Teaching programming for secondary school : a pedagogical content knowledge based approach

Teaching Programming for Secondary School: a Pedagogical Content Knowledge Based Approach

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen
op donderdag 2 februari 2012 om 16.00 uur

door

Mara Saeli

geboren te Messina, Italië

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. W.M.G. Jochems
en
prof.dr. G. Zwaneveld

Copromotor:
dr. J. Perrenet

This research was financially supported by Eindhoven School of Education and the Interuniversitaire Commissie Lerarenopleiding, The Netherlands.

This thesis was prepared with the LaTeX documentation system.
Cover Image: Mara Meo ©
Cover Design: Mara Meo
Printed by: Printserivce TU/*e*
A catalogue record is available from the Eindhoven University of Technology Library.

Teaching Programming for Secondary School: a Pedagogical Content Knowledge Based Approach / by Mara Saeli. – Eindhoven : Technische Universiteit Eindhoven, 2012
ISBN: 978-90-386-3084-7

You don't understand
Stanislav Kovalenko

Aknowledgment

I would like to thank you all, but some specific thanks...

A first thank is reserved to Prof. Wim Jochems and Prof. Bert Zwaneveld: to have assured everything was actually scientifically good and to provide me with the right framework to professionally grow. I would like also to thank Prof. Wim Jochems to dig problems out and to insist I would not quit the PhD in my most difficult period in Eindhoven.
I would like to thank dr. Jacob Perrenet, my supervisor, who has given me the right support to become an independent researcher and with whom I have had numerous interesting discussions.
I would like to thank Mary Beth Key (again) who, thanks to her charming and convincing way, she let me apply to the interview that lead to this, surprisingly, phd position (4 years, the Netherlands again!!!).
A special thank is devoted to my room-colleagues, Lesley and Migchiel, for never giving up in offering *iets uit de automaat*, to accept and at times encourage personalization and ergonomization of the *kantoor-ruimte* (e.g. giant balls, inflatable couches, big cushions), and to provide unexpected entertainment.
Thanks to Martina, for never giving up in talking in Dutch to me, though I made no sense when attempting to reply back in her native language. She kept on nodding all this time and managed to laugh with incredible timing.

All my friends who made my stay in Eindhoven really special. Among them: Lesley and the alternative working place (bouncing balls included), letting me experience the Netherlands from the inside and to always be a source of limitless suggestions and tips; Lorenzo i 5ini, le foto e le serate Eindhovenesi piene di sorprese e le piccole avventure; Agnese per aver affrontato con tanto coraggio i miei multipli esperimenti culinari; Ciccio e i suoi frullati, ma soprattutto per avere fornito innumerevoli fonti di risate e di scoppientante

materiale musicale per riempire le lunghe ore di digitalizzazione; Alessandro per darmi l'opportunita' di condividere la passione per gli aspetti nutrizionali e la cinofilia; Mamoun, to let me experience the *design vegetables* and to bring back memoirs as the inebriation of the first drive on a scooter; Jovana to have shared with me so many Georges and nice chats; Dario per le tue infinite traduzioni dei testi piu' impossibili, e per avere infuso il coraggio di contattare lui, il mitico Leo RatDad; Pina per la tua pentola gigante, le buste, i lasciti e soprattutto per le sveglie mattidine seguite dalle bussate; and thanks to Simon, to have introduced me to a new dimension of music, to have let me discover different approaches and for the multiple creatures.
Lana, I would like to thank you to be always there in the highest peaks of stress and to keep an eye on my little achievements.
Sushi, thank you to be so calming and entertaining.

Last but not least, i miei genitori, Angelo e Angela, e mia sorella Gaia. Devo a loro un supporto incondizionale, per darmi coraggio nei momenti di luna calante, per sopportare i miei sfoghi e per far si che i miei progetti si realizzassero. A loro e alla famiglia di Bruxelles, Gina e Gaetano Irato, devo la prontezza ad atterrare ad Eindhoven quando si sono presentate situazioni di emergenza. And Simon Shelley, my husband, to have had the patience to listen and to provide the fuel for the necessary creativity to distract my mind from this work. To have inspired me for the cover and to have provided intellectual, scientific and linguistic support to the extreme... and for the terrific amounts of white tea and delicacies.

Contents

1 Introduction **1**
1.1 Background to the study 1
1.2 Theoretical background 8
1.3 Definition of the problem and research questions 9
1.4 Structure of the dissertation 11

2 Literature Review **15**
2.1 Introduction 16
2.2 Programming Education 17
2.3 Pedagogical Content Knowledge 18
2.4 Methods and Aims 20
2.5 PCK of programming 21
2.6 Conclusions and Implications 29

3 Protraying PCK **31**
3.1 Introduction 32
3.2 Method 33
3.3 Results 38
3.4 Conclusions and discussion 46

4 Measuring the PCK of Textbooks **53**
4.1 Introduction 54
4.2 Methods 61
4.3 Results 69
4.4 Conclusions and Implication 73

5 Measuring Teachers' PCK **77**
5.1 Introduction 78
5.2 Methods 84

5.3 Results 92
5.4 Conclusions and Discussion 101

6 Conclusions **109**
6.1 Overview of the research 109
6.2 Summary of the outcomes 111
6.3 Critical reflections 117
6.4 Practical Implications 122
6.5 Suggestions for further research 123

References **127**

Appendices **137**
Appendix A - Sub-Domain Software 137
Appendix B - The OTPA 139
Appendix C - PCK about Algorithms 143

Summary **147**

Samenvatting **153**

Curriculum Vitae **159**

List of Pubblications **161**

Eindhoven School of Education **163**

Chapter 1

Introduction

1.1 Background to the study

New generation students, as well as other citizens, are placed in a society in which computers play an almost ubiquitous role. Students need to learn the world of computers, because even if their major interest resides in other disciplines, such as economics, arts or literature, they will most probably need to use computers and computer programs (Stephenson, Gal-Ezer, Haberman, & Verno, 2005). Computer Science (CS) is a relatively young discipline and still its definition is considered by some to be under construction. McGuffee (McGuffee, 2000) explored the different definitions proposed for this discipline in terms of its goal, such as the ones that follow. Dijkstra pointed out that "...the core challenge for computer science is hence a conceptual one: what (abstract) mechanism we can conceive without getting lost in complexities of our own making" (Dijkstra, 1987); while Long and colleagues argued that the role of CS is "the study and application of languages and methods for making precise and understandable descriptions of things" (Long et al., 1997). This discipline has also been defined, as a result of a task force effort to outline suggestions for a curriculum for computer science for secondary school, to be as "an integrated field of study that draws its foundations from mathematics, science and engineering" (Roberts, Shackelford, LeBlanc, & Denning, 1999) implying that whenever these three disciplines intersect you have computer science. A previous definition by the same task force (Comer et al., 1989)

states that CS is "the systematic study of algorithmic processes - their theory, analysis, design, efficiency, implementation, and application - that describes and transform information". Interestingly the task force included in its first definition the concept of 'information', abandoned then in their second definition. A recent definition (Baeten, 2009) defines CS as the study of discrete behaviour of interacting information processing agents. Information is considered by many as the core concept of CS, in terms of information transformation and processing. In other words, CS seems to still struggle for a clear definition that can describe it. This situation might suggest that teachers, students, headmasters, curriculum authors and others might have troubles in picturing this discipline. In the past most people would think that only computer enthusiasts were able to deal with computers. Nowadays, the advent of new 'smart' mobile phones and music players changed people's attitude and understating of computers, and educators' attitude for their use in classroom. An example is Abelson's note:

> "If your phone is going to be an influential force in your life, then you should be able to shape it to suit your needs whether or not you have a degree in computer science or electrical engineering." Description label posted at the MIT museum (MIT Museum - Hal Abelson, 2010)

This is just an example of the reasons we need to teach CS in schools. Other reasons are: learning to recognize when, how and why CS can be used to address and solve general problems (develop methods/instruments for concrete problems), viewing the possibilities and limits of CS, and understand the social and ethical aspects of users interacting with IT tools (Van Diepen, Perrenet, & Zwaneveld, 2011).

CS is a relatively young discipline, and Computer Science Education (CSE) is a new subject in the secondary school curriculum as well as in teacher education, in an international perspective as well as in the Dutch situation. Issues with young disciplines are different, as for example lack of agreement on its name or its content. This discipline is at times named as Computer Science, Computing, Informatics, IT (Information Technology) or ITC (Information, Technology and Communication). To underline the interchangeability of some of these terms, in this book both the therms Computer Science and Informatics are used. Also, it is not rare to find courses named as CS, but offering teaching on the use of computers, as for example use of Word, Excel or Powerpoint

(called Computer Literacy). Other problems regard the content to teach, as for example what topics or aspects of this discipline should be taught, or which approach is more suitable than other to facilitate students' learning.

This thesis, being the research based in the Netherlands, deals with the teaching and learning of CS at secondary school in the context of the Dutch scenario. In the Netherlands CS was introduced in secondary school very recently (1998/1999) and is not immune to problems relative to young disciplines. Among the problems outlined, as it will be further analysed, there is the relatively weak position CS covers in secondary education curriculum, where the subject is offered on an elective basis and is not centrally examined, as almost all other secondary school disciplines in the Netherlands. For this reason teachers have almost full freedom to choose which topics and targets to teach. On one hand this is an advantage for teachers, which gives them the opportunity to experiment and explore the subject with their students, on the other hand it gives little control on teaching quality. Another problem of this subject in the Netherlands is the lack of teachers with CS background (Schmidt, 2007b). Most CS Dutch teachers are licensed teachers from other disciplines (e.g. mathematics, physics, art) re-trained to teach CS, or they just teach CS with no special training. Only recently (2006) official Master's in CS teaching are offered in different universities, with the requirement that student teachers have completed at least a bachelor's in CS. More details on the Dutch scenario will follow.

The focus of this book is on the different aspects relative to the teaching at secondary school of one of the CS topics: programming, a topic of international interest. Particularly it is analysed the content and the quality of CS Dutch textbooks, to verify whether they can support teachers' needs, and Dutch teachers' knowledge is assessed, to sketch the actual situation and propose possible improvements.

A brief introduction to the Dutch education system, Dutch CS curriculum and the Dutch educational situation of CSE follows. Next, this chapter provides an overview of the theoretical framework, definition of the problem, research questions and the structure of the dissertation.

1.1.1 The Dutch education system

The Netherlands is a country where the teaching of CS for secondary school has been recently introduced (Grgurina, 2008), in the school year 1998/1999. At the moment CS is an elective course and is examined only at school level, while almost all other disciplines are centrally examined.

Dutch secondary education (Eurydice, 2007) follows on from eight years of compulsory primary education with pupils aging in average 12. There are three kinds of secondary education:

- pre-vocational secondary education (VMBO) which takes four years;
- senior general secondary education (HAVO) which takes five years;
- and pre-university education (VWO) which takes six years.

In the lower years, the emphasis of CSE is on acquiring and applying knowledge and skills, and delivering an integrated curriculum, in other words: computing literacy. Teaching is based on learning objectives which specify the knowledge and skills pupils must acquire. The aims of the upper years of HAVO and VWO are to provide a broad general education and to ensure cohesion between the various subjects and harmonization with the methods used in higher education.

All pupils entering the 4th year (HAVO and VWO) or the 5th year (VWO) have to choose one of the following four subject combinations:

- culture and society, with emphasis on language and history;
- economics and society, with emphasis on mathematics, economics and history;
- science and health, focusing on mathematics, biology and chemistry;

- science and technology, with emphasis on science and mathematics.

The common component to these four themes are: Dutch, English and mathematics (Eurydice, 2007). In addition to these compulsory subjects, there are some optional courses available to all students, one of these being CS.

1.1.2 Computer Science in the Dutch curriculum

CS is one of the elective courses for all mentioned subject combinations offered in the optional component, and for it are devoted 240 study hours for senior secondary education students, and 280 study hours for pre-university education students. Students' requirement to access such course is to have reached computer literacy, which consists in familiarity with common software (e.g. Word, Excel, Power Point, etc). Computer literacy in the Netherlands is taught in the lower grades. The curriculum for CS is divided into four themes (Schmidt, 2007a):

Theme A: Computer Science in perspective : CS is examined on its possible uses and scopes, as for example CS in society, science and technology, education and career perspectives, and from a personal perspective. Students are expected to reach a general overview of CS.

Theme B: Basic concepts and skills : for this theme students are ought to acquire adequate knowledge and skills pertaining to hardware, software, organization, as well as to data, and information and communication.

Theme C: Systems and their structures concerns general information issues, various types of data processing systems, and the situations where these are normally used.

Theme D: Application in context is devoted to practice. Students are introduced to system development and project management, including

their social aspects, information issues the development of IT applications at all kinds of institutions, enterprises and application areas.

These four domains are divided in 53 sub-domains, but the level of depth and understanding that students are supposed to achieve is not specified. A general guideline is that for HAVO students the emphasis should be placed on practical work, while for VWO students the approach should be more abstract and theoretical.

Dutch teachers are granted quite much freedom in following the guidelines for the curriculum, being the level of depth and understanding not really explicitly stated. Also, the subject has no central level examination (while all the major disciplines have), but only at school level. The school exam covers all the topics in the four themes.

1.1.3 Dutch secondary CS teachers

When CS was first introduced at secondary school level in 1999, one of the major concerns was the lack of trained teachers who could teach this subject in an adequate manner and at an adequate level. A solution was offered with the introduction of the so called CODI-traject (Consortium for Retraining Teachers towards Computer Science Education; in Dutch: Consortium Omscholing Docenten Informatica), a training program equivalent to a first year of a bachelor program, with content matter from CS and pedagogy of CS. In the period between 1998 and 2005 teachers wishing to teach CS, with the prerequisite to be computer literate at a sound level and only if licensed teacher in any discipline, could attend this two year course (roughly 45 ECTS) comprising the subjects listed in Table 1.1 (Grgurina, 2008).

From 2006 the first official Master programs of Computer Science Education were offered by five universities. To attend such course, prospective teachers are expected to have completed at least a Bachelor's degree in Computer Science, guaranteeing a solid disciplinary background.

Course	ECTS
Orientation on Informatics	3.5
Computer Architecture and Operating Systems	0.7
Visual programming with Java	5.7
Information Systems: Modeling and Specifying	5
Databases	0.7
Telematics	3.5
Software Engineering	5
Human-Machine Interaction	1.4
Programming Paradigms and Methods of Information System Development	1.4
Didactics of Informatics	5.7
Informatics Projects	2.8
Practical Teaching Assignment	10
TOTAL	45.4

Table 1.1: CODI program

1.1.4 Dutch scenario

At the moment the Netherlands counts on a total of 550 secondary schools around 350 CS teachers, most of whom did not receive a formal teacher training in CS, but attended the in-service CODI course (Schmidt, 2007b). There is also an unknown number of CS teachers who have not attend the CODI course, neither have a CS degree. The common scenario is that there is only one CS teacher per school (in case a school offers CS). Considering the total amount of secondary schools, it means that only around the 65% of schools provide a CS curriculum. The majority (67%) of these teachers in 2007 was 50 years old, or more, meaning towards the end of their career by far. Also the number of new prospective teachers registering at the Master program is not enough, around 10 new students per year in the whole country, to counterpart the national needs (Schmidt, 2008). And from the students completing the training is not certain whether they will all start a career as teachers.

Concluding, the situation of CS at secondary level in the Netherlands presents some weak points such as: a curriculum that does not describe the level of depth and understanding students are expected to attain, a lack of a national examination for this subject to guarantee its quality, a large number of Dutch CS teachers without a formal CS education (e.g. Bachelor's degree) and, in

the near future, an insufficient number of CS teachers to cover the national needs. Taking into consideration the situation described, actions for improvement are sought, such as improving the state of the subject in the curriculum, having the subject examined at central level, better formulated learning objectives, high quality learning materials, *tailor made* actions to support CS Dutch teachers' needs (Van Diepen et al., 2011).

1.2 Theoretical background

One of the ways of improving the teaching of a certain topic is to improve teachers' special knowledge of the content of that topic, which Lee Shulman (1986, 1987) defines as Pedagogical Content Knowledge (PCK). PCK is that expertise that allows teachers to represent, in an effective way, the subject to their students (Shulman, 1986) and is defined as:

> The ways of representing and formulating the subject that make it comprehensible to others (Shulman, 1986, p. 9).

There is a difference between knowing a topic (content knowledge) and being able to teach it. In a more detailed description, when a CS teacher has developed PCK it means that s/he knows 'why to teach CS', 'what CS topics should be taught', 'how CS could be taught' and 'the difficulties students encounter while learning CS' (Grossman & Howey, 1989; Grossman, 1990). This means that PCK is not just merely reformulating a subject, but taking into accounts also other aspects, such as extracurricular knowledge, students' difficulties or other factors influencing the teaching of a topic. Teachers need strong disciplinary background (content knowledge) to assure a full understanding of the subject. To be able to recognize different students' difficulties, misconceptions and learning needs specific to CS, the teacher will also need strong pedagogical knowledge. Also, because the teacher needs to find different forms of reformulation and representation teachers also need several years of teaching experience to be sure s/he'll gather sufficient knowledge about possible scenarios. Summarizing, PCK is that special amalgam of deep content knowledge and deep pedagogical knowledge, which results in a specialized knowledge around a subject that grows with the years of teaching experience, because:

> there are no single most powerful forms of representation, the teacher must have at hand a veritable armamentarium of alternative forms of representation, some of which derive from research whereas others originate in the wisdom of practice. Pedagogical content knowledge also includes an understanding of what makes the learning of specific topics easy or difficult: the conceptions and preconceptions that students of different ages and backgrounds bring with them to the learning of those most frequently taught topics and lessons. If those preconceptions are misconceptions, which they so often are, teachers need knowledge of the strategies most likely to be fruitful in reorganizing the understanding of learners, because those learners are unlikely to appear before them as blank slates (Shulman, 1986, p. 9)

The importance of this construct has been immediately recognized and explored in several disciplines, such as mathematics, science, chemistry, physics, language and also in computer science. (Carpenter, Fennema, Peterson, & Carey, 1988; Loughran, Gunstone, Berry, Milroy, & Mulhall, 2000; Van Driel, Verloop, & Vos, 1998; Grossman & Howey, 1989; Woollard, 2005). Knowledge of the PCK of a subject has several practical advantages, such as allowing scholars to improve teacher training courses, assess the quality of a subject, assess teachers' performance and help those who need support, improve and assess teaching material, and provide guidelines for curriculum and textbook authors. To be able to benefit from the knowledge of this construct in a subject, the first efforts should be deployed to portray such knowledge.

1.3 Definition of the problem and research questions

As introduced earlier, the Netherlands is facing big problems in delivering CS at secondary level. From a survey conducted by Schmidt (Schmidt, 2007b), mostly Dutch CS teachers have weak disciplinary background and quality monitoring of the subject is not achievable, because the subject is not centrally examined. Problems are so complex that Informatics education is at a crossroad, facing what Van Diepen and colleagues call the Dutch dilemma (Van Diepen et al., 2011). Dutch CS risks disappearing from the school curriculum, because no clear solutions to solve these problems are available, despite the efforts made. The approach proposed in this book is an attempt to specify the PCK of CSE, or at least part of it. The concept of PCK is then

used as a framework to assess the Dutch situation and provide local solutions. One reason to choose PCK is that a deep and broad PCK is important and necessary for effective teaching (An, Kulm, & Wu, 2004; Magnusson, Krajcik, & Borko, 1999). Another reason to study the PCK of a subject is to enable teacher training centers to improve their programs (Lapidot & Hazzan, 2003) or help improving the quality of textbooks, in order to provide help to those teachers who are missing a solid CS background. By providing better teaching material and by improving teacher trainer courses it is aimed to offer possible tailor made solutions to the problems described before.

In order to be able to use PCK as a framework to improve a subject, PCK should be first described. As aforementioned, CS is a relatively young discipline, and CSE is a new subject in the secondary school curriculum as well as in teacher education, in an international perspective as well as in the Dutch situation, therefore its PCK is merely undefined and only vaguely described. The first aim of the research described in this book is to understand to what extent the PCK of CS has already been explored, by analysing the literature. The second aim is to bridge the gap evidenced in the literature by portraying the PCK of this young discipline, by inviting experienced teachers with solid disciplinary background take part into this research. From the scenario described above, it is assumed that there are hardly any teachers with developed PCK of CS in the Netherlands, due to the lack of disciplinary background or to the lack of teaching experience, so the need to also include teachers from abroad. Because most Dutch teachers are assumed to have low PCK, interest is also focused on how teaching material can support teachers' developing PCK. Teaching material is the focus of the third study of this book, where knowledge of PCK of CS is used to analyze Dutch textbooks and to evaluate to what extent they can support teachers' developing PCK of CS. It is not implied that textbooks might have PCK, being this construct specific only to teachers, rather it is investigated whether textbooks authors represents the different aspects of PCK in the text. The last aim of this book is to assess Dutch teachers' PCK in order to portray the current scenario and compare it with the results of Schmidt's survey (Schmidt, 2007b), and finally suggest possible *ad hoc* possible solutions to one of the problems outlined: assumed Dutch CS teachers' weak PCK.

In this research project it has been decided to focus on only one of the different topics of CS: programming. Focusing on one topic would allow an in-depth analysis which seems more appropriate than a rather global description of a

number of topics. The reasons for choosing programming is that this is one of the core topics of a CS curriculum in both secondary and higher education, and is considered by many to be a difficult topic to learn and to teach. The term programming is used to refer to that topic through which secondary school students are introduced to computer programming, also present in the Dutch curriculum for CS (Appendix A). No specific programming languages (e.g. Java, Python, etc.) are referred, because these are considered to be means and tools to achieve the learning and teaching of programming. Secondary school students should be taught programming concepts independent of specific applications and programming languages (Stephenson et al., 2005; Szlávi & Zsakó, 2006).

Taking into consideration the different aims described earlier, this study focuses on the following research questions:

1. To what extent is it possible to recognize aspects of Pedagogical Content Knowledge of programming for secondary education in current literature?

2. What is the Pedagogical Content Knowledge of programming in the context of secondary school education?

3. To what extent is it possible to identify the Pedagogical Content Knowledge of programming in Dutch secondary school textbooks?

4. What is Dutch teachers' Pedagogical Content Knowledge of programming for secondary school?

These questions will be elaborated in the next chapters in greater detail.

1.4 Structure of the dissertation

The research project presented in this book consists of five chapters (see Table 1.2 for an overview). First a literature review study reveals to what extent previous studies uncovered the PCK of programming for secondary school.

The results of this study are presented in Chapter 2, with the title "Teaching programming in secondary school: a pedagogical content knowledge perspective". This chapter presents the answer to the first research question. In the second part a research instrument to portray PCK is introduced and the PCK of programming is unveiled. The results, gathered in an international scenario, are presented in Chapter 3, with the title "Portraying the pedagogical content knowledge of programming for secondary school", and describes the answer to the second research question. In following part of this study, the results of the second study are used to develop a research instrument to assess textbooks in respect to the different aspects of PCK that teachers might need in teaching material. The instrument is then used to assess three Dutch textbooks of secondary CS, in terms of PCK. The results of this study are presented in Chapter 4, with the title "Programming: teaching material and Pedagogical Content Knowledge", providing the answer to the third research question. In the last part of this study, the results of the second and third study are used to develop a research instrument to assess Dutch teachers' PCK of programming. The results, presented in Chapter 5, with the title "Programming: teachers and Pedagogical Content Knowledge" are the answer to the fourth research question. The general conclusions and discussion to this research project are examined in the last chapter, Chapter 6.

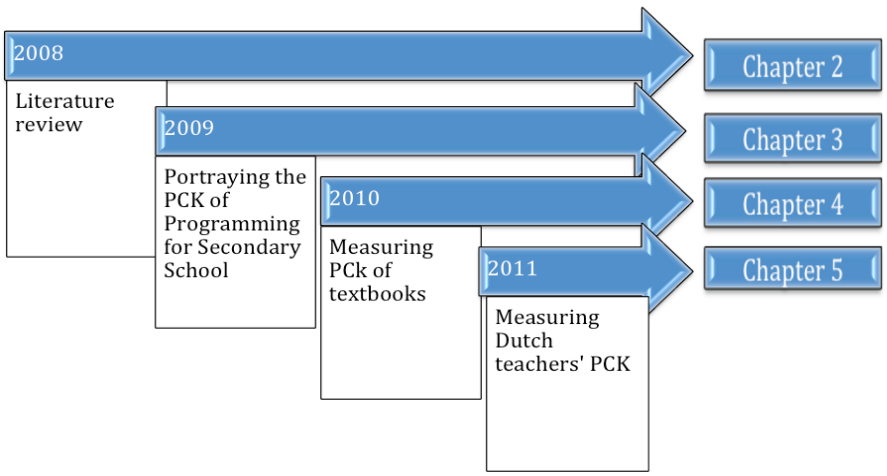

Table 1.2: Schema representing the four parts of research project, in respect of the time span and the chapters of the thesis

Chapter 2

Teaching Programming in Secondary School: a Pedagogical Content Knowledge Perspective [1]

Abstract

The goal of this literature study is to give some preliminary answers to the questions that aim to uncover the Pedagogical Content Knowledge (PCK) of Informatics Education, with focus on programming. PCK has been defined as the knowledge that allows teachers to transform their knowledge of the subject into something accessible for their students. The core questions to uncover this knowledge are: what are the reasons to teach programming; what are the concepts we need to teach programming; what are the most common difficulties/misconceptions students encounter while learning to program; and how to teach this topic. Some of the answers found are, respectively: enhancing students' problem solving skills; programming knowledge and programming

[1] Published as: Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: a Pedagogical Content Knowledge Perspective. *Informatics in Education*, vol. 10, no 1, 73-88.

strategies; general problems of orientation; and possible ideal chains for learning computer programming. Because answers to the four questions are in a way not connected with each other, PCK being an unexplored field in Informatics Education, we need research based efforts to study this field.

2.1 Introduction

The 21st century is characterized by the ubiquitous presence of technology in everyday life. New generation students are surrounded by computer related instruments and will possibly do a job that has not been invented yet. Computing succeeded to conquer most of the aspects of our society and, in order to fit in, people need to be versatile and adaptable to modern and future technology. This scenario emphasizes the need to provide an education that can offer students and future adults the ability to understand and work with computer related instruments. The aim of Computer Science Education Research (CSER) is to improve the quality of the teaching and learning of the topics relative to this computerized world. A review of the literature (Holmboe, McIver, & George, 2001) evidences the existing need to broaden the efforts of informatics educators to contribute to the knowledge of why informatics should be taught at all, how informatics should be taught, what topics of informatics should be taught, and for whom the teaching of informatics is meant. In this literature review we are particularly interested in answering these questions relative to a specific topic of informatics: programming.

In this article we refer with the term of informatics as the computer science education delivered to upper secondary school students (14 to 18 years old). There seems to be no distinction in the use of these two terms in the CSER community.

The answers to the four questions introduced lead to the understanding of the concept called Pedagogical Content Knowledge (PCK) (Shulman, 1986). PCK is a concept that combines the knowledge of the content (e.g. maths, informatics, etc.) to the knowledge of the pedagogy (e.g. how to teach maths, how to teach informatics, etc.), giving insights into educational matters relative to the learning and teaching of a topic. Teachers with good PCK are teachers who can transform their knowledge of the subject into something accessible for the learners. Studies portraying the PCK of programming will enable in-

formatics teacher trainings to improve their programs (Lapidot & Hazzan, 2003), boosting in this way their future teaching. There is evidence of the international interest (Stephenson et al., 2005; Ragonis, Hazzan, & Gal-Ezer, 2010; Woollard, 2005) on this topic, where the first efforts have been made to achieve the goal to portray the PCK of informatics.

PCK is a construct specific to teachers' knowledge; therefore teachers are the focus of this article. There are other aspects of the teaching and learning of programming that are as important as teachers' knowledge. Examples could be gender issues, mostly dealing with motivations that bring boys and girls to enroll in informatics courses; and students' motivation, which is part of general pedagogical knowledge. However, these topics of interest, despite very important, are not the focus of this paper.

2.2 Programming Education

Programming is only one of the topics concerning the teaching of informatics. In the Netherlands, informatics has been defined as a new generation discipline, because it is linked with Mathematics, Physics, Engineering, Linguistics, Philosophy, Psychology, Economy, Business, and Social Sciences in general (Mulder, 2002). If on one hand this complexity results in a relatively difficult job for researchers in this field, on the other it is possible to rely on the research achievements already obtained in the above mentioned disciplines. As Guzdial suggests, the basic mechanisms of human learning haven't changed in the last 50 years (Guzdial, 2004) and we can prevent the reinvention of the wheel by looking at research in education, cognitive science and learning sciences research (Almstrum, Hazzan, Guzdial, & Petre, 2005).

A popular definition is that programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source of code of computer programs (Wikipedia, 2007). We will later see that programming is a much broader topic than that described by the latter definition, as for example the ability to solve a complex problem with a top-down approach. Programming is a skill that is considered hard to learn and even after two years of instruction, the level of programming understanding is low (Kurland, Pea, Clement, & Mawby, 1989). However, if supported by suitable teaching strategies and

tools it can be mastered by pupils to some extent (Papert, 1980).

In this literature study we refer to programming as the topic used to introduce upper secondary school students to computer programming. We will not refer to specific programming languages (e.g. Java, Python, etc.), because we consider these as a mean/tool to achieve the teaching of programming. Secondary school students should be taught programming concepts independent of specific applications and programming languages (Stephenson et al., 2005; Szlávi & Zsakó, 2006).

2.3 Pedagogical Content Knowledge

Pedagogical Content Knowledge (PCK), a concept introduced by (Shulman, 1986, 1987), is defined as:

> The ways of representing and formulating the subject that make it comprehensible to others (Shulman, 1986, p.9).

There is in fact a difference between knowing how to program and being able to teach programming. The classroom, where learning and teaching occur, is a complex environment in which several processes and actions happen. But when talking about PCK a special attention should be spent on students' learning. An aspect of PCK concerns the need teachers have to represent and formulate the subject, so that comprehension can occur. From the literature we know that different learners have different learning styles (Rayner & Riding, 1997), and needs. This implies that:

> there are no single most powerful forms of representation, the teacher must have at hand a veritable armamentarium of alternative forms of representation, some of which derive from research whereas others originate in the wisdom of practice. Pedagogical content knowledge also includes an understanding of what makes the learning of specific topics easy or difficult: the conceptions

> and preconceptions that students of different ages and backgrounds bring with them to the learning of those most frequently taught topics and lessons. If those preconceptions are misconceptions, which they so often are, teachers need knowledge of the strategies most likely to be fruitful in reorganizing the understanding of learners, because those learners are unlikely to appear before them as blank slates (Shulman, 1986, p. 9).

An example in informatics could be the teachers' knowledge about the concept of programming structures, and the need to formulate their knowledge in a way that can be easily understood by their students. All research in this domain agrees on claiming that PCK is a knowledge that develops with years of teaching experience (Rovegno, 1992; Grossman, 1990; Loughran, Milroy, Berry, Gunstone, & Mulhall, 2001; Morine-Dershimer & Kent, 1999; Van Driel et al., 1998; Sanders, 1993), because teachers need to build up "a veritable armamentarium" of representations (Shulman, 1986).

The concept of PCK has been largely assimilated in educational research (Carpenter et al., 1988; Cochran, DeRuiter, & King, 1993; Van Driel et al., 1998; Peterson, Fennema, Carpenter, & Loef, 1989; Rich, 1993; Rovegno, 1992; Sanders, 1993) and some scholars have reformulated it (Grossman & Howey, 1989; Grossman, 1990; Hashweh, 2005; Marks, 1990; An et al., 2004; Turner-Bisset, 1999). A deep and broad PCK is important and necessary for effective teaching (An et al., 2004; Magnusson et al., 1999). Moreover, Hashweh (Hashweh, 2005) underlines how the teacher's approach or orientation to his or her discipline (personal beliefs) influences the teaching of a certain topic, and might influence her/his PCK. This means that each teacher's PCK is in a way personal.

For the purpose of this literature study we will use the reformulation of the concept of PCK proposed by Grossman (1989, 1990). We choose her reformulation because we think that it schematizes the PCK through simple and easy to use questions such as: why teach a certain subject?; what should be taught?; and what are learning difficulties?. In our study we add a fourth question, which refers to the teaching methodology: how to teach?. This last aspect has been also lately introduced by Grossman (1990). We think that the latter is an important aspect of the PCK of a topic because it gives an insight

of what teachers actually do. The question used to uncover this aspect of PCK is: how should the topic be taught?. By answering these four questions it will be possible to define the PCK of a certain subject. The four questions are all connected with each other, because the reasons to teach a topic (first question) will influence the contents chosen to be included in the curriculum. In addition the learning difficulties that students encounter will surely influence the way to teach it.

2.4 Methods and Aims

This literature study has been conducted by exploring the existent literature available. Sources include printed articles, books chapters and information found on the Internet. The research has been conducted by using searching engines such as Google Scholar, or by browsing references lists of other articles. The keywords used, sometimes in combination with each other, are: "Pedagogical Content Knowledge", "Teachers' education", "Programming (Education)", "Student's misconceptions/difficulties in learning to program", "Secondary education", "Why to teach programming", and "How to teach Programming". The choice of the articles has been mostly dictated by the search of papers published in research journals and presenting research results, as practice in science education research suggest. Also, papers from conference proceedings are considered, where relatively young subjects like informatics find the fastest way to share their results. Despite the call for people who are active in CSER to rely on scientific papers preferably not published in conference proceedings (Randolph, 2007; Lister, 2007), we found that still most of the up-to-date literature is shared through conference papers and therefore the need to rely on them.

The goal of this literature study is to sketch the PCK of programming, not of specific topics (e.g. variables, functions, etc.), but referring to programming as a subject. As instrument we use the framework introduced earlier, which consists of the four questions (see Table 2.1).

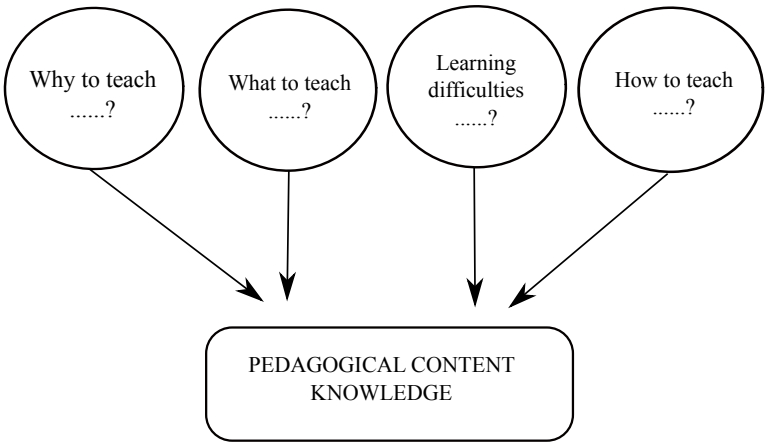

Table 2.1: Diagram based on Grossman's reformulation of PCK.

2.5 PCK of programming

As previously stated, the PCK of a subject is the knowledge that enables researchers and teachers to better understand the issues related to the teaching and learning of the subject, and consequently provide a better teaching. In this section we give preliminary answers to the core questions uncovering the PCK of programming, using the method described above.

2.5.1 Why teach programming?

What are the reasons to teach programming at high school level for non-major students? In a way, this question could also be reformulated as "why should students learn to program at all?". The answer, however, is interesting from a teacher's perspective. If this literature study would focus on students rather than teachers, than the question should be rephrased as "why should I, as student, learn to program?". This question, if properly answered, would also help teachers to motivate students to enrol in informatics courses in a first place. However, this is not the goal of this paper.

Soloway (1993) answers this question by reporting "respected folks' opinions" such as those by Seymour Papert and Alan Kay, arguing that in learning to program one learns powerful problem-solving/design/thinking strategies.

This is because when students program, they first need to find a solution to a problem, and then they need to reflect on how to communicate their solution to the machine, using syntax and grammar, through an exact way of thinking (Papert, 1980; Szlávi & Zsakó, 2006). The latter contributes to the students' natural language skills, because they are required to learn to tell, in an unambiguously way, what they want the computer - an unintelligent machine - to perform (Hromkovič, 2006). Programming involves the ability to generate a solution to a problem. Generating solutions means that one of the learning outcomes is the ability to solve problems and also, if the problem is a big problem, the ability to split the problem into sub problems and create a generalizable central solution. In addition, the student achieves the ability to create usable, readable and attractive solutions.

Problem solving skills can be deployed to solve 'realistic' problems in various domains together with the computing goals (Sims-Knight & Upchurch, 1993; Dagiené, 2005). Transferability of these and other skills is the argument that brought Feurzeig and his colleagues (Feurzeig, Papert, Bloom, Grant, & Solomon, 1970) to introduce programming as a way to help students to understand mathematics concepts such as: rigorous thinking, variable, function, decomposition, debugging and generalization. Syslo and Kwiatkowska (2006) went further by exploring those mathematics concepts that can benefice from programming, but which still have to be included in the (Polish) secondary school curriculum. Besides transferability of skills, when learning to program students also acquires a sense of mastery over a technological instrument and establishes contact with some of the deepest ideas of different disciplines such as: science, mathematics and the art of intellectual model building (Papert, 1980). Moreover, as we anticipated earlier on, programming is a new generation subject, which brings together pieces from different areas such as: linguistics, mathematics and economics (Mulder, 2002). This completeness gives students the opportunity to be faced with a multi-disciplinary subject that connects different aspects in a single class. Students could experience the opportunity to delve deeper into previously acquired knowledge, as for example Resnick and Ocko's students (1990) did with the physics concept of friction.

2.5.2 What should be taught?

By answering this question we aim to understand what are the core concepts of programming students need to learn. Decisions about what it is needed

to teach are usually taken by curriculum and examinations designers. In informatics efforts to define a suitable curriculum have been made since the late '60s (Atchison et al., 1968). However we should consider the different curricular representations (Akker & Voogt, 1994) including: the ideal curriculum, which refers the original ideas and intentions of the designers; the formal curriculum, denoting the written curriculum (documents, materials); the perceived curriculum, indicating the interpretation of the users, especially the teachers, of the curriculum; the operational curriculum, identifying the actual instruction process in the classroom; and the experiential curriculum, which represents students' reactions and outcomes. In this literature study we combine topics suggested from the ideal, the formal and the perceived curriculum (e.g. Gal-Ezer & Harel, 1999; Tucker et al., 2003; UNESCO, 2002; Tucker, 2010), because we think that these together form a more complete and realistic view of what happens in a class.

Rephrasing Romeike (2008), the core of programming is all about problem solving and creating a program as solution. In programming we can distinguish two kinds of knowledge, namely the program generation and the program comprehension (Van Merriënboer & Paas, 1987; Robins, Rountree, & Rountree, 2003; Mannila, 2007). In the first case, the programmer analyses the problem, produces and algorithmic solution, and then translates this algorithm into a program code. This means that students should be coached in the process of problem solving, reflection on this process, and in the development of algorithmic ways of thinking (Feurzeig et al., 1970; Resnick & Ocko, 1990; Sims-Knight & Upchurch, 1993; Dagiené, 2005; Breed, Monteith, & Mentz, 2005; Hromkovič, 2006; Futschek, 2006; Ginat, 2006). As for program comprehension, the programmer is asked to give a demonstration of her/his understanding of how a given program works. We consider then the teaching in secondary school of program generation and program comprehension very important.

Programs are a set of instructions that computers execute in order to perform a task and are written in a programming language. Usually curriculum designers leave the choice of the programming language to teachers, and among secondary teachers there seems to be heterogeneity in the choice of programming languages/paradigms. In the process of learning to program, Govender (2006) identifies, from a technical point of view, three main aspects students need to learn: data, instructions and syntax. Data refers to the concepts of

variables and data types for procedural programming, and objects involving attributes and actions for OO programming. As for instructions, the needed understanding is about control structures and subroutines for the procedural programming, and interacting objects and methods in the case of OO programming. Syntax denotes the group of rules that determine what is allowed and what is not within a programming language. Syntax rules determine what it is called the vocabulary of the language, how programs can be constructed using techniques such as loops, branches and subroutines. Govender's classification, however, does not take care of the modularity and abstraction aspects of programming, as for example Abelson and Sussman (1996) do. They identify three main aspects: primitive expressions, representing the simplest entities that a language is concerned with; means of combination, by which compound elements are built from simpler ones; and means of abstraction, by which compound elements can be named and manipulated as units. These three aspects deal with two kinds of elements: data and instructions. By using these three mechanisms in combination with each other it is possible to formulate complex programs, starting from simpler ones.

A final aspect, equally important, is the semantic of a program, also referred to as the meaning of a program. A semantically correct program is a program that performs the required task. Programs written with different syntax can perform the same semantic task.

2.5.3 What are the learning difficulties?

In this section we deal with students' different needs and difficulties. Because of the complexity of individuals, different students will have different needs and difficulties. For this reason some of the studies presented might result contradictory, but in fact they present the different realities of different students.

It has been stated several times that programming is a difficult task to achieve (Van Diepen, 2005; Govender, 2006) and often novice programmers hold misunderstanding and misconceptions. In early stages of the learning process a correct program often results as an unexpected surprise (DuBoulay, O'Shea, & Monk, 1989). By answering this question we aim to understand the most

common problems students have while learning to program. From this knowledge we can attempt to find solutions to prevent or guide these problems. A brief exploration of the most common problems is given.

DuBoulay (1989, p. 283-284) identifies five kinds of difficulties/areas which have a certain degree of overlap in programming learning/teaching, concerning aspects such as motivation and technical aspects. Students' difficulties are: 1) orientation, finding out what programming is useful for and what the benefits to learn to program are; 2) the notional machine (understanding the general properties of the machine that one is learning to control) and realizing how the behaviour of the physical machine relates to the notional machine; 3) notation, which includes the problems of aspects of the various formal languages such as syntax and semantics; 4) structures, understanding the schemas or plans that can be used to reach small-scale goals (e.g., using a loop); 5) mastering the pragmatics of programming (learning the skill to specify, develop, test and debug a program using the available tools).

From a relationship student-computer point of view, Pea (1986) identifies the existence of persistent conceptual language-independent 'bugs' in how novices program and understand programs. The starting point of the analysis of conceptual 'bugs' is that students have a tendency to converse with a computer as if it was a human (considered also as the superbug), with consequences such as expecting the computer to interpret students' conversations. Pea distinguishes three different kind of conceptual 'bugs': the parallelism bug refers to the assumption that different lines in a program can be active or somehow known by the computer at the same time, or in parallel. Another bug is the intentionality, for which students believe that computers "go beyond the information given" in the lines of programming code being executed when the program is run. The last bug, egocentrism, refers to students' assumption that there is more of their meaning for what they want to achieve in the program than is actually present in the code they have written (e.g. "Don't print what I say, print what I mean!"). Students' conceptions do not guide their attention to consider these problems as relevant reasons for their programs not to work as planned.

Another problem students could face is the paradigm shift (Kölling, 1999b, 1999a; Mazaitis, 1993) in cases where their teacher proposes them to learn more than one programming language with different paradigms (e.g. proce-

dural and object oriented), although this is not advisable in an introductory course at secondary school level. Students usually encounter problems in passing from one paradigm to another, especially from the procedural to the object oriented (but not the vice versa).

Regarding the acquisition of problem solving skills, several papers explore the different difficulties students encounter while trying to generate a solution for a given problem. Novices (Ginat, 2006) tend to maintain local, limited-insight points of view of the problem, leading often to undesirable, erroneous outcomes. It seems that novices fail to realize the importance of a global point of view. Also Weigend (2006) observed how, even when finding a mental or practical solution to a problem, students fail to write a correct program that does the job. The reason might be that students are not trained to translate mental intuitions in a communicative way, or might be connected with the semantic of a program. Semantic is also considered to be a problematic aspect of programming. This is because it requires the student to put together different parts of a program (variables, expressions, statements, control structure, objects and methods) into a working solution. Semantic is closely related to the debugging activity and the related correctness of a program (Pea & Kurland, 1983), a concept introduced by Dijkstra (1968, 1972). When teachers choose a programming language offering a more complex syntax, students will be faced with both semantic and syntax difficulties (Mannila, Peltomäki, & Salakoski, 2006).

2.5.4 How should the topic be taught?

By answering this question we aim to understand what the best approaches to introduce students into the learning of programming are, not only to prevent the above mentioned difficulties/misconceptions, but also to hook students' motivation in an effective and engaging way. As in the previous section, because of the complexity of individuals, different students will have different needs and difficulties. For this reason some of the studies presented might report contradictory results, but actually they propose different teaching approaches to meet different students' learning needs. We cannot conclude which one method is the best, but only highlight those methods which are considered best in different circumstances. This is also directly connected with Shulman's definition, which considers PCK as an armamentarium/repertoire of represen-

tations.

Hromović (2006) suggests that programming is seen as a skill to communicate, in an unambiguously way, a set of instructions to an unintelligent computer. If this process could take place by means of a relatively simple programming language (e.g. Python) offering a simpler syntax than other commonly used programming languages, students could focus more on the semantic aspect of the program and produce fewer syntax errors (Mannila et al., 2006). Another way to start this learning process could be the use of practical examples, such as rewriting recipes for cooking for a cooking machine (Hromkovič, 2006). The process should lead students to write at first simple programs, and then combine the simple solutions together to obtain solution to more complicated problems (Abelson & Sussman, 1996). This approach has the twofold purpose to let the student not only experience the historical development, but also learn the concepts of modularity and reusability. Writing a set of instructions to solve a problem is the definition of algorithm. In other words, writing code for a correct mental solution. To achieve algorithmic thinking students should solve many problems, which should be chosen independently from any programming language (Futschek, 2006), and should follow some pedagogical principles (Romeike, 2008). In fact, algorithmic thinking can be successfully introduced without the aid of a computer at all (Bell, Witten, & Fellows, 1998; Curzon & McOwan, 2008). However, it happens that students fail to translate their correct reasoning into an unambiguous set of instructions for the machine. To overcome this, students could be coached in analysing their intuitions and connecting them to the designated task (Weigend, 2006).

Linn and Dalbey (Linn & Dalbey, 1989, p. 58-62) suggest an ideal chain for learning computer programming, which gradually goes from program comprehension and ends with program generation. The chain has three main links: single language features, design skills, and general problem-solving skills. According to Linn and Dalbey (1989) the ideal chain should start with the understanding of the language features, knowledge that can be assessed by asking students to reformulate or change a language feature in a program so that the program does something slightly different. The second link of the chain consists of design skills, which are a group of techniques used to combine language features to form a program. This chain link also includes templates (stereotypical patterns of code that use more than a single feature) and procedural skills (used to combine templates and language features in order to solve

new problems, including planning, testing and reformulating). The third link of the chain, problem-solving skills, is useful for learning new formal systems. Problem-solving skills can be assessed by asking students to solve problems using an unfamiliar formal system such as a new programming language. Though this chain of cognitive accomplishment requires an extensive amount of time it forms a good summary of what could be meant by deep learning in introductory programming (Robins et al., 2003).

To provide novices with a framework for understanding, some model or description of the machine should be introduced, where a model should be designed around each group of novices, distinguished either for their age, background or kind of studies (DuBoulay, 1989). Students working with such models excelled at solving some kind of problem more than students without the model (Mayer, 1989). An example could be the metaphor of a black box inside the glass box as a way to present computing concepts to novices. The reason is that novices start programming with very little idea of the properties of the notional machine implied by the language they are learning.

The previous approaches mostly deal with the difficulties and misconceptions presented in the previous section. If we look at approaches which aim at teaching programming in an engaging way, we should refer to the family of programming environments and suited programming languages developed with the main goal to introduce students into the programming practice in active and motivating scenarios. These environments have been specially designed to answer the difficulties students usually encounter when learning to program with normal programming languages (Mannila et al., 2006). The list is quite long and the first efforts have been already made in the early '70s. Among the most popular we have Logo and its derivates (Feurzeig et al., 1970; Papert, 1980; Resnick & Ocko, 1990), initially designed to teach mathematics, which has the focus to enhance problem solving skills; Scratch (Resnick et al., 2009) which, based on a metaphor of building bricks and offering much of the same functionality as Logo, allows students to create syntactically correct program, and leaves the students to focus on the semantic aspect; and finally the more modern Alice, Greenfoot and Gamemaker (respectively Cooper, Dann, & Pausch, 2003; Kölling & Henriksen, 2005; Overmars, 2004). These learning environments find their basis in Piaget's model of children's learning, where students are fostered to build their own intellectual structures, if provided with the right material. It is then the teacher's task to find suitable sup-

port/stimuli/learning material to use with each of these tools. Some of these languages and environments, however, might not include some structures or topics important to the learning of programming (Murnane & McDougall, 2006).

2.6 Conclusions and Implications

In the previous section we gave the first preliminary answers to the questions that aim to uncover the PCK of programming. The first question aims to understand what the reasons to teach programming are. Preliminary answers to our first question are the following: enhancing students' problem solving skills and offering the students a subject, which includes aspects of different disciplines; use of modularity and transferability of the knowledge/skills; and the opportunity to work with a multi-disciplinary subject.
The second question aims to list the concepts/aspects that a programming curriculum should include. Preliminary answers point to the following concepts/aspects: programming knowledge, which refers to the knowledge of the data, instructions and syntax of a programming language, but also primitive expression, means of combination and means of abstraction; programming strategies, which identify the way syntax is used to create programs to solve problems; and programming sustainability, which refers to the ability to create user friendly and attractive program/software that takes care of ethical and privacy issues.
The third question aims to answer issues relative to the various difficulties students encounter while learning to program, such as a general problem of orientation; difficulty to instruct the machine about the solution of a problem; and tendency to converse with a computer as if it was a human. Regarding the solution of a problem, students tend to maintain a local, limited point of view, failing to find a suitable solution.
The fourth question addresses these difficulties, by discussing teaching methods such as possible and effective teaching sequences; offering a simple programming language so students can focus on the syntax; choosing several problems to solve, which should be carefully chosen, independently from any programming language, in order to achieve algorithmic thinking; and teaching by means of suited programming languages or programming environments.

In most of the cases these four 'answers' are not connected with each other, because no explicit attempt to uncover the PCK of programming has been done before, neither on higher or secondary education. The task in portraying the PCK of programming will be to find the answers not only from a general point of view (programming in general), but from the perspective of each of the most frequently taught topics, which are at the heart of learning to program (e.g. variables, functions, etc.). An example will be answering the four questions regarding the teaching of problem solving skills. Despite the fact that some of these answers are available for some concepts, most have still to be studied. Therefore we propose a call for research to portray the PCK of the most commonly taught programming topics.

This literature study constitutes the first phase of a PhD project, which is still in progress. In the second phase it will be attempted to uncover the PCK of programming for secondary education from an international perspective, through the use of research instruments already deployed in other subjects (Loughran et al., 2001). In the third phase an instrument will be developed to assess to what extent an informatics textbook can support teachers with low PCK. While the fourth phase of the project will consist of the formulation of an approach to assess Dutch teachers' PCK. The results of this project will be used to improve teacher training for the subject of programming.

Chapter 3

Portraying the Pedagogical Content Knowledge of Programming for Secondary School [1]

Abstract

In this article we present the results of an exploratory study to portray teachers' Pedagogical Content Knowledge (PCK) of programming for secondary education. PCK is teachers' knowledge about teaching and learning gained through disciplinary and pedagogical training and practice. Our research questions are: What is the PCK of programming in the context of secondary school education?; and is there a difference between teachers' PCK as portrayed in this study and the teaching theories and teaching practice found in the literature? Data was collected in four countries: Italy, Belgium, Lithuania and the Netherlands, using semi-structured group interviews. In terms of the first research question, the results constitute the first effort to portray the PCK

[1] This chapter has been submitted for publication as: Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. Portraying the Pedagogical Content Knowledge of Programming for Secondary School

of programming for secondary education regarding the following seven topics: control structures (with focus on loops), decomposition of the problem, problem-solving skills, parameters, algorithms, data structures and arrays. For each topic we revealed information such as the reasons for teaching that topic, students' required prior knowledge and difficulties, and teaching methods. Our results were partly confirmed by literature about teaching theories and practical advice, but also complemented the literature.

3.1 Introduction

In recent decades the topic of programming has been introduced in secondary school curricula in several countries. Because this is a relatively new topic, efforts are being made to provide research-based support to improve the understanding of its teaching and learning. One way to do this is by studying the Pedagogical Content Knowledge (PCK) of this subject. PCK has been defined as the possible ways to represent and formulate a subject that make it comprehensible to others (Shulman, 1986). PCK is the knowledge about the teaching and learning of a subject that teachers gain through their disciplinary and pedagogical training, and moreover through their teaching experience. Deep and broad PCK is important and necessary for effective teaching (An et al., 2004; Magnusson et al., 1999). One of the reasons to study the PCK of a subject is to enable teacher training centers to improve their programs (Lapidot & Hazzan, 2003).

In previous work (Saeli, Perrenet, Jochems, & Zwaneveld, 2011c) a literature study to sketch the PCK of programming for secondary school present in the literature has been conducted. The theoretical framework used is Grossman's reformulation of PCK (Grossman & Howey, 1989; Grossman, 1990), in which PCK is the answer to the following four questions: why to teach a certain subject?; what should be taught?; what about students?; and how to teach a certain subject? It was found that in most cases these four 'answers' are not connected with each other, because no explicit attempt to uncover the PCK of programming has been made before, neither on higher nor on secondary education, although the construct has already been introduced in computer science education research (Woollard, 2005; Ragonis & Hazzan, 2008). The task in portraying the PCK of programming will be to find the answers not only from a general point of view (programming in general), but from the per-

spective of each of the most frequently taught topics, which are at the heart of learning to program (e.g. variables, functions, etc.). An example will be answering the four questions regarding the teaching of problem-solving skills, or algorithmic thinking.

The goal of this study is to portray the PCK of programming for secondary school with the aim of bridging the gaps identified in the previous work (Saeli et al., 2011c). This leads us to a standard knowledge of the PCK for programming for secondary school. We also aim to indicate possible differences between teaching theories in the literature and teaching practices. The research questions are therefore:

- What is the Pedagogical Content Knowledge of programming in the context of secondary school education?
- Is there a difference between teachers' Pedagogical Content Knowledge portrayed in this study and the teaching theories and practical advice found in the literature?

In this literature study we refer to programming as the topic used to introduce upper secondary school students to computer programming. We do not refer to specific programming languages (e.g. Java, Python, etc.), because we consider these as a means/tool to achieve the teaching of programming. Secondary school students should be taught programming concepts independently of specific applications and programming languages (Stephenson et al., 2005; Szlávi & Zsakó, 2006).

3.2 Method

To answer the first research question and therefore to portray the PCK of programming for secondary school we use the research instrument described below. To answer the second research question, the results obtained through the research instrument are categorized and then compared with the teaching theories found in the literature.

3.2.1 Instrument

The research instrument used for this study is an adaptation of CoRe (Content Representation), and is designed to portray the PCK of science for secondary school (Loughran et al., 2000, 2001; Mulhall, Berry, & Loughran, 2003; Loughran, Mulhall, & Berry, 2004). CoRe is an instrument to capture the PCK of a specific topic, and gives a narrative account providing an overview of how teachers approach the teaching of the whole of a topic together with the reasons for that approach in the form of propositions. CoRe is both a research tool for accessing teachers' understanding of the content as well as a way of representing this knowledge, and is used by the researcher for schematic representation of a number of teachers' inputs.

As already stated, the research instrument is used to capture the PCK of a specific topic, in which the specific topic in the context of CoRes is referred to as the 'Big Idea' (for example Mulhall et al., 2003). The instrument consists of eight questions (Loughran et al., 2004) as listed below for each Big Idea explored.

The research instrument involves the following eight questions to be answered by expert teachers concerning a specific Big Idea:

1. What do you intend the students to learn about this Big Idea?
2. Why is it important for the students to know this Big Idea?
3. What else do you know about this Big Idea (and you don't intend students to know yet)?
4. What are the difficulties/ limitations connected with the teaching of this Big Idea?
5. What do you think students need to know in order for them to learn this Big Idea?
6. Which factors influence your teaching of this Big Idea?
7. What are your teaching methods (any particular reasons for using these to engage with this Big Idea)?

8. What are your specific ways of assessing students' understanding or confusion around this Big Idea?

In the course of this study it was decided to formulate question no. 5 in this way (S. Fincher, personal communication, August 2009), to gain further information from teachers. The question was posed instead of the following, which was originally included in the work by Loughran and colleagues (2001):

5a. Which knowledge about students' thinking influences your teaching of this Big Idea?

The reason for this choice is to be able to gain an important piece of students' knowledge which teachers with low PCK levels may find useful. Owing to time restriction it was decided to keep the total number of CoRe questions to eight.

Because one of the aims of this study is to fill the gaps found in the previous study, in which Grossman's theoretical framework was used, there was a need to regroup the questions of the research instrument. For the context of this study, the questions of the CoRe instrument are categorized to fall into Grossman's PCK reformulation according to the table below (see Table 3.1), which shows that Grossman's questions are answered by grouping together one or more of the CoRe questions.

3.2.2 Participants

Participants involved in this study belong to the category of secondary school teachers of computer science (CS). If necessary, secondary school teacher trainers can also be involved in the data collection process. If there are not enough teachers meeting the criteria in the country where the research is based, participants from other countries can be involved. The requirement to participate in this study is to have an already developed PCK of programming for secondary education. According to Shulman's definition of PCK, this knowledge grows with years of teaching experience, and must be supported by solid subject matter knowledge. For these reasons, teachers invited to take part in the study are required to have at least five years of teaching experience and to have at least a bachelor's degree in computer science.

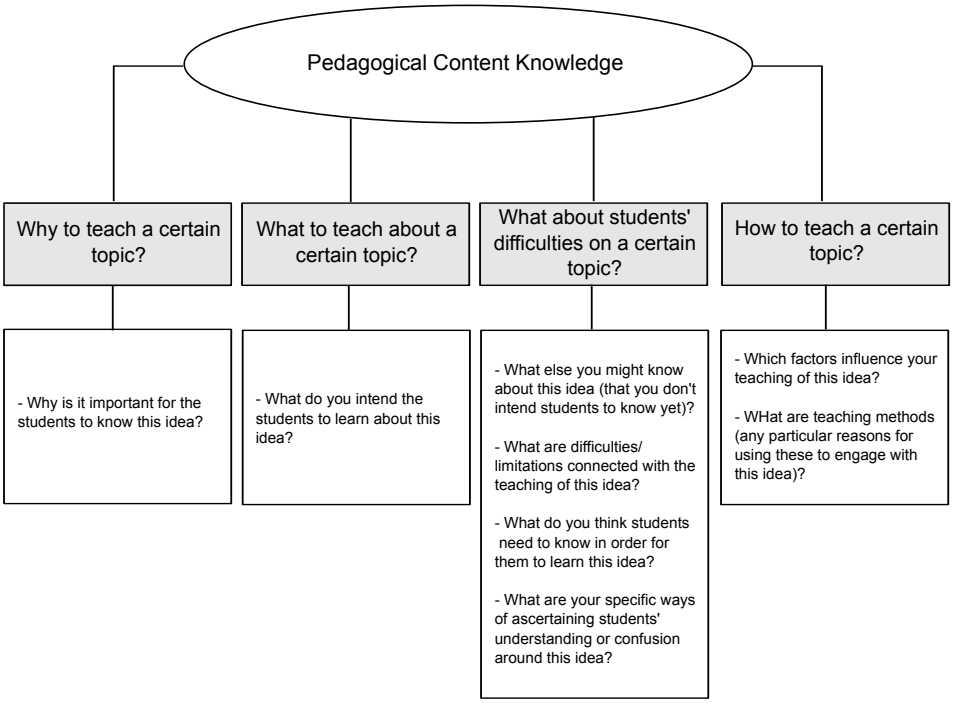

Table 3.1: The eight CoRe questions categorized according to Grossman's reformulation.

3.2.3 Procedure

Taking the point of view that PCK is something that resides in the body of CS teachers as a whole, a group approach is used. In this context teachers are encouraged to think about and to share their knowledge about how to teach particular CS content well, which they have developed through their practice (Loughran et al., 2000).

The data collection method consists of semi-structured group interviews organized in the context of workshops. Workshops have a twofold purpose: to serve as a method to gather data, and to provide in-service training for teachers (Van Driel et al., 1998). The reason to organize workshops is to encourage teachers to participate in a research study, by giving an opportunity of professional development. After running a pilot study to gain familiarity with the instrument, a time schedule to run workshops was defined. Invitations to

experienced teachers are sent, stating that the workshop would last for two hours. Time restriction is necessary to be able to organize group meetings efficiently.

The workshops, involving around five teachers at a time, are videotaped. Each workshop is divided into two parts. In the first part of the workshops, teachers list what in their opinion are the core concepts within a subject which are at the heart of the learning for that specific subject'. They first do this individually, and then in a group discussion. The answers are then called Big Ideas of programming in the context of secondary education. In the second part of the workshops teachers are given an A3 sheet representing the CoRe matrix; they choose one or two of the Big Ideas and then for each Big Idea they answer the eight CoRe questions introduced above (already in Table 3.1), first individually and then in group discussion. Groups are not required to reach consensus, and disagreements in the answers to these questions underline the differences between teachers' views of teaching and how these influence teachers' beliefs and practices. Teachers choose the Big Ideas according to their level of interest, and not the order of importance of the topics. Teachers should when possible be given the opportunity to use their native languages, so they can express their thoughts and ideas better.

3.2.4 Analysis

Each workshop is videotaped and transcribed. The data gathered through videos and teachers' notes is analyzed in order to fill in a group CoRe matrix for each workshop. Teachers' individual CoRes are also taken into consideration. As often happens in group discussions, participants do not systematically answer a list of questions, especially when these are related to each other, but jump from one question to another, or unintentionally answer one question while responding to another. Answers are transcribed and ordered according to the questions to which they belong. This process is done by analyzing all parts of the discussion among the teachers and comparing them with each of the eight questions. If the topic of a part of the discussion corresponds to one of the questions, it is then labeled as belonging to that question (even though it could actually come from the discussion of a different question). The data is finally organized in a matrix according to the CoRe sheet. Each cell includes the answers of the teachers given in a specific workshop. For

each workshop a list of Big Ideas and a group CoRe matrix for each Big Idea discussed is created. Disagreements are part of the PCK repertoire and give teachers examples of possible contrasting situations. This phase ensures that the first research question is answered, by organizing the data in a way that falls into the CoRe framework of PCK. Furthermore, the data obtained during the workshops is written down in text, analyzed and organized according to the figure presented earlier on (Table 3.1), in which the eight questions of the CoRe matrix are categorized under Grossman's theoretical framework. This last step shows the results in a way that can be compared with previous work (Saeli et al., 2011c), and that answers the second research question.

3.3 Results

In this section a selection of the data, collected in the period from May 2009 to November 2009 is reported. A selection is made because of space limits, and this is made by selecting the data that will be discussed later. A complete collection of all the results is presented in the technical report relating to this study (Saeli, Perrenet, Jochems, & Zwaneveld, 2010).

3.3.1 Participants

Because it was not possible to find teachers meeting the criteria in the Netherlands, where this research is based, teachers from different countries were invited. Teacher trainers were also asked to participate, to get an acceptable number of respondents. During this study a total of 6 workshops were organized in four different countries (Table 3.2), with a total of 31 participants. Participants in the workshops all had the opportunity to express their thoughts/ideas in their native language. During the two Lithuanian group interviews an interpreter helped the researcher to communicate with the teachers. Twenty-five available and suitable teachers were found in three countries: Italy, Lithuania and Belgium.

Country (cities)	No. of workshops	No. of teachers (t) or teacher trainers (tt)
Italy (Udine and Vicenza)	2	6t (Udine) and 4t (Vicenza)
Lithuania (Vilnius and Druskininkai)	2	4t (Vilnius) and 7t (Druskininkai)
Belgium (Hasselt)	1	3t+1tt
Netherlands (Utrecht)	1	6tt

Table 3.2: List of the countries, number of workshops per country and participants per workshop.

3.3.2 Big Ideas about programming

The Big Ideas about programming are listed below (Table 3.3), which show how many groups of teachers named each one. Locations are listed with the following abbreviations: Italy Udine as IT1; Italy Vicenza as IT2; Belgium as BE; Lithuania Vilnius as LT1; Lithuania Druskininkai as LT2; The Netherlands as NL. In some cases different groups named similar topics while using different terms (e.g. logical skills, logical thinking) and, where applicable, it was decided afterwards to use the same terminology.

3.3.3 CoRes

The Big Ideas chosen and discussed through CoRes are listed in Table 3.4. These lead to an answer to the first research question, in respect to these seven concepts (see section 3.1).

The following Big Ideas were collected before adapting the instrument, therefore answering question 5a (which knowledge about students' thinking influences your teaching of this Big Idea?): control structures for IT1, problem-solving skills for BE and algorithms for IT2. An extract of the CoRe about control structures with focus on loops is reported in Table 3.5.

BIG IDEA	Frequency	Location
Control structures: loops, conditions and sequence	6	IT1, IT2, BE, LT1, LT2, NL
Functions	5	IT1, IT2, BE, LT1, NL
Procedures and methods	5	IT1, IT2, BE, LT1, NL
Algorithms	5	IT1, IT2, LT1, LT2, NL
Variables and constants	4	IT1, IT2, LT1, NL
Parameters	4	IT1, IT2, BE, NL
Data structure	4	IT1, IT2, LT2, NL
Decomposition	4	IT1, IT2, BE, LT1
Reusability (one time named as generalization)	3	BE, LT1, NL
Arrays	2	LT1, NL
Logical thinking (one time named as logical skills)	2	IT1, IT2
Formal language grammar and syntax (one time named as formalism)	2	IT2, IT2
Input and output	2	IT2, LT1
Problem-solving skills	1	BE
User interface	1	NL
Thinking in modules	1	NL
Programming paradigms	1	NL
Recursion	1	NL
Direct and indirect referencing	1	NL

Table 3.3: Table summarizing the Big Ideas listed by the teachers of the different workshops, their frequencies and the locations in which the teachers named the referred Big Ideas referred to.

BIG IDEA discussed	Location
Control structures, focus: loops	IT1 and LT1
Data structures	NL
Arrays	LT1
Problem-solving skills	BE
Decomposition	NL
Parameters	IT2
Algorithms	IT2 and LT2

Table 3.4: List of Big Ideas discussed during the workshops and the corresponding locations.

Question	BIG IDEA Loops - from UDINE
1	GIOVANNA: Identify starting condition; groups of instructions to repeat; exit condition...
2	... SIMONE: One of the fundaments to understand the "miracle" of informatics: by combining in few ways a few elementary operations/actions, you can achieve amazing things....
3	... GIOVANNA: Invariant ...
4	... FRANCESCA: What is before loops, in loops and after loops; condition of loops; loops in true/false conditions. Students also do not manage to find their mistakes (e.g. when working with files, the condition EOF).
5a	... GAIA: Focusing the explanation on components of the loop more difficult to build.
6	GAIA: We try to organize the curriculum with other teachers and to organize combined examples. ...
7	... FRANCESCA: To teach the construct 'for' it is possible to start with a practical example, connected with everyday life such as: We go to the cinema, and we stand at the exit door. We want to know for how many people was the film good and for how many was it bad. I start like this. Then I ask them 'what should we do?'. And the answer is 'waiting until somebody will appear at the door.' ...
8	FRANCESCA: Asking them to explain aloud what would be the solution to a problem using the loop. In this way is possible to understand where the difficulty is, by analyzing the sequence of words and in her/his activity. ...

Table 3.5: Extract of the CoRe about control structures with focus on loops. The questions from 1 to 8 are those reported in the section 'Instrument'.

Similarly, six other CoRe matrixes have been made for the other six Big Ideas, and these are available in the technical report. The CoRe matrixes represent the answer to the first research question. In the next section we report on the analysis regarding the seven Big Ideas discussed during the workshops, leading to an answer to the second research question. From each Big Idea we report only the questions from Grossman's reformulation (see Table 3.1), and these will be compared later with the literature.

3.3.4 The seven Big Ideas

Big Idea: Control structure, focus on Loops

What about the students:
What else might you know about loops (that you don't intend students to know yet)? - knowledge about recursion and invariants; - more concise and efficient solutions; - knowledge about implementations of loops in different programming languages What are difficulties/ limitations connected with the teaching of loops? - identifying: what is before loops; identifying the group of instruction to repeat; and after loops. - condition of loops (true/false or non-trivial conditions); - explicit/implicit counters; zero iteration; limit cases (first and last iteration). - generalization of the problems; - to synthesize and to imagine the expected result of a loop; - difficulties in changing a term (e.g. modifying the initialization of a variable). What do you think students need to know in order for them to learn loops? - variables; - assignments; - conditions; - basic types, usually integers are used as counters in loops such as 'for'. What are your specific ways of ascertaining students understanding or confusion around loops?

- simulation with pencil and paper;
- explaining in words what happens in the algorithm used and reasons to use certain variables;
- a computational equivalence of iterative structures;
- giving almost identical loops and asking students to identify the loop that actually solves a specific problem;
- correcting students' own programs by asking them to execute it in a debugging context;
- giving a loop with a mistake and asking the student to spot it;
- measuring students' enjoyment in creating new and more complicated loops.

Big Idea: Decomposition

While answering the questions relating to the Big Idea decomposition, participants found disagreement among teachers on the use of a wrong algorithm. The reason not to use this methodology is the difficulty of the task because of the cognitive skills involved.

Why to teach:

Why is it important for the students to know this idea?

- students will manage to reach a higher level of abstraction;
- to solve bigger problems;
- it is a universal problem also proposed in mathematics education.

What about the students:

What else might you know about this idea (that you don't intend students to know yet)?

- choice of global or local variables;
- interface;
- parameters;
- formal specification.

What are the difficulties/ limitations connected with the teaching of this idea?

- often students immediately start to write code, without decomposing the problem.
- finding suitable problems to propose to students.

What do you think students need to know in order for them to learn this idea?

- it is necessary to give students prior knowledge of possible solutions to small problems.

- Students need experience of where a solutions can lead to.

What are your specific ways of ascertaining students understanding or confusion around this idea?

- asking students to read a program and describe what each part is doing.

Big Idea: Parameters

What about the students:

What else might you know about this idea (that you don't intend students to know yet)?

- the internal functioning of the computer in relation to parameters, for example types of linking.

What are the difficulties/ limitations connected with the teaching of this idea?

- identifying situations in which it is useful to pass quantities as parameters;

- deciding to pass quantities by value, by address, as parameter, through the result of a function, or keeping them as global or local variables.

What do you think students need to know for them to learn this idea?

MISSING: Teachers did not find a suitable answer to this question

What are your specific ways of ascertaining students understanding or confusion around this idea?

- formulating simple questionnaires in which students are asked parameters of input/output;

- asking students to realize and discuss programs, using different kind of parameters.

Big Idea: Arrays

What about the students:
What else might you know about arrays (that you don't intend students to know yet)? - Size limitations (or the possibility of modifying the size); - indirect sorting; - two-dimensional arrays. What are the difficulties/ limitations connected with the teaching of arrays? - there is only one name for several places in which to store values (correlated with the aspect of indexing); - range check (when the index is going out of array); - use of variables as index. What do you think students need to know for them to learn arrays? - basic types; - variables; - assignment; - the mathematical model of progression. What are your specific ways of ascertaining students understanding or confusion around arrays? - giving confusing examples, as for example a[[[1]]]; - asking to find the maximum value of the array (value); - finding the place of the maximum value (index)

Big Idea: Problem-solving skills

Why to teach:
Why is it important for the students to know this idea? - it is an expertise that can be reused in other domains than computer science;
What about the students:
What else might you know about this idea (that you don't intend students to know yet)? - working in circles rather than in a top-down approach, which is what students are usually confronted with. What are the difficulties/limitations connected with the teaching of this idea?

- students often find methods to follow quite boring;
- there is often not enough time to show students all the possible implementations of the same problem.

What do you think students need to know for them to learn this idea?

- Teachers did not find a suitable answer to this question.

What are your specific ways of ascertaining students' understanding or confusion around this idea?

- asking students to apply a method to solve a problem, and test its applicability by verifying if the problem is actually solved. The role of the students would be to recognize variables and constants.
- giving bigger problems to solve, assuming enough time is available, and letting students to work in groups on an assigned problem and come up with their solutions. The memebers of the groups will then mix, and there will be discussions in each group on the different implementations of the same problem.

How to teach:

Which factors influence your teaching of this idea?

Teachers did not find a suitable answer to this question.

What are the teaching methods (any particular reasons for using these to engage with this idea)?

- explaining the existence of methods to solve problems, which can save the time spent on ineffective attempts. It is important however to choose problems which are neither too difficult nor too easy; and they should especially be problems related to real-life situations.

These tables represent a first step to finding the answer to the second research question. The, complete answer is reached in the next section, in which these results are compared with the literature.

3.4 Conclusions and discussion

In this section we discuss the most interesting results found in the analysis in relation to the body of research found in the literature (Saeli et al., 2011c). We have analyzed the literature with respect to the Big Ideas listed by the teachers during the different group interviews, and made a closer comparison of the discussions of the participants in this study on the seven Big Ideas

(control structures, data structures, arrays, problem-solving skills, decomposition, parameters and algorithms). As will be seen, some of our results are a confirmation of what the literature already suggested, while others appear to be research-based results that are reported for the first time in this paper. It is interesting to note that we did not find any disconfirmation of our results within the literature.

3.4.1 The Big Ideas of programming

As stated earlier, during the first part of the group interviews teachers listed the topics that in their opinion are at the heart of teaching and learning programming (see Table 3.3). We now compare the literature (Saeli et al., 2011c) in search of agreement or disagreement with the content of a possible secondary school curriculum. Answers from the literature point to the following concepts and aspects: problem-solving skills are seen as the core concept of programming knowledge; from a technical point of view, students need to gain knowledge of the data, instructions and syntax of a programming language, but also primitive expression, means of combination and means of abstraction; programming strategies, which identify the way syntax is used to create programs to solve problems; and finally acquiring the skill to write semantically correct programs.

When comparing the literature with the results of this study, at first sight a striking outcome is that problem-solving skills, considered as core programming skills, were named by only a group of teachers. However, within the topic of problem-solving skills we find, as a sub-topic, decomposition of the problem (Soloway, 1986; Schoenfeld, 1979), which was named by four groups. In their answers, teachers discussing the topic of problem-solving skills gave as examples problems involving decomposition. This means that most teachers probably refer to problem-solving skills by just one of their sub-domains: decomposition. This may explain why the Big Idea of problem-solving skills was named by only one group, even though it is at the heart of teaching programming. The reason could be that the group included a teacher from high education, and this could have influenced the choice made by this term.

Concerning the other concepts listed by the participants in this study, we

note that these are in line with the suggestions found in the literature, covering knowledge of the data (variables, constants, arrays and data structures), instructions (functions, procedures and methods), syntax (formal language and syntax), means of combination (control structures and algorithms) and programming strategies (re-usability). The only concept that seems not be considered as a separate entity by teachers is semantics, although this is an issue that appears frequently in the transcripts.

3.4.2 The seven Big Ideas

During the second part of the group interviews teachers discussed the different aspects of PCK for certain topics (see Table 3.4). With these results we now compare the literature (Saeli et al., 2011c) in search of possible agreement between teachers' PCK and the teaching theories found in the literature. This process will lead us to an answer to the second research question.

Regarding the Big Idea control structures (with focus on loops), we find agreement with the results found in the literature. On the question 'What about the students?', the participants in this study identified some of the students' difficulties while learning this topic, for example issues with implicit counters (DuBoulay, 1989). This difficulty can be related to a more general issue: hidden and internal changes in the programming language. These internal changes are hidden from the students, and cause difficulty. Other difficulties are revealed in recognizing the different parts of the loops (before loops; the group of instructions to repeat; and after loops), also mentioned by Du Boulay (1989).

Regarding the Big Idea array and the question 'What about the students?' Du Boulay (1989) proposes an in-depth analysis of the problems students encounter while learning this topic. In this approach students' difficulties and misconceptions are classified into different areas, some of which correlate with those of this study. The participants in this study found that students have trouble understanding that there is only one name for several places in which values are stored (correlated with the aspect of indexing). On the indexing problems, Du Boulay links these difficulties to the more general problem of assignment understanding. Moreover the participants in our study reported other difficulties experienced by students, such as range check (when the index is going out of array) and the use of variables as index. On the latter,

Du Boulay associates this difficulty with the more general issues relating indirection' (e.g., pointers). Regarding the Big Idea parameters and the question 'What about the students?', Hristova and colleagues (Hristova, Misra, Rutter, & Mercuri, 2003) studied misconceptions and difficulties experienced by students learning this topic in the Java environment. Their findings reveal different aspects of misconceptions compared with those found in this study. They found that students are confused between declaring parameters of a method and passing parameters in a method invocation. While the participants in this study found identifying situations in which it is useful to pass quantities as parameters; and deciding to pass quantities by value, by address, as parameter, through the result of a function, or keeping them as global or local variables. Through our findings we can strengthen the knowledge about students' difficulties while learning parameters.

The Big Idea problem-solving skills is a topic on which it is possible to find answers in the literature to the question 'Why to teach it?'. Linn and Dalbey (1989) claim that problem-solving skills are useful for learning new formal systems, which constitute templates and procedural skills common to many or all formal systems or other domains (Schoenfeld, 1979; Mayer, 1989; Feurzeig et al., 1970). This to a certain extent agrees with the reasons given by the participants in this study, who referred to the potential re-usability of this expertise in other domains than computer science (which would consist of a formal system), such as mathematics education. To achieve re usability, the curriculum should be revised in a way that encourages students to transfer this knowledge from one domain to another (Soloway, 1986), for example by making the transferability explicit. Regarding the question 'How to teach it?', Linn and Dalbey do not directly address teaching methods for problem-solving skills in their paper, but refer to situations in which students can acquire this expertise, for example while attempting to apply templates or procedural skills learned in one system to a new system. Similarly, Dagiené (2005) suggests informal learning environments, such as competitions, to improve problem-solving skills. This could be considered complementary to our participants' teaching method, which consists in explaining to students the existence of methods to solve problems, thereby saving time spent on ineffective attempts.

Regarding the Big Idea decomposition of the problem, Soloway (1986), in agreement with our participants, explores some aspects of the PCK of problem solving in relation to the question 'Why to teach it?'. One of the purposes

of teaching decomposition of the problem is that this knowledge can be re-used in other disciplines Perkins and colleagues (1989), for example mathematics (as the participants in this study also suggested). On the question What about the students?', Soloway agrees by suggesting that students must already possess the primitives into which the problem will be decomposed in order to carry out a stepwise-refinement strategy. Obviously when students start learning to program, as our participants also underlined, they lack the knowledge of those possible primitives. On problems encountered while learning decomposition of the problem, Perkins and colleagues (1989) agree with our empirical data on the students' failure to recognize the need to break down problems, considering the programming language as the influential factor. Perkins and colleagues complete their discourse noting that students face difficulties in decomposition of the problem because they often try to deal with decomposition issues in the middle of coding, instead of preparatory planning.

Summarizing, the answer to the first research question is the body of knowledge collected through CoRes regarding the seven Big Ideas listed above, and can be considered as the first effort to portray the PCK of programming for secondary school. The second research question is answered by the close confrontation between the teachers' PCK and the teaching theories found in the literature. Because no disconfirmation of our results is found in the literature, we can suggest that teachers' PCK is in line with the teaching theories found in the literature.

The results of this study represent a source of information for both researchers and teachers. From the point of view of researchers, this study is the first structured effort to portray the PCK of secondary school programming. Seven topics of programming for secondary school are covered in this study: control structures (with focus on loops), decomposition of the problem, problem-solving skills, parameters, algorithms, data structures and arrays. From the point of view of a teacher, this study serves as a source of information for reflection, professional growth and inspiration. The knowledge gathered through such a study can be used for the writing of a textbook or guide for teachers, finding a suitable representation of this knowledge.

3.4.3 Limitations

A limitation of the instrument is that teachers sometimes have no answers to the questions, for example for 'problem-solving skills' (Saeli et al., 2010, p. 22). Addressing the question Which other factors influence your teaching of problem-solving skill?', the group interviewed discussed general teaching factors, such as time pressure and lack of teaching material. It should be noted that also the Australian researchers (see for example Mulhall et al., 2003, p. 21) were confronted with the situation of producing CoRes with empty cells. The reasons could be of a different nature and have not been investigated. A limitation regarding the choice of respondents is that teacher trainers also appeared to list Big Ideas with higher levels of difficulties (see Table 3.3). Although it was not found in this study, teachers from different countries could also report different teaching methods or teaching beliefs. However, this could be considered as a strengthening of the instrument in portraying a more complete PCK of a topic.

3.4.4 Suggestions for further research

In this study we decided not to use PaP-eRs (Pedagogical and Professional-experience Repertoires). PaP-eRs are narrative accounts of teachers' PCK for a particular piece of subject content. Each PaP-eR 'unpacks' the teacher's thinking around an element of PCK for that content, and is based on classroom observations (Mulhall et al., 2003). The PCK portrayed from such an instrument would therefore represent the PCK (Loughran et al., 2000) of an individual teacher. Because the main goal of this study is to sketch a standard for PCK of programming for secondary school, we leave the construction of PaP-eRs for further research, aimed for example at contextualizing the knowledge of PCK for a specific context (e.g. country, county, region, school or classroom). We hope to see more research following this path in order to provide the research community with more results on the PCK of programming, to extend the results of this study, and to find the PCK for other topics of programming and hopefully for other areas of CS. Knowledge of PCK could then be deployed to improve teacher training courses for CS teachers, to organize in-service training courses for teachers with low PCK, to assess teachers' PCK, or to analyze teaching material.

Acknowledgements

We would like to thank the Italian, Lithuanian, Belgian and Dutch participants in our workshops, and Prof. Dagienè, Dr. Mirolo and Dr. Gonnissen for helping us to organize the workshops.

Chapter 4

Programming: Teaching material and Pedagogical Content Knowledge [1]

Abstract

The scope of this article is to understand to what extent Computer Science teachers can find support for their Pedagogical Content Knowledge (PCK) in teaching material. We report the results of a study in which PCK is used as framework to develop a research instrument to examine three high school computer science textbooks, with special focus on programming. PCK is analysed in this study in its two components: pedagogical knowledge (PK) and content knowledge (CK). The results of the textbooks have been compared with the results of a previous study, in which experienced teachers from various countries were involved in semi-structured interviews to portray the PCK of programming for secondary school. Our expectations to find textbooks relatively strong on the CK, but weak on the PK aspect, is confirmed by the results.

[1] This chapter has been accepted for publication as: Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. Programming: Teaching material and Pedagogical Content Knowledge. *Journal of Educational Computing Research*

4.1 Introduction

Teachers, especially at the beginning of their career, use textbooks to determine what content to teach their students and to retrieve possible content representations (Wang, 1998; Chiappetta, Sethna, & Fillman, 1993). In disciplines such as science, textbooks are used most of the teaching time (Wang, 1998; Good, 1993). Also, guidelines of national curriculum and final exams provide an overview of the content to teach. But what could be the information a novice teacher needs to find in a textbook? And what is a suitable textbook from the perspective of a novice teacher? In this article we will try to answer these questions by describing a method to analyze textbooks which teachers, novice teachers and textbook authors can use. This method is exemplified using Computer Science (CS) as target discipline, but scholars from other disciplines can adapt it.
One of the aspects novice teachers need support for is the development of their Pedagogical Content Knowledge (PCK) of the subject they teach. PCK is a construct that has been firstly described by Shulman (1986, 1987) and then reformulated by other scholars (see for example Grossman, 1989; Hashweh, 2005) and is defined as:

> "the ways of representing and formulating the subject that make it comprehensible to others"(Shulman, 1986, p. 9).

In other words, PCK is that expertise that allows teachers to represent, in an effective way, the subject to their students, for example by using metaphors (Woollard, 2005). The more representations teachers have, the higher chance they have to suit the different students' learning needs. Teachers at the beginning of their career have no or low PCK (Rovegno, 1992; Van Driel, Verloop, & Vos, 1998) and could find content representation in teaching materials such as textbooks. It is then important to understand whether teachers can find support for their developing PCK in the textbook of their choice.
PCK is a special amalgam of content and pedagogy that is unique to experienced teachers (Shulman, 1987), thus we could see it as the combination of Pedagogical Knowledge (PK) and Content Knowledge (CK), because on one hand teachers need to have knowledge of the subject (CK) and on the other they need to have a suitable pedagogical approach for that subject (PK). This construct refers to teachers' knowledge and in this study is used as framework

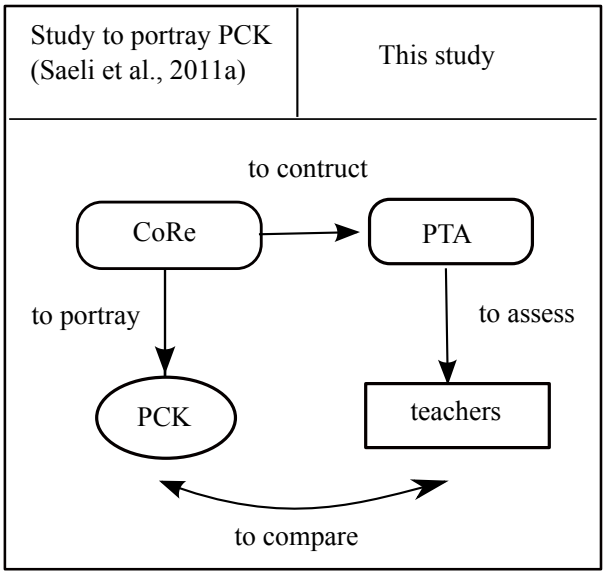

Figure 4.1: Schema representing the relation between this study and a previous one. The terms used are CoRe (Content Representation), PTA (PCK Textbook Analyzer) and PCK (Pedagogical Content Knowledge)

to analyze school textbooks.
In this article we describe our instrument, PTA (PCK Texbook Analyzer), which will be later tested on the subject of programming. The PTA is an adaptation of the research instrument CoRe (Loughran et al., 2001), which has been used in a previous study (Saeli et al., 2010) to portray the PCK of programming for secondary school. The results of the textbook analysis, using the PTA, are then compared with those of the previous study. In Figure 4.1 we find a schema representing the relations of the CORE instrument with the definition of PCK and the construction of PTA, and the relation between the results of the previous study with those of the textbook analysis. In the course of this study PTA will be used on Dutch textbooks, because this research is based in the Netherlands. This study is financed by the University of Eindhoven (the Netherlands), Eindhoven School of Education, in collaboration with ICL (Interuniversitaire Commissie Lerarenopleiding, Interuniversity Committee of Teacher Training).

The PTA is an instrument that can be used by teachers who need to choose a textbook, by textbook authors as a guideline, and by other researchers working

on the field of PCK or CS education.
An introduction to the PCK of Programming will now be given, followed by an analysis of textbook and a discussion about the situation in the Netherlands. Finally we will describe the research questions which have driven this work.

4.1.1 PCK of Programming

In a previous work the authors of this article reported the results of a study with the aim of portraying the PCK of programming in the context of secondary school (Saeli et al., 2010) in order to fill the gap evidenced in the literature (Saeli et al., 2011c). These results were obtained by using the research instrument CoRe (Content Representation), which has been already successfully used by Australian researchers in another domain (Loughran et al., 2001). PCK has been defined as the armamentarium of representations that teachers need to have at their disposal when teaching a certain subject. To initialize the process to create such a set, a total of 31 experienced teachers and teacher trainers have been asked to take part to semi-structured group interviews. These interviews had a length of roughly two hours, involving around five teachers at a time. Each interview was divided into two parts. In the first part teachers had to individually list what in their opinion the 'Big Ideas' of programming are (the CK component). Big ideas are those concepts within a subject which are at the heart of the learning for that specific subject. The results reveal that there are eleven Big Ideas in the context of learning to program (see Table 4.1).

In the second part of the interviews teachers were given a CoRe matrix (see Appendix C for an example) to discuss. They chose, based on their interests, one or two of the "Big Ideas" and then, for each Big Idea they answered the eight questions listed in Table 4.2.

Question 5 was firstly conceived as: knowledge about students' thinking that influences your teaching of this idea. But because this question could be included in question 6, it was decided to include the question 5 as it is now (S. Fincher, personal communication, August 2009).
The results, of an international nature, have been collected in four countries (Italy, Belgium, Lithuania and the Netherlands) and constitute a contribution to the efforts to portray the PCK of programming for secondary education.

Big Ideas
Control structures: loops, conditions and sequential
Functions, procedures and methods
Algorithms
Variables and constants
Parameters
Data structure
Decomposition
Reusability
Arrays
Logical Thinking
Formal language grammar and syntax

Table 4.1: List of the core topics within programming at the heart of its learning.

From the eleven Big ideas, seven topics have been explored, namely: control structures, with focus on loops; data structures; arrays; problem solving skills; decomposition; parameters; and algorithms. For each of these topics we now have available the knowledge about the PCK of these concepts.
An example on the PCK of algorithms is (Saeli et al., 2010):

> - **What to teach about algorithms**: Students need to learn to identify the sequence of actions that bring to the solution of a given problem
> - **Why to teach algorithms**: It is important that students learn the use and generation of algorithms because every programming problem has to be solved through the development of an algorithm, which will give reliability to the job also in presence of difficulty of verification. Without algorithms and algorithmic thinking it would not be possible to write in concrete programming language.
> - **Learning difficulties (what about the students) about algorithms**: Students do not require having knowledge about non deterministic algorithms, problems of complexity, variety of formalities or choice of particular structures. Students do not always recognize the need to develop an algorithm until they will have to solve complex problems. Moreover they have difficulty to autonomously work and to identify the elementary actions of the executor. Students will be facilitated in the learning of algorithms

	Questions
1.	What do you intend the students to learn about this Big idea?
2.	Why is it important for the students to know this Big idea?
3.	What else you might know about this Big idea (and you don't intend students to know yet)?
4.	What are difficulties/ limitations connected with the teaching of this Big idea?
5.	What do you think students need to know in order for them to learn this Big idea?
6.	Which factors influence your teaching of this Big idea?
7.	What are teaching methods (any particular reasons for using these to engage with this Big idea)?
8.	What are your specific ways of ascertaining students understanding or confusion around this Big idea?

Table 4.2: The eight questions of the CoRe instrument

if they have a good background of mathematical logic and motivation. It is possible to ask students to formalize algorithms from different domains, as for example cooking recipes, or to propose examples in which they have to autonomously reach to identify the algorithm.
- **How to teach algorithms**: The interactive environment of programming languages gives the opportunity to obtain results also without a proper preparatory development. This promotes the tendency students have of skipping the analysis of the problem. A way to introduce students to algorithms in programming education is to first identify algorithms in the everyday life before working with algorithms to calculate something more abstract. The process could be assisted by the teacher, who analyzes examples with the students and suggests possibilities of implementation.

4.1.2 Textbook analysis

Textbook analysis is a research method that has been broadly used in different science areas. The reasons to conduct textbook analysis can be various and from different perspectives (e.g. researcher, textbook author, teacher, etc.). Examining the space devoted to interactions among different subjects and topics, understanding the content covered in the textbook, finding sources for

students' misunderstanding, finding the relation between national curriculum and textbooks, finding exercises or tests for students, etc. (Ahtineva, 2005; Chiappetta et al., 1993; Good, 1993; Stylianidou, Ormerod, & Ogborn, 2002). Textbook analysis can be conducted either with a qualitative or a quantitative approach. In the first case an in-depth analysis of the content offered in the textbook is performed, while in the quantitative approach statistics such as number of concepts, number of pages or other quantities are taken into account. In our analysis we will use both approaches, because on one hand we want to know which concepts are covered in a textbook, and on the other hand we want to understand the quality of the concepts offered in the textbook. In the context of CS we have found three studies which analyze textbook in different natures. Means (1988) analyzed ten introduction-to-programming textbooks with the purpose of finding possible differences in the content with earlier texts, possible trends in the content of different books and to provide a framework to allow comparisons between past and recent textbooks. Wu and colleagues (Wu, Lin, & Lin, 1999) explored the nature and the presentation styles of programming examples; and WU and colleagues (Wu, Lee, & Lai, 2004) explored the use of concept maps to examine concepts presented in textbooks. It appears that there is no previous study in CS literature in which PCK has been used to analyze textbooks. Moreover Wang (1998) proposes to design teacher-friendly content analysis methods. In this way student teachers, during their teaching preparation, can be instructed on how to choose a suitable textbook according to their needs.

4.1.3 The Dutch Scenario

The Netherlands is a country in which the teaching of CS for secondary school has been recently introduced (Grgurina, 2008) in the school year 1999/2000, and at the moment is an elective course, examined only at school level. Dutch teachers can choose among three textbooks available in commerce for computer science: Fundament of Computer Science (original title: Fundament Informatica), Enigma and Active Computer Science (original title: Informatica actief). The textbook Instruct is available in combination with extra modules, with a focus on programming. Usually teachers use the main book Instruct with one extra module (e.g. Java) dedicated to programming (Instruct, 2011). In the results section the textbooks will be referred as A (Instruct), B (Active informatics) and C (Enigma). Regarding the textbook Instruct, its modules are referred as: a (Java), b (Delphi), c(PHP), d (BlueJ), e (VB.net) and f (VisualBasic). The results relative to the textbook Instruct and its modules

are presented in combination with each of the modules, as for example Aa (Instruct + Java), Ab (Instruct + Delphi), etc.
The authors of the three Dutch books claim that, in general, the content of the exam program has been inspirational for the writing of the textbooks (Schmidt, 2007b). The exam program for programming refers to the subdomain of Software (domain B, basic knowledge and skills, retrieved from the website examenblad.nl, 2010). The guidelines for the school exam propose that: the candidate masters simple data types, program structures and programming techniques. In Appendix A there is a detailed table with the guidelines for the school exam, proposed by Schmidt (2007a). It should be noted that in the Dutch curriculum, a maximum of quarter of the total time is devoted for programming (which means around 60 to 70 teaching hours per year), despite the fact that the worldwide computer science community considers this topic at the heart of computer science education (Grgurina, 2008). This implies that in the Dutch school scenario, programming is not seen at the heart of learning CS.

4.1.4 Research Questions

The aim of this study, as mentioned earlier, is to develop a research instrument, called PTA, to analyze textbooks using PCK as framework of reference, and to test PTA on Dutch CS textbooks. The research questions therefore are:

- Is it possible to apply the concept of PCK to the analysis of a textbook with the use of the PTA?
- To what extent can we identify the PCK of programming in Dutch secondary textbooks?

We expect to find textbooks relatively strong on the CK, but weak on the PK aspect. The reason is because in general, the content of the exam program has been inspirational for the writing of the textbooks (Schmidt, 2007), which is highly a CK aspect, and no reference is made about the PK. Therefore there seems to be no explicit effort to support teachers' developing PCK in the design of the textbooks.
The method used for this study will now be described, followed by the results, a discussion and the conclusion.

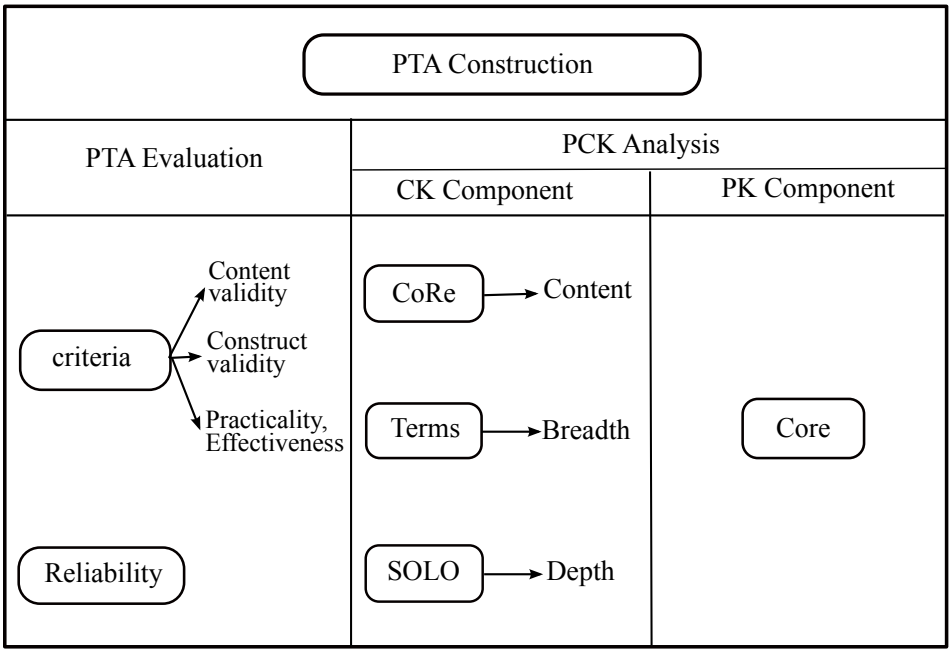

Figure 4.2: Scheme summarizing the different stages of the method. The terms used are CoRe (Content Representation), SOLO (Structure of the Observed Learning Outcomes)

4.2 Methods

In this section we describe two methods, summarized in Figure 4.2: the evaluation of the PTA instrument (left most column) and the assessment of the PCK of Dutch textbooks (centre and right most column).

4.2.1 PTA Evaluation

In order to answer the first research question, about the quality of the instrument PTA, we use Nieveen's quality assessment (Nieveen, 1999), which has been primarily designed for educational products. Its applicability in various domains of educational product development, such as for example learning material and computer support systems, has also been proved (Nieveen,

1999). We extend the list of possible target products, using Nieveen's quality assessment on the PTA, a textbook content analysis instrument. Nieveen's framework for product quality consists of check lists on the following criteria (Figure 4.2, left most column): validity, which refers to its content and its construct; practicality, focusing on the easiness of the instrument; and effectiveness, referring to the time requirement for the use of the instrument.

In order to evaluate these different aspects the following steps will be covered. As for the content validity, the theoretical framework of the PTA will be compared with the state-of-art of PCK; while for the construct validity we will verify if the components of the instrument are consistently linked to each other. Regarding the practicality and the effectiveness of the PTA, two student teachers and a teacher trainer will be involved in the study. A last step will be concerned with the evaluation of the reliability of the instrument. To do so a second researcher will be asked to use the PTA on one topic using two different textbooks. Their results will be compared with each other and the percentage of agreement calculated (Figure 4.2, left most column, lower part).

4.2.2 Quality Evaluation

The goal of this phase is to answer the research question: Is it possible to apply the concept of PCK to the analysis of a textbook?

Content Validity

The CoRe instrument has been successfully used in different subjects(Loughran et al., 2001; Saeli et al., 2010) and has been positively assessed on how the eight questions actually cover the different aspects of Grossman's reformulation of PCK (Saeli, Perrenet, Jochems, & Zwaneveld, 2011a).

Construct Validity

The instrument covers both pedagogy and content through the PK and CK components respectively. Moreover, the instrument is to be used in combination with the results already available about the PCK of a subject, which is the result of group interviews made with experienced teachers. This in order to assure that the knowledge assessed is indeed that special amalgam unique to teachers.

Practicality and Effectiveness

In order to evaluate the practicality and the effectiveness of the PTA, two student teachers and a teacher trainer are questioned regarding the easiness and the time consumption of the instrument, reporting a score between low, medium or optimal for easiness and for effectiveness.

Reliability

In order to test the reliability of the instrument, a pilot round has been run. Two researchers (the first and second author of this article) independently analysed two textbooks on the topic 'control structures' using the following parts of the PCK analysis: step two and three of the CK component and the entire procedure of the PK component. Their scores have been compared, with the following statistical results: the step two of the CK component scored a percentage of agreement (POA) of .83; the third step scored a POA of 1; the PK component scored a POA=.83. After discussing step two, full agreement on the method was found. Because some of the scores are binary numbers (namely CK breadth) it was not possible to perform correlation analysis.

4.2.3 Textbook Analysis

The second research question, assessing the PCK of a textbook, can be answered by using our research instrument. PTA is based on the CoRe instrument introduced earlier, with some adjustments, and by using the results of a previous study (Saeli et al., 2010). Similarly to the CoRe, PTA focuses on Big Ideas (core concepts of a subject), which could be considered as the CK

(content knowledge) aspect of PCK. For each of these Big Ideas, an analysis of its PK (pedagogical knowledge) aspect is conducted through the eight questions introduced earlier. We will now describe the details of the two parts of the PTA (illustrated in Figure 4.2, centre and left most column).

CK Component

Assessing the CK component of a textbook means finding the answer to the first question listed in Table 4.2: "1. What do you intend the students to learn about this Big Idea?". To do so, we divide the CK analysis into three steps: coverage, breadth and depth (Figure 4.2, centre column).
The first step, measuring the CK coverage of a textbook, consists of verifying whether in the table of content and/or in the index of the textbook the topics shown in Table 4.1 are available. If available a '1' will be assigned for the corresponding topic, otherwise a '0' is assigned (see Figure 4.3, left column, as example). In the case of digital textbooks it is possible, when available, to use a search engine to browse the content of the textbook.

This first step assures whether the topics are named or not. The second step, the CK breadth, consists in analyzing the chapters relative to each topic of Table 4.1, verifying whether the terms of the concept tables (Table 4.3 for an example) are covered. As long as at least a paragraph is spent on the topic (or the concept of the topic), a '1' will be assigned (see Figure 4.3, middle column as example). The concept tables are produced in a preparatory step in which a group of computer science researchers and teachers are asked to list the terms related to those topics, in the context of secondary school learning, and reach a consensus. Summarizing, the criteria to understand to what extent a topic is covered is to examine the percentage of terms covered.

There is a final third step, the depth of coverage for each topic. This step is performed by using the Structure of the Observed Learning Outcomes (SOLO) taxonomy (Fuller et al., 2007) for each topic (e.g. control structures, functions, etc.). The SOLO taxonomy has been used to assess students' learning, but we extend its use and adopt it to assess not the learning outcome, but the learning aid provided by the textbook. The SOLO taxonomy has 5 levels of understanding, that in our case will become level of coverage: (1) pre-structural, not

Terms - Control Structures
conditions
logical operator
boolean type
selection
iteration
sequence
if then - if then else
while - do while
for
choice - switch - case

Table 4.3: Example of table of terms

related to topic, disjoint, missed the point; (2) uni-structural, simple meaning, naming, focussing on one issue in a complex case; (3) multi-structural, 'shopping list', disorganised collection of items; (4) relational, understanding, using a concept that integrates a collection of data, understanding how to apply the concept to a familiar data set or to a problem; and (5) extended abstract, relating to existing principle, so that unseen problems can be handled, going beyond existing principles. The chapters relative to each topic are analyzed to verify which of these 5 levels the topic belongs, and mark it with the relative level number. Only one level, the highest, will be chosen, in case more than a value is present.

Summarizing, in order to measure the CK of a textbook we first analyze whether the Big Ideas of programming are mentioned, then we analyze the breath of its coverage and finally its depth (an example in Figure 4.3).

PK Component

As for the PK analysis of the PCK of a textbook, we use the results of a previous study (Saeli et al., 2010) in which the research instrument CoRe (Content Representation) has been used to portray the PCK of programming for secondary school. At the moment seven topics, in the context of programming, have been studied, namely: loops, data structures, arrays, problem solving skills, decomposition, parameters and algorithms. The chapters of the textbooks where those topics are covered are analysed. The CoRe is an instrument

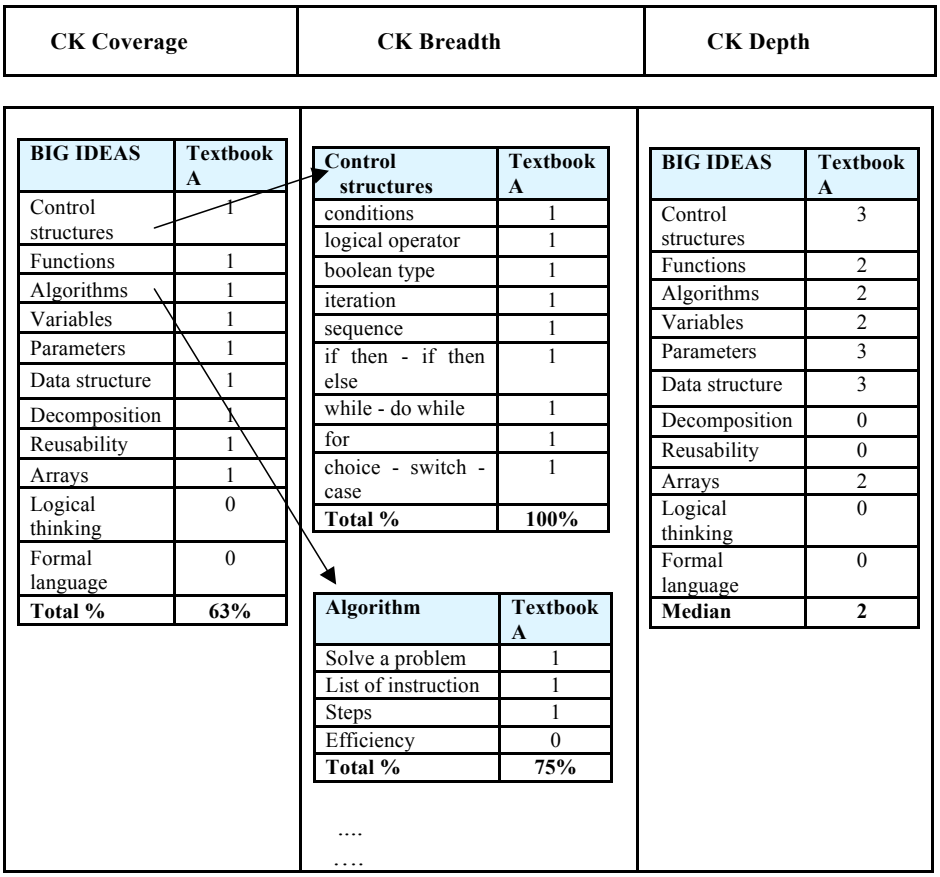

Figure 4.3: Example of partial results of the CK component relative to a textbook analysis using PTA.

that through the use of questions unpacks the PCK of a subject. We use the answers to these questions from the previous study from which to compare the content of the textbook with (Figure 4.2, right most column). We analyse the textbook in search of answers to questions 2 to 8, each answer will be then assessed in terms of comparison with the results of Saeli and colleagues (2010) as: absent (0), low (1), sufficient (2), high (3) in relation to their quality and quantity (for an example see Table 4.5). Below we present the questions and the methods to measure the presence and the quality of each answer found in the book.

2. Why is it important for the students to know this Big idea?: References to the importance to learn/use this specific topic (e.g., it is important to learn control structures because it is necessary to solve more abstract problems).

3. What else you might know about this Big idea (and you don't intend students to know yet)?: A list of topics that will be covered in the future (next school year, for example) by accomplishment of the relative chapter, or in the teachers' guide.

4. What are difficulties/ limitations connected with the teaching of this Big idea?: Hints or suggestions about the difficulty of the topic (e.g. when we want to use a loop, it is often difficult to identify the group of actions to repeat).

5. What do you think students need to know in order for them to learn this Big idea? : Guidance about how to read the book, looking for a hierarchy or sequence to read the different chapters (e.g. in order to be able to use loops, it is advisable to first learn about variables), or some sort of requirements to access the relative chapter.

6. Which factors influence your teaching of this Big idea?: This answer can be found in the teacher guide of the textbook, when available (e.g., when teaching loops, it would be advisable to work in collaboration with the sport teacher, because in sport loops are practically used).

7. What are teaching methods (any particular reasons for using these to engage with this Big idea)?: Different approaches to teach the topic proposed in the book (e.g. in order to understand which construct it is possible to ask students 'do you know how many times you want to do it?').

Extract from a textbook
Algorithms are a set of instructions in a specific order, which serve to solve a problem. The algorithm to a problem to reach a program, looks like follows: 1. definition of the problem 2. analysis 3. schematic solution 4. the making of the source code 5. compiling

Table 4.4: An extract from one of the textbooks listed above, paragraph about Algorithms

8. What are your specific ways of ascertaining students understanding or confusion around this Big idea?: This answer can be found in the diagnosis/assessment section of the book, as for example the problems proposed at the end of the chapter (e.g. by asking students to solve an exercise using loops and asking them to write next to the solution an explanation).

Summarizing, in order to analyse the PK component of the PCK of a textbook, we compare the answers of the questions listed above, with the results found by Saeli and colleagues (Saeli et al., 2010). We now propose an example on how to use the PTA instrument by using an extract from one of the textbooks (Table 4.4). Concerning the CK depth, if we compare the text with the list of terms (see Figure 4.3, centre column, below), we see that this textbook scores, in terms of CK coverage, 75%, because all the terms ('solve a problem', 'list of instruction' and 'steps') except one (efficiency) have been named. In respect to CK breadth, this textbook will score a SOLO level of 3, because the text looks like a 'shopping list' and no relation is made with familiar data.

As for the PK analysis, we read the paragraph (Table 4.4) in search of the answers above listed and compare them with the results found in Appendix C .
The analysis will be:
2. There is no reference to the importance to learn/use algorithms. Score: 0
3. There is no reference to what can be achieved in the future, nor is a teachers' guide is available. Score: 0
4. There are no hints about students' difficulties. Score: 0
5. There is no reference of prerequisites. Score: 0

Textbook	2.	3.	4.	5.	6.	7.	8.
A	0	0	0	0	0	1	0

Table 4.5: Example of PK analysis of a textbook

6. There is a teacher's guide available on this topic. Score: 0
7. The algorithm structure is given. Score: 1
8. Students are asked to think of situation where to apply algorithms, but no real application is proposed, as from the results in Appendix C . Score: 0

These results are then summarized in the compact form showed in Figure 4.5.

4.3 Results

In this section we report the results obtained in this study relative to the analysis of the textbooks.

4.3.1 PCK Analysis

The goal of this phase is to answer the research question: "To what extent can we identify the PCK of programming in Dutch textbooks?" PTA, the method to assess the PCK of a textbook, focuses on two different aspects: the CK component, including its coverage, breadth and depth; and the PK component.

CK Coverage

The goal of the first step of this analysis is to evaluate whether the Big Ideas relative to the teaching and learning of Programming are covered. As we can see in Table 4.6, last row, on average 64% of the topics are covered. The Textbook C (second last column, last row) covers the majority of the topics, with 91%, while the Textbook Aa (second column, last row) scored the least, with 36%. Considering the Big Ideas, on average 63% of the books cover

Big Ideas	Aa	Ab	Ac	Ad	Ae	Af	B	C	Total
Control Structures	1	1	1	1	1	1	1	1	100%
Function	1	1	1	1	1	1	1	1	100%
Algorithms	1	1	1	1	1	1	1	1	100%
Variables	1	1	1	1	1	1	1	1	100%
Parameters	0	1	1	0	1	1	1	1	75%
Data structure	0	0	0	0	0	0	0	0	0%
Decomposition	0	1	0	0	1	1	1	1	62%
Reusability	0	0	0	0	0	0	0	1	12%
Arrays	0	1	1	1	1	1	1	1	87%
Logical thinking	0	0	0	0	0	0	0	1	12%
Formal language	0	0	1	1	1	0	0	1	5%
Average	36%	64%	64%	54%	73%	64%	64%	91%	64%

Table 4.6: Scores of the CK Coverage analysis, divided per textbook.

these Big Ideas (last column), as seen in the last column. The Big Ideas that are covered by every textbook are (100%): control structures, function and algorithm (third to fifth row). The Big Idea that is not covered at all is: data structure (eighth column).

CK Breadth

The goal of the second step is to assess the breadth of coverage for each topic. For each Big Idea a group of experts in the field have been asked to develop table of terms related to those topics. A total of six experts were involved in this phase, producing tables of terms for 8 out of 11 concepts. The concepts for which terms are missing are logical thinking and formal language. The concept problem solving skills has been incorporated with decomposition, because one is a subtopic of the other.
The scores relative to the breadth (percentage of terms present in the book) are reported (see Table 4.7), in relation with the table of terms introduced in the method section.

On average the textbooks cover 62% of the designed terms (see last row). The Textbook B (eighth column) is the one that covers the most of the designed terms with an average of 80% , while the Textbook Aa (second column) scores the least with an average of 45% . Considering the Big Ideas (last column), on average the 62% of the terms is covered in the textbooks. The Big idea that has been the most broadly covered is control structures (on the third row),

Big Ideas	Aa	Ab	Ac	Ad	Ae	Af	B	C	Total
Control Structures	100	100	100	100	100	100	100	100	100%
Function	86	86	71	71	43	43	71	43	64%
Algorithm	75	75	75	75	75	75	75	100	78%
Variable	57	86	71	14	57	57	100	57	63%
Parameters	0	80	80	0	60	60	60	100	55%
Arrays	0	14	29	57	57	57	86	71	46%
Decomposition	0	17	0	0	50	50	67	50	29%
Average	45%	65%	61%	45%	63%	63%	80%	73%	62%

Table 4.7: Scores relative to the CK Breadth analysis, expressed in percentage (%)

scoring 100%, while the least covered is decomposition (see the second last row), with 29%.

CK Depth

The goal of the third step is to understand the depth of coverage for each topic. For each Big Idea the scores relative to the depth (score of the SOLO taxonomy level between 1 and 5) are reported, as shown in Table 4.8.
The median value of the SOLO levels of the textbooks is 4 (see last row). The Textbooks with a higher SOLO level are Textbooks: Ac, Ae, Af, B and C, all scoring a median of 4, while the lowest SOLO level belongs to Textbooks: Aa, Ab and Ad,all scoring a median of 3.
Regarding the Big ideas, the median value of the SOLO level is 4. The Big Ideas that have been most deeply covered are: control structures, functions/ procedures/ methods, variables, and parameters, all scoring a median of 4. While the least deeply covered is decomposition, with a median value of the SOLO levels of 3.

PK Component

This last part of the textbook analysis aims at assessing the PK component of the general PCK per book. The scores relative to the answers of the seven questions (Q2 to Q8) are reported in Table 4.9 Table 4.10. In the first table the scores are the result of the sum of the seven answers (scoring between 0, absent, and 3, high) of each textbook, presented per Big Idea. In each cell

Big Ideas	Aa	Ab	Ac	Ad	Ae	Af	B	C	Total
Control Structures	4	4	4	4	4	4	5	4	4
Function	4	4	5	4	3	3	4	4	4
Algorithm	3	3	3	3	4	4	4	5	3.5
Variable	4	4	4	3	4	4	4	4	4
Parameters	0	3	4	0	4	4	4	4	4
Arrays	0	3	2	3	4	4	5	4	3.5
Decomposition	0	3	0	0	4	4	3	4	3
Median	3	3	4	3	4	4	4	4	4

Table 4.8: Scores relative to the CK Depth analysis.

values can vary between a minimum of 0 and a maximum of 21 (seven times the highest score). In the second table the scores are the result of the sum of each answer (scoring between 0, absent, and 3, high) of the five Big Ideas per textbook, presented per Question. In each cell values can vary between a minimum of 0 and a maximum of 15 (five times the highest score).

Big Ideas	Aa	Ab	Ac	Ad	Ae	Af	B	C	Total (out of 168)
Control Structures	4	4	4	4	4	4	6	7	37
Arrays	0	0	0	2	3	3	6	7	21
Decomposition	0	1	0	0	2	2	0	3	8
Parameters	0	3	4	0	3	3	2	5	20
Algorithm	1	1	1	1	3	5	5	9	26
Total (out of 105)	5	9	9	7	15	17	19	31	

Table 4.9: Results of the PK analysis per Big Idea, on a maximum score of: 168 per Big idea and 105 per Textbook

From the Big Idea point of view all topics are scoring quite low (Table 4.9, last column). The topic with less support for PK is decomposition, scoring an 8 out of a maximum of 168, while the topic scoring the best is control structures, with a value of 37.
From a Question point of view (Table 4.10), there are three questions that have not been answered by any of the textbooks, namely: question 3, 4 and 6. Of the other questions, the best answered one is question 7, scoring 54 out of 120; while the worst one is question 2, scoring 20.

Regarding the textbooks (Table 4.9 and Table 4.10, last rows), all books score quite low. The textbook providing less support for the PK component is Aa, scoring 5 out of 105; while the most supportive is textbook D, scoring 31.

Big Ideas	Aa	Ab	Ac	Ad	Ae	Af	B	C	Total (out of 120)
Q2	1	1	2	1	1	2	3	9	20
Q3	0	0	0	0	0	0	0	0	0
Q4	0	0	0	0	0	0	0	0	0
Q5	0	0	0	0	0	1	3	0	4
Q6	0	0	0	0	0	0	0	0	0
Q7	3	6	5	4	9	9	6	12	54
Q8	1	2	2	2	5	5	7	10	34
Total (out of 105)	5	9	9	7	15	17	19	31	

Table 4.10: Results of the PK analysis per Question, on a maximum total score of: 120 per question and 105 per Textbook

4.4 Conclusions and Implication

In this paper we discussed the use of PCK as framework to analyse secondary school textbooks in terms of how a textbook can support teachers' in-development PCK. The two research questions leading this study concern the use of the PTA and the assessment of a textbook PCK.
Regarding the first research question, "is it possible to apply the concept of PCK to the analysis of a textbook?", the results suggest that it is indeed possible in general to use PCK as a framework to analyse a textbook. An exception is made for some topics, such as formal language and logical thinking, for which the terms to measure the CK breadth were missing, because the group of experts did not find suitable terms for those concepts. This represents a limitation for the PTA instrument, which cannot be used with all concepts.

Regarding the second research question, "to what extent can we identify the PCK of programming in Dutch textbooks?", we found a confirmation of our expectations, the textbook having scored relatively high in the CK component, and low in the PK component. In terms of CK coverage more than a half of the Big Ideas listed are covered by the books. Also, per Big Idea, there is a connection between the frequency of the topic in the textbooks and the frequency of the groups naming the same topic (Saeli et al., 2011a). For example Control Structures is a topic covered in each of the textbooks, and is also a topic named by all the groups in Saeli and colleagues' study. The striking re-

sult is relative to data structure, which has been named by 66% of the groups, but not covered in any of the textbooks; and Reusability, named by half of the groups, and covered by only one of the eight books. This means that teachers with a weak background in CS and who are willing to teach programming cannot find content related support on these Big Ideas, and therefore no pedagogical information. On the other end the Big Idea formal language, covered by half of the textbooks has been named by only of the 33% of the groups from the previous study. In terms of the CK breadth, more than half of the textbooks quantitatively covers these concepts. The only remarkable result is relative to the concept "decomposition", which is poorly covered. Though, it should be considered that this concept is not included in the Dutch curriculum. A surprising choice if we consider that from the previous study 66% of the groups interviewed found it at the heart of learning programming.

Regarding the CK depth, we can notice that there is not much variation in the scores, which on average are quite high. This means that teachers can find a relatively good support in terms of the quality of the content.

As anticipated earlier, the PK component of the textbooks is very poor, with the exception of teaching methods and ways to ascertain students' understanding. Three of the questions have not been answered at all, namely: further knowledge teachers could know about the topic; students' difficulties/misconceptions around the topic; and possible factors influencing the teaching of the topic. These three questions find potential answers in the teachers' guide to the textbooks. None of the three textbooks had any teachers' guidelines on the teaching of these topics, hence the low scores. From the Big Idea point of view, also the PK component analysis is in line with the CK analysis, finding the best support for the topic "control structures", while "decomposition" scores the worse.

On the use of PTA for other purposes than research (e.g. teaching), we suggest an alternative and more practical use of the PTA, to reduce time consumption, though it has been positively scored in terms of time consumption. Also in terms of practicality, the instrument needs some revision, the participants having scored the instrument as medium in terms of easiness of use. Suggestions for an alternative use of the PTA are however relative to the Dutch situation, being developed from the results of this study. We think that it is important to complete the step referring to the CK coverage, because it shows if the text-

book is covering the most important concepts within programming. As for the CK breadth, we suggest to analyse three concepts: control structures (scoring the highest), parameters (scoring the middle) and decomposition (scoring the lowest). Regarding the CK depth there is not much variation in the results, therefore we suggest to randomly pick two or three concepts. As for the PK component the results suggest that some questions are completely omitted by the books, and in general there is not much variation of the scores per book. We suggest to randomly pick two or three concepts and to analyse the PK component.

For those textbook authors who want to target teachers with low PCK, we suggest to take into account our results when designing their products. However, it should be noted that the results of this study do not point at the general quality of a textbook, but only in terms of their PCK. Also, not all textbooks are designed to support teachers' developing PCK, therefore not all textbook should score high from a PCK analysis.

Because this study is the first in its kind, we cannot compare our results with those of others studying the PCK of textbooks. Our results suggest that teachers with low PCK, either because of lack of content knowledge or because of lack of teaching experience, are not able to find suitable support for the development of their PCK. This might explain the reason why some teachers do not relate to textbooks for the teaching of CS (Schmidt, 2007b) and refer instead either to teaching material they develop themselves or to material uploaded on a portal for CS teachers (www.informaticaVO.nl).

Acknowledgements

We would like to thank Elisa, Dennis and Frank, the student teachers who took part into this research.

Chapter 5

Programming: Teachers and Pedagogical Content Knowledge[1]

Abstract

In this article we report about a study to assess Dutch teachers' Pedagogical Content Knowledge (PCK), with special focus on programming as a topic in secondary school informatics education. For this research we developed an online research instrument: Online Teacher PCK Analyser (OTPA). The results show that Dutch teachers' PCK is scored between low and medium. Also we enquired whether there is any relation between teachers' PCK and the textbooks they use by comparing the results of this study with those of a previous one in which the PCK of textbooks was assessed. The results show that there is no strong relation. Finally, we looked for trends between teachers' PCK and their educational backgrounds, as most of the Dutch teachers have a different background than Informatics. The results show that also in this case there is no strong relation.

[1]This chapter has been accepted for publication as: Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. Programming: Teachers and Pedagogical Content Knowledge in the Netherlands. *Informatics in Education*

5.1 Introduction

The goal of this study is to measure Dutch teachers' Pedagogical Content Knowledge (PCK) in informatics education using an instrument developed for this purpose. The reason to underpin such a study is the concern about the quality of Informatics education in secondary schools (Van Diepen et al., 2011). In order to assess the current Dutch scenario, we choose to analyse Dutch teachers' PCK. PCK is that expertise that allows teachers to present, in an effective way the subject to their students (Shulman, 1986) and can be seen as the special combination of content knowledge (CK) and pedagogical knowledge (PK), and grows with the years of teaching experience. The instrument used, OTPA (Online Teachers' PCK Analyzer) reported in Appendix B , is an adaptation of the research instrument CoRe, used to portray the PCK of chemistry education (Loughran et al., 2001) and programming education (Saeli et al., 2011a).

Teachers' answers to the online questionnaire are then compared with the standard PCK of programming portrayed in a previous study (Table 5.1, left column), fully available in the technical report (Saeli et al., 2010) and briefly represented in Appendix C . Further, possible relations between teachers' PCK and the textbooks they use are sought, by comparing teacher's results with those of another study (Table 5.1, right column), in which Dutch textbooks were analysed in search of aspects of PCK (Saeli, Perrenet, Jochems, & Zwaneveld, 2011b). Below (Table 5.1) a scheme is provided representing the relations of the CoRe instrument with the definition of PCK and the construction of OTPA, and the relation between the results of this study with those of two previous ones. These two studies will later be referred as the 'international study' (Saeli et al., 2011a) and the 'textbook analysis study' (Saeli et al., 2011b). A last goal of the study described in this paper is to understand whether teachers' disciplinary background (e.g. literature, mathematics, arts, etc.) is related with teachers' PCK.

5.1.1 PCK Development

Having good PCK means that teachers have several representations of the most commonly taught topics within a certain subject. The more representations teachers have at their disposal and the better they recognize learning difficulties, the more effectively they can deploy their PCK (Van Driel et al., 1998).

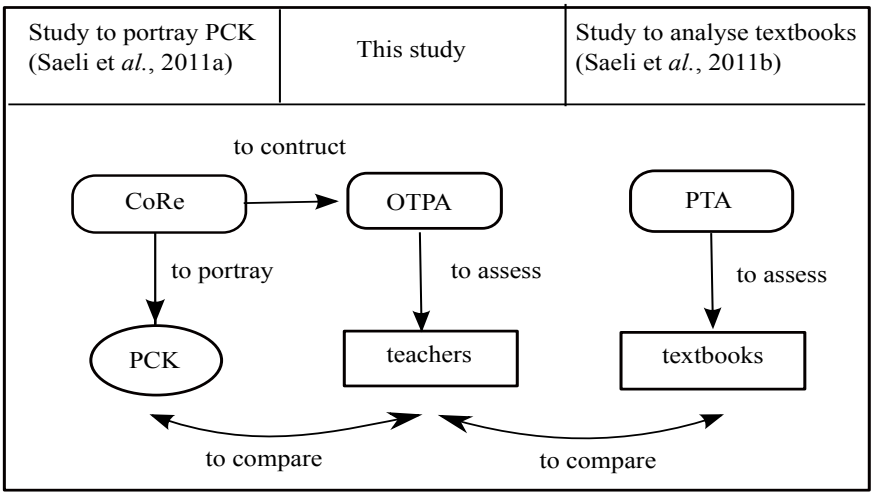

Table 5.1: Scheme representing the relation between this study and two previous ones. The terms used are CoRe (Content Representation), OTPA (Online PCK Teachers' Analyzer), PCK (Pedagogical Content Knowledge) and PTA (PCK Textbook Analyzer)

This implies that PCK is knowledge that grows with the years of teaching experience and can be almost absent at the beginning of the teaching career. Research in fact shows how novice teachers' PCK was inadequate to support teaching in field experiences (Rovegno, 1992). However, teacher training provides a framework on which novice teachers can build their PCK on (Grossman, 1990).

A different scenario is presented when teachers, with several years of teaching experience, are teaching a subject outside of their certification. A study (Sanders, 1993) shows that teachers, when teaching a topic outside their science specialty, sometimes acted like novices (e.g., difficulties in answering students' questions; determining how deep and how much of the topic to present to the students) and sometimes as experts. The conclusion is that PCK is knowledge that is transferable, but not fully. It seems that experienced teachers with strong PCK can reuse their knowledge to teach subjects outside of their certification. Their PCK helps them to recognize the need to transform the knowledge for the students, even though there might be the difficulty determining how much to present at a given time and how to sequence presentations. Through their PCK they can recognize the need to deal with students'

input and try to determine students' background knowledge.

5.1.2 Measuring PCK

Educators and researchers have developed several techniques and methods to study PCK (An et al., 2004; Carpenter et al., 1988; Kromrey & Renfrow, 1991; Carlson, 1990; Rohaan, Taconis, & Jochems, 2009). Baxter and Lederman (1999) give a description of the most general techniques used and their criticisms. They organize the different methods into three groups: convergent and inferential techniques; concept mapping, card sorts and pictorial representations; and multi-method evaluation. *Convergent and inferential techniques* include Likert self-report scales, multiple-choice items and short answer formats. These techniques seem to be an economical means of improving general teacher tests, but it is unclear if these tests are actually tapping new domains of knowledge. The assessment and measurement of PCK concerns the study of a teacher's ability to deal with the unusual, non-generalizable aspects of teaching. Accordingly these techniques seem to be inadequate, because they are too restrictive. *Concept mapping, card sorts, and pictorial representations* are tools that have been largely used to study teachers' knowledge and beliefs, and to measure short-term changes. These tools are not suitable to study the persistence of changes and (therefore) they have little value in understanding the development and change of a teacher's PCK (where PCK involves changes that take place through the years). *Multi-method evaluations* include a variety of techniques to collect data such as interviews, concept maps, and video-prompted recall (Magnusson et al., 1999). Studies conducted with multi-method evaluations are effective in assessing PCK, but they are time- and energy-consuming. For certain studies (Hashweh, 1987) difficulties can be the feasibility to replicate the measurements. In some cases there is the need to make difficult decisions as to which data sources are needed to build a global profile of PCK. The description of the multi-method evaluations suggest that the assessment of PCK is neither simple nor obvious.

Methods and instruments to measure and assess PCK are being studied and experimented with. The most common trend is to rely on qualitative approaches. However, these produce results that do not allow one to generalize concepts about teaching and PCK, because they often consist of case studies. It seems that quantitative approaches have been rarely adopted and their results give a partial view of a teachers' PCK. Both methods are effort and time

consuming. The qualitative methodology requires time for the data analysis part (e.g., interviews transcripts), and in contrast the quantitative method requires time for the development of the research instruments (e.g., adequate multiple-choice items design).

In this study we use an instrument that has both qualitative and quantitative aspects, because we think that a combination of the two techniques will lead to a stronger measurement.

5.1.3 PCK of Programming

The results of the first effort to portray the PCK of programming are reported in the international study (Saeli et al., 2010, 2011a). These results were obtained by using the research instrument CoRe (Content Representation), which has already been successfully used by Australian researchers in chemistry education (Loughran et al., 2001). PCK has been defined as "the armamentarium of representations that teachers need to have at their disposal when teaching a certain subject" (Shulman, 1987). To initialize the process to create such a set, a total of 31 experienced teachers and teacher trainers have been asked to take part in semi structured group interviews, organized in the format of workshops. These interviews had a length of roughly two hours, involving around five teachers at a time. Each interview was divided into two parts. In the first part teachers had to individually list what in their opinion the "Big Ideas" of programming are (the CK component). Big ideas are those concepts within a subject which are at the heart of the learning for that specific subject according to well-educated and experienced teachers. The Big Ideas in the context of learning to program that have been named by more than two groups are reported in Table 5.2. In the second part of the workshops teachers chose, based on their interests, one or two of the Big Ideas and then, for each Big Idea they answered the eight questions listed below (Table 5.3). These results comprise the standard against which the PCK Dutch teachers will be measured.

The data has been collected in four countries (Italy, Belgium, Lithuania and the Netherlands) and constitute the first contribution to the efforts to portray the PCK of programming for secondary education. Having the possibility to freely choose from the eleven Big ideas, the teachers chose seven topics according to their interest, namely: control structures, with focus on loops;

Big Ideas
Control Structures: loops, conditions and sequence
Functions, procedures and methods
Algorithms
Variables and constants
Parameters
Data structure
Decomposition
Reusability
Arrays
Logical Thinking
Formal language grammar and syntax

Table 5.2: List of the core topics within programming at the heart of its learning

data structures; arrays; problem solving skills (named by only one group); decomposition; parameters; and algorithms. The PCK of these topics is now available integrally in the technical report (Saeli et al., 2010), and in Appendix C only the known PCK of the Big Idea algorithms is reported.

5.1.4 The Dutch Situation

In the Netherlands Informatics at secondary schools first landed in 1998, when the Dutch Ministry of Education had the content and quality of all existing courses controlled and new ones introduced. Informatics was one of these new courses. Concerning the content of Informatics courses it should be noted that in the Netherlands, lower grades students (up to 14 years old) are all expected to become IT literate (Hulsen et al., 2005), which is to achieve the minimal level of familiarity with technological tools like word processors, e-mail, and Web browsers (on Information Technology Literacy, 1999). This means that by attending secondary Informatics education students should foster higher achievements. Recently (fall 2007) secondary education underwent two modifications, which for Informatics implied a simplification of the curriculum. It became less detailed and schools were granted more autonomy and choice in the way they organize education. Informatics is at the moment an elective subject for students and for schools as well.

	Questions
1.	What do you intend the students to learn about this Big idea?
2.	Why is it important for the students to know this Big idea?
3.	What else you might know about this Big idea (and you don't intend students to know yet)?
4.	What are difficulties/ limitations connected with the teaching of this Big idea?
5.	What do you think students need to know in order for them to learn this Big idea?
6.	Which factors influence your teaching of this Big idea?
7.	What are teaching methods (any particular reasons for using these to engage with this Big idea)?
8.	What are your specific ways of ascertaining students understanding or confusion around this Big idea?

Table 5.3: The eight questions of the CoRe instrument

At the moment, on a total of roughly 550 secondary schools, there are about 350 Informatics teachers (Schmidt, 2007b) in the Netherlands, most of whom did not receive a formal teacher training in Informatics, but were offered an in-service training spread over two school years, known as the CODI course (Consortium Omscholing Docenten Informatica - Consortium for the schooling of Informatics teachers). The content of this course covers about half of the first year of a Bachelor of Science in Informatics and the pedagogical aspect of teaching Informatics in the last two years of secondary education (ages 16 to 18). The common scenario is that there is only one Informatics teacher per school, in case a school offers Informatics. Considering the amount of Dutch schools, it means that only around the 65% of schools could adequately provide an Informatics curriculum. Further, the majority (67%) of these teachers in 2007 were 50 years old, or more, which means towards the end of their career. It seems that Informatics in the Netherlands is at a crossroad, where there is even the risk to withdraw the teaching of Informatics from schools (Van Diepen et al., 2011). The curriculum is divided into four domains, namely, Informatics in perspective (possible uses and scope of Informatics), basic concepts and skills, systems and their structures, and applications in connection. These four domains are divided into 18 sub-domains (Schmidt, 2007a), but the level of understanding and depth students are expected to achieve are not specified.
Three textbooks are available in the Netherlands: Fundament of Computer Science (original title: Fundament Informatica), Enigma and Active Com-

puter Science (original title: Informatica actief). The authors of the three Dutch books claim that, in general, the content of the exam program has been inspirational for the writing of the textbooks (Schmidt, 2007b). The exam program is divided in several subdomains, programming is mentioned in the subdomain Software (domain B, basic knowledge and skills, retrieved from the website www.examenblad.nl, 2010). The results of the textbook analysis study (Saeli et al., 2011b) suggest that these books sufficiently support teachers' PCK on the content component (CK), but fail in supporting the pedagogical component (PK).

5.1.5 Research Questions

The aim of this study is to measure Dutch Informatics teachers' PCK. To do so we develop a research instrument, called OTPA. Moreover we are interested in exploring the relation between teachers' PCK on the one hand and on the other hand their background studies and the textbook they use. Therefore the research questions are:

- Is it possible to assess teachers' PCK with the use of OTPA?
- What is Dutch teachers' PCK of programming for secondary school?
- To what extent is teachers' PCK related to the textbook they use?
- To what extent is teachers' PCK related to their disciplinary background?

5.2 Methods

In this section we describe the methods to answer the four research questions. The methods for the first two questions are summarized in Table 5.4: the evaluation of the OPTA instrument (left most column) and the assessment of Dutch teachers' PCK (centre and right most column). Regarding the third research question, finding a relation between teachers' PCK and the textbooks they use, we will use the results the textbook analysis study to find any trend to suggest whether a textbook with high PCK could support teachers' PCK. The fourth question is answered by grouping teachers according to their disciplinary backgrounds.

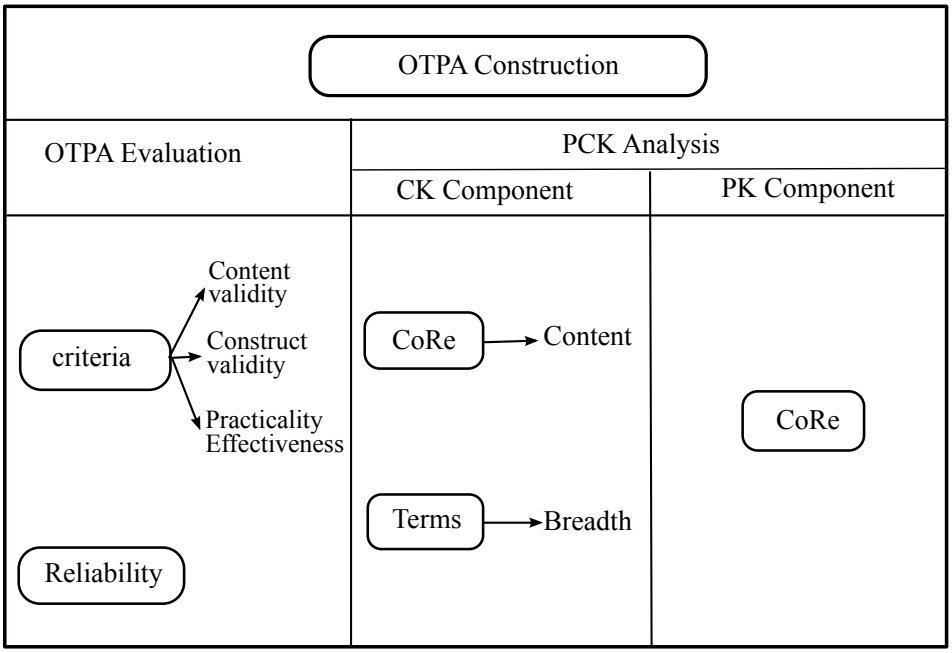

Table 5.4: Scheme summarizing the different stages to answer the two first research questions. The acronym used is CoRe (Content Representation). Details of the method used follow in the next sections.

5.2.1 Participants Teachers' PCK

The OTPA has been available online between January 2011 and April 2011, and has been filled in by 92 teachers, but only 69 of those reached the end of the questionnaire. The other 23 teachers abandoned the process, though they were given the opportunity to complete the questionnaire in a later stage by anonymously signing in again. We report the results of those teachers who reached the end of the questionnaire. The majority of the participants are either at the beginning of their teaching (Table 5.5) - less than 10 years experience - or quite experienced - between 20 and 40 years of teaching. As for the teaching experience of Informatics education, the majority of teachers have less than 10 years of teaching experience (Table 5.6). Of these teachers 37 also teach another discipline, and the majority of those teaches a scientific oriented subject (e.g. mathematics, physics) (Table 5.7). In the Netherlands subjects are categorized as follows: *alpha* are subject such as Dutch and English; *beta* are subjects such as mathematics and physics; *gamma* are subjects such eco-

nomics or geography; and *delta* is the category for informatics (Mulder, 2002). This categorization sees Informatics as the only delta discipline because this subject is unique in that it has in its nature different aspects of other disciplines.

Teaching experience	Frequency
<10 years	26
10 to 20	5
20 to 30	18
30 to 40	20
TOTAL	69

Table 5.5: General teaching experience

Informatics Teaching experience	Frequency
<10 years	37
10 to 20	30
20 to 30	1
30 to 40	1
TOTAL	69

Table 5.6: Informatics teaching experience

5.2.2 OTPA (Online Teacher's PCK Analyser) Evaluation

In order to measure Dutch teachers' PCK we developed an instrument by adapting the CoRe, an instrument used to portray the PCK of programming (Saeli et al., 2011a). The adaptation is made into two directions, the content and the form. Originally the CoRe instrument was designed to portray the PCK of chemistry education (Loughran et al., 2001), whereas this study deals

Teaching also another discipline	Frequency
alpha	26
beta	5
gamma	18
other	20
TOTAL	37 (of 69)

Table 5.7: Teaching also other discipline

with the topic of interest is Informatics education, with a focus on programming. As for the form, CoRe was initially designed to be used in the context of semi-structured interviews, while in this study it is adapted to be used in the format of an online questionnaire. Also questions about teachers' disciplinary background and teaching experience are added.

In order to answer the first research question, the quality of the instrument OTPA, we used Nieveen's quality assessment (Nieveen, 1999), which has been primarily designed for educational products. Its applicability in various domains of educational product development, such as for example learning material and computer support systems, has also been proven (Nieveen, 1999). The use was further extended to evaluate a research instrument to analyze textbooks (Saeli et al., 2011b). We extend the list of possible target products, using Nieveen's quality assessment on the OTPA. Nieveen's framework for product quality consists of checklists on the following three criteria (Table 5.4, left most column): validity, which refers to its content and its construct; practicality, focusing on the easy of use of the instrument; and effectiveness, referring to the time requirement for the use of the instrument.

In order to evaluate the different aspects of the OTPA evaluation, the following steps are covered. For the content validity, the theoretical framework of the OTPA is compared with the theoretical framework of PCK, to verify whether the instrument assesses all aspects of PCK; while for the construct validity it will be verified if the components of the instrument are consistently linked to each other. Regarding the practicality and the effectiveness of the OTPA, an independent researcher is asked to assess the easiness and the time consumption of the instrument, in terms of low, medium or optimal for easiness and for effectiveness.

A further step regards the evaluation of the reliability of the instrument. To evaluate the instrument reliability, another independent researcher is asked to use the OTPA to analyse three randomly chosen teachers' responses to the online questionnaire. The two researchers' results are compared with each other and the percentage of agreement calculated (Table 5.4, left most column, lower part).

5.2.3 Quality evaluation

The goal of this phase is to answer the research question: Is it possible to assess teachers' PCK with the use of the OTPA?

The CoRe instrument, which has been readapted to build OTPA, has been successfully used in different subjects (Loughran et al., 2001; Saeli et al., 2011a) and has been positively assessed on how the eight questions actually cover the different aspects of PCK (Saeli et al., 2011b). There is an almost one-to-one correspondence between the OTPA and the results obtained with the CoRe.

The instrument analyses both aspects of pedagogy (PK) and content (CK), two of the three main components of PCK. The data obtained with this instrument is then compared with the known PCK.

In order to measure the practicality and effectiveness a second researcher participated into this study. The second researcher commented that: *"Controlling the urge to be too interpretive and give meaning where none was obvious was not easy. I do not think this is a reflection of the instrument. I believe this is probably a common factor in any qualitative research where respondent's words represent the data. Where the respondent's comments were of good quality and quantity, the scoring was easy"*. The second researcher's comment reveals a difficulty of analyzing qualitative data that is not specific to this study. Therefore her positive score in terms of the practicality of PTA is at a medium level in terms of easiness. When asked to score its effectiveness, the second researcher reported the instrument to be high in terms of time consumption, commenting that *"It is operational and perfectly usable for scoring the respondents' answers"*.

Moreover, regarding the reliability of the instrument, a pilot round has been run. Two researchers (the first author of this article and a second external researcher) independently analysed two teachers' answers to the open-ended questions (CK3 and PK2 to PK8, reported in Appendix B). The answers to be compared have been randomly and blindly chosen by the second research. The scores have been compared and resulted in a percentage of agreement (POA) of .81. After discussing the answers that produced different scores, full agreement on the method was found.

5.2.4 Teachers' PCK Analysis

The second research question, assessing Dutch teachers' PCK, is tackled by using our research instrument. The data obtained are compared with the standard of PCK for programming, obtained through workshops, fully available in the technical report (Saeli et al., 2010) and partially reported in Appendix C . Similarly to the CoRe, OTPA focuses on Big Ideas (core concepts of a subject), considered as the CK (content knowledge) aspect of PCK. For each of these Big Ideas, an analysis of its PK (pedagogical knowledge) aspect is conducted through the eight questions introduced earlier (Table 5.3). Below the details of the CK and PK components of the OTPA (Table 5.4, centre and left most column) follow.

Content Knowledge Component

Assessing the CK component is done in two phases (Table 5.4). In the first phase we analyse the answers to two multiple-choice questions (see Appendix B), which are labelled as CK1 and CK2. Question CK1 has been constructed using the aforementioned Big Ideas of programming, obtained from the international study (Saeli et al., 2011a); while CK2 is constructed using experts' opinion regarding which terms are related to the teaching of control structures, obtained from the textbook analysis study (Saeli et al., 2011b). In the list of choices of these two questions there are both correct and incorrect answers, therefore teachers have to recognize the right choices and cross them. For each correct choice they get 1 point, up to 11 for CK1 and up to 8 for CK2. In order to be consistent with the scales of questions CK3 and PK2 to PK8, the scores of questions CK1 and CK2 are then labeled as 'low', 'medium' and 'high'. To do produce such labels CK1 and CK2 scores are divided into thirds (e.g. for CK2: a 0 to 2 score is labelled as 'low', 3 to 5 as 'medium', and 6 to 8 as 'high').

In the second phase the answer to the question: "1. What do you intend the students to learn about this Big Idea?" (Table 5.3) is found. This is the first of the PK part of the questionnaire, but because it refers to content knowledge is labelled as CK3. The answer is compared with experienced teachers' PCK, considered as the standard and reported in (Saeli et al., 2011a). It is evaluated as: blank (0) if answer is not given; low (1) if only 1/3 of the standard is listed by the participant; sufficient (2), if 2/3 of the standard is named by the participant; and high (3) if the full standard is listed by the participant. Because

of the qualitative nature of this analysis one does not find exact matching answers, but similar concepts are also positively scored.

An example of the CK3 could be comparing a teacher's answer on the topic 'algorithm' with the standard (Appendix C and Table 5.8):

Teacher: *'Algorithm' = plan of steps/recipe. Writing an algorithm = solving a problem. You can write simple algorithms, using parameters, iterations, conditions and simple data structures (variables/arrays).*

This teacher's answer is scored with a 'medium', because it indeed lists topics such as sequence of actions (*plan of steps*) and combination of structures to reach the solution of a problem (*parameters, iterations, etc.*), but fails in naming its representation and realization and the necessity for algorithms.

What do you intend the students to learn about this idea?
-To start programming in concrete programming languages for the future Foundation for the future programming - Managing to spot the sequence of actions which bring to the solution of the problem - A correct description of algorithms allows to: boost (sharpens) logic skills; communicate the results of the job. - Utility to clarify a fixed sequence of actions which allows to obtain specific, concrete and correct results - What it is, for what is it useful, representation, realization. The goal is to teach students that you need instructions to solve a problem, independently that it will be a computer to execute your instructions

Table 5.8: An extract of the CoRe about algorithms fully reported in Appendix B

Pedagogical Knowledge Component

As for the PK (Pedagogical Knowledge) aspect of teachers' PCK, as term of comparison the standard of the PCK of seven topics in the context of programming is used, namely: loops, data structures, arrays, problem solving skills,

decomposition, parameters and algorithms. We offered teachers, in the context of the OTPA, the choice of one of the seven topics listed above. They were then prompted with the eight questions introduced in Table 5.3 and had the possibility to skip them or leave them blank. As question 1 is used to analyse the CK component, we only use questions 2 to 8 for the PK component. Each answer is then assessed for comparison with the results of the international study (Table 5.4, right column) as: blank (0), low (1), sufficient (2), or high (3) in relation to their quality and quantity. Similarly to the CK3, teachers' answers are compared and scored (see example in previous paragraph). Because of the qualitative nature of this analysis, it is not always possible to find exact matching answers, but more often similar concepts can be found. When teachers' answers are different from the standard, but still seem to be of good quality, an expert has been asked to assess these questions.

In the course of the international study (Saeli et al., 2010) the PCK of some Big Ideas has been collected using a different set of questions to Table 5.3. For the Big Ideas 'problem solving skills' question 5 was conceived as follows: Which knowledge about your students' thinking influences your teaching of this Big Ideas? Consequently for those teachers answering that question of the OTPA on the concept of problem solving skills an expert has been asked to assess teachers' answers.

Relation with Textbook

In order to explore whether there is any relation between teachers' PCK and the PCK of the textbook they use, the results of this study are compared with those of the textbook analysis study, in which the PCK of textbooks have been measured (Saeli et al., 2011b). The resulting histogram graphs are studied in search of relations and trends (see Tables 5.16 and 5.17).

Relation with Background

In order to test whether there is any relation between teachers' PCK and their background, the results of teachers' PCK will be sorted according to teachers' disciplinary backgrounds, according to the alpha, beta, gamma, delta categorization introduced earlier. Similarly to the previous section, the resulting histogram graphs are studied in search for relations and trends (see Tables

5.20 and 5.21).

5.3 Results

This section is divided into three parts: the results relative to the analysis of teachers' PCK; the results of possible relation between teachers' PCK and the textbook they use; and finally the results of a possible relation between teachers' PCK and their background (alpha, beta, gamma and delta).

5.3.1 PCK Assessment

In this section the results relative to the CK and PK component are reported.

Content Knowledge Component

Regarding the CK1 and CK2 components, the majority of teachers scored medium values in both multiple choice questions (Table 5.9 and Table 5.10). The method used to obtain these scores is reported in section 5.2.4. In the CK3 component there is not much variation between low and medium scores, though a consistent number of teachers skipped this question (Table 5.11).

Score	Frequency
high	13
medium	54
low	2
TOTAL	69

Table 5.9: Scores of the CK1, first multiple choice question

Score	Frequency
high	11
medium	42
low	16
TOTAL	69

Table 5.10: Scores of the CK2, second multiple choice question

Score	Frequency
high	6
medium	22
low	24
empty	17
TOTAL	69

Table 5.11: Score of the CK3, first open-ended question

Pedagogical Knwoledge Component

In this part the results of the PK component of teachers' PCK is reported. Teachers had the opportunity to freely choose one topic from the available seven concepts. The majority chose to answer the questions for Problem Solving Skills and Algorithms (Table 5.12), while only 2 teachers chose to discuss Decomposition. The scores relative to the answers of the seven questions (PK2 to PK8) are reported in Table 5.13. Here the scores reported are the result of the sum of each answer (scoring between 0, blank, and 3, high) for the 69 teachers, which means that the maximum score can be 207. The question with the highest score is PK2 (93 out of 207), concerning the reasons to teach a certain concept. The questions with the lowest score are PK3, concerning extracurricular knowledge about a concept, and PK6, concerning other factors influencing the teaching of a concept, scoring relatively, 51 and 55 out of 207.

Concept	Frequency
Loops	11
Data structures	6
Arrays	2
Problem solving skills	33
Decomposition	2
Parameter	4
Algorithms	11
TOTAL	69

Table 5.12: Choice of the concept to discuss

Table 5.14 provides an overview of the blank answers, either because the questions were not answered, or because the answer was out of context. Teachers who chose to answer the questions relative to the Big Idea "data structures" actually understood the term data structure to mean databases, therefore the

Question	Sum out of 207	Score
PK2	93	medium
PK3	51	low
PK4	73	medium
PK5	75	medium
PK6	55	low
PK7	67	low
PK8	63	low

Table 5.13: Summary of the scores relative to the questions. Low (sum up to 69), Medium (sum between 70 and 138), High (sum between 139 and 207).

6 teachers have been given the score '0'.

Questions	PK2	PK3	PK4	PK5	PK6	PK7	PK8
Blank	17	40	23	25	34	27	30

Table 5.14: Number of teachers who did not answer the questions

5.3.2 Relation with Textbooks

The textbooks used in this study are three: Instruct (29 teachers), Active informatics (20 teachers) and Enigma (only 6 teachers). Because a relevant number of teachers do not report to use textbooks (13 teachers), their results are also reported. However, because we have no details about the content of teachers' own material, it is not possible to infer or speculate from the results relative to these teachers. Also, the research question leading the analysis of these data concerns only the possible relation between teachers' PCK and the textbooks they use. The scores relative to the CK (Table 5.16) and PK (Table 5.17) components are reported below. As for the use of textbooks, the majority of the respondents, 29 teachers, use the Instruct textbook (Table 5.15). This book is designed to be used with one or more extra modules. From the data obtained we now that from a total of 69 teachers, 16 teachers didn't answer the questions at all, from those 7 use Informatica actief, 5 Instruct, 2 teachers using no textbook, 1 Enigma and 1 other material (the only representative of this group). In Table (5.15) the averages of teachers' scores using these textbooks are also reported, relative only to the PK component, later analysed in detail in this section.

Textbook	Frequency	blank	low	medium	high
Instruct	29	35%	30%	30%	6%
Enigma	6	33%	31%	31%	5%
Informatica actief	20	45%	15%	35%	5%
No textbook	13	45%	30%	22%	3%
Other	1	100%	0%	0%	0%
TOTAL	69				

Table 5.15: Textbook used in classroom by 69 teachers and the relative average scores with respect to the PK component.

Note that the following results have been scaled up to 100%, though it should be considered that sample groups for the different categories (textbook or disciplinary background) have very different sizes, reported in the graphs' captions. This choice was necessary to be able to compare the different groups.

Regarding the CK1 and CK2 (Table 5.16), there is little variation among the teachers using different textbooks, where the most frequent score is 'medium'. An exception is found in CK2 for teachers using the Enigma textbook are the only ones not to have any 'low' scores. As for the CK3, where teachers were given the opportunity to leave blank answers, the scenario is different. There is no consistency in the different groups. The only teachers scoring, in a very small portion, 'high' are those using Instruct and Informatics-actief. Enigma teachers score mostly 'low', while teachers not using textbooks are almost equally spread between 'low' and 'medium'.

On the questions relative to the PK component (Table 5.17), teachers were given the opportunity to give blank answers. On average 40% of teachers skipped these questions (not considering teachers using 'other' kind of teaching material) and the most skipped question were PK3, PK6 and PK8. Those who answered are almost equally distributed between 'low' and 'medium' (see Table 5.15), with the exception of those teachers using Informatica-actief, who mostly score 'medium' (35% of them). We can see the biggest variation of scores in questions PK2 and PK3 (relatively reasons to teach a concept and extracurricular knowledge), where some more 'high' scores are also found in all textbooks, except for teachers not using textbooks. On these last two questions, teachers using Informatica-actief are those having the smallest percentage of 'low' scores. For the other questions, there is seems to be no book

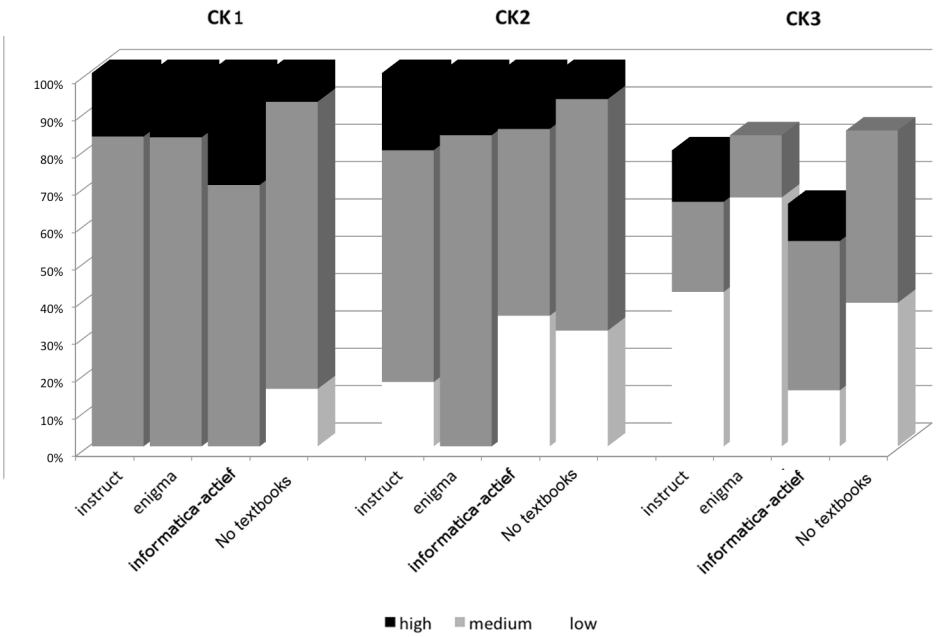

Table 5.16: Results relative to teachers using the same textbook or no textbook, divided by question (CK1, CK2 and CK3) and showed as percentage. Instruct N=29, Enigma N=6, Informatica-actief N=20, No textbooks N=13. Gaps between the top of the bars and 100% represent teachers who left blank answers.

where teachers score remarkably higher than others, though it is possible to notice some high scores (PK4, PK7 and PK8). The answers to questions PK4 (difficulties connected with the teaching) present less difficulties to teachers using the book Inofrmatica-actief. Question PK5 (students' prior knowledge needed) results in a similar distribution of scores irrespective of the textbook, with most scoring 'medium' and a smaller percentage 'low'. While for PK6 (factors influencing the teaching of a concept), teachers using the Enigma book score the highest. The answers to question PK7 (teaching methods) are in general 'low' for Instruct and Enigma teachers, while mostly 'medium' for teachers using Informatica-actief and teachers not using textbooks. Lastly, regarding the question PK8 (ways of ascertaining students' understanding) there are more 'low' scores, although Instruct teachers and teachers not using textbooks score slightly better than the others.

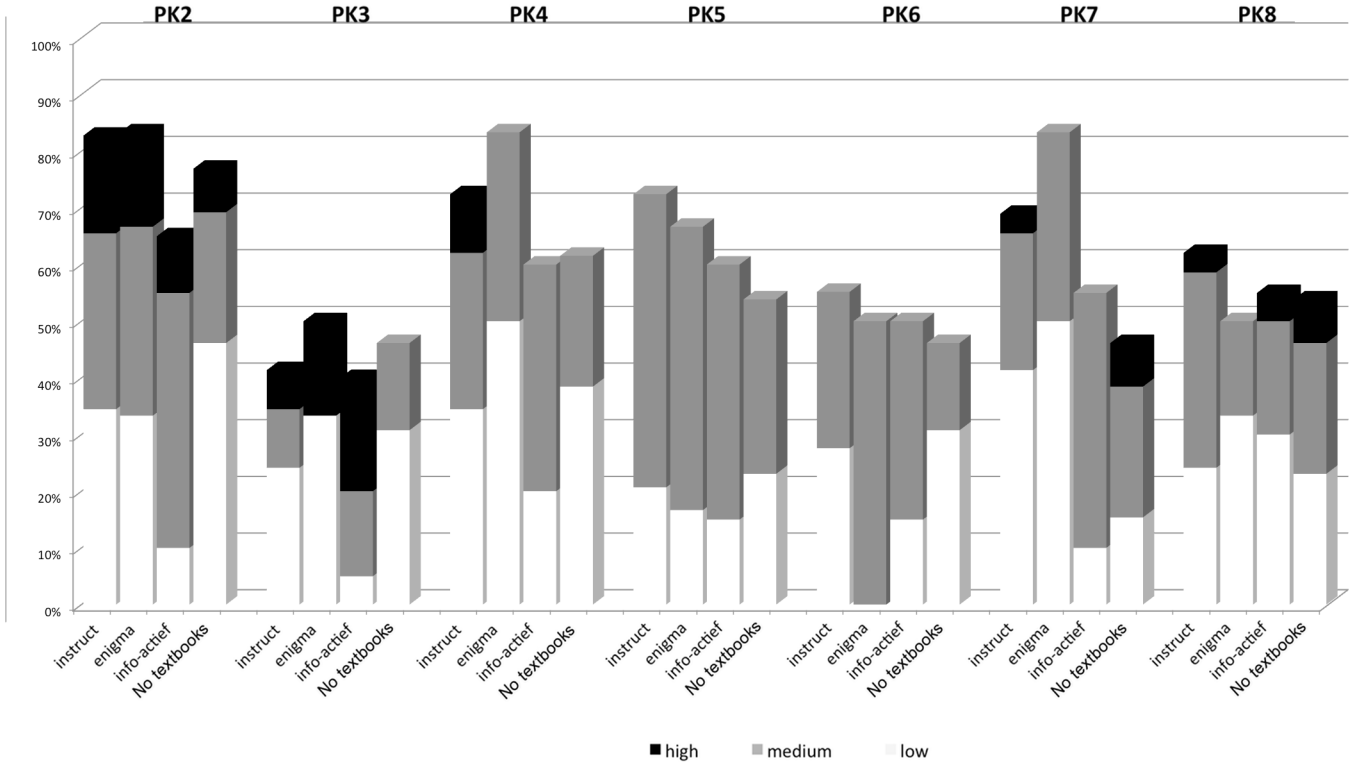

Table 5.17: Results relative to teachers using the same textbook or no textbook, divided by question (PK2 to PK8) and showed as percentages. Instruct N=29, Enigma N=6, Informatica-actief N=20, No textbooks N=13. Gaps between the top of the bars and 100% represent teachers who left blank answers.

5.3.3 Relation with Disciplinary Background

The teachers participating in this study have all different disciplinary backgrounds: alpha (16 teachers), beta (26 teachers), gamma (only 6 teachers) and delta (15 teachers). The other 6 teachers did not specify their background studies, therefore will not be shown in the results. The scores relative to the CK (Table 5.20) and PK (Table 5.21 and Table 5.18) components are reported below.

Background	Frequency	blank	low	medium	high
alpha	16	52%	28%	16%	4%
beta	26	30%	30%	36%	4%
gamma	6	50%	14%	31%	5%
delta	15	38%	22%	30%	10%
not specified	6	52%	17%	31%	0%
TOTAL	69				

Table 5.18: Background studies of 69 teachers and the relative average scores with respect to the PK component.

	Frequency
Yes	34
No	35
TOTAL	69

Table 5.19: Did you attend the CODI course?

In Table 5.18 we can see that most teachers have a beta (26 teachers), while only 6 teachers have a gamma background. Roughly half of the teachers have attended the CODI course (Table 5.19), the in-service training for teachers with another disciplinary background to get basic knowledge for teaching Informatics.

Regarding the CK component (Table 5.20), in the context of multiple-choice questions, teachers from different disciplines score quite similary ('medium'), except for gamma teachers in question CK2. A completely different scenario is presented in the context of the open-ended question, were there is a definite distinction between delta teachers (followed by the gamma teachers) and the others.

Regarding the PK component (see Table 5.18), teachers with alpha back-

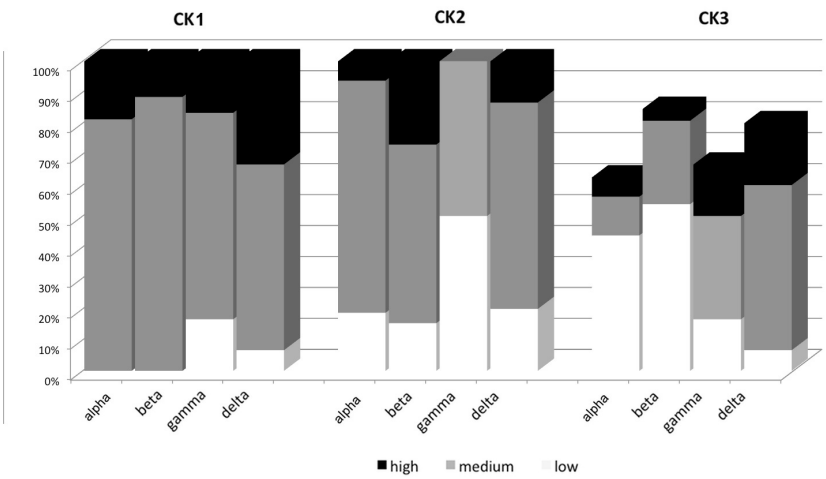

Table 5.20: Results relative to teachers having the same background, divided per question (CK1 to CK3) and showed as percentage. Alpha N=16, Beta N=26, Gamma N=6, Delta N=15. Gaps between the top of the bars and 100% represent teachers who left blank answers.

ground are those scoring the worst out of those teachers answering the questions. While the group of delta teachers is performing slightly better than the others. Teachers with beta background are evenly spread between 'low' and 'medium'. As for the number of blank answers, teachers who answered most of the questions (Table 5.18) are those with beta background, while the ones answering the least have a gamma and alpha background.
Further (Table 5.21), in questions PK2 (reasons to teach a concept) and PK3 (extracurricular knowledge) there is more variation between 'low', 'medium' and 'high' scores spread among teachers of different backgrounds. On the same two questions, teachers with delta background also have the largest percentage of 'high' scores.

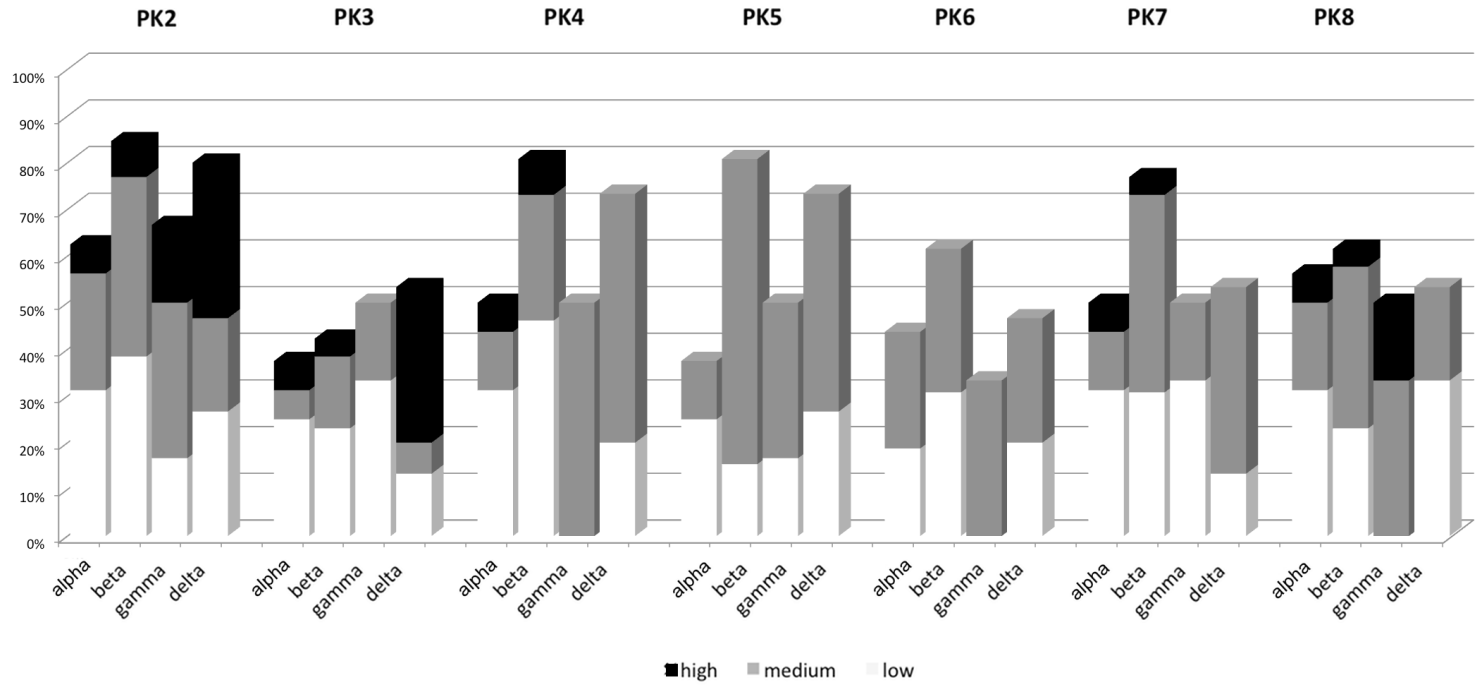

Table 5.21: Results relative to teachers having the same background, divided per question (PK2 to PK8) and showed as percentage. Alpha N=16, Beta N=26, Gamma N=6, Delta N=15. Gaps between the top of the bars and 100% represent teachers who left blank answers.

In questions PK4, about difficulties connected with the teaching of a concept, gamma and delta teachers generally scored 'medium', while alpha and beta had consistent 'low' scores. A similar scenario is presented for questions PK5 (students' prior knowledge needed), except for alpha teachers (mostly scoring 'low'); while for PK6 (factors influencing the teaching of a concept) scores are between 'low' and 'medium', except gamma teachers (mostly scoring 'medium'). For questions PK7, relative to teaching methods, alpha and gamma teachers mostly score 'low', while beta and delta teachers have a larger proportion of 'medium' scores. Lastly, regarding question PK8 (ways of ascertaining students' understanding), gamma teachers perform distinctively better than the others.

5.4 Conclusions and Discussion

In this paper we discussed the use of PCK as framework to assess secondary school teachers. The four research questions leading this study concern the use of the OTPA, teachers' PCK assessment, possible relation between teachers' PCK and the textbooks they use and possible relation between teachers' PCK and their disciplinary background.

5.4.1 About the instrument

Regarding the first research question, "is it possible to assess teachers' PCK with the use of the OTPA?", the results suggest that it is indeed possible in general to use PCK as a framework for this kind of assessment. A second researcher used the instrument and positively assessed it in terms of ease of use and practicability. Though, for the qualitative aspect of the instrument, the interpretation of teachers' answers is at times difficult. However, this is a common difficulty of those interpretative processes specific of qualitative methods.

5.4.2 Dutch teachers' PCK

The goal of the second research question of this study is to assess Dutch teachers' PCK from a general perspective. The results show that Dutch teachers have generally scored 'medium' on the content knowledge component, when

given multiple choice questions. In the context of open ended questions, teachers perform less well and scores are almost equally distributed between 'low' and 'medium'. These results might indicate that Dutch teachers have not enough disciplinary background to answer open ended questions, which involve knowledge production. However, when they are in the condition of recognizing knowledge (multiple choice questions) their performance improves. A possible explanation of this phenomenon might be found in the difference between knowledge recognition and knowledge production (synthesis). According to the hierarchical classification of cognitive processes (Krathwohl, 2002) knowledge recognition is considered of a lower difficulty than knowledge (re)production. Teachers answering multiple-choice questions are actually recognizing knowledge, because they need to choose between correct and incorrect answers. While when teachers answer open-ended questions they are in the process of producing knowledge. One might argue that clicking on a multiple choice question is faster than answering an open-ended question, influencing teachers' performance (better with multiple choice questions and worse with open ended questions). However, if the results of the other open ended questions are considered (Table 5.13) it is possible to note that scores vary between 'low' and 'medium' scores, suggesting that some aspects of Dutch teachers' PCK is stronger than other and is sufficient to answer open ended questions, according to the hierarchical classification described above.

As for the pedagogical component, Dutch teachers have scored 'medium' on the questions relative to the reasons to teach a certain Big Idea, on the difficulties connected with the teaching of a certain Big Idea and on students' prior knowledge required to learn a certain Big Idea. At the same time they score poorly on questions relating to extracurricular knowledge around the Big Idea of their choice, factors influencing their teaching of that Big Idea, teaching methods and ways to ascertain students' understanding. The 'low' scores on these domains are also influenced by the consistent number of teachers skipping these questions. Especially on the extracurricular knowledge, where more than half of the teachers skipped the question relating to this topic. This result evidences how Dutch teachers may be lacking of a solid disciplinary knowledge in programming ('low' CK).

Summarizing, the answer to the second research question is that Dutch teachers perform sufficiently on the content knowledge component, especially when in the condition of recognizing knowledge, like with multiple-choice questions.

One might have expected Dutch teachers to poorly score on the CK component, because most of them have a disciplinary background different than Informatics. However, Dutch textbooks (Saeli et al., 2011b) have been positively assessed in terms of the CK component and could have supported Dutch teachers weak disciplinary background. We could speculate that these teachers would benefit from well designed teaching material to further improve their performance. These outcomes are confirmed by Dutch teachers' results on the pedagogical knowledge component, especially on the question relative to extracurricular knowledge, where scores were poor both for teachers and textbooks. Also on the teaching methods, teachers seem to need more support. This result is actually a confirmation of the need for more teaching materials and teaching examples, also underlined by participants to the international study (Saeli et al., 2011a). It is reassuring on the other hand, to see that Dutch teachers score 'medium' on topics such as reasons to teach, students' prior knowledge and difficulties relative to the teaching of a topic. The results of this study could help Dutch scholars (Van Diepen et al., 2011) in the process of revising the whole subject. From the results of this study we can conclude that Dutch teachers are indeed neither strong on the CK nor the PK component, though they perform much better when knowledge is made explicit (e.g. multiple choice questionnaires). Effort should therefore be made in the direction of producing more teaching material and guidebooks for teachers.

5.4.3 PCK and textbooks

To what extent is there a relation between teachers' PCK and the textbooks they use? From the results of this study (see Table 5.16) there seems to be no relation regarding the CK component, in the sense that teachers using the same textbook do not score better than teachers using another textbook, but generally score 'medium' in the context of multiple-choice questions, and between 'low' and 'medium' (except teachers using the book Informatica-actief) in the context of open-ended questions. A reason for this homogeneity can be attributed to the CK component of Dutch textbooks, which has been positively assessed in the textbook analysis study.

Regarding the PK component, the first sign of a relation between textbooks and teachers' PCK is on the choice of Big Ideas to discuss. Teachers mainly chose those concepts that are also present in the textbooks. Of the few participants who decided to discuss the 'data structures' concept, all teachers

actually answered as if the Big Idea was referring to 'databases'. The latter is a concept included in the Dutch curriculum (Schmidt, 2007a), while the concept 'data structures' is the only concept that is not explicitly addressed in any of the Dutch textbooks. Also it should be noted that none of these teachers had an Informatics background. Interestingly, most of the teachers (33 out of 69) decided to discuss the concept 'problem solving skills', which is considered to be at the heart of teaching programming (Saeli et al., 2011c). 11 teachers decided to discuss either Loops and another 11 Algorithms, which are both concepts largely covered in the textbooks.

When considering the different parts of the PK component, it is possible to see from the graphs (see Table 5.17) that there is no real difference between teachers using different textbooks, except for those that use Informatica-actief in questions PK2 and PK3 (content to teach and reasons to teach that content). Dutch textbooks were assessed in a previous study as weak on the PK component. When comparing the results of these two studies, Dutch teachers obtained better scores in comparison with the results of the textbooks they use. One exception is made on the question regarding the teaching methods, where some more consistent 'low' and 'medium' scores were found from Dutch teachers, while their textbooks scored generally 'medium'. Although in some aspects of PCK (reasons to teach a certain Big Idea and extracurricular knowledge) there is more variation in teachers' answers, showing also some 'high' results, we cannot identify any relation with the textbooks they use, which all had poor scores on the same questions. Remarkably, teachers scored mostly 'medium', but also sometimes 'high', on those questions that found no answers in the textbook analysis. These questions concern difficulties connected with the teaching/learning of a concept and factors influencing the teaching of a concept. These questions usually would find answers in a teacher's guide to the textbook, which is not available for any of the Dutch textbooks. As for the question concerning extracurricular knowledge, mostly teachers scored low, except for teachers using Informatica-actief. This is the only relationship between teachers' low performance and the PCK assessed in the textbook study.

Summarizing, the answer to the second research question is that, in the Dutch scenario, there seems to be no strong relation between teachers' PCK and the textbooks they use, though for some aspects teachers performance might be linked with the quality of the textbooks (e.g. CK component).

5.4.4 PCK and disciplinary background

The goal of the fourth research question is to understand whether there is a relation between teachers' PCK and their disciplinary background. Disciplinary backgrounds are divided according to the Dutch categorization: alpha (e.g. Dutch, English), beta (e.g. Mathematics, physics), gamma (e.g. Geography, Economics) and delta (Informatics). From the results there seems to be no relation regarding the content knowledge component, when teachers are given the opportunity to answer multiple choice questions. When answering open ended questions delta teachers scored remarkably better than the other teachers, while alpha teachers scored worse. This is a confirmation that teachers with a solid disciplinary background have better knowledge of the subject and manage to reproduce their knowledge in the context of open ended questions.

Regarding the pedagogical knowledge component, one might expect teachers teaching another discipline to actually be supported on some aspects of their PCK from their teaching experience, as research has shown (Sanders, 1993). Also from the results of this study it is possible to see how teachers with different disciplinary backgrounds than Informatics, do actually score at times 'medium', for example on questions regarding reasons to teach a certain concept, students' prior knowledge needed, and factors influencing the teaching. Regarding the question about teaching methods, the teachers that seem to score better are those with a beta or a delta disciplinary background. As for question regarding the methods to ascertain students' understanding, all different disciplines seem to have equally distributed results between 'low' and 'medium', with the exception of gamma teachers (mostly scoring 'medium'). Only for teachers with alpha and gamma background is there a noticeable tendency to skip open ended questions, and unfortunately the reasons for this are unclear.

Summarizing, the answer to the fourth and last question is that there is no strong relation between teachers' PCK and their disciplinary backgrounds, except in the context of content knowledge reproduction (CK3), where Informatics teachers clearly scored better than other teachers. Also teachers with alpha and gamma background are those who mostly skipped questions, though we cannot infer the reasons for such choice. Quite surprisingly teachers with Informatics background scored quite similarly to teachers with a non-Informatics background on the content knowledge component in the context of multiple

choice questions. This might be due to the fact that textbooks have been positively assessed on their CK component and might support teachers with weak disciplinary background. As for the pedagogical knowledge aspect, it is not possible to evidence a single disciplinary background scoring better than the other. However teachers with alpha background scored on average the worse and skipped the most questions. Beta and delta teachers are those that, in average, scored better. One might have expected teachers of non-Informatics disciplines to score better on the PK, because they are supported, in some cases, by the PCK developed through their teaching experience.

5.4.5 Limitations of the study

The instrument OTPA, intended as teachers' PCK measurement instrument, has the limitation that it does not give the teacher the chance to discuss his/her own knowledge. Though teachers were given the opportunity to write their answers in open-ended questions, they might have not had the chance to fully express their knowledge. Another limitation of the instrument is that several teachers skipped the open-ended questions. Reasons might be different, as for example time pressure, lack of knowledge or interest. Also, all teachers answering the questions relative to the teaching of data structures actually answered the questions referring to databases.

One might argue that the PCK assessed in this study does not reflect a teachers' actual PCK, because teachers had the chance to choose the topic to discuss. Probably teachers chose the topics in which they felt more confident. A consequence could be that even if a teachers' PCK on her/his topic of preference is good, it does not automatically imply that her/his PCK of programming is good. However, the goal of this study is not to measure a single teachers' PCK, but to assess the Dutch scenario using PCK as framework. In order to be able to generalize our results from single teachers' PCK it was chosen to have a large sample size.

One final observation should be made regarding the quality of the sample of teachers. Participants of this study were invited to fill in the questionnaire anonymously, either through e-mail, advertisement on the portal for Dutch speaking informatics teachers (www.informaticavo.nl) or through mailing lists. Though the participants to this study consist of almost 1/5 of the Dutch Informatics teachers, we suspect a bias in our results owing to the fact that

only those teachers who feel confident in programming might have completed the questionnaire.

Acknowledgement

We would like to thank all the teachers who kindly filled in the questionnaire. Without them this study would have not been possible. Also we would like to thank Cynthia C. Selby and Peter van Mill for their professional help.

Chapter 6

Conclusions

6.1 Overview of the research

The research described in this book has been guided by the general aim of exploring the teaching of Computer Science (CS) for secondary education, in terms of Pedagogical Content Knowledge (PCK). CS is a discipline that has only recently been introduced in secondary education in the Netherlands as well as in other countries, therefore its PCK is still merely unknown. The work conducted through the studies reported in this book has on one hand the important and pioneering scientific relevance of portraying the PCK of this discipline, developing an instrument to measure CS teachers' PCK and developing an instrument to recognise aspects of PCK in textbooks. On the other hand it has the practical relevance of improving the quality of CS education in the Netherlands, by portraying the current situation from a Pedagogical Content Knowledge perspective.

The need to investigate the recent Dutch CS Education (CSE) scenario is due to the recent critics Dutch CSE received. The problems evidenced in the Netherlands are multiple (see chapter 1), but among these the fact that most Dutch CS teachers have weak disciplinary background (Schmidt, 2007b). This situation is the result of an action to introduce CS in secondary education, when no teachers of CS were available in 1998. The solution opted at that

time was to organise in-service training for teachers certified in other disciplines (e.g. economics, art, mathematics, Dutch, language, etc.). This resulted in a population of Dutch CS teachers willing to teach CS, but having a weak disciplinary background. Now CS risks to be erased from the CS curriculum for secondary school, because no clear solutions are available to the problems outlined, though efforts are being made. The approach adopted in this research is to focus on the Pedagogical Content Knowledge (PCK) of CS as a framework to assess the Dutch scenario and establish a basis knowledge to propose local solutions.

In order to measure Dutch teachers' PCK it was first necessary to explore the current literature in search for possible knowledge about PCK of CS through a literature review study (chapter 2). Possible gaps and lacks of the PCK of CSE evidenced in the literature were then bridged through an exploratory study aimed at portraying teachers' PCK of CS inside and outside the Netherlands (chapter 3). Also, different aspects of PCK of CS are searched in Dutch textbooks (chapter 4), to later find possible relations between Dutch teachers' PCK and the information found in textbooks in a study aimed at measuring Dutch teachers' PCK (chapter 5).

In this research project it has been decided to focus on only one of the different topics of CS: programming. Focusing on one topic would allow an in-depth analysis which seems more appropriate than a rather global description of a number of topics. The reasons to choose programming is that this is one of the core topics of a CS curriculum in both secondary and higher education, and is considered by many to be a difficult topic to learn and to teach.

The research questions guiding this research are:

1. To what extent is it possible to recognise aspects of Pedagogical Content Knowledge of programming for secondary education in current literature? (Chapter 2)

2. What is the Pedagogical Content Knowledge of programming in the context of secondary school education? (Chapter 3)

3. To what extent is it possible to identify the Pedagogical Content Knowledge of programming in Dutch secondary school textbooks? (Chapter 4)

4. What is Dutch teachers' Pedagogical Content Knowledge of programming for secondary school? (Chapter 5)

In the next section an overview of the answers to these questions are given, drawing the conclusions from the results of the studies reported in in chapters 2 to 5.

6.2 Summary of the outcomes

6.2.1 Pedagogical Content Knowledge of programming

In chapters 2 and 3 the PCK of programming for secondary education is studied in terms of its presence in current literature and its conceptualisation from current practice. One of the reasons to conduct the literature review reported in chapter 2 is the common consensus on the need to understand the PCK of CS (Holmboe et al., 2001). In their paper, Holmboe and colleagues underline the need for CS education research to focus on the traditional *didactical* questions of teaching *why, what, how* and *for whom*. In other words they ask for research focused on the PCK of CS, a construct still unexplored in this discipline. In the literature review reported in chapter 2, the current literature was first explored using as framework of reference Grossman's reformulation of PCK (Grossman & Howey, 1989; Grossman, 1990). In her reformulation of the concept, PCK is seen as the answer to the following four questions: why to teach a certain subject?; what should be taught?; what are learning difficulties?; and how to teach? Answers to these questions were firstly sought in the literature.

Because no explicit attempt to uncover the PCK of programming has been done before, either on higher or secondary education, in most of the cases the four 'answers' of Grossman's reformulation found in chapter 2 are not connected with each other. Among the reasons to teach programming, the first question, it was found: enhancing students' problem solving skills (Soloway, 1993) and offering the students a subject, which includes aspects of different disciplines; use of modularity, reusability and transferability of the knowledge/skills (Sims-Knight & Upchurch, 1993; Dagiené, 2005); and the opportunity to work with a multi-disciplinary subject (Mulder, 2002).

The second question of Grossman's reformulation, aimed at listing the concepts/aspects that a programming curriculum should include, is answered in the literature with: programming knowledge, which refers to the knowledge of the data, instructions and syntax of a programming language (Govender, 2006), but also primitive expression, means of combination and means of abstraction (Abelson & Sussman, 1996); programming strategies, which identify the way syntax is used to create programs to solve problems; and programming sustainability, which refers to the ability to create user friendly and attractive program that takes care of ethical and privacy issues (Hromkovič, 2006).

The third question, aimed at defining the various difficulties students encounter while learning to program, is answered by: general problem of orientation, in terms of finding out what programming is useful for and what the benefits to learn to program are (DuBoulay et al., 1989); difficulty to instruct the machine about the solution of a problem; and tendency to converse with a computer as if it was a human (Pea, 1986). Regarding the solution of a problem, students tend to maintain a local, limited point of view, failing to find a suitable solution (Ginat, 2006).

The last aspect of PCK, teaching methods, is dealt in the literatures as: offering a simple programming language so students can focus on the syntax (Mannila et al., 2006); choosing several problems to solve, which should be carefully chosen, independently from any programming language, in order to achieve algorithmic thinking (Futschek, 2006); and teaching by means of suited programming languages or programming environments (Mannila et al., 2006; Feurzeig et al., 1970; Papert, 1980; Resnick & Ocko, 1990; Resnick et al., 2009; Cooper et al., 2003; Kölling & Henriksen, 2005; Overmars, 2004).

Though it was possible to find some aspects of PCK of programming in the literature, these are incomplete and and not always connected with each other. Therefore the need to conduct an explorative study to portray the PCK of programming.

These results have been compared in chapter 3 with teachers' practice in an international context, through the results of an exploratory study aimed at unpacking the PCK of CS for secondary school. Using a research instrument previously deployed to portray Australian teachers' PCK of science education (Loughran et al., 2000, 2001; Mulhall et al., 2003; Loughran et al., 2004), the PCK of programming for secondary school was captured in the context of

semi-structured group interviews. These were organized in different countries (Belgium, the Netherlands, Italy and Lithuania), with experienced teachers or teacher trainers, with a total of 31 participants spread over 6 groups. On the first aspect of PCK, "what to teach about programming", when comparing the results of the literature review with those of chapter 3, at first sight a striking outcome is that problem-solving skills, considered as the core concept of programming, were named by only one group of teachers. By problem-solving skills it is meant the ability and knowledge of solving a problem using certain techniques, such as spiral approach or decomposition (Soloway, 1986; Schoenfeld, 1979). The latter was named by four groups. In their answers, teachers discussing the topic of problem-solving skills gave as examples problems involving decomposition. This means that most teachers probably refer to problem-solving skills by just one of their sub-domains: decomposition, explaining why the concept of problem-solving skills was named by only one group, though it is at the heart of teaching programming. Concerning the other concepts listed by the participants in this study, these are in line with the suggestions found in the results of chapter 2.

During the second part of the group interviews teachers discussed the four different aspects of PCK according to Grossman's reformulation (Grossman & Howey, 1989; Grossman, 1990) for certain topics, which teachers chose according to their interests, namely: control structures (with a focus on loops), decomposition of the problem, problem-solving skills, parameters, algorithms, data structures and array. These results constitute the first effort to portray the PCK of programming for secondary school and can be found in the technical report (Saeli et al., 2010). When compared with the results of chapter 2, it is possible to note that some aspects of teachers' PCK are in line with the teaching theories found in the literature, while other aspects complement and bridge the gap evidenced in the literature (chapter 2). Below some examples on the different aspects of PCK explored where comparison with literature was possible.

- Control structures (with a focus on loops): students' difficulties on the learning of this concept are addressed to implicit counters and recognizing the different parts of loops (before loops, the group of instructions to repeat and after loops). These difficulties have also been recognized by DuBoulay (DuBoulay et al., 1989).
- Arrays: according to the participants of the study, students seem to have troubles in understanding that in the context of arrays there is

only one name for several places in which values are stored. On the indexing aspect of this problem Du Boulay (1989) connects this problem with a more general assignment issue. Also there are problems related to 'indirection' (e.g. pointers) connected with range check and the use of variables as index.

- Parameters: on this concept findings from the literature and those of this study do not focus on the same aspects. In the literature (Hristova et al., 2003) issues are focused on students' confusion between declaring parameters of a method and passing parameters in a method invocation. While teachers' of this study pointed at students' difficulties such as passing quantities as parameters, keeping parameters global or local, and whether to pass quantities by address or value.

- Decomposition of the problem: being decomposition of the problem a subdomain of problem solving skills, reasons to teach it are similar for the two topics, as for example re-usability cited by both the respondents of the study and in the literature (Perkins et al., 1989). Again agreement is found on the difficulties students' encounter when learning this topic, as for example students' failure to recognize the need to break down problems (Perkins et al., 1989)

- Problem solving skills: seen as at the heart of learning programming, a reason to learn this skill is that it enables students to learn new formal systems, which constitute templates to reuse. Participants to the study gave reasons to teach such skill because of its re-usability of this expertise in other domains than computer science. To achieve such re-usability the curriculum should be revised in a way that encourages students to do so, as also suggested by Soloway (1986). Methods to foster such skills suggested by teachers differ with those found in the literature. Teachers suggest to propose students methods to solve problems, saving time spent on ineffective attempts, while in the literature (Dagiené, 2005) suggestions focus on informal learning environments, such as competitions, to improve these skills.

The above constitutes only a fraction of the full contribution of this study to the understanding of PCK of programming, more is available in the technical report of the study (Saeli et al., 2010).

6.2.2 Teaching material and PCK

Teachers, especially at the beginning of their career, use teaching material to determine the content to teach their students and to retrieve possible content representations (Wang, 1998; Chiappetta et al., 1993), in other words they might look for support for their developing PCK. In chapter 4 the results of a content analysis study are reported with the twofold goal of describing a method of looking at teaching material from the perspective of PCK, and to draw the Dutch scenario from the perspective of Dutch textbooks. PCK in this study has been analysed from its two main components: content knowledge (CK) and pedagogical knowledge (PK). Textbooks were found to score relatively high in the CK component, and low in the PK component. This means that teachers seeking for help can find relatively good support in terms of the quality of the content in Dutch textbooks. As for the PK component, in the textbooks it was possible to find only support for teaching methods and aspects related to students' understanding. Others aspects are not covered, such as further knowledge teachers could know about the topic; students' difficulties/misconceptions around the topic; and possible factors influencing the teaching of the topic. These three aspects find potential support in the teachers' guide to the textbooks. None of the three Dutch textbooks had any teachers' guidelines on the teaching of these topics, thus the low scores. Because the PK component of CS is an aspect that is more of interest to teachers than students, suggestions to textbook authors were oriented in writing teachers' guidelines to the textbooks.

6.2.3 Teachers and PCK

Methods of measuring teachers' PCK is a topic of concern of different disciplines (An et al., 2004; Carpenter et al., 1988; Kromrey & Renfrow, 1991; Carlson, 1990; Rohaan et al., 2009; Baxter & Lederman, 1999), because deep and broad PCK is linked to effective teaching (An et al., 2004; Magnusson et al., 1999). In chapter 5 the results of an empirical study are reported with the twofold goal of describing a method to measure teachers' PCK and to draw the Dutch scenario from the perspective of Dutch teachers. Using an online anonymous questionnaire with multiple-choice and open-ended questions, it was possible to measure 69 Dutch teachers's PCK. Again PCK in this study has been analysed from its two main components: content knowledge (CK) and pedagogical knowledge (PK). The results of Sanders and colleagues (1993)

about the reusability of PCK in a discipline other than a teachers' specialty are echoed in this study, in which Dutch teachers even when only having disciplinary background and teaching experience in other domains than CS, they manage to reuse their PCK from other domains and disciplines, especially on the reasons to teach certain topics. The results show that Dutch teachers perform poorly on the CK component when answering open-ended questions, but improve their performance when placed in the condition of recognizing knowledge, like with multiple-choice questions. As for the PK component, Dutch teachers score adequately on questions relative to the reasons to teach a certain topic, on the difficulties connected with the teaching of a certain topic and on students' prior knowledge required to learn a certain topic. While their performance worsen on the questions relative to extracurricular knowledge, factors influencing their teaching, teaching methods and ways of ascertain students' understanding.

In the same study, possible relations between teachers' PCK and their disciplinary background or textbook were sought. As for the disciplinary background, the results show a very weak relation between teachers' PCK and their disciplinary background. Teachers with beta or delta background are those who in average scored the best. As for the alpha and gamma teachers, there are those who mostly skipped questions especially on the pedagogical component, though it is not possible to infer the reasons for such choice. Moreover, alpha teachers are those who in average scored the worse. Quite surprisingly teachers with Informatics background scored quite similarly as teachers with a non-Informatics background on the Content Knowledge component in the context of multiple-choice questions. A reason could be that textbooks support those teachers on the CK component. Results of the study reported in chapter 4 in fact show that textbooks are strong on the CK component. However, in the context of open-ended question, Informatics teachers clearly score better than the other teachers, probably because their content knowledge is better and allows them to reproduce their knowledge. As for the relation between teachers' PCK and the textbook they use, there seems to be no strong relation, probably because the different textbooks had similar results with each other (chapter 4).

Summarizing, the results of this research project concern an understanding of the PCK of CS, first by analysing what is already present in literature, and then by portraying in depth the different aspects of the PCK of seven

topics, namely: control structures (with a focus on loops), decomposition of the problem, problem-solving skills, parameters, algorithms, data structures and array. Making use of this knowledge, it was possible to analyse Dutch textbooks, finding that the PCK reported in the text was satisfying, in terms of content knowledge, but poor in terms of pedagogical knowledge. Last, Dutch teachers' PCK was assessed, giving in this way a picture of the actual scenario and the prerequisites to suggest further development. Dutch teacher's PCK was found to be poor, especially in the context of open ended questions. Finding Dutch teachers' PCK poor is not of too much surprise, especially when considering the Dutch scenario described in chapter 1. These results echo the results found in Schmidt's survey about Dutch Informatics (Schmidt, 2007b), and contribute to evidencing where support can be offered.

6.3 Critical reflections

In this section critical reflections on two different aspects of this research project are presented. First reflection on the methodological choices are reported, by analyzing the different research methods and instruments of the studies reported in chapters 2 to 5. Lastly, reflection on the theory and how the research described in this research project contributes to a better understanding of the construct of PCK.

6.3.1 Methodological reflections

In hindsight of this research, reflection on the different methods used to answer the different research questions can be made, highlighting how the methodological choices, though vulnerable to critics, were taken to maximize sample size, analysis and results. One of the major assets of this research project is that a combination of mixed methods, both qualitative and quantitative, grants generalizability of the results (Baxter & Lederman, 1999), in contrast with the general trend in PCK research of focusing on a single method or small sample size. Details of the critical reflection on the methodology and the different methods follow.

The first methodological choice for the literature review reported in chapter 2, was to rely on scientific papers published on referred journals, as people who are active in CS education research suggest (Randolph, 2007; Lister, 2007).

However, the study was conducted by taking into account also papers published in conference proceedings. One might argue this choice, but it should be considered that for young disciplines, such as CS, conference proceedings papers are considered the fastest way to share results. Therefore, to embrace a wider and up to date choice of papers the method to choose literature needed to be broadened.

In chapter 3, the method used to portray the PCK of programming included the use of a research instrument, called CoRe (Content Representation) developed by Australian researchers (Loughran et al., 2000, 2001; Mulhall et al., 2003; Loughran et al., 2004), in the context of science education. This instrument is generally used in combination with PaP-eRs (Pedagogical and Professional-experience Repertoires) therefore someone might oppose that only the combination of a CoRe and more PaP-eRs, tackling the different aspects of a CoRe, would be considered as the full representation of PCK for a specific topic. PaP-eRs are narrative accounts of teachers' PCK for a particular piece of subject content. Each PaP-eR 'unpacks' the teacher's thinking around an element of PCK for that content, and is based on classroom observations (Mulhall et al., 2003) and interviews. The PCK portrayed from such a combination of instruments would though represent the PCK of an individual teacher (Loughran et al., 2000), being PaP-eRs narrative accounts of a single teacher's experience. But because the main goal of chapter 3 is to portray the PCK of programming for secondary school in a generalizable way, it was decided not to use PaP-eRs. Another reason not to include PaP-eRs was to keep the sample size considerably big, which could have not been assured in the context of an in-depth qualitative approach, such with PaP-eRs.

Also, one might argue that PCK is contextualized knowledge, therefore should be portrayed considering cultural backgrounds, as involving teachers from the same country. However, because the final goal of this research project is to measure Dutch teachers' PCK and because from Shulman's definition it was assumed that in the Netherlands there would be no teachers' with strong PCK of programming - whether because of lack of strong disciplinary background, because of lack of consistent teaching experience or both - it was needed to internationalize and generalize the results of chapter 3 by assuring a satisfying sample size, which is to be found abroad. The method of internationalizing taken was to have a heterogeneous group of participants taking part to semi-structured interviews organized in different countries. In this way it was assured that the resulting PCK would not be contextualized to a specific coun-

try, but generalized and reusable to measure Dutch teachers' PCK.

The content analysis study described in chapter 4 is the first in its kind in CSE research, in terms of the use of PCK as a framework to analyse teaching material. This means that no other research results were to be used to compare the results of chapter 4, implying that no stronger claims on the validity of the instrument are possible. The instrument has been assessed in terms of its content validity, construct validity (in terms of whether the components of the instrument are consistently linked to each other), practicality and effectiveness, and reliability (Nieveen, 1999). One might argue that the quality assessment method chosen is not complete, because is not possible to assess the instrument validity, but considering that there are no results from similar studies the choice can be considered justifiable.

The results of the study reported in chapter 5 are drawn from a combination of the data of the studies and the results of chapters 2 to 4. One might criticize that data gathered from semi-structured group interviews and online questionnaire are not comparable, because of the different nature of the two methods. A different way of assessing Dutch teachers' PCK could have been to conduct semi-structured interviews, individually or in group. However, this could have jeopardized attendance of a consistent number of teachers, both for practical and personal reasons. From a motivational point of view, Dutch CS teachers have mostly a different disciplinary background than CS, meaning that probably teachers could have feared taking part to an interview, whether individually or in a group format, and confront their knowledge with that of their colleagues. Also, it was taken into consideration that teachers' schedule is too busy and they might have not found the time to participate to meetings. Online questionnaires give teachers the opportunity to decide when and where to fill in the questionnaire. Also, it was possible to fill in the questionnaire at different moments. These choices were taken to give teachers as much freedom and confidence as possible, and to give the best prerequisites to obtain a suitable sample size for the study, so that results could be generalized. However, a risk about the heterogeneity of the sample might have been run, because it is possible that only teachers feeling confident in "programming" might have participated to the survey. However there is no data available to confirm this assumption.

As for the instrument, a comparison of teachers' PCK with that of the text-

books analysed in chapter 4 reveals that textbooks resulted in a relatively strong CK, while teachers scored mostly medium. A speculation on these results could be that the calibration of the instrument is not good enough to discern the difference between strong and medium, and therefore a possible relation between teachers' small PCK and the textbooks they use is missed. However, because no other instrument to measure CS teachers' PCK is available, it is not possible to assess the validity of the instrument from this perspective.

Though PCK has been defined more than 20 years ago, and literature around this topic and its importance has been proliferating, critics are made on what little is known about this concept in terms of its definition, content (Hill, Ball, & Schilling, 2008; Ball, Thames, & Phelps, 2008), and its operationalization to achieve measurement of this knowledge. As also confirmed in chapter 3, it is not easy to find literature on methods to capture and portray such knowledge and often only narrow content areas are explored. The important methodological contribution of this book is to present empirical results, obtained with mixed methods (semi-structured interviews, content analysis and online questionnaire), conceptualizing the PCK of a wide range of concepts within programming, an important topic of CS. Also the sample size and selection of this research project allows the generalizability of the results, which can be used in the context of different countries, as for example using the results of the international study reported in chapter 3, to the national study reported in chapter 5. The same cannot be said of the generalizability within CS, being the results of this research project specific of programming, for most aspects. The results of the studies reported in this book on the PCK cannot be generalized to other topics of CS. However the methodological approach can be generalized allowing further exploration and assessment of PCK of CS topic (e.g. ethics in CS, history of CS, databases, etc).
Also, because efforts need to be made in representing such knowledge to other actors (e.g. teachers, textbook authors, teacher trainers), other approaches could be incorporated in the process of representing such complex knowledge, as for example the "dimension doughnut" (Kinnunen, 2009). This is a model that helps to visualize complex educational realities and phenomena, and helps to highlight the different viewpoints of instructional processes by means of dimensions (e.g. teachers, students, topics to teach and teaching methods).

6.3.2 Theoretical reflections

Since its first definition in 1986, PCK has been attracting attention. In an interview, more than 20 years later, Shulman (Berry, Loughran, & Van Driel, 2008) renews his vision of this construct, reminding how teachers need strong disciplinary background in order to develop their PCK. One of the consequences of weak subject matter knowledge and a sense that one is weak in it, is that teachers might tend to be rigid in his/her teaching, leaving students little space to explore. In the context of this research project it has not been assessed whether students' taught by teachers' with weak PCK are affected in their learning, in comparison with those with stronger PCK. The data in chapter 5 suggests that teachers' with weak disciplinary background in general are also weak in PCK. However PCK is not the result of only disciplinary background. Though one might have a PhD in a subject, it does not automatically imply that the PCK would be strong (Berry et al., 2008). General pedagogical knowledge and years of experience are the other two key ingredients for the recipe. The latter is also suggested by the results presented in chapter 5, in which teachers, though with strong CS background, resulted in weak PCK. A reason could be these teachers' few teaching experience.

PCK has already been cited in CS education research (Ragonis & Hazzan, 2008), in the context of pre-service teaching trainings. In their approach, Ragonis and Hazzan suggest that student teachers should be fostered to broaden their PCK by expanding their perspectives on the field, and consequently, enhance the quality of their teaching. Among the issues that they mention is the question "what is CS?", a bird's-eye view of the discipline and familiarity with tools and methods for teaching. The major critics to such approach is that it refers to PCK of CS as if it was known, neglecting the need to conduct explorative research towards an understanding of it. A counterexample is Woollard's interest in metaphors as an expression of PCK (2005). In his paper he explores the role that metaphor plays in the teaching of computing, around a number of difficult topics, and relates the different kinds of metaphors to different kind of difficulties. The contribution of his research is that by combining his results with those reported in this book, CS education research community can move forward towards a full understanding of the PCK of computing.

A last theoretical reflection is given on the nature of PCK, which is considered to be of a complex nature and whose details are difficult to discern from each

other (Rohaan, 2009). In this research project it was shown that actually different aspects of a teacher's PCK are discernible, both in a unveiling (chapter 3) and in an assessment (chapter 5) context. In the different studies it was possible to recognize the content from the pedagogical knowledge, but also more specific aspects (e.g. extracurricular knowledge). However, if taken one by one, these different aspects would not represent a teachers' PCK. It is the combination of these that makes that special construct, known as Pedagogical Content Knowledge. PCK is indeed a complex construct, but its different aspects are recognizable and teachers can be assessed on these.

6.4 Practical Implications

Important practical implications derived from the results of this research project are the possibility for textbook and teaching material authors to use the knowledge about PCK of programming to (re-)design the content of their material. Considering that the scenario of having CS teachers with weak disciplinary background is not only a situation specific to the Netherlands, also teaching materials authors from other countries could be interested. Representation of the PCK of programming could be suitably arranged in teaching material in a way that teachers could retrieve such knowledge.

Practical implications arose from the results reported in chapter 5 would be directed at in-service Dutch teacher trainers. These could design tailor made solutions for Dutch CS teachers, aiming at the aspects of PCK which need support the most (e.g. extra-curricular knowledge, teaching methods and ways of ascertain students' understanding). An *ad hoc* solution could be the designing of guidelines for teachers to be found in teaching material (or support for teaching material). Textbook authors could use the results of this research to strengthen and widen the content of their teaching material. Also a web portal could be the devised where the information gathered from the results of this research project is organized in such a way that teachers can retrieve material and knowledge to strengthen their PCK of programming.

Another important practical implication of this thesis is the clarification of what should be included in a curriculum for a programming course at secondary school. In chapter 3 expert teachers from different countries agreed on the core topics that should be offered students when learning to program. These results can be used by Dutch curriculum authors and consequently also by textbook authors. As seen in chapter 4, Dutch textbooks authors write

their content following the Dutch curriculum guidelines, though they seemed not to cover some important topics within programming. Having a clear picture of the core concepts to teach, and have them reported in the curriculum and textbooks might help those teachers with weak disciplinary background in finding their way in the difficult path of teaching programming.
Also, teaching material authors targeting teachers with not well developed PCK could find in chapter 4 source of inspiration in terms of using the framework of PCK when writing textbooks, while in chapter 3 and in the technical report (Saeli et al., 2010) they can find useful resources about the different aspects of PCK of different programming concepts.

An important implication of this research project is the contextualization of CS education in secondary school and an understanding of its importance. In chapter 2 the important question of "why should we teach programming in secondary school" is answered. Though programming is only one of the topics of CS in an international context, it is one of the core subjects in which the important aspect of problem solving skills is learned, a skill that can be reused also in other domains. In the Netherlands, as introduced earlier, Informatics education is at crossroad, being its future in danger. The results reported in chapter 2 renew once more the importance of offering CS as school topic. In order to improve the teaching of this important subject, Dutch policy makers could direct future efforts in developing and improving teaching material, because as seen in chapter 4, teachers seem to benefit from support.

6.5 Suggestions for further research

Part of the research described in this book represents the very first effort made to portray the PCK of CS, which produced pioneering results. However, the work reported cannot be considered to be an exhaustive representation of PCK of CS, because in this research the only topic of CS explored is programming. A logical next step in research would be to portray the PCK of other topics within CS from an international perspective, using the research methods described in this book. Also there is the need to complete the work initiated in this book by portraying the PCK of other programming concepts, having only seven concepts been explored until now (chapter 3).
Also, the methodology used in this research results in a representation of PCK

that is decontextualized from any specific culture, country, district or school. A next step would be to conduct research to contextualize this knowledge to specific needs, as for example producing teaching material or teaching training courses material. A way of doing this could be to produce PaP-eRs in collaboration with teachers having strong PCK of the subject.
Because this study its the first of this explorative nature in CS, research should also be directed in developing other methods of portraying the PCK of CS, so that the results of this study can be compared and instruments validated.

In chapter 4 a content analysis study to assess teaching material was described. This study is the first in its nature, in terms of using PCK as framework to analyse teaching material. A next step would be to develop a method to assess the PCK of online teaching resources, which have a different format than textbooks, because CS teachers have often access to digital resources as for example the infrmaticavo.nl portal in the Netherlands. The method presented in chapter 4 explores index, table of content and chapters. Online resources might not have designed around these framework, therefore the need to develop an alternative method for such resources. One might suggest to develop an automated system to analyse textbooks in search for PCk representations. Though such a system would be very useful in the context of a country with several textbooks, the feasibility seems quite limited, due to the combined nature of this approach (quantitative and qualitative). Also, research might be also directed at developing an automated system that could analyse teaching material in search of PCK aspects.
A last aspect of this study that could be explored is whether teachers using teaching material designed around the PCK framework would be effectively helped in their teaching. The goal of this study was to assess whether it would be possible to find aspects of PCK of programming in teaching material. Whether and how teachers would use this information was not a goal and could be explored in further research, assuming teaching material is available.

The study reported in chapter 5 represents one of the several studies aimed at measuring teachers' PCK. Suggestion for those willing to conduct research using the same or similar instrument is to take into consideration other ways of involving participants to the study. The study described in this research involves voluntary and anonymous participants, implying that no follow up interviews, useful for example to deepen understanding for unclear answers or class observations were possible.

The practical implication relative to the study reported in chapter 5 is to provide research based results in order to design *ad hoc* solutions to improve the Dutch scenario. Research is needed in the direction of how to design *ad hoc* solutions to improve the Dutch scenario, as for example how to design such material.

A final suggestion is to involve students in similar research projects. It would be especially important to understand what the link is between teachers' PCK and students' understanding and motivation.

References

Abelson, H., & Sussman, G. J. (1996). *Structure and Interpretation of Computer Programs* (second ed.). Cambridge, Massachusetts: The MIT Press. Hardcover. Available from `http://mitpress.mit.edu/sicp/full-text/book/book.html`

Ahtineva, A. (2005). Textbook analysis in the service of chemistry teaching. *Universitas Scientiarum*, *10*, 25-33.

Akker, J. Van den, & Voogt, J. (1994). The use of innovation and practice profiles in the evaluation of curriculum implementation. *Studies in Education Evaluation*, *20*, 503–512.

Almstrum, V. L., Hazzan, O., Guzdial, M., & Petre, M. (2005). Challenges to computer science education research. In *Proceedings of the 36th sigcse technical symposium on computer science education* (pp. 191–192). New York, NY, USA: ACM.

An, S., Kulm, G., & Wu, Z. (2004). The Pedagogical Content Knowledge of Middle School, Mathematics Teachers in China and the U.S. *Journal of Mathematics Teacher Education*, *7*(2), 145–172.

Atchison, W. F., Conte, S. D., Hamblen, J. W., Hull, T. E., Keenan, T. A., Kehl, W. B., et al. (1968). Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on Computer Science. *Communications of the ACM*, *11*(3), 151–197.

Baeten, J. (2009). *Models of Computation: Automata and Processes.* Internal publication. Retrieved October 2011, from `http://www.win.tue.nl/~andova/education/2IT15/apbook11ver2009.pdf`

Ball, D. L., Thames, M. H., & Phelps, G. (2008). Content knoweldge for teaching - what makes it special? *Journal of Teacher Education*, *59*(5), 389-407.

Baxter, J. A., & Lederman, N. G. (1999). Assessment and Measurement of Pedagogical Content Knowledge. In J. Gess-Newsome & N. G. Lederman (Eds.), *Examining pedagogical content knowledge* (pp. 147–161). Dordrecht, the Netherlands: Kluwer Academic Publishers.

Bell, T., Witten, J. H., & Fellows, M. (1998). *Computer Science Unplugged... Off-line activities and games for all ages.* Retrieved April 2011, from `http://csunplugged.org/sites/default/files/activity_pdfs_full/CS_Unplugged-en-10.2006.pdf`

Berry, A., Loughran, J., & Van Driel, J. H. (2008). Revisiting the roots of pedagogical content knowledge. *International Journal of Science Education*, *30*(10), 1271-1279.

Breed, E. A., Monteith, & Mentz, E. (2005). *Effective Learning in Computer Programming: the Role of Learners' Reflective Thinking.* Proceedings of the 35 Years of Computers in Education: What Works? Proceedings of IFIP 8th World Conference on Computers in Education - WCCE 2005.

Carlson, R. E. (1990). Assessing Teachers' Pedagogical Content Knowledge: Item Development Issues. *Journal of Personnel Evaluation in Education*, *4*, 157–173.

Carpenter, T. P., Fennema, E., Peterson, P. L., & Carey, D. A. (1988). Teachers' Pedagogical Content Knowledge of Students' Problem Solving in Elementary Arithmetic. *Journal for Research in Mathematics Education*, *19*(5), 385-401.

Chiappetta, E., Sethna, G., & Fillman, D. (1993). Do middle school life science textbooks provide a balance of scientific literacy themes? *Journal of Research in Science Teaching*, *30*(7), 787-797.

Cochran, K. F., DeRuiter, J. A., & King, R. A. (1993). Pedagogical Content Knowing: An Integrative Model for Teacher Preparation. *Journal of Teacher Education*, *44*(4), 263–272.

Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., & Young, P. R. (1989). Computing as a discipline. *Commun. ACM*, *32*, 9–23.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. In *Proceedings of the 34th sigcse technical symposium on computer science education* (pp. 191–195). Reno, Navada, USA: ACM.

Curzon, P., & McOwan, P. (2008). Engaging with Computer Science through Magic Shows. *Paper presented at the 13th Annual Conference on Innovation and Technology in Computer Science Education*.

Dagiené, V. (2005). Teaching Information Technology in General Education: Challenges and Perspectives. In R. T. Mittermeir (Ed.), *From computer literacy to informatics fundamentals* (Vol. 3422, pp. 53–64). Berlin /

Heidelberg: Springer.

Dijkstra, E. W. (1968). A constructive approach to the problem of program correctness. *BIT Numerical Mathematics*, *8*(3), 174–186.

Dijkstra, E. W. (1972). Notes on Structured Programming. In O. J. Dahl, E. W. Dijkstra, & C. A. R. Hoare (Eds.), *Structured programming* (pp. 1–82). London and New York: Academic Press.

Dijkstra, E. W. (1987). Mathematicians and computing scientists: The cultural gap. *Abacus*, *4*(4), 26-31.

DuBoulay, B. (1989). Some Difficulties of Learning to Program. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 283–299). London: Lawrence Erlbaum Associates.

DuBoulay, B., O'Shea, T., & Monk, J. (1989). The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 467–446). London: Lawrence Erlbaum Associates.

Eurydice. (2007). *The education system in the netherlands 2007.* Retrieved August 2011, from `http://english.minocw.nl/documenten/en_2006_2007.pdf`

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1970). Programming-languages as a conceptual framework for teaching mathematics. *SIGCUE Outlook*, *4*, 13–17.

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., et al. (2007). Developing a computer science-specific learning taxonomy. *SIGCSE Bull.*, *39*, 152–170.

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. In R. Mittermeir (Ed.), *Informatics education – the bridge between using and understanding computers* (Vol. 4226, pp. 159–168). Berlin / Heidelberg: Springer.

Gal-Ezer, J., & Harel, D. (1999). Curriculum and Course Syllabi for a High-School CS Program. *Computer Science Education*, *9*(2), 114–147.

Ginat, D. (2006). On Novices' Local Views of Algorithmic Characteristics. In R. Mittermeir (Ed.), *Informatics education – the bridge between using and understanding computers* (Vol. 4226, pp. 127–137). Berlin / Heidelberg: Springer.

Good, R. (1993). Science textbook analysis - editorial. *Journal of Research in Science Teaching*, *30*(7), 619.

Govender, I. (2006). *Learning to Program, Learning to Teach Programming: Pre- and In-service Teachers' Experiences of an Object-oriented Language.* Unpublished doctoral dissertation, University of South Africa.

Grgurina, N. (2008). The first decade of informatics in dutch high schools.

Informatics in Education, *7*(1), 55-74.

Grossman, P. L. (1990). *The making of a teacher: Teacher knowledge and teacher education.* New York [etc.]: Teachers College Press, Columbia University.

Grossman, P. L., & Howey, K. R. (1989). A Study in Contrast: Sources of Pedagogical Content Knowledge for Secondary English. *Journal of Teacher Education*, *40*(5), 24–31.

Guzdial, M. (2004). Programming Environments for Novices. In S. Fincher & M. Petre (Eds.), *Computer science education research* (pp. 127–153). Lisse, The Netherlands: Taylor & Francis.

Hashweh, M. (1987). Effects of Subject-Matter Knowledge in the Teaching of Biology and Physics. *Teaching & Teacher Education*, *3*(2), 109–120.

Hashweh, M. (2005). Teacher pedagogical constructions: a reconfiguration of pedagogical content knowledge. *Teachers and Teaching*, *11*, 273-292.

Hill, H. C., Ball, D. L., & Schilling, S. G. (2008). Unpacking pedagogical content knwoledge: Conceptualizing and measuring teachers' topic-specific knowledge of students. *Journal for Research in Mathematics Education*, *39*(4), 372-400.

Holmboe, C., McIver, L., & George, C. (2001). *Research Agenda for Computer Science Education.* 13th Workshop of the Psychology of Programming Interest Group.

Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). *Identifying and correcting java programming errors for introductory computer science students* (Vol. 35) (No. 1). SIGCSE Proceedings of the 34th technical symposium on Computer science education, Reno, Nevada, USA. ACM Press.

Hromkovič, J. (2006). Contributing to General Education by Teaching Informatics. In R. Mittermeir (Ed.), *Informatics education – the bridge between using and understanding computers* (Vol. 4226, pp. 25–37). Berlin / Heidelberg: Springer.

Hulsen, M., Wartenbergh-Cras, F., Smeets, E., Uerz, D., Neut, Sontag, L., et al. (2005). *ICT in Cijfers - ICT-onderwijsmonitor studiejaar 2004/2005* (Tech. Rep.). Nijmegen/Tilburg: IVA - ITS.

Kinnunen, P. (2009). *Challenges of teaching and studying programming at a university of technology - viewpoints of students, teachers and the university.* Unpublished doctoral dissertation, Helsinki University of Technology.

Kölling, M. (1999a). The Problem of Teaching Object-Oriented Programming, Part 2: Environments. *Journal of Object-Oriented Programming*, *11*(9), 6-12.

Kölling, M. (1999b). The problem of teaching object-oriented programming, Part I: Languages. *Journal of Object-Oriented Programming*, *11*, 8-15.

Kölling, M., & Henriksen, P. (2005). Game programming in introductory courses with direct state manipulation. *SIGCSE Bulletin*, *37*, 59–63. Available from `http://dx.doi.org/10.1145/1151954.1067465`

Krathwohl, D. (2002). A revision of bloom's taxonomy: An overview. *Theory into practice*, *41*(4), 212-218.

Kromrey, J. D., & Renfrow, D. D. (1991). *Using Multiple Choice Examination Items to Measure Teachers' Content-Specific Pedagogical Knowledge.* Paper presented at the Annual Meeting of the Eastern Educational Research Association.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1989). A Study of the Development of Programming Ability and Thinking Skills in High School Students. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 83–112). Lawrence Erlbaum Associates.

Lapidot, T., & Hazzan, O. (2003). Methods of teaching a computer science course for prospective teachers. *SIGCSE Bulletin*, *35*(4), 29–34.

Linn, M. C., & Dalbey, J. (1989). Cognitive Consequences of Programming Instruction. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 58–62). London: Lawrence Erlbaum Associates.

Lister, R. (2007). The Randolph thesis: CSEd research at the crossroads. *SIGCSE Bulletin*, *39*, 16–18.

Long, T. J., Weide, B. W., Hollingsworth, J., Sitaraman, M., Edwards, S., Bucci, P., et al. (1997). *Providing intellectual focus to cs1/cs2.* Online Publication. Retrieved April 2011, from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.1947`

Loughran, J., Gunstone, R., Berry, A., Milroy, P., & Mulhall, P. (2000). Science Cases in Action: Developing an Understanding of Science Teachers' Pedagogical Content Knowledge. *Paper presented at the Annual Meeting of the National Association for Research in Science Teaching (73rd, New Orleans, LA, April 28-May 1).*

Loughran, J., Milroy, P., Berry, A., Gunstone, R., & Mulhall, P. (2001). Documenting Science Teachers' Pedagogical Content Knowledge Through PaP-eRs. *Research in Science Education*, *31*, 289–307.

Loughran, J., Mulhall, P., & Berry, A. (2004). In Search of Pedagogical Content Knowledge in Science: Developing Ways of Articulating and Documenting Professional Practice. *Journal of Research in Science Teaching*, *41*(4), 370–391.

Magnusson, S., Krajcik, J., & Borko, H. (1999). Nature, Sources, and Development of Pedagogical Content Knowledge for Science Teaching. In

J. Gess-Newsome & N. Lederman (Eds.), *Examining pedagogical content knowledge* (Vol. 6, pp. 95–132). Dordrecht: Springer Netherlands.

Mannila, L. (2007). Novices' progress in introductory programming courses. *Informatics in education*, *6*(1), 139–152.

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, *16*(3), 211–227.

Marks, R. (1990). Pedagogical Content Knowledge: From a Mathematical Case to a Modified Conception. *Journal of Teacher Education*, *41*(3), 3–11.

Mayer, R. E. (1989). The Psychology of How Novices Learn Computer Programming. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 129–159). London: Lawrence Erlbaum Associates.

Mazaitis, D. (1993). The Object-Oriented Paradigm in the Undergraduate Curriculum: A Survey of Implementations and Issues. *SIGCSE Bulletin*, *25*(3), 58–64.

McGuffee, J. W. (2000). Defining computer science. *SIGCSE Bulletin*, *32*, 74–76.

Means, W. H. (1988). A content analysis of ten introduction to programming textbooks. *ACM SIGCSE Bulletin*, *20*(1), 283-287.

MIT Museum - Hal Abelson. (2010). *Google app inventor.* Retrieved October 2011, from `http://museum.mit.edu/150/29`

Morine-Dershimer, G., & Kent, T. (1999). The Complex Nature and Sources of Teachers' Pedagogical Knowledge. In J. Gess-Newsome & N. G. Lederman (Eds.), *Examining pedagogical content knowledge* (pp. 21–50). The Netherlands: Kluwer Academic Publishers.

Mulder, F. (2002). INFORMATICA: van BETA- naar DELTA-discipline. *TINFON*, *11*(2), 48.

Mulhall, P., Berry, A., & Loughran, J. (2003). Frameworks for representing science teachers' pedagogical content knowledge. *Asia-Pacific Forum on Science Learning and Teaching*, *4*(2).

Murnane, J. S., & McDougall, A. (2006). Bad computer science in beginners programming courses: "considered harmful?". In D. Watson & D. Benzie (Eds.), *Proceedings of ifip-conference on imagining the future for ict and education.* Alesund, Norway.

Nieveen, N. (1999). Prototyping to reach product quality. In J. V. den Akker, R. M. Branch, K. Gustafson, N. Nieveen, & T. Plomp (Eds.), *Design approaches and tools in education and training* (p. 125-135). Dordrecht: Kluwer academic publishers.

on Information Technology Literacy, N. (1999). *Being Fluent with Information*

Technology. National Academy Press.

Overmars, M. (2004). Teaching computer science through game design. *Computer*, *37*(4), 81–83.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York, NY, USA: Basic Books, Inc.

Pea, R. D. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of educational computing research*, *2*(1), 25–36.

Pea, R. D., & Kurland, D. M. (1983). *On the cognitive Prerequisite of Learning Computer Programming* (Tech. Rep.).

Perkins, D., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1989). Conditions of learning in novice programmers. In E. S. J. Spohrer (Ed.), *Studying the novice programmer* (p. 261-279). Hillsdale, NJ: Lawrence Erlbaum.

Peterson, P. L., Fennema, E., Carpenter, T. P., & Loef, M. (1989). Teacher's Pedagogical Content Beliefs in Mathematics. *Cognition and Instruction*, *6*(1), 1–40.

Ragonis, N., & Hazzan, O. (2008). Disciplinary-pedagogical teacher preparation for pre-service computer science teachers: Rational and implementation. In R. T. Mittermeir (Ed.), *Lncs* (Vol. 5090, p. 253-264). Berlin / Heidelberg: Springer.

Ragonis, N., Hazzan, O., & Gal-Ezer, J. (2010). *A survey of computer science teacher preparation programs in israel tells us: 'computer science deserves a designated high school teacher preparation'.* SIGCSE'10 proceedings of the 41st ACM technical symposium on Computer science education.

Randolph, J. J. (2007). *Computer Science Education Research at the Crossroads: A Methodological Review of Computer Science Education Research: 2000-2005*. Unpublished doctoral dissertation, Utah State University.

Rayner, S., & Riding, R. (1997). Towards a Categorisation of Cognitive Styles and Learning Styles. *Educational Psychology*, *17*(1), 5–27.

Resnick, M., Maloney, J., Monroy-Hermández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Epistemology and Learning Group, E & L Memo No. 8. *Communications of the ACM*, *52*(11), 60-67.

Resnick, M., & Ocko, S. (1990). *Lego/logo: Learning though and about design.* (Vol. 8). Cambridge: MIT Media Laboratory.

Rich, Y. (1993). Stability and Change in Teacher Expertise. *Teacher & Teacher Education*, *9*(2), 137–146.

Roberts, E., Shackelford, R., LeBlanc, R., & Denning, P. J. (1999). Curriculum 2001: interim report from the acm/ieee-cs task force. *SIGCSE*

Bulletin, *31*, 343–344.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172.

Rohaan, E. J. (2009). *Testing Teacher Knowledge for Technology Teaching in Primary Schools.* Unpublished doctoral dissertation, Eindhoven University of Technology.

Rohaan, E. J., Taconis, R., & Jochems, W. (2009). Measuring teachers' pedagogical content knowledge in primary technology education. *Research in Science and Technological Education*, *27*(3), 327-338.

Romeike, R. (2008). What's my challenge? the forgotten part of problem solving in computer science education. In R. T. Mittermeir (Ed.), *Issep 2008, lncs 5090* (p. 122-133). Springer Berlin / Heidelberg.

Rovegno, I. C. (1992). Learning to teach in a field-based methods course: the development of pedagogical content knowledge. *Teaching & Teacher Education*, *8*(1), 69–82.

Saeli, M., Perrenet, J., Jochems, W., & Zwaneveld, B. (2010). *Portraying the Pedagogical Content Knowledge of Programming - The Technical Report.* Internal publication. Retrieved June 2011, from `http://teachingprogramming.esoe.nl/TechnicalReport/SPJZ_Technical_Report.pdf`

Saeli, M., Perrenet, J., Jochems, W., & Zwaneveld, B. (2011a). *Portraying the pedagogical content knowledge of programming.* (Submitted)

Saeli, M., Perrenet, J., Jochems, W., & Zwaneveld, B. (2011b). *Programming: Teaching material and pedagogical content knowledge.* (Submitted)

Saeli, M., Perrenet, J., Jochems, W., & Zwaneveld, B. (2011c). Teaching programming in secondary school: a pedagogical content knowledge perspective. *Informatics in Education*, *10*(1), 73-88.

Sanders. (1993). Secondary Science Teachers' Knowledge Base when Teaching Science Courses in and out of Their Area of Certification. *Journal of Research in Science Teaching*, *30*(7), 723–736.

Schmidt, V. (2007a). *Handreiking schoolexamen informatica havo/vwo - tweede fase (guidelines for sescondary school exam - second phase)* (Tech. Rep.). Enschede, The Netherlands.

Schmidt, V. (2007b). *Vakdossier 2007 informatica (dossier of the subject computer science 2007)* (Tech. Rep.). Enschede, The Netherlands: Slo, Stichting L.

Schoenfeld, A. H. (1979). Explicit heuristic training as a variable in problem-solving performance. *Journal for Research in Mathematics Education*, *10*, 173-187.

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, *15*, 4-14.

Shulman, L. (1987). Knowledge and teaching: Foundations of the new reform. *Harvard Educational Review*, *57*, 1-22.

Sims-Knight, J., & Upchurch, R. (1993). *Teaching software design: A new approach to high school computer science.* The Annual Meeting of the American Educational Research Association.

Soloway, E. (1986). Learning to program = learning to contruct mechanisms and expanations. *Communications of the ACM*, *29*, 9.

Soloway, E. (1993). Should We Teach Students to Program? *ACM Communications*, *36*(10), 21–24.

Stephenson, C., Gal-Ezer, J., Haberman, B., & Verno, A. (2005). *The New Educational Imperative: Improving High School Computer Science Education* (Tech. Rep.). Final Report of the CSTA Curriculum Improvement Task Force - February 2005.

Stylianidou, F., Ormerod, F., & Ogborn, J. (2002). Analysis of science textbook pictures about 'energy' and pupils' readings of them. *International Journal of Science Education*, *24*(3), 257-283.

Syslo, M., & Kwiatkowska, A. (2006). Contribution of informatics education to mathematics education in schools. In R. T. Mittermeir (Ed.), *Issep 2006, lncs 4226* (p. 209-219). Berlin / Heidelberg: Springer.

Szlávi, P., & Zsakó, L. (2006). Programming versus application. In R. T. Mittermeir (Ed.), *Issep 2006, lncs 4226* (p. 48-58). Berlin / Heidelberg: Springer.

Tucker, A. (2010). K-12 computer science: aspirations, realities and challenges. In *Issep 2010, lncs* (p. 22-34). -.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stepherson, C., & Verno, A. (2003). *A Model Curriculum for K-12 Computer Science* (Vol. Final Report of the ACM K-12 Education Task Forse Computer Science Curriculum Committee; Tech. Rep.).

Turner-Bisset, R. (1999). The Knowledge Bases of the Expert Teacher. *British Educational Research Journal*, *25*(1), 39–55.

UNESCO. (2002). *Information and communication technology in education. a curriculum for schools and programme of teacher development.* (J. Anderson & T. V. Weert, Eds.). France: Author. Paris.

Van Diepen, N. (2005). Elf redenen waarom programmeren zo moeilijk is (Eleven reasons why programming is so difficult). *TINFON*, *14*(4), 105–107.

Van Diepen, N., Perrenet, J., & Zwaneveld, B. (2011). Which Way with Informatics in High Schools in the Netherlands? The Dutch Dilemma.

Informatics in Education, *10*(1), 123-148.

Van Driel, J. H., Verloop, N., & Vos, W. de. (1998). Developing Science Teachers' Pedagogical Content Knowledge. *Journal of Research in Science Teaching*, *35*(6), 673–695.

Van Merriënboer, J. J. G., & Paas, F. G. W. C. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science*, *16*(3), 251–285.

Wang, H. (1998). *Science textbook studies reanalysis: Teachers "friendly" content analysis methods?* Annual Meeting of the National Association for Research in Science Teaching (71st, San Diego, CA, April 19-22, 1998).

Weigend, M. (2006). From intuition to programme. programming versus application. In R. T. Mittermeir (Ed.), *Issep 2006, lncs 4226* (p. 117-126). Berlin / Heidelberg: Springer.

Woollard, J. (2005). The implications of the pedagogic metaphor for teacher education in computing. *Technology, Pedagogy and Education*, *14*(2), 189-204.

Wu, C., Lee, G., & Lai, H. (2004). Using concept maps to aid analysis of concept presentation in high school computer. *Education and Information Technology*, *9*(2), 185-197.

Wu, C., Lin, J., & Lin, K. (1999). A Content Analysis of Programming examples in high School Computer Textbooks in Taiwan. *Journal of Computers in Mathematics and Science Teaching*, *18*(3), 225-244.

Appendices

Appendix A - Sub-Domain B3: SOFTWARE

Goal: The candidate possesses simple data types, program structures and programming techniques. For the core program (in Dutch 'kernprogramma', contains all domains and almost all sub-domains of the CS curriculum) students develop software needed for the functionality of a simple web application form. A simple web application contains at least one function for the data entry and at least one function for the presentation of the stored data, whether or not in edited form (sorted, grouped or aggregated). The corresponding programming language or languages have the following possibilities:
- a connection with a database - database instructions can gather from and offer to the database management system
- the result of a selection assignment in the database. */+
- The layout of a web page with (dynamic) in- and output possibilities.
The scripting languages for Web applications that are prevailing nowadays have at their disposal the named options, and are thus sufficiently suitable. As for the layout of a webpage HTML is sufficient. Furthermore, the core program comprehends an introduction to the object-oriented programming. From this programming paradigm the students controls the following concepts:
- Object
- Object-class
- Attributes and methods.

Students can apply these concepts in the development of an object-oriented or visual program. With the latter concept a program is designed with a graphical user interface. This interface contains screen objects as buttons, text boxes, combo boxes, and so on. The screen objects have attributes such as position, colour and label and methods that can handle user events. It is sufficient that in the core program students can connect the concept of object and can write a visual program. For vwo-students there a deepen program also learn:
- to write a OO-programme with logical objects (which are only in the memory of the computer and not as visible screen objects)
- to use collection of objects in the program (which consist of a collection of other objects, such as the array and the array list (Java).

In addition, vwo-students learn to 'neatly' program. This means that these students learn to program in a (OO)-language that enforces flexible typing.

Appendix B The OTPA

Background information

Please fill in the following fields:

TEACHING

For how many years have you been a teacher?

..... years

Are you teaching Informatics this school year?

☐yes

☐no

If yes, for how many years have you been teaching Informatics?

..... years

Are you teaching other subjects?

☐yes

☐no

If yes, which ones?

.....

EDUCATION

Did you study Informatics at university level?

☐yes

☐no

If yes, which level?

If no, which other subject did you study and its respective level?

.....

☐yes

☐no

Textbooks

Which one of these textbooks do you use?

☐Instruct
- ☐Java
- ☐Delphi
- ☐PHP
- ☐BlueJ
- ☐Visual Basic
- ☐VB. net

☐Enigma

☐Informatica Actief

☐other:.....

CK1 Component

Please, tick the concepts that you find are at the heart of learning programming for students at VO level. If you find the need to add more, please use the reserved space below the list.

- ☐Control structures: loops, conditions and sequential
- ☐Functions, Procures and methods
- ☐Algorithms
- ☐Variables and constants
- ☐Parameters
- ☐Pointers
- ☐HTML
- ☐Data structures
- ☐Decomposition (Problem solving)
- ☐Reusability
- ☐Arrays
- ☐ Logical thinking
- ☐Formal language: grammar and syntax
- ☐.....

CK2 Component

Please, cross the terms that you find are at the heart of learning control structures. If you would like to add more, please use the reserved space below the list

- ☐ conditions
- ☐ tag
- ☐ parameters
- ☐ selection
- ☐ typing
- ☐ iteration
- ☐ sequence
- ☐ database
- ☐ pointer
- ☐ if then if then else
- ☐ while - do while
- ☐ for choice switch case
- ☐ ...

PK Component

The following questions will help us understanding the different possible ways to teach one of the following topics: loops, data structures, arrays, problem solving skills, decomposition, parameters and algorithms. Please, give your answers to the following questions, but first choose the topic (only one) of your preference:

- ☐ loops
- ☐ data structures
- ☐ arrays
- ☐ problem solving skills
- ☐ decomposition
- ☐ parameters
- ☐ or algorithms

PK Component - open ended questions

Now please, provide an answer to the following questions:

- (CK3) What would you like your students to learn about this Big Idea?
Insert answer

- (PK2) Why is it important for the students to know this Big Idea?
Insert answer

- (PK3) What else you might know about this Big Idea (and you don't intend students to know yet)?
Insert answer

- (PK4) What are difficulties/ limitations connected with the teaching of this Big Idea?
Insert answer

- (PK5) What do you think students need to know in order for them to learn this Big Idea?
Insert answer

- (PK6) Which factors influence your teaching of this Big Idea?
Insert answer

- (PK7) What are teaching methods (any particular reasons for using these to engage with this Big Idea)?
Insert answer

- (PK8) What are your specific ways of ascertaining students understanding or confusion around this Big Idea?
Insert answer

Appendix C - PCK about Algorithms

Core about Algorithms
1. What do you intend the students to learn about this idea? To start programming in concrete programming languages for the future - Foundation for the future programming. - Managing to spot the sequence of actions which bring to the solution of the problem. - A correct description of algorithms allows to: boost (sharpens) logic skills; communicate the results of the job. - Utility to clarify a fixed sequence of actions which allows to obtain specific, concrete and correct results. - What it is, for what is it useful, representation, realization. The goal is to teach students that you need instructions to solve a problem, independently that it will be a computer to execute your instructions
4. Why is it important for the students to know this idea? - To write good programs. - Foundation for the future. - To autonomously cope with the analysis of a problem and propose a possible solution. - To give reliability to the job also in presence of difficulty of verification. - Because every programming problem has to be solved through the development of an algorithm. - To solve problems.
3. What else you might know about this idea (that you don't intend students to know yet)? - Difficult algorithms. - variety of formalities, definitions and specific tools. - Choice of particular structures. - Several personal exercises and teaching/professional experience. - Problems of complexity for resolving algorithms; non deterministic algorithms.
4. What are difficulties/ limitations connected with the teaching of this idea? - Algorithmic thinking is something new that student needs to understand. They will also find it in other subjects.

CoRe about Algorithms (continued)
- Autonomous ability of the student to analyze a problem. - The need to realize the job into a working experience. - Often students do not understand the need to develop an algorithm, until they will have to solve complex problems. At that point it will be too late to start clarifying their ideas. - To spot elementary actions of the executor
5. What do you think students need to know in order for them to learn this idea? - Mathematics logic knowledge. - Limits the curriculum, because there is the need to spend time on helping students understanding difficult topics. - Students aim to immediately obtain results. - Students think they are unimportant. There is the need to show them the actual utility of algorithms. - Choice of concrete cases to propose.
6. Which factors influence your teaching of this idea? - Motivation, Success, Feedback. - The use of formalism to present an algorithm. - Possibility to compare other areas of study. - Logical skills, knowledge of specific students reasons. - The interactive environments of programming languages give the opportunity to obtain results also without a proper preparatory development. In the past, the slowness of the machine would force programmers plan their actions, because they had only one chance per day. Nowadays students have possibility to try out their code as many times as they want.
7. What are teaching methods (any particular reasons for using these to engage with this idea)? - Give problem solving to good students, more examples,rRead algorithmic structures. - Analyzing examples with the students and suggestion of implementation. - Development of well known subjects to demonstrate the functionality of the method to represent the algorithm.

CoRe about Algorithms (continued)
- The idea would be to start for algorithms as they are and not relative to informatics. First of all students should have an idea of what the automatic functioning is. That in a prefixed way, from an input it will come an output. For example students are proposed some data and the output of an operation. Then students are asked to guess how the results have been obtained. This could be a way to let students discover that there is an internal logic, and they have to understand how it works. - Identifying algorithms in the everyday life before to work with algorithms to calculate something more abstract. TWO DIFFERENT APPROACHES - Flowcharts are not suitable for a better understanding of algorithms. - Flowcharts used next to the code help students to visualize what is the flow of the program
8. What are your specific ways of ascertaining students understanding or confusion around this idea? - Reactions, asking to write what happens in a given algorithm. - Exercises in full autonomy. - From small particulars and by asking specific questions. - Formalize algorithms from different domains, as for example cooking recipes. - Propose other examples in which students have to autonomously reach to identify algorithms.

Summary

Teaching Programming for Secondary School: a Pedagogical Content Knowledge Based Approach

Computer science is a subject that is increasingly being taught in schools at secondary level in the past twenty years. One of the reasons for this choice is that citizens of this century are surrounded by products and other results of Computer Science (CS) and are asked to develop an attitude and skills that allows them to be able to use them properly. Also, with the recent evolution of social media, music players and mobile phones, users are inspired to develop their own applications and the Internet offers a multitude of online tutorials. Teaching CS at secondary school means enabling new generation citizens to acquire skills to develop applications, but also much more such as: learning to recognize when, how and why CS can be used to address and solve all sort of problems (develop methods/instruments for concrete problems), viewing the possibilities and limits of CS, and understand the social and ethical aspects of users interacting with IT tools.

Because the teaching of CS at secondary school has only been relatively recently introduced, still discussion is open on agreeing what to teach, how to teach, with which reasons to teach and what students' difficulties to cope with. In other words the Pedagogical Content Knowledge (PCK), a construct

defined by Lee Shulman (1986). For Computer Science it is still in its very early infancy. This thesis focuses on this topic, with special attention on programming. PCK is that expertise that allows teachers to represent, in an effective way, the subject to their students, it is the special amalgam between general and specific pedagogical knowledge and content knowledge. It is a knowledge that grows with the years of teaching experience.

The general research aim of this thesis is to understand what the PCK of CS for secondary school is, with a special focus on the subject "programming", being programming one of the core subjects of CS. Knowledge of PCK will then be used to assess the Dutch situation, with special focus on Dutch CS textbooks and Dutch teachers. The final aim is to find tailor made solutions for the Dutch CS scenario, where CS risks to disappear from the secondary school curriculum due to several problems. Among the problems evidenced: most Dutch teachers have no solid disciplinary background, being mostly teachers from other disciplines re-trained to teach CS. To support these teachers *ad hoc* solution, it is necessary to understand the PCK of CS. To do so a preparatory literature study reveals to what extent the PCK of programming has been explored. Because no real effort to portray such knowledge has been done in CS before, an exploratory study has been designed and conducted with expert CS teachers in order to unveil this knowledge. With the knowledge about PCK gathered through this exploratory study, the PCK of textbooks and teachers in the Dutch scenario is evaluated. Summarizing, the research questions addressed in this thesis are:

1. To what extent is it possible to recognize aspects of Pedagogical Content Knowledge of programming for secondary education in current literature?

2. What is the Pedagogical Content Knowledge of programming in the context of secondary school education?

3. To what extent is it possible to identify the Pedagogical Content Knowledge of programming in Dutch secondary school textbooks?

4. What is Dutch teachers' Pedagogical Content Knowledge of programming for secondary school?

In chapter 2, with the title "Teaching programming in secondary school: a pedagogical content knowledge perspective" the first research question is answered. The goal of this literature study is to find a preliminary sketch of what the Pedagogical Content Knowledge of CS, with focus on programming, is. PCK has been defined as the knowledge that allows teachers to transform their knowledge of the subject into something accessible for their students. The four core questions to uncover this knowledge are: what are the concepts we need to teach programming?; how to teach this topic?; what are the reasons to teach programming?; and what are the most common difficulties/misconceptions students encounter while learning to program? Some of the answers found are, respectively: enhancing students' problem solving skills; programming knowledge and programming strategies; general problems of orientation; and possible ideal chains for learning computer programming. Because answers to the four questions are not connected with each other, PCK being an unexplored field in CS, there is need of research based efforts to study this field.

In chapter 3, entitled "Portraying the pedagogical content knowledge of programming for secondary school", the second research question is answered. The exploratory study aims at portraying teachers' PCK of programming for secondary education, from an international setting. Also possible differences between teachers' PCK portrayed in this study and the teaching theories and teaching practice found in the literature (see chapter 2) are investigated. Because of the insufficient number of Dutch teachers to participate in the study, it was necessary involving teachers from other countries, therefore the international setting. The data have been collected in four countries: Italy, Belgium, Lithuania and the Netherlands, using semi-structured group interviews. The results constitute the first effort to portray the PCK of programming for secondary education regarding the following seven topics: control structures (with focus on loops), decomposition of the problem, problem solving skills, parameters, algorithms, data structures and arrays. For each topic information such as reasons for teaching it, students' required prior knowledge and difficulties, teaching methods is revealed. The results were partly confirmed by literature about teaching theories and practical advice, but also complemented it.

In chapter 4, with the title "Programming: textbooks and Pedagogical Content Knowledge", the third research question is addressed. The scope of this study is to understand to what extent teachers with low PCK can find support

for their Pedagogical Content Knowledge (PCK) in textbooks. The results of a study in which PCK is used as framework to develop a research instrument to examine all three high school Dutch computer science textbooks, with special focus on programming, is reported in this chapter. PCK is in this study analysed in terms of its two components: pedagogical knowledge (PK) and content knowledge (CK). The results with respect to the textbooks have been compared with the results of the previous study (see chapter 3), in which experienced teachers from various countries were involved to portray the PCK of programming for secondary school. The expectations to find textbooks relatively strong on the CK, but weak on the PK aspect, are confirmed by the results.

In chapter 5, entitled "Programming: teachers and Pedagogical Content Knowledge", the answer to the fourth and last research question is found. In this chapter the research to assess Dutch teachers' PCK, with special focus on programming, is reported. For this study an online research instrument called Online Teacher PCK Analyzer (OTPA) was developed. It consists of an adaptation of Content Representation (CoRe) already introduced in chapter 3. The results show that Dutch teachers' PCK is scored between low and medium (in a scale between low, medium and high). Also, it is enquired whether there is any relation between teachers' PCK and the textbooks they use, by comparing the results of this study with those of a previous one (see chapter 4) where the PCK of textbooks was assessed. The results show that there is no strong relation, indicating that teachers using different textbooks generally have similar PCK. Finally, trends between teachers' PCK and their educational backgrounds, having most of the Dutch teachers a different background than Computer Science, is searched. The results show that also in this case there is no strong relation.

In chapter 6 the conclusions of this research project are reported, which include the standard of PCK for programming, a method to assess textbooks and an overview of the actual Dutch situation regarding the teaching of programming. An overview of the outcomes of this research is given, from the perspective of the different studies. Further, some critical reflections are presented with respect to the methodological choices which, though vulnerable to critics, were taken to maximize sample sizes or data collection, as for example: including conference papers in the literature review study, though in science education is good practice to refer only to journal papers; or preferring an

online questionnaire to measure Dutch teachers' PCK above interviews, which give a more in depth view of teachers' reasoning. Also theoretical reflections are given, in hindsight of the research conducted, such as the exclusion, from this research, to explore whether students benefit when taught by teachers with strong PCK. Lastly practical implications of this research and suggestions for further research are proposed, such as the possibility for textbook and teaching material authors to strengthen their material with the results of this research. Also Dutch teacher training centers can use the results of this study to understand which areas of Dutch teachers' PCK need more support.

Samenvatting

Onderwijs in programmeren in het voortgezet onderwijs: een benadering vanuit de Pedagogical Content Knowledge

Informatica is een vak dat de laatste 20 jaar meer en meer onderwezen wordt in het voortgezet onderwijs. Een van de redenen hiervoor is dat men in de huidige maatschappij omgeven is door producten en andere resultaten van informatica en dat men gevraagd wordt een houding en vaardigheden te ontwikkelen om deze op de juiste wijze te gebruiken. Bovendien, met de recente ontwikkeling van sociale media, muziekspelers en mobiele telefoons, worden de gebruikers genspireerd hun eigen applicaties te ontwikkelen, waarvoor op Internet een veelheid aan handleidingen online beschikbaar is. Informatica onderwijzen in het voortgezet onderwijs betekent een nieuwe generatie vaardig maken in het ontwikkelen van applicaties. Maar het betekent nog veel meer, bijvoorbeeld dat deze generatie leert te herkennen wanneer, hoe en waarom informatica gebruikt kan worden om allerlei soorten problemen op te lossen (methoden/instrumenten te ontwikkelen voor concrete problemen), dat deze generatie leert de mogelijkheden en beperkingen van informatica te zien en leert de sociale en ethische aspecten te begrijpen van gebruikers in interactie met ICT-middelen.

Aangezien het informaticaonderwijs recentelijk in het voortgezet onderwijs is gentroduceerd, is de discussie nog gaande over wat precies onderwezen moet worden, hoe precies onderwezen moet worden en om welke redenen, en met welke moeilijkheden voor de leerlingen rekening gehouden moet worden. Met andere woorden: de Pedagogical Content Knowledge (PCK), een begrip gedefinieerd door Lee Shulman (1986). Voor het vak informatica staat de PCK nog in haar kinderschoenen. In deze dissertatie staat de PCK van informatica centraal, specifiek gericht op programmeren. PCK is die expertise waardoor docenten het vak aan hun leerlingen op effectieve wijze kunnen laten zien; het is die speciale samensmelting van algemene en specifieke pedagogische kennis en vakinhoudelijke kennis. Het is kennis die toeneemt met de jaren van onderwijservaring.

Het algemene onderzoeksdoel van deze dissertatie is te begrijpen wat de PCK voor informatica voor het voortgezet onderwijs inhoudt, waarbij de focus ligt op een van de centrale onderwerpen van de informatica, programmeren. De vergaarde kennis van de PCK van informatica zal dan gebruikt worden om de Nederlandse situatie te meten, in het bijzonder de Nederlandse schoolboeken en de Nederlandse docenten. Het uiteindelijke doel is goed passende oplossingen te vinden voor het Nederlandse informaticascenario, waar door verscheidene problemen het risico bestaat dat informatica uit het curriculum verdwijnt. Een van de gebleken problemen is dat de meeste Nederlandse docenten geen stevige disciplinaire achtergrond hebben, daar ze voor het merendeel uit een andere discipline afkomstig zijn en voor het onderwijs in de informatica alleen een nascholing hebben gevolgd. Het begrijpen van de PCK van de informatica is een directe oplossing om deze docenten te ondersteunen. Daartoe is een literatuurstudie gedaan om te laten zien in hoeverre de PCK van programmeren is onderzocht. Omdat er nog geen pogingen waren ondernomen om deze kennis te portretteren is een exploratief onderzoek opgezet en uitgevoerd met ervaren docenten informatica om deze kennis te openbaren. Met de door deze exploratieve studie vergaarde kennis van de PCK, werd de PCK van schoolboeken en van docenten in de Nederlandse context gevalueerd. Samengevat zijn de onderzoeksvragen van deze dissertatie de volgende:

1. In hoeverre is het mogelijk aspecten van Pedagogical Content Knowledege voor programmeren in het voortgezet onderwijs te vinden in de actuele literatuur?

2. Wat houdt de Pedagogical Content Knowledge in voor programmeren in het voortgezet onderwijs?

3. In hoeverre is het mogelijk de Pedagogical Content Knowledge voor programmeren te identificeren in Nederlandse schoolboeken voor het voortgezet onderwijs?

4. Wat is de Pedagogical Content Knowledge van docenten voor programmeren in het Nederlands voortgezet onderwijs?

In hoofdstuk 2, getiteld "Het onderwijzen van programmeren in het secundair onderwijs vanuit een pedagogical content knowledge perspectief", wordt de eerste onderzoeksvraag beantwoord. Het doel van deze literatuurstudie is een voorlopige schets van wat de Pedagogical Content Knowledge (PCK) met de focus op programmeren inhoudt. PCK is hier gedefinieerd als de kennis die het docenten mogelijk maakt hun vakinhoudelijk weten zodanig om te vormen dat het voor leerlingen toegankelijk wordt. De vier kernvragen om deze kennis aan het licht te brengen, zijn achtereenvolgens: welke begrippen zijn er nodig om programmeren te onderwijzen?; hoe moet programmeren onderwezen worden?; wat zijn de redenen om programmeren te onderwijzen?; en welke zijn de meest voorkomende moeilijkheden/misconcepties bij leerlingen die leren programmeren?. Enkele van de gevonden antwoorden zijn achtereenvolgens: het vergroten van de probleemoplosvaardigheden van de leerlingen; programmeerkennis en programmeerstrategien; algemene orintatieproblemen; en mogelijke ideale leerlijnen voor programmeren. De antwoorden op de vier vragen kunnen niet met elkaar verbonden worden, omdat de PCK in de informatica nog niet onderzocht is; gedegen onderzoek is op dit gebied nodig.

In hoofdstuk 3, getiteld "Een schets van de pedagogical content knowledge voor programmeren in het voortgezet onderwijs", wordt de tweede onderzoeksvraag beantwoord. Deze exploratieve studie is erop gericht de PCK van docenten voor programmeren in het voortgezet onderwijs te portretteren vanuit een internationale setting. Ook worden mogelijke verschillen onderzocht tussen de PCK van docenten uit deze studie en onderwijstheorien en onderwijspraktijk uit de literatuur (zie hoofdstuk 2). Omdat voor deze studie te weinig Nederlandse docenten geschikt waren, was het nodig docenten van andere landen erbij te betrekken, vandaar de internationale setting. De gegevens

zijn verzameld in Itali, Belgi, Litouwen en Nederland, door middel van semigestructureerde groepsinterviews. De resultaten van deze eerste poging de PCK van programmeren in het voortgezet onderwijs in kaart te brengen, betreffen de volgende zeven onderwerpen: controlestructuren (met de nadruk op loops), probleemdecompositie, probleem-oplosvaardigheden, parameters, algoritmes, datastructuren en arrays. Voor ieder onderwerp is informatie verzameld, zoals de redenen om het topic te onderwijzen, de bij de leerlingen vereiste voorkennis, moeilijkheden van leerlingen en onderwijsmethoden. De resultaten kwamen deels overeen met onderwijstheorien en praktische adviezen uit de literatuur, deels waren ze aanvullend.

In hoofdstuk 4 met de titel "Programmeren: leerboeken en Pedagogical Content Knowledge" wordt de derde onderzoeksvraag behandeld. Het doel van deze studie is te begrijpen in hoeverre docenten met geringe PCK daarin door leerboeken ondersteund kunnen worden. In dit hoofdstuk worden de resultaten beschreven van een studie waarin met PCK als raamwerk een instrument wordt ontwikkeld om de drie Nederlandse schoolleerboeken voor informatica te onderzoeken met de focus op programmeren. PCK wordt in deze studie gesplitst in haar twee componenten: pedagogische kennis (PK) en vakinhoudelijke kennis (CK). De resultaten met betrekking tot de leerboeken zijn vergeleken met die van de eerdere studie (zie hoofdstuk 3) waarin ervaren docenten van diverse landen participeerden bij het in kaart brengen van de PCK van programmeren in het voortgezet onderwijs. De verwachting dat leerboeken relatief sterk zouden zijn op het CK-aspect, maar zwak op het PK-aspect, werd door de resultaten bevestigd.

In hoofdstuk 5, getiteld "Programmeren: docenten en Pedagogical Content Knowledge", wordt het antwoord gevonden op de vierde en laatste onderzoeksvraag. In dit hoofdstuk wordt verslag gedaan van het meten van de PCK, met de focus op programmeren, van Nederlandse docenten. Voor deze studie werd een online instrument ontwikkeld, de Online Teacher PCK Analyzer (OTPA). Het bestaat uit een aanpassing van het al in hoofdstuk 3 beschreven instrument Content Representation (CoRe). De resultaten laten zien dat Nederlandse docenten op PCK tussen laag en gemiddeld scoren (op een schaal van laag-gemiddeld-hoog). Ook werd onderzocht of er een verband was tussen de PCK van een docent en het gebruikte leerboek door de resultaten van deze studie te vergelijken met die van de voorgaande (zie hoofdstuk 4) waarin de PCK van leerboeken werd gemeten. De resultaten geven aan

dat er geen sterke relatie is; docenten die verschillende leerboeken gebruiken hebben een in het algemeen vergelijkbare PCK. Aangezien de meeste Nederlandse docenten een andere achtergrond hebben dan informatica konden verbanden onderzocht worden tussen deze achtergrond en de PCK. De resultaten gaven aan dat er ook in dit geval geen sterke relatie is.

In hoofdstuk 6 worden de algemene conclusies van dit onderzoeksproject gerapporteerd, onder meer de standaard PCK voor programmeren, een methode om leerboeken te meten en een beeld van de actuele Nederlandse situatie ten aanzien van het programmeeronderwijs. Er wordt een overzicht gegeven van de resultaten vanuit het perspectief van de verschillende studies. Vervolgens worden enkele kritische reflecties gepresenteerd betreffende de gemaakte keuzes voor de onderzoeksmethode. Alhowel men kritiek kan hebben op de gemaakte keuzes, zijn ze vooral gemaakt met het oog op maximalisatie van zowel de steekproefgrootte als van hoeveelheid vezamelde data. Voorbeelden van deze keuzes zijn het opnemen van conferentiepapers in de review van de literatuur, hoewel het in het onderzoek van science onderwijs goed gebruik is alleen naar tijdschriftartikelen te verwijzen, en het prefereren van een online vragenlijst om de PCK van Nederlandse docenten te meten, boven interviews die meer diepgaand inzicht zouden hebben gegeven op de gedachtegang van de docenten. Er wordt, terugkijkend op het uitgevoerde onderzoek, ook een theoretische reflectie uitgevoerd, bijvoorbeeld op het niet meenemen van onderzoek naar het profijt wat leerlingen hebben van onderwijs door docenten met een hoge PCK. Tenslotte worden praktische implicaties van het onderzoek genoemd en suggesties gegeven voor verder onderzoek. Auteurs van leermateriaal kunnen bijvoorbeeld hun materiaal versterken met de resultaten van dit onderzoek. Ook kunnen centra voor docententraining uit de resultaten van dit onderzoek afleiden welke gebieden van de PCK van Nederlandse docenten meer ondersteuning nodig hebben.

Curriculum Vitae

Mara Saeli was born on 26 February 1981 in Messina, Italy. After finishing degree high school in 1999 at Liceo Scientific "'G. Seguenza"' in Messina (Italy), she studied a Computer Science Bachelor at Universita' di Ferrara in Ferrara (Italy) and a Mathematics and Science Education Master at Universiteit van Amsterdam (The Netherlands). Her Master dissertation on Action Research was carried out in a Secondary School in Messina (Italy). From 2008 she started a PhD project at Technische Universiteit van Eindhoven at Eindhoven (The Netherlands) of which the results are presented in this dissertation. Since 2008 she is employed at Technische Universiteit van Eindhoven.

List of Pubblications

Hamer, J., Cutts, Q.I., Jacková, J., Luxton-Reilly, A., McCartney, R., Purchase, H.C., Riedesel, C., Saeli, M., Sanders, K. & Sheard. J. (2008). Contributing student pedagogy. SIGCSE Bulletin, 2008: 194 212.

Saeli, M. (2008). International innovations in Computer Science Education: impressions of the ITiCSE conference (original title: Internationale innovatie in informatica-onderwijs: impressie van de ITiCSE-conferentie). TINFON, 17, 3, 68-70.

Saeli, M. (2009). Planting the seeds of Action Research for the revitalization and professionalism of Mathematics teachers. Quaderni di Ricerca in Didattica (original title: Notebooks of Research in Education), 19, 83-100.

Saeli, M. (2009). How to teach programming in secondary education: first results of a PhD project. In Proceedings of ITiCSE'2009.

Saeli, M., Perrenet, J., Jochems, W.M.G.m Zwaneveld, B. (2010). Portraying the Pedagogical Content Knowledge of Programming The Technical Report. Internal publication. Available at: teachingprogramming.esoe.nl/TechnicalReport/SPJZ_Technical_Report.pdf

Saeli, M., Perrenet, J., Jochems, W.M.G.m Zwaneveld, B. (2011). Teaching Programming in Secondary School: a Pedagogical Content Knowledge Perspective. Informatics in Education, vol. 10, no 1, 73-88.

Eindhoven School of Education

PhD dissertation series

Sande, R.A.W. van de. (2007). *Competentiegerichtheid en scheikunde leren: over metacognitieve opvattingen, leerresultaten en leeractiviteiten.*

Hooreman, R. (2008). *Synchronous coaching of trainee teachers: an experimental approach.*

Rajuan, M. (2008). *Student teachers perceptions of learning to teach as a basis for supervision of the mentoring relationship.*

Raessens, B.A.M. (2009). *De E-kubus: een analysemodel voor curricula.*

Rohaan, E.J. (2009). *Testing teacher knowledge for technology teaching in primary schools.*

Oemar Said, E. (2009). *De Da Vinci Case: een onderzoek naar de relaties tussen vernieuwende leeromgevingen en de motivatie en regulatievoorkeuren van leerlingen in het MBO.*

Koopman, M. (2010). *Students' goal orientations, information processing strategies and knowledge development in competence-based pre-vocational secondary education.*

Mittendorff, K.M. (2010). *Career conversations in senior secondary vocational education.*

Crasborn, F.J.A.J. en Hennissen, P.P.M. (2010). *The skilled mentor. Mentor teachers' use and acquisition of supervisory skills.*

Van Bragt, C.A.C. (2010). *Students' educational careers in Higher Education: A search into key factors regarding study outcome.*

Bakker, G. de. (2010). *Allocated only reciprocal peer support via instant messaging as a candidate for decreasing the tutoring load of teachers.*

Vos, M.A.J. (2010). *Interaction between teachers and teaching materials: on the implementation of context-based chemistry education.*

Bruin-Muurling, G. (2010). *The development of proficiency in the fraction domain.*