs complained more about time during the sessions than the time to prepare. Q4, on the influence of subgoals on UTAs' preparation is a bit better in the second survey. This is, like Q2, interpreted by the amount of practice and experience UTAs have gained with using the strategy by the end of the course. In their open answers, the UTAs who used the labels specified they were useful to structure the session, prepared them to know how to answer students' questions or to help them remember the content of the course. Q5 on whether they declare using the subgoals during their sessions is roughly the same. Three out of four declared using the subgoals during their classroom sessions

Figure 9.6: Correlation between UTAs' reported familiarity and frequency of use of the SL strategy.



and the rest is neutral (neither agrees nor disagrees with the statement). T06 says on this account: *"In my opinion, I used the optimum number of steps; using more would have been counterproductive"*. Q6 asked about whether they at least *thought about* using the labels even if in the end they didn't; here also the results got better with two thirds declaring agreeing with this item. This shows at the same time more reflection on the UTAs part, but also that they either forgot or decided not using it during their sessions. Some UTAs said explicitly that they sometimes chose not to use the strategy: *"I used it less for simpler or mastered concepts, in order to not confuse students"* (T08). UTAs mentioned other obstacles too. Four mentioned not enough time during sessions and three mentioned that they themselves did not learn with them: *"I haven't learned to code with subgoals, I sometimes forget the point and realise it when a student gets stuck on something that shouldn't be too complicated"* (T05). Finally, T03 also mentioned that a lot of exercises do not use all subgoals of a concept but only a subset.

Questions S2Q7–13 interrogate the perceived usefulness of the strategy. S2Q7–9 concern the student audience the strategy helps. UTAs think the strategy is mainly useful for weaker students and one in four do not agree it

helps stronger students. Answers to open questions refine this. While one in four totally agrees that the strategy if useful for all concepts, three out of four agree that it's at least useful for some concepts. Finally, UTAs think that SL is more suited to introduce concepts while half of them still think it can also be useful for restructuring concepts.

In the open questions, T03 suggested that *"the first exercises for each concept should nearly force students to use the subgoals or to add multiple choice questions on the subgoals"*, but that is something the teachers explicitly forbade. T18 also suggested adding subgoals for unit tests.

9.2.3 Crossing the Data Sources

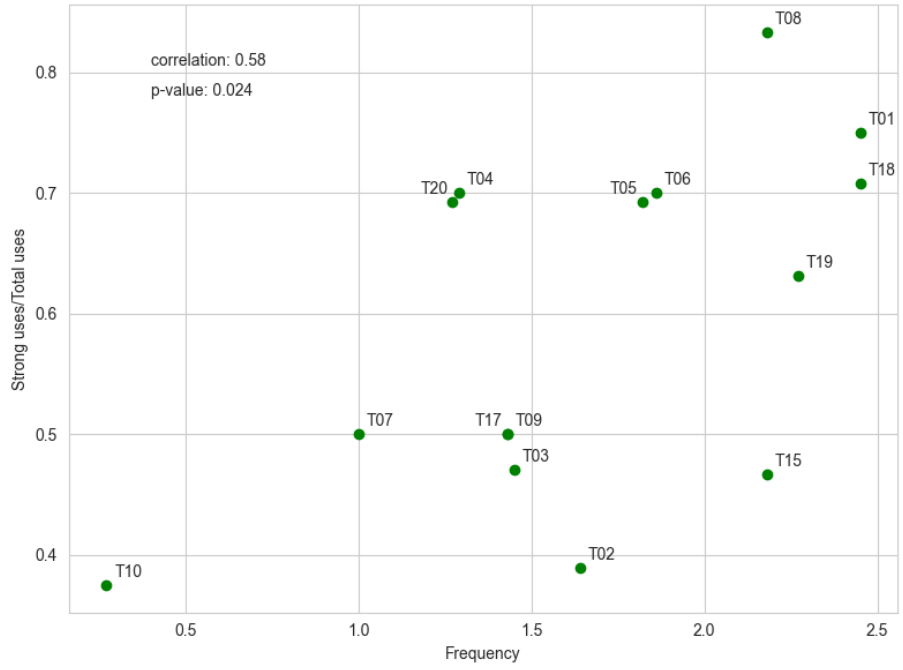
UTAs' opinions on when to use the strategy corresponded to their actual use of the strategy which happened more often during introduction sessions than sum-up sessions. T01 said *"When students have trouble getting started ..., I show everyone what to do in general at the beginning (the steps to follow)."* which matches both triggers t3 and s1 from Table 9.4. T18 said something similar: *"To help students if the student cannot start, or it they start writing code directly, to highlight the subgoals (sometimes with the help of the classroom) in colour"*. UTAs also reported using the strategy more for some concepts, especially the more structured and advanced ones. This is in line with the higher use of the strategy during w7m6-1 (introduction to file operations) and w12-13m11-1 (introduction to linked list operations) as seen in Table 9.5. They agreed that the concept of file operations was more structured in the sense that the structure was more visible and therefore more easily mapped to named subgoals. UTAs also agreed they used the strategy more to illustrate a concept when used for the first time, to introduce it before a session or before a specific exercise. This also aligns with the observed higher use for introducing concepts than during sum-up sessions (session 2 in Table 9.5).

We did find a significant moderately positive correlation (Pearson's correlation $r = 0.58, p = 0.024$) between UTAs self-reported frequency and the ratio of their observed number of "stronger" (i.e. at least two criteria or an explicit mention of the labels) uses to the total uses, see Figure 9.7. We don't know yet if that correlation is just accidental or could be strengthened by giving UTAs even more guidance.

9.2.4 Students' Awareness of the Strategy

An optional survey was sent to the students before the last course mission. It was answered by 48 students only, which is less than 10% of all students. The survey consisted of eight questions on three themes. Overall, regarding awareness and frequency of mentioning the catalogue and the subgoals,

Figure 9.7: Correlation between UTAs' reported frequency of strategy use and their observed "stronger" use ratio.



a majority of the respondents seemed unaware of this strategy. On their awareness of the subgoal strategy integration in the course material, surprisingly around 70% of the respondents declared not having explicitly heard of the strategy by the teaching staff nor of its integration in the course. When provided with a modified slide or an exercise with subgoals prompts, however, more than 70% acknowledged they had noticed those. On the frequency at which they heard about the catalogue, 60% declared not having heard of it at all neither during lectures nor the lab sessions. This falls to 30% for the subgoals during those same moments. The subgoals have been noticed weekly by one out of four respondents during lab sessions and nearly one in three for the lectures. 30% had regularly seen the catalogue mentioned in the exercises syllabus and a bit more for the subgoals.

Overall, regarding the usefulness of the catalogue, even though the strategy seemed not to be seen that frequently by the students, most of them did consider it moderately to very helpful. Around 65% declared it being not or only a little bit useful while it did seem useful for nearly 15% of the responding students. The subgoals during the lecture and in the exercises are considered very useful by nearly 30% of the students and by 30% in the lab sessions. If we add the students declaring the subgoals moderately useful,

we reach nearly 60% of the students for the lectures, labs and exercises.

Regarding the different concepts finally, the majority of the students declared knowing the five first concepts (from assignment to function) by heart. For the four next concepts asked (file manipulation, dictionary and class) one in three declared knowing them. Overall, it seems students think they know the first concepts treated by the subgoal labels well and the later ones a bit less.

9.2.5 Threats to validity

While we ensured that UTAs were comfortable with the observation process by having a training week and letting them opt out, at least one tutor did stop participating during the process. Meanwhile, at least three UTAs asked for feedback on their teaching practice during their lab session when they knew they had been recorded. This shows a real interest to become better tutors.

Another limitation, is that teachers have not been trained or observed in this study, since the focus of this study was on UTAs. While we had discussions with the teachers of the course, training was deemed unnecessary for them. We did not check how they discussed the labels since they had been integrated in their slides. It might be that teachers did not properly highlight the labels or did not discuss them at all. However, we could confirm teachers discussed them through indirect information like students mentioning hearing about the labels in the courses, and informal feedback from the teachers.

Since there were 6 observers, newly trained for coding the recordings segment, a threat is the inter-rater reliability of the actual coding. To mitigate this threat, the first author verified and harmonised the coding of the different observations. This process was eased by regular meetings with the observers to discuss what and how to code different observations made during the week.

9.3 Conclusion

Subgoal learning as an instructional strategy highlights generic solving steps when learning and using programming concepts. It is a promising teaching strategy for CS1 programming courses to enhance students' learning experience and outcomes. Through a detailed analysis of a large-scale CS1 course in which we fully integrated this strategy, and studied its fidelity of implementation among undergraduate teaching assistants (UTAs), our study revealed key insights.

We answered **RQ9.1** by acknowledging the effect of SL integration throughout the course, in particular the effect of exercise prompts on UTAs' strategy uses. Training and follow-up sessions with UTAs played a crucial role in the successful deployment of this strategy. As the course progressed, UTAs reported increased familiarity and frequency of use of the labels for the different concepts taught. This may indicate that the subgoal learning method helped streamline their instructional process and improved their teaching practice. We also found a moderately positive correlation between their self-reported frequency and the ratio of their observed number of stronger uses over their total strategy uses.

We use fidelity criteria to answer **RQ9.2** and introduced the notion of "stronger" use of the strategy. The mention of subgoals remained the most observed use while around 40% of the strategy uses have been classified as "stronger" uses which seems correlate with UTAs reported frequency of use.

Answering **RQ9.3** to classify UTAs use of the subgoal learning strategy, we observed it was used more by UTAs during sessions introducing new concepts than during sum-up sessions, and more for structured and advanced concepts than for simpler ones. This structured dimension of concepts that would better fit with the strategy is a lead for further research. The program of the students audience had little impact on UTAs use of the strategy.

Another working point is to make the strategy's use more visible to the students. Despite students' infrequent perception of the strategy's mention during lab sessions and lectures, a majority found the subgoals to be moderately to very helpful. This suggests that even limited exposure to subgoal learning can positively influence student understanding and retention of programming concepts.

Overall, the findings from this design research study provide interesting insights on how subgoal learning can and is used as a tool by UTAs for reducing cognitive load and promoting learning transfer in introductory programming courses.

Part III

Discussion and Future Directions

Overall Results 10

This chapter first summarises (10.1) the results obtained in the previous five chapters of Part II. Answers to the main research questions of this dissertation (cf. Section 1.1) are then proposed (10.2). Since we want the results of our research to be actionable, we reformulate these answers as advice to practitioners (10.3) and to UTAs (10.4). CS educators wanting to integrate instructional strategies in their course can follow this advice and get informed and inspired by our research results. We then discuss some limitations to our work. To conclude, I share some opinions and lessons learned (10.6) on the methods I used in this dissertation.

10.1 Summary of the Results

In the five previous chapters, we covered the analysis of a CS1 course based on two processes of learning transfer: knowledge encoding and knowledge accessibility in a situation of transfer. This analysis yielded three instructional intervention proposals: (P1) make the organisation of the course's learning objectives visible; (P2) highlight transfer conditions and opportunities when teaching knowledge; (P3) Propose explicitly recall strategies. Following a design research approach, we intervened on the design of the course through three iterations. We selected four evidence-based explicit strategies aligned with cognitive load theory and our learning transfer instructional recommendations: explicit tracing, subgoal learning, Parsons problems and explicit problem solving.

During our exploration in our first iteration on the design of the course, we introduced the four selected strategies to four tutors, we followed-up on their strategy usage through weekly focus groups and collected their perceptions on the explicit instructional strategies. Based on their reporting, we extracted four criteria that make them prefer a strategy: easy to understand, straightforward to apply, useful for students on the long term and supported by literature. Nevertheless a main limitation reported was the time it took before or during lab sessions to prepare and use the strategies.

In order not to overwhelm our UTAs and in order to better focus on less strategies, we used only explicit tracing and subgoal labelling in our second iteration. For this intervention on the design of the course, we

used subgoals tailored to the concepts and programming language of the course. We also selected a standardised external memory representation for the explicit tracing strategy. We handed out to the participating tutors support documents we prepared with information on how and when to use the strategies, with Python examples based on actual exercises from the course. We trained seven tutors by presenting, explaining, demonstrating and exercising both strategies at the beginning of the course. We recorded and coded observations of the tutors during their lab sessions. Using mixed-methods, we analysed the recordings using thematic analysis and fidelity of implementation in order to characterise tutor's usage of the strategies. The analysis yielded three categories of triggers: students, tutors and prompts in exercises. We quantified the fidelity of implementation of both strategies. We also documented adaptations of the strategies made by the tutors.

Based on the lessons learned from that characterisation study and the advice to practitioners proposed, we presented a third intervention: a full integration of the subgoal learning strategy throughout the course. For this intervention on the design of the course, all resources of the course have been impacted by the integration of the strategy. They have been modified in order to gain time for tutors so that they did not need to introduce the subgoals themselves and could focus on using them in their lab sessions. We analysed their strategy usage through observations during three weeks and two surveys using mixed-methods: thematic coding and fidelity of implementation. In that iteration, we acknowledged the effect of exercise prompts, training and follow-up sessions on tutors' strategy usage. We introduced the notion of stronger use of the strategy and observed that 2 out of 5 strategy use were "stronger" in that sense. We found a positive correlation between tutors' self-reported frequency of use and their normalised stronger SL uses. Finally, tutors used SL more often to introduce new concepts and specifically for more structured and advanced concepts.

Overall, these results indicate that it is possible to integrate explicit instructional strategies in an introductory programming course at university to improve tutors' teaching practice. These results show that training, personalised follow-up and exercise prompts where important factors to favor adoption of the strategy by tutors. In all three iterations, tutors used (or reported using) strategies more often in order to introduce a concept. Globally, tutors' self-reported frequency of uses can be used as a proxy for their actual frequency of uses.

10.2 Main Answers

Based on our results from the previous chapters summarised above, we can now answer the main RQs from Section 1.1. More specific and actionable

advice based on our research results are then presented in the next section.

10.2.1 (RQ1) Strategy Choice

How and what instructional programming strategies to select to integrate in a CS course?

Our answer to this question is rooted in the methodology we followed in Chapter 5 and in literature. First, the choice of a conceptual model on which to base our analysis was determinant in the intervention proposals that led to how we selected evidence-based instructional programming strategies. In our case, we based our analysis on the conceptual model of learning transfer. This allowed us to identify a need for explicit instructional programming strategies that would emphasise transfer opportunities and help recall previously learned material. Then, the actual selection of strategies was done according to cognitive load theory inspired instructional recommendations and our own learning transfer intervention guidelines. Proper alignment between selected strategies and the conceptual model is important (cf. 6.1). A partial literature review allowed us to identify four evidence-based strategies recently used in CS education. Such strategies help learners recognise similar problems where they will need to reuse previously learned knowledge. The proposed strategies also help learners fetch that learned knowledge in their long term memory. For example, in our research, explicit tracing helped as a debugging strategy to showcase how the state of a program evolves when executing it line-by-line. It helps students remember the semantics of the different constructs introduced in the course. Subgoal learning gives a name – and labels – to the important concepts and the intermediate solving steps to apply when using them in a program. How we chose to proceed with the selected strategies, in particular which one to iterate with, is answered in the next section.

10.2.2 (RQ2) Strategy Usage: Pros and Cons

What makes explicit programming instructional strategies easily usable and applicable by UTAs? What are the obstacles to use these strategies?

Chapter 6, even though it presented an exploratory study based on a small sample, allowed us to learn early on that tutors with little pedagogical background preferred strategies that are easy to understand and to apply. The evidence-based character of the strategies convinced and motivated them to apply it to help students learn, on the long term if possible. Strategies that are less meta or abstract were considered easier to understand. Strategies were considered easier to apply if they didn't need too much preparation, took less time or seemed already familiar to the tutors (like tracing which was close to debugging for example).

Thanks to the tutors' feedback collected across all our interventions, we identified as obstacles that they were overwhelmed by too many strategies, needed more support and training and lacked time during lab sessions. In Chapter 6, tutors reported it was not always clear how and when to apply strategies. In Chapter 9, tutors also mentioned that not having learned themselves with a specific strategy was an obstacle. The training and support we decided to put into place based on this feedback is detailed in the next section.

Globally, integrating strategies for tutors who are novice educators was a challenge for them. Some of their difficulties were already reported in literature on CS TAs like the classroom management challenges or the lack of time [Mir+19; Rie+21]. Based on our observations, the lack of experience with teaching and with applying specific instructional strategies due to a lack of dedicated training was a major brake to strategy use. While UTAs made an explicit choice to take the job, we also observed difficulties for some of them to learn the subgoals or to anticipate and prepare when to use the strategies during sessions.

10.2.3 (RQ3) Training and Support

How to train and support UTAs to integrate explicit instructional programming strategies in their teaching practice?

In order to train UTAs, we converged after our different iterations to propose a dedicated training session at the beginning of the course where we explained and demonstrated the strategies, their motivation and effect, when tutors should use them and how. We modeled the strategy for the tutors so they could observe how it should be used and also provided tutors with practice exercises. In order to support UTAs, we proposed a constant follow-up to discuss, remind, share and ask about their strategy usage. We adapted the strategies to tailor them to the specific content and concepts of our course in Chapters 7 and 8. Based on our observed list of triggers and adaptations, we integrated presentations and prompts to the strategy throughout the course material: in the slides, in exercises and as a catalogue of all the covered concepts. We observed that UTAs benefited from sharing their own practice during tutor meetings and that it inspired them to try out the proposed strategies. We also observed that proper training and integration of the strategies benefited UTAs and allowed them to better use the strategies and more often. Overall, our research results show that it is not trivial for UTAs to properly integrate instructional strategies in their tutoring and that carefully designed training and regular follow-up was needed.

Literature on peer teaching or peer tutoring – another term referring to

the usage of undergraduate teaching assistants [JH20] – also recommends doing peer observations as a training strategy [BM08; BM15] even for UTAs in CS [Kir06]. We found little literature on (U)TAs training programs dedicated to didactical knowledge or pedagogical content knowledge while the importance of PCK in education is well recognised [Abe08]. In their review on peer tutors and the learning they gain from tutoring, Roscoe et al. [RC07] also show the importance of proper tutor training to develop their capacity to explain in detail the solving steps of a problem or giving meaningful worked-out examples (*"knowledge-building"*) instead of just *"knowledge-telling"* interactions [RC07]. Our findings that there is a need and positive impact for tutor follow-up and training, seem coherent with the literature that we found.

10.2.4 (RQ4) Usage and Adaptions

How do tutors use and adapt explicit instructional programming strategies in their teaching practice?

In the studies on explicit tracing [XNK18] and subgoal learning [MMD19] on which we based our research, the teaching was given by the researchers themselves and we found little detail on how to train educators to use these strategies. This gap was filled by our research in which we designed handout documents and material to train UTAs and corresponding fidelity criteria to assess their fidelity of implementation of the strategies. We also cataloged and reported on triggers and adaptations for both these strategies. The categories and lists of triggers and adaptations are the results of our analyses of observations and recordings of tutors' actual usage of the strategies during their tutoring sessions.

Collecting this information was done through observations, exchanges during the tutor briefings and meetings, surveys and open questions. We now know that UTAs reporting higher frequency of use of a strategy are those who will report higher familiarity and who will demonstrate a stronger use of the strategy. We also found that given sufficient time, tutors would use the strategy more during the semester. We observed that subgoal learning integration in the course pushed tutors to integrate more subgoals in their practice. Chapter 8 also presented in Section 8.2.3 different categories of adaptations made by the tutors. We observed a lot of impact of the strategy on tutors' practice which we called *effects*. We observed that tutors, while integrating the strategies, proposed some *additions* in line with the strategies and that we thought could prove beneficial. We also observed some *misuses* of the strategy but in retrospect, most of the reported misuses are either part of a fading of the strategy usage or susceptible to be corrected through more support and training. That being said, the fact

remains that an incomplete usage of the subgoal learning strategy means that tutors will not leverage its full power. Catrambone [Cat98] already mentioned that incomplete steps knowledge will lessen the chances for the learner *"to identify what prior knowledge he or she possesses that might be useful"*. Further research should definitely pay attention to proper subgoal labels learning for UTAs. Globally, for both explicit tracing and subgoal learning, we seldomly observed fully accurate implementations. While partial usage is already a step in the right direction and while adaptations might have benefits, a conclusion of our work is that tutors have difficulties to implement the strategies as well as we would have hoped for. Partial tracing, not mentioning the subgoal catalogue, etc. remain areas of improvement for better fidelity of implementation.

10.3 Advice to Practitioners

In this section, we reformulate the results of our research in actionable advice to practitioners willing to integrate instructional strategies from research in their own teaching staff practice.

10.3.1 How to Choose Instructional Strategies?

Aligned with learning objectives. Practitioners should choose a conceptual model aligned with their learning objectives. Chapter 5 saw how a thorough analysis of a course material based on an educational conceptual model can help identify recommendations and amelioration opportunities. In our case, we wanted to enhance learning transfer between the different concepts seen along the course and our analysis of the course hinted towards three proposals that guided our strategy selection.

Evidence-based. The chosen strategies should be rooted in literature and proven effective. From Chapter 6, we learned that tutors liked and were motivated by the evidence-based character of the proposed strategies. In our second iteration presented in Chapter 8, we integrated some theoretical context and evidence from literature to make the case of our strategies stronger for our UTAs.

Simple to apply. Our advice to practitioners is to select simple strategies. Chapter 6 already highlighted that too abstract or complex strategies were harder for tutors to grasp and put into practice. However, we also showed that by integrating subgoal learning in the course, a strategy that was at first considered difficult to apply by tutors might still be

used when more effort was put in supporting and simplifying strategy adoption.

A couple at a time. Only introduce introduce strategies one at a time, before introducing them with another strategy. It is better that tutors have a few strategies that work for them, as opposed to having seen so many that they don't remember when to use them, or apply them wrongly because they start to confuse them with other strategies. The trend in the research presented in this dissertation was to go from more to less strategies. While one of the reasons was to focus the research to deepen our understanding of tutors' adoption of subgoal learning, another reason was that tutors had been a bit overwhelmed in the first iteration by the different possibilities offered.

10.3.2 How to Train UTAs?

Provide evidence. Linked to the evidence-based character of the strategies, the training should be showing to UTAs the reason behind the strategy efficacy. For example, in our research, we told them how it will support students learning, foster learning transfer, lower their mental load, scaffold the learning of students that needed it, potentially lower student's drop rates. UTAs needed to be convinced by those evidences in order to apply the strategies.

Provide training material. Tutors asked for more training and support in Chapter 6. We advise providing a good training supported by training documents. We included in our training material results of our research like hints on when to use the strategy based on our identified trigger for example.

Model strategy use. The tutors' training should include demonstrating and modeling of proper strategy use. This was requested by tutors in order not to be lost with only a paper or a training document and it gave tutors a better idea on how to apply it themselves. This could be done through small videos.

Allow for adaptations. We advise to give the liberty to the teaching staff to integrate a strategy in their practice as they see fit while also modeling, supporting and defending proper advised use. As we have seen in Chapter 8, tutors will adapt the strategy anyway and we have seen that most of these adaptations stay in line with the big ideas of the strategies and some might even be beneficial.

Provide practice exercises. Again, based on UTAs' feedback in Chapter 6 where they asked for more training and support, we decided to let

them practice the strategy beforehand. We advise to integrate practice into UTAs' training to give them the opportunity to try out the strategy, ensure proper understanding and boost confidence before applying it into the classroom.

Be patient. We found in Chapter 9 that giving tutors time to get used to the strategy was beneficial as the further in the course semester they were, the more they used it. Therefore, practitioners should give sufficient time to tutors to assimilate and try out the strategies.

10.3.3 How to Support Instructional Change?

Experiment. Strategies should be integrated progressively through small experiments if one can afford the time. An iterative integration process definitely allowed us to recognise tutors' needs and difficulties (like the time it took them to present SLWEs in Chapter 8) and to fine tune our interventions accordingly.

Integrate the strategy. When documenting the main triggers for strategy use in Chapter 8 and 9, strategy prompts integrated into the course material played a big part. Chapter 9 specifically showed that an integration throughout the course, with presentation of the subgoals in the slides and prompts in the exercises favored tutors' strategy usage. Therefore, our advice is to integrate hints to the strategy in the course material to support and trigger strategy use.

Adapt the strategy to the course. It is important to note that the strategy should be adapted to the course if needed. We applied this advice by creating our own SLWEs adapted to the Python programming concepts or our course in Chapter 7 or when we chose an appropriate external memory representation in Chapter 8.

Follow-up tutors. An opportunity should be provided to tutors for sharing, between them and with the teaching staff, their practice and perception on strategy usage. Our research showed that personalised follow-up of our UTAs allowed them to share how they used the strategies, to give us feedback and discuss difficulties. We are convinced that some tutors, hearing how others did, were led to try out more strategy usages and adaptations in their practice.

Collect feedback. We learned in Chapter 9 that UTAs' feedback correlates with how they used the strategy. Therefore, we advise collecting feedback on usage and frequency of use through surveys and open questions. This will allow some reflection on the tutors side, it will

serve as proxy to how tutors are adopting the new practice and might highlights difficulties and suggestions to adapt the strategy integration.

10.4 Advice to UTAs

In this section, based on our observations and mentoring of tutors during our research, we formulate some advice to tutors to apply strategies in their teaching practice.

Prepare. Tutors should come prepared to the classroom. The more tutors know about the content of a tutored session as well as the possible moments to use an instructional strategy, the better they will be able to anticipate how and when to apply it. We observed tutors who tried to improvise subgoal labeling on the fly or who traced code poorly, and we believe that with more preparation, they would have performed better.

Try. Tutors should give it a chance. We have seen tutors who needed more examples of how to use the strategy, or to listen to experience from other tutors, before daring using it by themselves. We also observed that, given time, tutors will use the strategy more.

Share. Related to the previous one, it is important for tutors to share good and bad experiences. This can help their peers to figure out other possible usage of the strategy. It also leads to community building which we consider very useful and relevant.

Adapt. Tutors should feel free to adapt strategies to their own teaching style. While adaptations have to remain reasonable and in line with the strategy motivations, we have observed promising adaptations in tutors' practice. It is also a good indication of ownership of the strategy.

Understand. UTAs should understand well the strategy motivations and principles. We observed this motivated them to apply it. It also will push them to apply and adapt it properly.

Discuss. When in doubt on how or when to best apply the strategy, UTAs should refer back to the provided documentation or training, or their peers' experience, or even a specialist or dedicated member of the teaching staff.

Refine. Tutors should give feedback to the teaching staff. It can help the teaching staff to know what is hindering or helping them and what they can do to better adapt and integrate the strategy in the course if needed.

10.5 Threats to Validity

Inherent to qualitative research methods and in particular to observations, it is difficult to assess the impact of observation on tutors actual strategy usage. We observed tutors mentioning they applied the strategy *"to please the observer"* during the training week of observations of the third iteration. The training week was put in place specifically to make tutors used to being observed and to mitigate the effect of observations on their behavior. We also made it clear all along to tutors that they could withdraw from the research. Some tutors opted out of the research from the beginning and one tutor actually withdrew during the semester, mentioning observations put too much pressure on them. We believe others would have done so too if they would have felt the need for it. It is also interesting to mention that as a side effect some UTAs asked for personal feedback on their teaching practice during their lab session when they knew they had been observed. This shows a real interest to become better tutors. Overall, we think this limitation has been mitigated in our study design.

While for our two first iterations one might argue that our sample size was quite limited, it was deliberate in the design research approach we chose for this thesis to start small and grow bigger. Those two first iterations still allowed us the develop some deep insights into tutors' strategy use and their perspective on it. This approach allowed us to refine our design principles and to better anticipate tutors' needs for the next iteration. Finally, in our third iteration, we no longer suffered from that limitation since all tutors had to apply subgoal learning.

Since we only worked on one specific CS1 course, it is difficult to judge whether our results and advice are generalisable to other courses or course settings. Nevertheless, as discussed in Section 2.6, it is common in CS1 courses to use (U)TAs and in our opinion, the training setups and advice to practitioners we proposed can be integrated in different kind of CS1 setups.

10.6 Lessons Learned on the Methodology

In retrospect we do think that mixing research methods from different research paradigms (cf. Section 3.1) has been beneficial to the overall research work and conclusions presented in this dissertation. In particular, crossing results and analysis from observation and interviews with more deductive

tools such as fidelity of implementation criteria allowed to both have a more synthetic normative appreciation of tutors' usage of the strategies as well as some deeper knowledge on how they actually used the strategies and their opinion and perspective on the subject.

Qualitative research, and coding interview transcripts or recording segments always has a subjective part, and while this might be intimidating at first, we really enjoyed the process of coding and "discovering" themes from the data. While the coding itself can be quite tedious when confronted to a lot of data, being able to offload part of this on hired observers has really been efficient in our opinion. However, this did not come at no cost. Dividing the work between multiple observers also holds its own challenges. The main costs were the training and the logistic to coordinate. Regarding observers' training, the combination of theoretical elements from literature on the strategy motivations and usage, and actual observational training on old recordings and during a training week proved enough in the end to ensure the quality of the observation. Regarding logistics, we had to setup a shared server to host all the recording, to assign observers to tutors during specific weeks and to organise follow-up meetings. These meetings allowed us to discuss operational and methodological issues with the observers and to ensure methodological reliability in how the coded their video segments. We would definitely recommend this multi-observers approach for CEd researchers that have the time to set it up properly.

On design-based research as a whole, we think it proved to be a really nice method that allowed us to progressively better understand the studied tutors' need and feeling about our strategy integration. It also allowed us to focus more and more on less strategies and more tutors while at the same time refining and tailoring our instructional design interventions. While this obviously takes time, it also allowed to make trials and more exploratory work at first to finally integrate subgoal learning in the course material the best we could, one informed by previous iterations. We would definitely recommend this research methodology for similar long term interventions.

A last element on literature, we found it quite challenging at times to have to combine different research traditions from different countries and languages (in particular, English- and French-speaking (CS)Ed research). There was sometimes quite a disconnection between the research interests, the theories and conceptual models used in research (e.g., more didactical approaches in the French-speaking Didapro[1] community) or the "influential" sources which are not the same (e.g., the learning transfer references in French). We ended up mixing references to both literature when appropriate.

[1]See https://www.didapro.org/ for more informations on Didapro.

Conclusion and Future Directions

11

This chapter presents some possible future directions for this dissertation. We first consider the possibility to train school teachers with explicit instructional programming strategies (11.1), we then propose some other future work (11.2) and finally we conclude (11.3).

11.1 Another Audience: School Teachers

Revisiting my initial thesis proposal in hindsight, it is clear that training school teachers had always been an implicit long term objective. This still perspires in the title of my ICER doctoral consortium submission [Gol21] ("Promoting Learning Transfer in Computer Science Education by Training Teachers to Use Explicit Programming Strategies"), even though this dissertation eventually targeted an audience of UTAs. In this section, we reflect back upon this initial idea and try to identify some elements from literature to assess whether an introduction of explicit instructional programming strategies could make sense and be provided for school teachers in the Belgian context.

In order to discuss the training of school teachers, we decided to compare both the UTA and teacher audiences using the concepts of *professional identity* and *motivation* to teach. We will look at the main factors influencing these fundamental concepts and how close or different to each other both audiences are in that regard.

Before comparing both UTA and teacher audiences, here is a summary of what worked and did not work in UTAs' usage of explicit instructional programming strategies. Some of these elements come from observations and interview segments not necessarily reported on explicitly earlier in previous chapters:

- Tutors often have close to no experience with teaching. An impact of this is they had trouble identifying when using a strategy. To compensate for this, we provided them with training and follow-up during the course.
- Tutors never seem to have enough time. Their time is scarce outside of

the lab since they have other courses to follow and need to prepare for the tutored sessions. To compensate this, we tailored the strategies to the course content and integrated subgoals presentation in the course material in order for them not to have to introduce them themselves.

- Tutors have class management issues. We observed this, UTAs reported on this and literature also states this [LBG00]. Such issues were discussed during tutors weekly briefings and in their required faculty-level training.

- Tutors need training for their "professional development" and to be able to better know how and when to use instructional strategies. We supported UTAs with in-person training and training material.

The questions that guided us in this post-facto analysis are:

- How different are both audiences' (the studied UTAs and school teachers) professional identities and motivations?

- How feasible and realistic would it be to train school teachers with explicit instructional programming strategies in the Belgian context?

We first introduce the French-speaking Belgian context, then compare the professional identities and motivations of school teachers and UTAs, and finally we give our own take on the feasibility of training school teachers with explicit instructional programming strategies.

11.1.1 The Context in Belgium

After multiple calls internationally for more computer science for all in schools [Gan+13; Cas+18] and following a European trend to reintroduce computer science in school curricula [Roy12; Vah+17], the French speaking part of Belgium — "FWB", for "Fédération Wallonie-Bruxelles", the language community responsible for french speaking education — is reintroducing some CS elements as part of a reform[1] of the grades 1-9 curricula[2]. Since CS wasn't officially part of the general curriculum before this reform [HJ16], the questions of who will teach this subject and of which background they will have are still unanswered. In France or England, it was observed early into the reintroduction of CS into schools that teachers at lower-secondary school/middle school level came largely from other

[1] See https://pactepourunenseignementdexcellence.cfwb.be for more information on the reform.

[2] See http://www.enseignement.be/index.php?page=23827&do_id=17242&do_check=CNEJLFQGEC for more information on the "manual, technical, technological and digital" curriculum

disciplines such as mathematics, sciences and technologies and had low confidence in their knowledge of computing subjects [Bra23; FV21]. It seems realistic to expect the same in FWB. The CER community is well aware that such policy changes ask for more teacher training and has already proposed guidelines on how to train teachers [Cas+18; DeL+18].

Pedagogical content knowledge (PCK) [Shu86] in computing education [Sae+11; Hub18] was up until recently not taught in higher education in FWB [HS18]. Before the enactment of FWB's primary and lower-secondary school reform, school teachers in FWB were already interested in training for teaching computing [HS18]. Contrary to UTAs, in-service teachers already have pedagogical knowledge. When pursuing professional development in order to teach computing topics, they will be in need for both computing content knowledge and pedagogical content knowledge [HS18; MD19].

11.1.2 Identity and Motivation

Teachers *Teacher identity* is the sense of being recognised as a teacher by oneself and by others [BT09]. Many factors of teacher motivation seem to be linked to their identity. Education research talks about the concept of teacher identity. Teacher's identity is a continuous process that shifts over time under the influence of a range of both internal and external factors [BT09]. Cattonar states teacher identity has three main components [Cat05]:

- a collective one, coming from shared experience like training.
- an individual one, linked to their personal story and experiences.
- a contextual one, depending on their school environment and social relations.

Cattonar also states: *"[Teachers] chose the profession primarily for the content of the work (working with pupils, teaching the subject or pedagogy) and not for favourable employment conditions such as free time or job security."* [Cat05]. Cattonar also mentions that teachers in FWB are more and more pushed towards a model of *"reflexive practitioners"*.

Regarding motivation, Han and Yin distinguish *initiating motivation* and *sustaining motivation* [HY16]. They report on pre-service teachers stating that *"intrinsic and altruistic motivations were crucial for satisfying and enduring career in the classroom in developed countries"*. For teachers, they also mention Thoonen et al. [Tho+11] who presented a model that assumes that teacher motivation indirectly influences the quality of teaching practice through their engagement in professional learning activities.

For computing teachers specifically, research identified major challenges for computer science teachers. Ni et al. mention [Ni+21]: unstable and new curriculum, lack of subject knowledge especially for teachers for whom CS is not the main subject, isolation since CS teachers are often alone with the few CS hours being taught.

It is quite clear already that being a computing teacher will be a challenge and that support will be needed for teachers retention and identity. This could be done through building communities and providing opportunities for professional development [NG12].

Since it seems that secondary school teachers prefer to learn from direct peers and informal contexts [Lec+19], models of communities of practice for computer teachers could foster teachers' professional development and building their identity [Ni+11]. There are examples such as the Computing At School efforts in England [JMH13] or the high-school CS teacher community we documented during our Erasmus+ project [Bac+23].

UTAs We did not find the equivalent amount of research on (U)TA's identity or motivations to teach. Mostly, literature speaks about intrinsic (student interactions, helping others, societal usefulness) and extrinsic factors (salary, ECTS, CV building) to motivation. Effective feedback seems to be related to intrinsic motivation [Rod+14].

Meyers et al. mention three major themes as motivation for enrolling as a TA [MGG12]: *"(1) helping others — many indicated that they wanted to help First-Year students in their transition into engineering since it was difficult for them, (2) resume-builder — others also indicated they needed a job anyway and it was related to engineering, and (3) they looked up to their student assistant when they were first year students"*. Prior experience with a teaching method and beliefs in this teaching method are also mentioned by Wheeler et al. [Whe+19]. In our context, tutors also gain ECTS and are paid for the job once trained.

Comparison The main difference between teachers and UTAs in our view and based on the previous research-informed sections is that teachers are mainly teaching while UTAs are mainly students. Both audiences seem to share an intrinsic motivation to help and interact with students. However, the scarcity of UTAs' identity literature leaves this mainly as a conjecture.

While literature on teachers' identity already mentions multiple identities, it seems in the case of new CS teachers the lack of a CS teacher identity might hinder their motivation to train in CS teaching. If the future teachers in charge of CS identify mainly as other subject teachers, this main aspect might be a major hindrance. It seems to us that Ni et al.'s idea of

building teachers communities could really help in that regard and might also be an opportunity to provide professional development.

Another big difference is that CS teachers will likely lack in CS content knowledge while they will have more pedagogical knowledge, classroom management skills and their own subject pedagogical content knowledge. This makes them quite different that UTAs that are in a sense subject experts but with very little teaching experience and little pedagogical knowledge.

11.1.3 Discussion

Education literature on the gap between education research and teachers' practice advocates for an effort from both sides [McI05]. Suggested steps to bridge this gap include school participation and practice-informed research [McI05; Ryc22]. We believe efforts to bring evidence-based practice through teacher training goes in the right direction.

While both audiences have different identities and motivations to teach, our research and literature show that both UTAs and school teachers benefit from professional development, are motivated by being included in research and by their interactions with students. Education literature also shows that teachers will adapt professional development content and weave it into their practice, like the studied UTAs did.

Overall, the idea of teaching using explicit instruction programming strategies would also need to fit teachers own *subjective educational theory* [Kel09]. In fact, regarding our own strategies, we already had the opportunity to present our own research results to pre-service and in-serve teachers in multiple professional development workshops [GM22]. While this remains anecdotal, it shows at least some interest of the teacher community. We also supervised a master thesis on bringing research-supported instructional practice in schools: the integration of PRIMM-inspired block programming sequences [Deb22].

We can also expect difficulties when transitioning from block-based to text programming. This has been described in detail in literature, by Weintrop's series on the subject [WH17; WW19] or in France by Branthôme in his dissertation [Bra23]. Nevertheless, Tshukudu's series on transfer when switching between two programming languages suggest that explicitly teaching for transfer could help [TJ20; Tsh+21].

Computing education research from school-focused venues like ACM's WiPSCE — Workshop in Primary and Secondary Computing Education — and ISSEP — Informatics in Schools: Situation, Evolution and Perspectives — will continue to produce research that is more practice-informed. It could then make for a source of (explicit) instructional (programming) strategies

used in teacher's professional development.

Teachers will have less classroom management issues, more time at hand and overall more pedagogical expertise than UTAs, but they will (at first) lack in CS content knowledge. We hypothesise for teachers that explicit strategies might be of help to increase their self-efficacy to teach programming concepts. It they are less confident on the content but are provided with research-informed strategies to teach it, this might actually boost their motivation and help them cope with the new challenge of being CS teachers.

We hypothesise that the strategies presented in this dissertation could be adapted to school teaching and presented to CS teachers during their initial training or during professional development training. The strategies would have to be adapted and aligned to the learning objectives. The programming language in lower-secondary school will be block-based at first. Depending on the notional machine, the explicit tracing steps could be adapted for memory representation but the main steps would still remain. In a school context, we also think subgoal learning would maybe be more interesting for simpler concepts than perceived by the UTAs.

11.2 Future Directions

A (big) question we deliberately decided not to study in this dissertation is the impact on students' learning outcomes. While we already collected some feedback and students seem to find subgoal learning useful, it would make sense to measure the actual gain of students having had tutors using the strategies. As discussed before, the benefits might not be limited to learning outcomes, but a reduction in student drop-rates or variance in students' results to tests could be measured as well. Does it help them? (How) do they use and adapt them? Such questions seem relevant questions to address in future follow-up research.

Another interesting question worth investigating is the long term impact of our design intervention on the course and on UTAs' practice. We put a lot of effort in the integration of subgoal learning in the course and produced companion training documents that could be used to keep new tutors informed of the way to use explicit tracing and subgoal learning. Will trained tutors and future UTAs continue to use the strategies? Will the course teachers remain sufficiently motivated to train the tutors?

Other strategies, integration ideas or research informed practices could be considered for integration in CS1 courses. We actually already tried some ideas at small scale in the scope of master theses like: the variation of type of feedback given to students [Ver23] or the integration of Parsons problems in the online auto-grader for the course [Len24].

Finally, a last research avenue is to apply the outcomes of this research to other institutions. We would be interested in feedback on other higher education faculty members using our advice to practitioners (cf. Section 10.3) to introduce the same or other strategies in their own setup. Can our results be reproduced in other courses and university setups? What other strategies are integrated by other practitioners?

11.3 Conclusion

In this dissertation, we researched how different iterations of design interventions in an introductory programming course can encourage undergraduate teaching assistants to make use of evidence-based strategies in their practice. After three main iterations, we showed that the combination of UTA training, follow-up and the integration of explicit strategies and especially subgoal learning in the course material and in companion documents can indeed foster the use of such strategies by the studied UTAs. We believe our research also demonstrated how different flavours of instructional practice and in particular how more guided strategies could blend together with the problem-based setup of a CS1 course. Since most higher-education institutions with CS1 courses will need to scale and since the number of UTAs can easily scale with the number of students, we believe our approach can be applied in other CS1 setups. To facilitate the integration of explicit strategies in other course setups, we provided to willing practitioners a set of actionable advice informed by our own experience and research. We have also showed how mixed research methods from different paradigms and epistemologies can be combined for a richer and deeper analysis of computing education research questions. While acknowledging the limitations of our work, we also hope our results will prove valuable contributions to the field and would be pleased to see more integration of explicit instructional programming strategies discussed in computing education research.

Bibliography

[Abe08] S. K. Abell. "Twenty Years Later: Does pedagogical content knowledge remain a useful idea?" In: *International Journal of Science Education* 30.10 (2008), pp. 1405–1416. DOI: 10.1080/09500690802187041.

[AD13] R. K. Atkinson and S. J. Derry. "Computer-Based Examples Designed to Encourage Optimal Example Processing: A Study Examining the Impact of Sequentially Presented, Subgoal-Oriented Worked Examples 1". In: *International Conference of the Learning Sciences*. Psychology Press. 2013, pp. 132–133. ISBN: 9780203763865.

[AEJ17] T. E. Adams, C. Ellis, and S. H. Jones. "Autoethnography". In: *The International Encyclopedia of Communication Research Methods*. John Wiley & Sons, Ltd, 2017, pp. 1–11. ISBN: 978-1-118-90173-1. DOI: 10.1002/9781118901731.iecrm0011.

[AKS20] G. Ashman, S. Kalyuga, and J. Sweller. "Problem-Solving or Explicit Instruction: Which Should Go First When Element Interactivity Is High?" In: *Educational Psychology Review* 32.1 (Mar. 2020), pp. 229–247. ISSN: 1573-336X. DOI: 10.1007/s10648-019-09500-5.

[AP20] H. H. Alharahsheh and A. Pius. "A Review of Key Paradigms: Positivism VS Interpretivism". In: *Global Academic Journal of Humanities and Social Sciences* 2.3 (2020), pp. 39–43. DOI: 10.36348/gajhss.2020.v02i03.001.

[Bac+23] S. Bachy, F. Chambon, P. Corieri, O. Goletti, S. Hoarau, V. Komis, T. Massart, K. Mens, G. Parriaux, C. Poulmaire, M. Romero, M. Rafalska, and T. Viéville. "Création d'une Communauté d'Apprentissage de l'Informatique". In: *Adjectif : analyses et recherches sur les TICE* (Nov. 2023).

[Bak18] A. Bakker. *Design research in education: A practical guide for early career researchers*. London and New York: Routledge (Taylor & Francis), 2018. ISBN: 9781138574489.

[Bar96] H. S. Barrows. "Problem-based learning in medicine and beyond: A brief overview". In: *New Directions for Teaching and Learning* 1996.68 (1996), pp. 3–12. DOI: 10.1002/tl.37219966804.

[BC19] V. Braun and V. Clarke. "Reflecting on reflexive thematic analysis". In: *Qualitative Research in Sport, Exercise and Health* 11.4 (2019), pp. 589–597. DOI: 10.1080/2159676X.2019.1628806.

[BM08] A. Bell and R. Mladenovic. "The Benefits of Peer Observation of Teaching for Tutor Development". In: *Higher Education* 55.6 (June 2008), pp. 735–752. ISSN: 1573-174X. DOI: 10.1007/s10734-007-9093-1.

[BM15] A. Bell and R. Mladenovic. "Situated learning, reflective practice and conceptual expansion: effective peer observation for tutor development". In: *Teaching in Higher Education* 20.1 (2015), pp. 24–36. DOI: 10.1080/13562517.2014.945163.

[Bor+13] M. Borrego, S. Cutler, M. Prince, C. Henderson, and J. E. Froyd. "Fidelity of Implementation of Research-Based Instructional Strategies (RBIS) in Engineering Science Courses". In: *Journal of Engineering Education* 102.3 (2013), pp. 394–425. ISSN: 2168-9830. DOI: 10.1002/jee.20020.

[BP04] N. Brouillette and A. Presseau. "Expérimentation En Contexte Scolaire d'un Modèle Axé Sur Le Transfert Des Apprentissages". In: *Le Transfert Des Apprentissages: Comprendre Pour Mieux Intervenir*. Presses Université Laval, 2004, pp. 161–212. DOI: 10.2307/jj.8816087.10.

[Bra04] D. Bracke. "Un Modèle Fonctionnel Du Tranfert Pour l'éducation". In: *Le Transfert Des Apprentissages: Comprendre Pour Mieux Intervenir*. Presses Université Laval, 2004, pp. 77–106. DOI: 10.2307/jj.8816087.7.

[Bra23] M. Branthôme. "Apprentissage de la programmation informatique : analyses et ressources pour accompagner la transition collège-lycée". Theses. Université de Bretagne occidentale - Brest, Oct. 2023.

[BT09] C. Beauchamp and L. Thomas. "Understanding Teacher Identity: An Overview of Issues in the Literature and Implications for Teacher Education". In: *Cambridge Journal of Education* 39.2 (June 2009), pp. 175–189. ISSN: 0305-764x. DOI: 10.1080/03057640902902252.

[Cas+18] M. E. Caspersen, J. Gal-Ezer, A. McGettrick, and E. Nardelli. *Informatics for All The Strategy*. Tech. rep. Acm, Feb. 2018. DOI: 10.1145/3185594.

[Cat05] B. Cattonar. "Convergence et diversité de l'identité professionnelle des enseignants du secondaire en Communauté française de Belgique. Tensions entre le vrai travail et le sale boulot". In: *Education et Francophonie* 1 (2005).

[Cat11] R. Catrambone. "Task Analysis by Problem Solving (TAPS): Uncovering Expert Knowledge to Develop High-Quality Instructional Materials and Training". In: *Learning and Technology Symposium, Columbus, GA*. 2011.

[Cat98] R. Catrambone. "The subgoal learning model: Creating better examples so that students can solve novel problems." In: *Journal of experimental psychology: General* 127.4 (1998), p. 355. DOI: 10.1037/0096-3445.127.4.355.

[CH89] R. Catrambone and K. J. Holyoak. "Overcoming contextual limitations on problem-solving transfer." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15.6 (1989), p. 1147. DOI: 10.1037/0278-7393.15.6.1147.

[Chi+21] L. Chiodini, I. Moreno Santos, A. Gallidabino, A. Tafliovich, A. L. Santos, and M. Hauswirth. "A Curated Inventory of Programming Language Misconceptions". In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. Virtual Event Germany: Acm, June 2021, pp. 380–386. ISBN: 978-1-4503-8214-4. DOI: 10.1145/3430665.3456343.

[Cho+22] K. Choi, H. Shin, M. Xia, and J. Kim. "AlgoSolve: Supporting Subgoal Learning in Algorithmic Problem-Solving with Learnersourced Microtasks". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. Chi '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: 10.1145/3491102.3501917.

[Cla+08] R. Clark, D. Feldon, J. J. G. Van Merrienboer, K. Yates, and S. Early. "Cognitive Task Analysis". In: *Handbook of Research on Educational Communications and Technology*. Routledge, Jan. 2008. Chap. 43, pp. 577–593. ISBN: 9780203880869.

[Cor+20] P. Corieri, M. Romero, T. Massart, O. Goletti, K. Mens, M. Rafalska, T. Viéville, L. Meziane, J. Christophe, S. Hoarau, et al. "Enjeux dans la création d'une communauté d'enseignants engagés dans l'apprentissage de l'informatique". In: *Didapro 8-DidaSTIC-Colloques francophones de didactique de l'informatique*. Poster. 2020.

[Cre12a] J. W. Creswell. "Analyzing and Interpreting Qualitative Data". In: *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. 4th ed. Boston: Pearson, 2012. Chap. 8, pp. 236–246. ISBN: 978-0-13-261394-1.

[Cre12b] J. W. Creswell. "Collecting Qualitative Data". In: *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. 4th ed. Boston: Pearson, 2012. Chap. 7, pp. 212–217. ISBN: 978-0-13-261394-1.

[Cre12c] J. W. Creswell. "Survey Designs". In: *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. 4th ed. Boston: Pearson, 2012. Chap. 12, pp. 375–422. ISBN: 978-0-13-261394-1.

[Cun+17] K. Cunningham, S. Blanchard, B. Ericson, and M. Guzdial. "Using Tracing and Sketching to Solve Programming Problems: Replicating and Extending an Analysis of What Students Draw". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 164–172. ISBN: 9781450349680. DOI: 10.1145/3105726.3106190.

[Dan+17] H. Danielsiek, J. Vahrenhold, P. Hubwieser, J. Krugel, J. Magenheim, L. Ohrndorf, D. Ossenschmidt, and N. Schaper. "Undergraduate Teaching Assistants in Computer Science: Teaching-related Beliefs, Tasks, and Competences". In: *2017 IEEE Global Engineering Education Conference (EDUCON)*. Athens, Greece: Ieee, Apr. 2017, pp. 718–725. ISBN: 978-1-5090-5467-1. DOI: 10.1109/educon.2017.7942927.

[DD16] T. Dragon and P. E. Dickson. "Memory Diagrams: A Consistant Approach Across Concepts and Languages". In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. SIGCSE '16. New York, NY, USA: Acm, 2016, pp. 546–551. ISBN: 978-1-4503-3685-7. DOI: 10.1145/2839509.2844607.

[DD21] P. E. Dickson and T. Dragon. "A Memory Diagram for All Seasons". In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE '21. New York, NY, USA: Association for Computing Machinery, June 2021, pp. 150–156. ISBN: 978-1-4503-8214-4. DOI: 10.1145/3430665.3456317.

[DDL17] P. E. Dickson, T. Dragon, and A. Lee. "Using Undergraduate Teaching Assistants in Small Classes". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 165–170. ISBN: 9781450346986. DOI: 10.1145/3017680.3017725.

[Deb22] C. Debongnie. "Didactique de l'informatique : la méthodologie PRIMM". MA thesis. UCL - Ecole polytechnique de Louvain, 2022.

[deJ+22] T. de Jong, A. W. Lazonder, C. A. Chinn, F. Fischer, J. Gobert, C. E. Hmelo-Silver, K. R. Koedinger, J. S. Krajcik, E. A. Kyza, M. C. Linn, M. Pedaste, K. Scheiter, and Z. C. Zacharia. "Let's talk evidence – The case for combining inquiry-based and direct instruction". In: *Educational Research Review* 39 (2023), p. 100536. ISSN: 1747-938X. DOI: 10.1016/j.edurev.2023.100536.

[DeL+18] L. A. DeLyser, M. Guzdial, J. Goode, Y. Kafai, and A. Yadav. *Priming the CS Teacher Pump. Integrating Computer Science Education into Schools of Education*. Tech. rep. CSforALL, 2018.

[Dem24] A. Demblon. "Intégration de l'apprentissage par étapes dans les ressources d'un cours d'introduction à la programmation". MA thesis. UCL - Ecole polytechnique de Louvain, 2024.

[DeP22] F. De Pierpont. "Intégration des stratégies Subgoal Labeled Worked Exemples (SLWEs) et Explicit Tracing aux travaux pratiques du cours d'introduction à la programmation". MA thesis. UCL - Ecole polytechnique de Louvain, 2022.

[Der+15] G. Derval, A. Gego, P. Reinbold, B. Frantzen, and P. Van Roy. "Automatic grading of programming exercises in a MOOC using the INGInious platform". In: *European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'15)* (2015), pp. 86–91.

[DLD20] Y. Du, A. Luxton-Reilly, and P. Denny. "A Review of Research on Parsons Problems". In: *Proceedings of the Twenty-Second Australasian Computing Education Conference*. ACE'20. Melbourne, VIC, Australia: Association for Computing Machinery, 2020, pp. 195–202. ISBN: 9781450376860. DOI: 10.1145/3373165.3373187.

[DLS08] P. Denny, A. Luxton-Reilly, and B. Simon. "Evaluating a new exam question: Parsons problems". In: *Proceedings of the Fourth International Workshop on Computing Education Research*. ICER '08. Sydney, Australia: Association for Computing Machinery, 2008, pp. 113–124. ISBN: 9781605582160. DOI: 10.1145/1404520.1404532.

[duB86] B. du Boulay. "Some Difficulties of Learning to Program". In: *Journal of Educational Computing Research* 2.1 (Feb. 1986), pp. 57–73. ISSN: 0735-6331, 1541-4140. DOI: 10.2190/3lfx-9rrf-67t8-uvk9.

[DZS22] R. Duran, A. Zavgorodniaia, and J. Sorva. "Cognitive Load Theory in Computing Education Research: A Review". In: *ACM Transactions on Computing Education* 22.4 (Sept. 2022), 40:1–40:27. DOI: 10.1145/3483843.

[EFR18] B. J. Ericson, J. D. Foley, and J. Rick. "Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems". In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ICER '18. New York, NY, USA: Acm, 2018, pp. 60–68. ISBN: 978-1-4503-5628-2. DOI: 10.1145/3230977.3231000.

[EMR17] B. J. Ericson, L. E. Margulieux, and J. Rick. "Solving Parsons Problems Versus Fixing and Writing Code". In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. Koli Calling '17. New York, NY, USA: Acm, 2017, pp. 20–29. ISBN: 978-1-4503-5301-4. DOI: 10.1145/3141880.3141895.

[Eri+16] B. J. Ericson, K. Rogers, M. Parker, B. Morrison, and M. Guzdial. "Identifying Design Principles for CS Teacher Ebooks Through Design-Based Research". In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ICER '16. New York, NY, USA: Acm, 2016, pp. 191–200. ISBN: 978-1-4503-4449-4. DOI: 10.1145/2960310.2960335.

[Eri+22] B. J. Ericson, P. Denny, J. Prather, R. Duran, A. Hellas, J. Leinonen, C. S. Miller, B. B. Morrison, J. L. Pearce, and S. H. Rodger. "Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs". In: ITiCSE-WGR '22 (2022), pp. 191–234. DOI: 10.1145/3571785.3574127.

[ET17] F. J. Estrada and A. Tafliovich. "Bridging the Gap Between Desired and Actual Qualifications of Teaching Assistants: An Experience Report". In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '17. New York, NY, USA: Association for Computing Machinery, June 2017, pp. 134–139. ISBN: 978-1-4503-4704-4. DOI: 10.1145/3059009.3059023.

[FHE22] C. J. Felege, C. J. Hunter, and S. N. Ellis-Felege. "Personal Impacts of the Undergraduate Teaching Assistant Experience". In: *Journal of the Scholarship of Teaching and Learning* 22.2 (June 2022). ISSN: 1527-9316. DOI: 10.14434/josotl.v22i2.31306.

[Fin+20] S. Fincher, J. Jeuring, C. S. Miller, P. Donaldson, B. du Boulay, M. Hauswirth, A. Hellas, F. Hermans, C. Lewis, A. Mühling, J. L. Pearce, and A. Petersen. "Notional Machines in Computing Education: The Education of Attention". In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '20. Trondheim, Norway: Association for Computing Machinery, 2020, pp. 21–50. ISBN: 9781450382939. DOI: 10.1145/3437800.3439202.

[For+17] J. Forbes, D. J. Malan, H. Pon-Barry, S. Reges, and M. Sahami. "Scaling Introductory Courses Using Undergraduate Teaching Assistants". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 657–658. ISBN: 9781450346986. DOI: 10.1145/3017680.3017694.

[Fre+07] M. Frenay, B. Galand, E. Milgrom, and B. Raucent. "Project- and Problem-Based Learning in the Engineering Curriculum at the University of Louvain". In: Leiden, The Netherlands: Brill, 2007, pp. 93–108. ISBN: 9789087900922. DOI: 10.1163/9789087900922_008.

[FV21] B. Fowler and E. Vegas. *How England Implemented Its Computer Science Education Program*. Tech. rep. Center for Universal Education at The Brookings Institution, Jan. 2021.

[Gan+13] W. Gander, A. Petit, G. Berry, B. Demo, J. Vahrenhold, A. McGettrick, R. Boyle, A. Mendelson, C. Stephenson, and C. Ghezzi. *Informatics Education: Europe Cannot Afford to Miss the Boat*. Tech. rep. Informatics Europe & ACM Europe Working Group on Informatics Education, 2013.

[GDM22] O. Goletti, F. De Pierpont, and K. Mens. "Création d'exemples résolus avec objectifs étiquetés pour l'apprentissage de la programmation avec Python". In: *Didapro 9–DidaSTIC*. 2022.

[GFR12] B. Galand, M. Frenay, and B. Raucent. "Effectiveness of problem-based learning in engineering education: a comparative study on three levels of knowledge structure". In: *The International journal of engineering education* 28.4 (2012), pp. 939–947.

[GM22] O. Goletti and K. Mens. "Atelier : Utiliser des stratégies d'instruction explicites dans l'enseignement de la programmation". In: *Didapro 9 – DidaSTIC*. 2022.

[GMH21] O. Goletti, K. Mens, and F. Hermans. "Tutors' Experiences in Using Explicit Strategies in a Problem-Based Learning Introductory Programming Course". In: *ITiCSE '21*. Virtual Event, Germany: ACM Press, June 2021. DOI: 10.1145/3430665.3456348.

[GMH22] O. Goletti, K. Mens, and F. Hermans. "An Analysis of Tutors' Adoption of Explicit Instructional Strategies in an Introductory Programming Course". In: *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research*. 2022, pp. 1–12. DOI: 10.1145/3564721.3565951.

[GMH24] O. Goletti, K. Mens, and F. Hermans. "An Observational Study of Undergraduate Teaching Assistants' Use of Subgoal Learning Integrated in an Introductory Programming Course". In: *Proceedings of the 2024 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E '24)*. Pasadena, CA, USA: ACM Press, Oct. 2024. DOI: 10.1145/3689493.3689986.

[Gol19] O. Goletti. *En quoi le dispositif mis en œuvre dans le cours d'introduction à l'informatique en BAC1 ingénieur civil basé sur l'apprentissage par problèmes soutient les processus du transfert des apprentissages : l'encodage et l'accessibilité aux connaissances ?* Tech. rep. http://hdl.handle.net/2078.1/245579. UCLouvain, 2019.

[Gol21] O. Goletti. "Promoting Learning Transfer in Computer Science Education by Training Teachers to Use Explicit Programming Strategies". In: *ICER '17*. Virtual Event USA: Acm, Aug. 2021, pp. 411–412. DOI: 10.1145/3446871.3469776.

[Gol24] O. Goletti. "Subgoal Learning Integration in a CS1 Course". In: *Colloque Didapro 10 Sur La Didactique de l'informatique et Des STIC*. 2024, pp. 131–135.

[GSH19] K. Geldreich, A. Simon, and P. Hubwieser. "A Design-Based Research Approach for Introducing Algorithmics and Programming to Bavarian Primary Schools". In: *MediaEducation: Journal for Theory and Practice of Media Education* (Feb. 2019), 53–75 Seiten. DOI: 10.21240/mpaed/33/2019.02.15.x.

[GSM11] D. Ginat, E. Shifroni, and E. Menashe. "Transfer, Cognitive Load, and Program Design Difficulties". In: *Informatics in Schools. Contributing to 21st Century Education*. Ed. by I. Kalaš and R. T. Mittermeir. Vol. 7013. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 165–176. ISBN: 978-3-642-24722-4. DOI: 10.1007/978-3-642-24722-4_15.

[Guz03] M. Guzdial. "A media computation course for non-majors". In: *SIGCSE Bull.* 35.3 (June 2003), pp. 104–108. ISSN: 0097-8418. DOI: 10.1145/961290.961542.

[Guz16] M. J. Guzdial. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*. Synthesis Lectures on Human-Centered Informatics 33. San Rafael?: Morgan & Claypool Publishers, 2016. ISBN: 978-1-62705-351-8.

[Haz+06] O. Hazzan, Y. Dubinsky, L. Eidelman, V. Sakhnini, and M. Teif. "Qualitative research in computer science education". In: SIGCSE '06 (2006), pp. 408–412. DOI: 10.1145/1121341.1121469.

[HDC07] C. E. Hmelo-Silver, R. G. Duncan, and C. A. Chinn. "Scaffolding and Achievement in Problem-Based and Inquiry Learning: A Response to Kirschner, Sweller, And". In: *Educational psychologist* 42.2 (2007), pp. 99–107. DOI: 10.1080/00461520701263368.

[HHB20] M. Hennink, I. Hutter, and A. Bailey. *Qualitative Research Methods*. Sage, 2020. ISBN: 9781473903913.

[HJ13] M. Hertz and M. Jump. "Trace-based teaching in early programming courses". In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE '13. Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 561–566. ISBN: 9781450318686. DOI: 10.1145/2445196.2445364.

[HJ16] J. Henry and N. Joris. *Informatics at Secondary Schools in the French-speaking Region of Belgium: Myth or Reality?* Tech. rep. accepted for the country reports track, Informatics in Schools: Improvement of Informatics Knowledge and Perception (ISSEP'16). UNamur, 2016.

[HMF16] F. Heintz, L. Mannila, and T. Farnqvist. "A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K-12 Education". In: *2016 IEEE Frontiers in Education Conference (FIE)*. Erie, PA, USA: Ieee, Oct. 2016, pp. 1–9. ISBN: 978-1-5090-1790-4. DOI: 10.1109/fie.2016.7757410.

[HS18] J. Henry and A. Smal. "«Et Si Demain Je Devais Enseigner l'informatique?» Le Cas Des Enseignants de Belgique Francophone". In: *Didapro 7–DidaSTIC. De 0 à 1 Ou l'heure de l'informatique à l'école*. 2018.

[Hub18] A. Hubbard. "Pedagogical Content Knowledge in Computing Education: A Review of the Research Literature". In: *Computer Science Education* 28.2 (2018), pp. 117–135.

[HWC13] M. Hu, M. Winikoff, and S. Cranefield. "A process for novice programming using goals and plans". In: *Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136*. ACE '13. Adelaide, Australia: Australian Computer Society, Inc., 2013, pp. 3–12. ISBN: 9781921770210.

[HY16] J. Han and H. Yin. "Teacher Motivation: Definition, Research Development and Implications for Teachers". In: *Cogent Education* 3.1 (Dec. 2016). Ed. by M. Boylan, p. 1217819. ISSN: 2331-186x. DOI: 10.1080/2331186x.2016.1217819.

[HY17] J. R. Hollingsworth and S. E. Ybarra. *Explicit Direct Instruction (EDI): The Power of the Well-Crafted, Well-Taught Lesson*. Corwin Press, 2017. ISBN: 978-1506337517.

[IK11] P. Ihantola and V. Karavirta. "Two-dimensional parson's puzzles: The concept, tools, and first observations". In: *Journal of Information Technology Education. Innovations in Practice* 10 (2011), p. 119. DOI: 10.28945/1394.

[IM21] C. Izu and C. Mirolo. "Learning Transfer in Novice Programmers: A Preliminary Study". In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE '21. Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 178–184. ISBN: 9781450382144. DOI: 10.1145/3430665.3456336.

[JH20] F. D. F.-M. José L. Arco-Tirado and M. Hervás-Torres. "Evidence-based peer-tutoring program to improve students' performance at the university". In: *Studies in Higher Education* 45.11 (2020), pp. 2190–2202. DOI: 10.1080/03075079.2019.1597038.

[JMH13] S. P. Jones, B. Mitchell, and S. Humphreys. "Computing at School in the UK". In: *CACM Report* (2013).

[JS87] W. Johnson and E. Soloway. "Intention-Based Diagnosis of Novice Programming Errors". In: *IEEE Expert* 2 (Oct. 1987), pp. 94–94. DOI: 10.1109/mex.1987.4307101.

[Kal17] M. Kallia. "Assessment in Computer Science Courses: A Literature Review". In: *Royal Society* (2017).

[Kau16] J.-C. Kaufmann. *L'entretien Compréhensif-4e Éd.* Armand Colin, 2016. ISBN: 9782200613976.

[Kel09] G. Kelchtermans. "Who I Am in How I Teach Is the Message: Self-understanding, Vulnerability and Reflection". In: *Teachers and Teaching* 15.2 (Apr. 2009), pp. 257–272. ISSN: 1354-0602. DOI: 10.1080/13540600902875332.

[Kir02] P. A. Kirschner. "Cognitive Load Theory: Implications of Cognitive Load Theory on the Design of Learning". In: *Learning and Instruction* 12.1 (Feb. 2002), pp. 1–10. ISSN: 0959-4752. DOI: 10.1016/s0959-4752(01)00014-7.

[Kir06] M. Kirley. "Supporting casual tutors and demonstrators: a case study in computer science and software engineering". In: *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*. ACE '06. Hobart, Australia: Australian Computer Society, Inc., 2006, pp. 109–115. ISBN: 1920682341.

[KJH18] H. Keuning, J. Jeuring, and B. Heeren. "A Systematic Literature Review of Automated Feedback Generation for Programming Exercises". In: *ACM Trans. Comput. Educ.* 19.1 (Sept. 2018). DOI: 10.1145/3231711.

[Ko+19] A. J. Ko, T. D. LaToza, S. Hull, E. A. Ko, W. Kwok, J. Quichocho, H. Akkaraju, and R. Pandit. "Teaching Explicit Programming Strategies to Adolescents". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. New York, NY, USA: Acm, 2019, pp. 469–475. ISBN: 978-1-4503-5890-3. DOI: 10.1145/3287324.3287371.

[Kom+22] V. Komis, S. Bachy, O. Goletti, G. Parriaux, M. Rafalska, and K. Lavidas. "Connaissances du contenu et connaissances technologiques des enseignants en Informatique en milieu francophone". In: *Review of Science, Mathematics and ICT Education* 16.2 (2022), pp. 105–133. DOI: 10.26220/rev.4080.

[KSC06] P. A. Kirschner, J. Sweller, and R. E. Clark. "Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching". In: *Educational Psychologist* 41.2 (June 2006), pp. 75–86. ISSN: 0046-1520, 1532-6985. DOI: 10.1207/s15326985ep4102_1.

[LaT+20] T. D. LaToza, M. Arab, D. Loksa, and A. J. Ko. "Explicit Programming Strategies". In: *Empirical Software Engineering* 25.4 (July 2020), pp. 2416–2449. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-020-09810-1.

[LBG00] J. Luo, L. Bellows, and M. Grady. "Classroom Management Issues for Teaching Assistants". In: *Research in Higher Education* 41.3 (June 2000), pp. 353–383. ISSN: 1573-188X. DOI: 10.1023/A:1007042911919.

[Lec+19] A. Lecat, I. Raemdonck, S. Beausaert, and V. März. "The what and why of primary and secondary school teachers' informal learning activities". In: *International Journal of Educational Research* 96 (2019), pp. 100–110. ISSN: 0883-0355. DOI: 10.1016/j.ijer.2019.06.003.

[Len24] C. Lengelé. "Integrating Parsons problems in a CS1 programming course autograder". MA thesis. UCL - Ecole polytechnique de Louvain, 2024.

[LFT09] R. Lister, C. Fidge, and D. Teague. "Further evidence of a relationship between explaining, tracing and writing skills in introductory programming". In: *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '09. Paris, France: Association for Computing Machinery, 2009, pp. 161–165. ISBN: 9781605583815. DOI: 10.1145/1562877.1562930.

[Lis+04] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas. "A multi-national study of reading and tracing skills in novice programmers". In: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '04. Leeds, United Kingdom: Association for Computing Machinery, 2004, pp. 119–150. ISBN: 9781450377942. DOI: 10.1145/1044550.1041673.

[Lis16] R. Lister. "Toward a Developmental Epistemology of Computer Programming". In: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. WiPSCE '16. Münster, Germany: Association for Computing Machinery, 2016, pp. 5–16. ISBN: 9781450342230. DOI: 10.1145/2978249.2978251.

[LK16] D. Loksa and A. J. Ko. "The Role of Self-Regulation in Programming Problem Solving Process and Success". In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ICER '16. Melbourne, VIC, Australia: Association for Computing Machinery, 2016, pp. 83–91. ISBN: 9781450344494. DOI: 10.1145/2960310.2960334.

[Lok+16] D. Loksa, A. J. Ko, W. Jernigan, A. Oleson, C. J. Mendez, and M. M. Burnett. "Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 1449–1461. ISBN: 9781450333627. DOI: 10.1145/2858036.2858252.

[Lux+18] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo. "Introductory Programming: A Systematic Literature Review". In: *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, July 2018, pp. 55–106. ISBN: 978-1-4503-6223-8. DOI: 10.1145/3293881.3295779.

[Lux16] A. Luxton-Reilly. "Learning to Program is Easy". In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '16. Are-

quipa, Peru: Association for Computing Machinery, 2016, pp. 284–289. ISBN: 9781450342315. DOI: 10.1145/2899415.2899432.

[Mar+22] L. E. Margulieux, P. Enderle, P. J. Clarke, N. King, C. Sullivan, M. Zoss, and J. Many. "Integrating Computing into Preservice Teacher Preparation Programs across the Core: Language, Mathematics, and Science". In: *Journal of Computer Science Integration* 5.1 (Nov. 2022). ISSN: 2574-108x. DOI: 10.26716/jcsi.2022.11.15.35.

[McC+01] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students". In: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '01. Canterbury, UK: Association for Computing Machinery, 2001, pp. 125–180. ISBN: 9781450373593. DOI: 10.1145/572133.572137.

[MCG13] L. Margulieux, R. Catrambone, and M. Guzdial. "Subgoal Labeled Worked Examples Improve K-12 Teacher Performance in Computer Programming Training". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 35. 2013.

[McI05] D. McIntyre. "Bridging the Gap between Research and Practice". In: *Cambridge Journal of Education* 35.3 (Nov. 2005), pp. 357–382. ISSN: 0305-764X. DOI: 10.1080/03057640500319065.

[MD19] P. Marquet and C. Declercq. "DIU Enseigner l'informatique Au Lycée". In: *1024 : Bulletin de la Société Informatique de France* 13 (Apr. 2019), pp. 47–56.

[Men15] M. Menekse. "Computer Science Teacher Professional Development in the United States: A Review of Studies Published between 2004 and 2014". In: *Computer Science Education* 25.4 (2015), pp. 325–350. DOI: 10.1080/08993408.2015.1111645.

[MGC12] L. E. Margulieux, M. Guzdial, and R. Catrambone. "Subgoal-Labeled Instructional Material Improves Performance and Transfer in Learning to Develop Mobile Applications". In: *Proceedings of the Ninth Annual International Conference on International Computing Education Research*. ICER '12. New York, NY, USA: Association for Computing Machinery, Sept.

2012, pp. 71–78. ISBN: 978-1-4503-1604-0. DOI: 10.1145/2361276.2361291.

[MGG12] K. Meyers, V. E. Goodrich, and N. Gedde. "Participation in an Undergraduate Teaching Assistantship: Experiences, Influences, and Outcomes". In: *2012 ASEE Annual Conference & Exposition*. 2012, pp. 25–1026.

[Mir+19] D. Mirza, P. T. Conrad, C. Lloyd, Z. Matni, and A. Gatin. "Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review". In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ICER '19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 31–40. ISBN: 9781450361859. DOI: 10.1145/3291279.3339422.

[MMD19] L. E. Margulieux, B. B. Morrison, and A. Decker. "Design and Pilot Testing of Subgoal Labeled Worked Examples for Five Core Concepts in CS1". In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '19*. Aberdeen, Scotland Uk: ACM Press, 2019, pp. 548–554. ISBN: 978-1-4503-6895-7. DOI: 10.1145/3304221.3319756.

[MMD20] L. E. Margulieux, B. B. Morrison, and A. Decker. "Reducing Withdrawal and Failure Rates in Introductory Programming with Subgoal Labeled Worked Examples". In: *International Journal of STEM Education* 7.1 (May 2020), p. 19. ISSN: 2196-7822. DOI: 10.1186/s40594-020-00222-7.

[MMG15] B. B. Morrison, L. E. Margulieux, and M. Guzdial. "Subgoals, Context, and Worked Examples in Learning Computing Problem Solving". In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. ICER '15. Omaha, Nebraska, USA: Association for Computing Machinery, 2015, pp. 21–29. ISBN: 9781450336307. DOI: 10.1145/2787622.2787733.

[Moo+13] A. Moon, H. Jung, F. Marbouti, K. Rodgers, and H. Diefes-Dux. "Undergraduate and Graduate Teaching Assistants' Perceptions of Their Responsibilities - Factors That Help or Hinder". In: *2013 IEEE Frontiers in Education Conference (FIE)*. Oct. 2013, pp. 1576–1578. DOI: 10.1109/FIE.2013.6685103.

[Mor14] D. L. Morgan. "Pragmatism as a Paradigm for Social Research". In: *Qualitative Inquiry* 20.8 (2014), pp. 1045–1053. DOI: 10.1177/1077800413513733.

[Mow+03] C. T. Mowbray, M. C. Holter, G. B. Teague, and D. Bybee. "Fidelity Criteria: Development, Measurement, and Validation". In: *American Journal of Evaluation* 24.3 (2003), pp. 315–340.

[MS05] J. J. G. van Merriënboer and J. Sweller. "Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions". In: *Educational Psychology Review* 17.2 (June 2005), pp. 147–177. ISSN: 1573-336x. DOI: 10.1007/s10648-005-3951-0.

[MS23] F. Muzny and M. D. Shah. "Teaching Assistant Training: An Adjustable Curriculum for Computing Disciplines". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 430–436. ISBN: 9781450394314. DOI: 10.1145/3545945.3569866.

[Nae+23] M. Naeem, W. Ozuem, K. Howell, and S. Ranfagni. "A Step-by-Step Process of Thematic Analysis to Develop a Conceptual Model in Qualitative Research". In: *International Journal of Qualitative Methods* 22 (2023), p. 16094069231205789. DOI: 10.1177/16094069231205789.

[Nat00] National Research Council. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. Washington, DC: The National Academies Press, 2000. ISBN: 978-0-309-07036-2. DOI: 10.17226/9853.

[NG12] L. Ni and M. Guzdial. "Who AM I? understanding high school computer science teachers' professional identity". In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. SIGCSE '12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 499–504. ISBN: 9781450310987. DOI: 10.1145/2157136.2157283.

[Ni+11] L. Ni, M. Guzdial, A. E. Tew, B. Morrison, and R. Galanos. "Building a Community to Support HS CS Teachers: The Disciplinary Commons for Computing Educators". In: *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. New York, NY, USA: Acm, 2011, pp. 553–558. ISBN: 978-1-4503-0500-6. DOI: 10.1145/1953163.1953319.

[Ni+21] L. Ni, T. McKlin, H. Hao, J. Baskin, J. Bohrer, and Y. Tian. "Understanding Professional Identity of Computer Science Teachers: Design of the Computer Science Teacher Identity Survey". In: *Proceedings of the 17th ACM Conference on International Computing Education Research*. ICER 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 281–293. ISBN: 978-1-4503-8326-4. DOI: 10.1145/3446871.3469766.

[NK18] G. L. Nelson and A. J. Ko. "On Use of Theory in Computing Education Research". In: *Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18*. Espoo, Finland: ACM Press, 2018, pp. 31–39. ISBN: 978-1-4503-5628-2. DOI: 10.1145/3230977.3230992.

[ODo08] C. L. O'Donnell. "Defining, conceptualizing, and measuring fidelity of implementation and its relationship to outcomes in K–12 curriculum intervention research". In: *Review of educational research* 78.1 (2008). Publisher: Sage Publications, pp. 33–84.

[Pal90] D. B. Palumbo. "Programming Language/Problem-Solving Research: A Review of Relevant Issues". In: *Review of Educational Research* 60.1 (Mar. 1990), pp. 65–89. ISSN: 0034-6543. DOI: 10.3102/00346543060001065.

[Pap80] S. Papert. *Mindstorms: children, computers, and powerful ideas*. Usa: Basic Books, Inc., 1980. ISBN: 0465046274.

[Pat12] E. Patitsas. "A Case Study of Environmental Factors Influencing Teaching Assistant Job Satisfaction". In: *Proceedings of the Ninth Annual International Conference on International Computing Education Research*. 2012, pp. 11–16.

[Pat13] E. Patitsas. "A case study of the development of CS teaching assistants and their experiences with team teaching". In: *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. Koli Calling '13. Koli, Finland: Association for Computing Machinery, 2013, pp. 115–124. ISBN: 9781450324823. DOI: 10.1145/2526968.2526981.

[Pea+07] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. "A Survey of Literature on the Teaching of Introductory Programming". In: *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '07. New York, NY, USA: Association for Computing Machinery, Dec.

2007, pp. 204–223. ISBN: 978-1-4503-7842-0. DOI: 10.1145/1345443.1345441.

[Pea86] R. D. Pea. "Language-Independent Conceptual "Bugs" in Novice Programming". In: *Journal of Educational Computing Research* 2.1 (Feb. 1986), pp. 25–36. ISSN: 0735-6331, 1541-4140. DOI: 10.2190/689t-1r2a-x4w4-29j2.

[Per+86] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons. "Conditions of learning in novice programmers". In: *Journal of Educational Computing Research* 2.1 (1986), pp. 37–55.

[PH06] D. Parsons and P. Haden. "Parson's programming puzzles: a fun and effective learning tool for first programming courses". In: *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*. ACE '06. Hobart, Australia: Australian Computer Society, Inc., 2006, pp. 157–163. ISBN: 1920682341.

[PRS03] F. Paas, A. Renkl, and J. Sweller. "Cognitive load theory and instructional design: Recent developments". In: *Educational psychologist* 38.1 (2003), pp. 1–4.

[PS13] L. Porter and B. Simon. "Retaining nearly one-third more majors with a trio of instructional best practices in CS1". In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE '13. Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 165–170. ISBN: 9781450318686. DOI: 10.1145/2445196.2445248.

[QL17] Y. Qian and J. Lehman. "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review". In: *ACM Trans. Comput. Educ.* 18.1 (Oct. 2017), 1:1–1:24. ISSN: 1946-6226. DOI: 10.1145/3077618.

[Rau+04] B. Raucent, J.-M. Braibant, M. N. d. Theux, C. Jacqmot, E. Milgrom, C. Vander Borght, and P. Wouters. "Devenir ingénieur par apprentissage actif: compte rendu d'innovation/How to become an engineer through active learning: report of innovation". In: *Didaskalia* 24.1 (2004), pp. 81–101.

[Rau04] B. Raucent. "What Kind of Project in the Basic Year of an Engineering Curriculum". In: *Journal of Engineering Design* 15.1 (2004), pp. 107–121.

[RC07] R. D. Roscoe and M. T. Chi. "Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions". In: *Review of educational research* 77.4 (2007), pp. 534–574.

[Ren+98] A. Renkl, R. Stark, H. Gruber, and H. Mandl. "Learning from Worked-out Examples: The Effects of Example Variability and Elicited Self-Explanations". In: *Contemporary educational psychology* 23.1 (1998), pp. 90–108.

[Rie+21] E. Riese, M. Lorås, M. Ukrop, and T. Effenberger. "Challenges Faced by Teaching Assistants in Computer Science Education Across Europe". In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE '21. Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 547–553. ISBN: 9781450382144. DOI: 10.1145/3430665.3456304.

[RK22] E. Riese and V. Kann. "Training Teaching Assistants by Offering an Introductory Course". In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. Vol. 1. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 745–751. ISBN: 978-1-4503-9070-5. DOI: 10.1145/3478431.3499270.

[RLR95] E. Roberts, J. Lilly, and B. Rollins. "Using undergraduates as teaching assistants in introductory programming courses: an update on the Stanford experience". In: *SIGCSE Bull.* 27.1 (Mar. 1995), pp. 48–52. ISSN: 0097-8418. DOI: 10.1145/199691.199716.

[Rod+14] K. J. Rodgers, F. Marbouti, A. Shafaat, H. Jung, and H. A. Diefes-Dux. "Influence of Teaching Assistants' Motivation on Student Learning". In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. Oct. 2014, pp. 1–8. DOI: 10.1109/fie.2014.7044004.

[Roy12] Royal Society. *Shut Down Or Restart?: The Way Forward for Computing in UK Schools*. Tech. rep. Royal Society (Great Britain), 2012.

[RTW07] M. de Raadt, M. Toleman, and R. Watson. "Incorporating programming strategies explicitly into curricula". In: *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88*. Koli Calling '07. Koli National Park, Finland: Australian Computer Society, Inc., 2007, pp. 41–52. ISBN: 9781920682699.

[Ryc22] L. Rycroft-Smith. "Knowledge Brokering to Bridge the Research-practice Gap in Education: Where Are We Now?" In: *Review of Education* 10.1 (Apr. 2022), e3341. ISSN: 2049-6613, 2049-6613. DOI: 10.1002/rev3.3341.

[Sae+11] M. Saeli, J. Perrenet, W. M. Jochems, and B. Zwaneveld. "Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective." In: *Informatics in education* 10.1 (2011), pp. 73–88.

[SAK11] J. Sweller, P. Ayres, and S. Kalyuga. *Cognitive Load Theory, Volume 1 of Explorations in the Learning Sciences, Instructional Systems and Performance Technologies*. Springer, New York, 2011.

[Sal+10] R. J. Salden, K. R. Koedinger, A. Renkl, V. Aleven, and B. M. McLaren. "Accounting for Beneficial Effects of Worked Examples in Tutored Problem Solving". In: *Educational Psychology Review* 22.4 (2010), pp. 379–392.

[Sch+07] H. G. Schmidt, S. M. Loyens, T. Van Gog, and F. Paas. "Problem-Based Learning Is Compatible with Human Cognitive Architecture: Commentary on Kirschner, Sweller, And". In: *Educational psychologist* 42.2 (2007), pp. 91–97.

[SDW17] P. Sobieski, D. Ducarme, and V. Wertz. "Renforcer l'analyse réflexive des tuteurs en formation". In: *Actes - Questions de pédagogie dans l'enseignement supérieur, QPES*. 2017.

[Sen21] S. Sentance. "Teaching computing in school: is K-12 research reaching classroom practice?" In: *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*. Koli Calling '21. Joensuu, Finland: Association for Computing Machinery, 2021. ISBN: 9781450384889. DOI: 10.1145/3488042.3491040.

[Shu86] L. S. Shulman. "Those who understand: Knowledge growth in teaching". In: *Educational researcher* 15.2 (1986), pp. 4–14.

[SL14] B. Skudder and A. Luxton-Reilly. "Worked Examples in Computer Science". In: *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148*. Ace '14. Aus: Australian Computer Society, Inc., Jan. 2014, pp. 59–64. ISBN: 978-1-921770-31-9.

[Sol86] E. Soloway. "Learning to Program= Learning to Construct Mechanisms and Explanations". In: *Communications of the ACM* 29.9 (1986), pp. 850–858.

[Sor12] J. Sorva. "Visual Program Simulation in Introductory Programming Education". PhD thesis. 2012. ISBN: 978-952-60-4626-6.

[SvP19] J. Sweller, J. J. van Merriënboer, and F. Paas. "Cognitive Architecture and Instructional Design: 20 Years Later". In: *Educational Psychology Review* 31 (2019). Publisher: Springer, pp. 1–32.

[Swe88] J. Sweller. "Cognitive load during problem solving: Effects on learning". In: *Cognitive science* 12.2 (1988), pp. 257–285.

[Sza+19] C. Szabo, N. Falkner, A. Petersen, H. Bort, K. Cunningham, P. Donaldson, A. Hellas, J. Robinson, and J. Sheard. "Review and Use of Learning Theories within Computer Science Education Research: Primer for Researchers and Practitioners". In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '19. New York, NY, USA: Association for Computing Machinery, Dec. 2019, pp. 89–109. ISBN: 978-1-4503-7567-2. DOI: 10.1145/3344429.3372504.

[Tar99] J. Tardif. *Le Transfert Des Apprentissages*. Editions logiques, 1999.

[Tho+11] E. E. J. Thoonen, P. J. C. Sleegers, F. J. Oort, T. T. D. Peetsma, and F. P. Geijsel. "How to Improve Teaching Practices: The Role of Teacher Motivation, Organizational Factors, and Leadership Practices". In: *Educational Administration Quarterly* 47.3 (Aug. 2011), pp. 496–536. ISSN: 0013-161x, 1552-3519. DOI: 10.1177/0013161x11400185.

[TJ20] E. Tshukudu and S. A. M. Jensen. "The Role of Explicit Instruction on Students Learning Their Second Programming Language". In: *United Kingdom & Ireland Computing Education Research Conference*. 2020, pp. 10–16.

[TR93] J. G. Trafton and B. J. Reiser. "Studying examples and solving problems: Contributions to skill acquisition". In: *Proceedings of the 15th conference of the Cognitive Science Society*. Citeseer. 1993, pp. 1017–1022.

[Tsh+21] E. Tshukudu, Q. Cutts, O. Goletti, A. Swidan, and F. Hermans. "Teachers' Views and Experiences on Teaching Second and Subsequent Programming Languages". In: *Proceedings of the 17th ACM Conference on International Computing Education Research*. ICER 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 294–305. ISBN: 978-1-4503-8326-4. DOI: 10.1145/3446871.3469752.

[TW01] E. L. Thorndike and R. S. Woodworth. "The Influence of Improvement in One Mental Function upon the Efficiency of Other Functions.(I)." In: *Psychological review* 8.3 (1901), p. 247.

[Vah+17] J. Vahrenhold, M. Caspersen, G. Berry, J. Gal-Ezer, M. Kölling, A. McGettrick, E. Nardelli, C. Pereira, and M. Westermeier. *Informatics Education in Europe: Are We All In The Same Boat?* Tech. rep. New York, NY, USA: ACM and Informatics Europe, 2017.

[VAW14] A. Vihavainen, J. Airaksinen, and C. Watson. "A systematic review of approaches for teaching introductory programming and their influence on success". In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*. ICER '14. Glasgow, Scotland, United Kingdom: Association for Computing Machinery, 2014, pp. 19–26. ISBN: 9781450327558. DOI: 10.1145/2632320.2632349.

[Ver23] C. Verstraete. "Increasing the diversity of feedback types in a CS1 course". MA thesis. UCL - Ecole polytechnique de Louvain, 2023.

[VKK03] J. J. G. Van Merrienboer, P. A. Kirschner, and L. Kester. "Taking the Load Off a Learner's Mind: Instructional Design for Complex Learning". In: *Educational Psychologist* 38.1 (Jan. 2003), pp. 5–13. ISSN: 0046-1520, 1532-6985. DOI: 10.1207/s15326985ep3801_2.

[vPS10] T. van Gog, F. Paas, and J. Sweller. "Cognitive Load Theory: Advances in Research on Worked Examples, Animations, and Cognitive Load Measurement". In: *Educational Psychology Review* 22.4 (Dec. 2010), pp. 375–378. ISSN: 1573-336x. DOI: 10.1007/s10648-010-9145-4.

[VS07] V. Vainio and J. Sajaniemi. "Factors in novice programmers' poor tracing skills". In: *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '07. Dundee, Scotland: Association for Computing Machinery, 2007, pp. 236–240. ISBN: 9781595936103. DOI: 10.1145/1268784.1268853.

[WH17] D. Weintrop and N. Holbert. "From Blocks to Text and Back: Programming Patterns in a Dual-Modality Environment". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*. Seattle, Wash-

ington, USA: ACM Press, 2017, pp. 633–638. ISBN: 978-1-4503-4698-6. DOI: 10.1145/3017680.3017707.

[Whe+19] L. B. Wheeler, J. L. Chiu, J. L. Maeng, and R. L. Bell. "An Exploratory Study of Teaching Assistants' Motivation for Inquiry-Based Teaching in an Undergraduate Laboratory Context". In: *Chemistry Education Research and Practice* 20.1 (Jan. 2019), pp. 53–67. ISSN: 1756-1108. DOI: 10.1039/c8rp00157j.

[WS90] M. Ward and J. Sweller. "Structuring effective worked examples". In: *Cognition and instruction* 7.1 (1990), pp. 1–39.

[WW19] D. Weintrop and U. Wilensky. "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms". In: *Computers & Education* 142 (Dec. 2019), p. 103646. ISSN: 0360-1315. DOI: 10.1016/j.compedu.2019.103646.

[XNK18] B. Xie, G. L. Nelson, and A. J. Ko. "An Explicit Strategy to Scaffold Novice Program Tracing". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 344–349. ISBN: 9781450351034. DOI: 10.1145/3159450.3159527.