ANKARA UNIVERSITY 1946

Local Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution and Perspectives

ISSEP 2014

Selected Papers

22-25 September 2014 İstanbul,Turkey

Yasemin Gülbahar
Erinç Karataş
Müge Adnan (Eds.)

Local Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution and Perspectives

ISSEP 2014

Selected Papers

22-25 September 2014, Istanbul, Turkey

Yasemin Gülbahar
Erinç Karataş
Müge Adnan (Eds.)

Ankara University 2014
http://en.ankara.edu.tr/

Print: Ankara University Press
http://basimevi.ankara.edu.tr/

EDITORS
Yasemin GÜLBAHAR, Ankara University
Erinç KARATAŞ, Ankara University
Müge ADNAN, Muğla University

COVER DESIGN
Çağrı BÜYÜKÇAYLI

ISBN 978-605-136-171-0
e-ISBN 978-605-136-172-7

Typesetting: Camera-ready by author, data conversion by Ankara University Press, Ankara, Turkey

Available online at http://www.issep2014.org/lp-book/

TÜBİTAK

This conference is supported by TUBITAK (The Scientific & Technological Research Council of Turkey) 2223-B Support Program for Scientific Activities with applied number of 1929B021401286 (http://www.tubitak.gov.tr/sites/default/files/2223b-2014-3.pdf)

Baskı
ANKARA ÜNİVERSİTESİ BASIMEVİ
İncitaşı Sokak, No: 10
06510 Beşevler, Ankara / TÜRKİYE
Tel: +90312 213 66 05
Basım Tarihi:

”This proceeding is dedicated to Roland Mittermeir, our dear colleague
and mentor who passed away too early”

Preface

The International Conference on Informatics in Schools: Situation, Evolution and Perspective (ISSEP), co-hosted by Ankara University and Istanbul University in Istanbul, Turkey, is the seventh in the series of ISSEP conferences. Although the terms 'Information and Communication Technologies' (ICTs) and 'Informatics' are used interchangeably by different countries, ICT is the more acknowledged term used in Turkey, and the main focus of this conference is informatics and computer science education. Hence, the aim of this conference is to set up collaboration between researchers and practitioners in the areas of informatics education and computer science education in schools, with a different focus each year.

We live in a culturally rich world, which brings different approaches for teaching similar concepts. The focus of this conference, either labelled as computer science or informatics, is to provide different insights about teaching and learning perspectives of the phenomenon. Thus, this local proceedings will be including papers about Computer Science Education, Competence Measurement for Informatics, Emerging Technologies and Tools for Informatics, Teacher Education in Informatics and Curriculum Issues.

The book consists of 10 contributed papers, selected from 33 submissions, seven short papers as poster presentations, and four workshop descriptions. Altogether, ISSEP 2014 encompassed presentations covering research and theoretical papers, best practice and country reports, and work in progress and discussion papers, together with short papers and workshops from 12 different countries.

Proceedings under the "**Computer Science Education**" section has two papers. The first paper, by Maciej Borowiecki and Katarzyna Oledzka, is about a coding school planned for teachers who need support in conducting classes on the basics of programming. The second paper, by Mareen Przybylla and Ralf Romeike, reveals the findings of a survey investigating students' perceptions of their informatics classes, computing systems and embedded systems as informatics devices.

In the section on "**Competence Measurement for Informatics**", there are also two papers. Thomas Schiller and Peter Micheuz present concrete approaches for teaching information security within existing frameworks and reference models, and then focus on the current situation in Austria. Then Ján Gunčaga and Mária Karasová evaluate several issues regarding informatics education in primary schools in Slovakia; showing examples and discussing their positive and negative aspects.

Innovative use of technologies and tools are addressed in the section named "**Emerging Technologies and Tools for Informatics**". In their paper, Mahdi Miled, Christophe Reffay and Mona

Laroussi share preliminary results of a large-scale experiment about the use of a prototype called HiPPY (an epistemic HyPermedia to learn PYthon language). This is a tool which is currently integrated in the France-IOI platform to study solving strategies mainly within French high schools. Then we have Michael Weigend, who discusses how to use metaphors from everyday life to elaborate spreadsheet-related concepts, and presents findings from two paper-pencil-and-pencil exercises.

In this book, specific issues about "**Teacher Education in Informatics**" are discussed by Erik Barendsen, Valentina Dagienė, Mara Saeli and Carsten Schulte. The researchers focus on the Content Representation (CoRe) format as an instrument to elicit teachers' Pedagogical Content Knowledge (PCK) in the area of computer education.

The last topic is "**Curriculum Issues**". Based on successful lesson scenarios for computer science described as a pattern network, Bernhard Standl and Wilfried Grossmann combine these patterns with the Austrian standards, and propose a structure for how the patterns can be used for the application of pedagogical-content knowledge in computer science teaching. Then Sergei Pozdniakov and Svetlana Gaisina analyse perspectives of the integration processes of mathematics and informatics, and ways for a smooth transition to the Russian informatics curricula. In the last paper of this section, Filiz Kalelioğlu, Yasemin Gülbahar, Sümeyra Akçay and Dilek Doğan provide implementation suggestions for the integration of Scratch into existing ICT curriculum, based on prior research of the phenomenon, and discuss Scratch in terms of its possible contribution to students' computational thinking skills.

The book includes another section for "**Poster Presentations**", where short summaries of posters are provided. We have seven poster presentations, with various interesting topics in this section. Kadir Burak Olgun, Gonca Kizilkaya Cumaoğlu and Sevinç Gülseçen investigate the effects of a programming course on middle school students' reflective thinking skills towards problem solving, and then Mehmet Fatih Erkoç and Sevinç Gülseçen research the Effect of Collaborative Game Design on Critical Thinking, Problem Solving and Algorithm Development Skills. In her presentation, Sebnem Özdemir discusses the contributions of Universities to Children's Informatics Education in Turkey, and Gaisina Svetlana Valer'evna analyses the quality of teaching computer science and ICT in St. Petersburg. Following those, Andreas Grillenberger and Ralf Romeike focus on data management issues, underlining that it is more than a matter of computer science. Then Ilya Gosudarev writes about broadcasting web-development, mobile teaching and learning, while Cem Turan looks at quality in education by questioning if it is technology by mechanicity, or methodology by humanity.

Short summaries of the workshops held are provided under the "**Workshops**" section. Peter Micheuz offers a workshop about "Educational Standards in Informatics/ICT at Lower Secondary Level", whereas Orçun Madran provides an insight about "Robotic Programming for Teaching Programming Languages". Natasa Grgurina and Erik Barendsen share information about "The State of Affairs in Dutch Informatics Education" and Michael Weigend presents hands-on activities for teaching "Python".

This conference event and book have been organized to highlight the importance of Informatics Education. It is my sincere pleasure to thank all those who have contributed to ISSEP 2014 for their academic insights. In addition to those already mentioned, there are many people who also contributed to the preparation, organisation, communication, as well as the reviewing and publishing processes for this conference. I would like to extend my gratitude and sincere thanks to members of the organising and programme committee for their time and conscientious work. Special thanks also to Erinç Karataş and Müge Adnan for their help with the proceedings, and finally, a special thanks to Ira Diethelm, who provided guidance throughout the whole conference process.

June 25 2014 Yasemin Gülbahar

Conference Organization

Conference Chair

Yasemin Gülbahar (Co-Chair)
Sevinç Gülseçen (Co-Chair)

Program Committee

Müge Adnan	Mugla University, Turkey
Ayfer Alper	Ankara University, Turkey
Bahar Baran	Dokuz Eylul University, Turkey
Torsten Brinda	Universität Duisburg-Essen, Germany
Valentina Dagiene	Vilnius University, Lithuania
Ira Diethelm	Oldenburg University, Germany
Çiğdem Erol	Istanbul University, Turkey
Nuray Gedik	Akdeniz University, Turkey
David Ginat	Israel Institute of Technology, Israel
Yasemin Gulbahar	Ankara University, Turkey
Jan Guncaga	Catholic University in Ruzomberok, Slovakia
Sevinç Gülseçen	İstanbul University, Turkey
Juraj Hromkovič	ETH Zürich, Switzerland
Ivan Kalaš	Comenius University in Bratislava, Slovakia
Erinç Karataş	Ankara University, Turkey
Janka Majherova	Catholic University in Ruzomberok, Slovakia
Roland Mittermeir	Alpen-Adria-Universität Klagenfurt, Austria
Ferhan Odabasi	Anadolu University, Turkey
Malgorzata Pankowska	University of Economics in Katowice, Poland
Zerrin Ayvaz Reis	Istanbul University, Turkey
Ralf Romeike	Friedrich-Alexander University Erlangen-Nürnberg, Germany
Carsten Schulte	Freie Universität Berlin, Germany
Sue Sentence	Computing At School Cambridge, UK
Simon Simon	University of Newcastle, UK
Maciej Syslo	UMK Torun, U. Wroclaw, Poland
Erkan Tekinarslan	Abant Izzet Baysal University, Turkey
Sacip Toker	Mehmet Akif Ersoy University, Turkey
Pelin Yüksel	Inonu University, Turkey
Erman Yükseltürk	Kirikkale University, Turkey
Soner Yildirim	Middle East Technical University, Turkey
Recep Çakir	Amasya University, Turkey

Additional Reviewers Winczer, Michal.

Organizing Committee

Yasemin Gülbahar Co-chair Ankara University, Turkey
Sevinç Gülseçen Co-chair Istanbul University, Turkey
Soner Yildirim Middle East Technical University, Turkey
Erinç Karataş Ankara University, Turkey
Müge Adnan Mugla University, Turkey
Orçun Madran Hacettepe University, Turkey

Secretariat

Hale Ilgaz Ankara University, Turkey
Şebnem Özdemir Istanbul University, Turkey

Table of Contents

Computer Science Education

Coding School for 10-12 aged students

Maciej Borowiecki and Katarzyna Olędzka

Computer Assisted Education and Information Technology Centre, Warsaw, Poland
{maciej.borowiecki, katarzyna.oledzka}@oeiizk.waw.pl

Abstract. There are many examples all over the world which show that learning programming is possible from early stages. However, it is not enough to equipped schools with modern IT devices, but teachers need support in conducting classes on how to introduce the basics of programming. Project *Coding School* tries to meet this expectations.

The 8 lessons with applications in Scratch have been prepared not only to show the basics of programming, but also to engage students to think creatively. At first, teachers created these applications during courses, next they conduct programming classes in their schools sharing knowledge and enthusiasm with students. The project is realised in cooperation with Computer Assisted Education and Information Technology Centre, Centre for Citizenship Education, Association Parents in Education with support of Samsung Company. In first part of the project 1300 students have participated. Teachers have been from 34 schools from all over Poland.

Keywords: programming, primary school, Coding School, Scratch

1 Introduction

Polish students won many awards in international competitions and Olympiads in informatics, but they are only an elite. An average student has less ability [1]. There is a separate subject called computer classes (primary school) and informatics (middle schools and high schools) in Polish schools. Each curriculum contains records of algorithms and programming, but by many teachers it is treated as difficult materials and they frequently reduce time spend on these topics or even omit this part. The reason is that they do not feel to be prepared to teach algorithms and programming.

The main objective of the *Coding School* project [6] is popularization programming in Polish schools, with particular attention to primary school pupils. In addition, there is assumption that these activities can be conduct both by computer classes teachers and by teachers of different subjects. The third principle of this project is that these lessons should be attractive to students.

Starting from these assumptions, the initiator of the project Samsung company, invited partners from both public organizations and non-governmental ones. The project is realised in cooperation with Computer Assisted Education and Information Technology Centre, Centre for Citizenship Education, Association Parents in Education. The project is under patronage of the Ministry of National Education and the Ministry of Administration and Digitization.

Fig. 1. Logo of *Coding School* (in Polish *Mistrzowie kodowania*)

2 About the project

The pilot edition of the project was organized for primary school students aged 10 to 12 years. In the project also participated younger students. Students have been invited from 34 schools all over Poland – 2 from each province and 2 additional ones from Warsaw. They represented both urban and rural schools, large and small – the smallest school had less than 100 students.

The Scratch was chosen as a programming language for this project because of its features. It is design for visual programming to prepare students' own interactive stories, games, and animations. Students share them with others in the online community. As we can read on Scratch's website [7], Scratch helps young people learn to think creatively, reason systematically, and work collaboratively — essential skills for life in the 21st century. We decided to prepare materials for 8 lessons of approximately 90 minutes each. Teachers were obligated to make this lessons in 2-3 months. The first module was an introduction into programming environment. In the next modules students design simple games. In the last one students create multimedia project associated with the season of the year. The realization of this module came out in December, so children prepared Christmas cards.

We know that it is not enough to prepare materials and deliver them to schools. This is why two teachers from each school were invited on a three-day together training programme. From each school one teacher specialized in computer classes whereas another one in other subject – mathematics, science, history or even physical education. During workshop teachers learned basics of the Scratch. They also took on a role of students to create games and activities according to the given scenarios. They were encouraged to show a lot of flexibility in modifications of the scenarios because we want them to do the same in classrooms to let children implement their own ideas. Moreover, the role of programming in education was presented and its impact on children's creativity, as well as the role of independent investigation into knowledge. There was a special time to integrate in the schedule of meeting, because we wanted to help teachers to cooperate with each other, even if there was a big distance between schools. During the training there was also organized a meeting with a representative of the Association of Parents in Education.

The project organizers offered support during the implementation of activities in schools. There was a contact with trainers by email or phone and the opportunity to participate in discussion forum. This last form was the most popular one. Moreover, the implementation of the activities in schools was continuously monitored. Teachers were asked to write short reports after the completion of each module, which were published on this forum. At the end teachers along with the 3-person representations of students were invited for festival of projects. During one day meeting students presented their works.

3 Scenarios and lessons

As it was mention above there were prepared 8 scenarios for 2-hours lessons. Their titles are:

1. Introduction
2. A Cat Chasing a Mouse
3. A Cat in a Maze
4. A Cat Sets a Trap for a Mouse
5. Bouncing Balls
6. Guessing the Number
7. Racing Sprites
8. Multimedia Card

As an example, we describe one of the scenarios *A Cat in a Maze*. In this game a cat moves forward, turns left and right to find the right path in a maze. We want students after this lesson to know how to construct a simple script that responds to pressed key. Students also should know how to use a conditional statement, a simple loop and control behaviour of sprite based on its location. Before lesson teacher prepares the board.

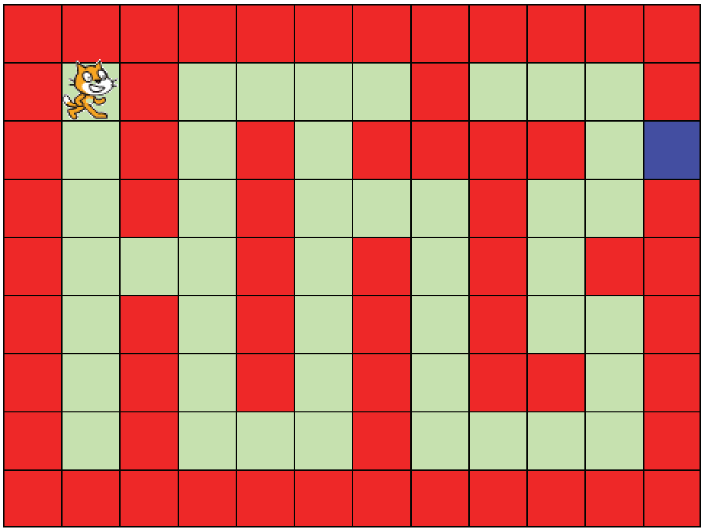
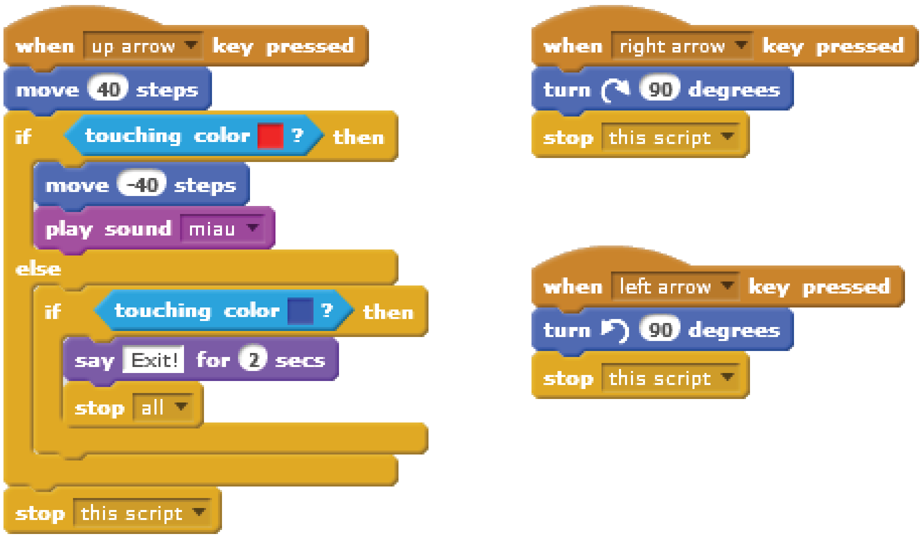


Fig. 2. The board for the game *A Cat in a Maze*

In the first part of the lesson teacher can ask different questions about the game. Exemplary question are:

- How the game starts?
- How can you control the sprite?
- What should happen when the sprite collides with a wall of the maze?
- How to rotate sprite to move it in a specific direction?
- What is going to happen when the sprite comes to the end?



```
when up arrow key pressed
move 40 steps
if touching color ? then
    move -40 steps
    play sound miau
else
    if touching color ? then
        say Exit! for 2 secs
        stop all
stop this script
```

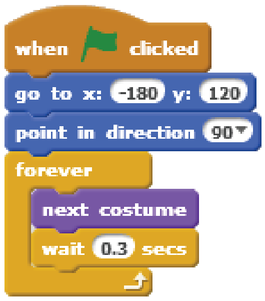
```
when right arrow key pressed
turn ↻ 90 degrees
stop this script
```

```
when left arrow key pressed
turn ↺ 90 degrees
stop this script
```

Fig. 3. Scripts for controlling the movement of the sprite

These question will help students to realize what the rules of game are. Moreover, we will stimulate children's thinking to programming issues hidden in this task. Next, there are many simple scripts that should be prepared – for example to control the sprite to move forward, turn left and right. There are many possibilities to program such tasks, for example – when we discuss recognition of the background colour of the sprite we want the sprite when it touches a red colour, not to move forward 40 steps but to stay in the same place. It can be realized by moving forward, checking the background colour and if it is red moving it backward, otherwise do nothing. In this part of the lesson all main difficulties of the project should be presented.

The next step is to merge the whole project. Students import the background – a maze which consists of green and red squares. In this project the sprite is a little bite smaller than usual and it is placed on the top left corner. Students should also check if the length of sprite's movement is the length of squares. The last step is to prepare the beginning script which will start the whole game.



```
when [flag] clicked
go to x: -180 y: 120
point in direction 90
forever
    next costume
    wait 0.3 secs
```

Fig. 4. The beginning script

After finishing the basic core of game students can add something special. They can do some modification. For example, if the sprite hits the red square it should not stay in the same place but go to the beginning. Another idea is to add more boards like in many games to make it more attractive. It is more complicated task to program it but much more fun while playing.

Generally, such projects are not very complicated but extremely engaging for students. They have a lot of fun, but what is more important they learn much. First steps of programming are defeated and general impression is undoubtedly positive. It is important for their future.

4 The project evaluation

The project evaluation was conducted successively from different perspectives. Firstly, teachers filled up a questionnaire after the workshop, secondly there was a monitoring of the work conducted in schools and thirdly by the end of classes in schools external evaluation was conducted [2]. Teachers highly assessed the prepared materials and the training itself. The report [5] of the evaluation was prepared on the basis of 52 evaluation questionnaires and telephone interviews with 10 teachers, who together led classes for 949 students. In this report it is written that: The project *Coding school* was very highly rated. Teachers found the program very well prepared. In their opinion it can help to efficiently and friendly familiarizes with programming. They stressed that the project helps student in creative and strategic thinking. Students can learn much more that simply programming in the Scratch. They learn about reasoning thinking in terms of cause-effect relationships, conditions and rules, practicing patience, accuracy and cooperation with others. In surveys and interviews there are many expressions about the pleasure of using the Scratch : it is fun, it is cool, students and teachers are in some way seduced. From teacher's relations it is an attractive activity, engaging and forces concentration. The lesson time seemed to flow very quickly and students often ask to extend such lessons. Both students and teachers fill that they achieved great success. We may notice that the project is described as pleasant and satisfactory on the one hand and connected with fear and difficulty on the other.

5 Summary

The pilot edition was attended in 34 schools, 70 teachers who teach about 1,300 students. The project was highly appreciated both by teachers and educational authorities. It is now continued for more than 100 schools from different parts of Poland. It should be emphasized that all schools from the pilot edition, want to continue it with their students. It is planned to extend the program through development of further materials for primary schools and to extend idea for more mature students.

References

1. Eurostat (2012), Computer skills in the EU27 in figures, http://europa.eu/rapid/press-release_STAT-12-47_en.htm
2. Filiciak,, M. et al.: Nauka programowania w szkołach. Czas na upgrade? Centrum Cyfrowe, Warszawa (2013).
3. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., New York, NY, USA (1980).
4. Papert, S.: The Connected Family: Bridging the Digital Generation Gap. National Book Network (1996).
5. Tarkowski, A., Mazgal, A.: Nauka programowania w szkołach, Czas na upgrade – perspektywa 2014. Centrum Cyfrowe, Warszawa (2014).
6. Mistrzowie Kodowania, http://mistrzowiekodowania.pl.
7. Scratch – Imagine, Program, Share, http://scratch.mit.edu.

Overcoming Issues with Students' Perceptions of Informatics in Everyday Life and Education with Physical Computing

Suggestions for the Enrichment of Computer Science Classes

Mareen Przybylla[1], Ralf Romeike[2]

[1]Didactics of Computer Science, University of Potsdam, Germany
[2]Didactics of Computer Science, University of Erlangen-Nuremberg, Germany
[1]przybyll@cs.uni-potsdam.de
[2]ralf.romeike@fau.de

Abstract. The same way that interactive computing systems gain more relevance in our society, the development of easily programmable micro controllers offers a new way of creatively designing computing systems at school. Physical computing comprises the development of interactive objects and installations. In this paper findings of a survey investigating students' perceptions of their informatics classes and of computing systems and embedded systems as informatics devices are presented. Suggestions are made, how the identified issues can be improved by implementing physical computing activities into computer science classes and experiences described.

Keywords. physical computing, students' perceptions, informatics classes, interest, creativity, constructionism

1 Introduction

Physical computing is very popular among hobbyists and makers and starts becoming increasingly popular in extracurricular computer science education contexts, such as afternoon clubs or summer camps. However, it only plays a little role in classroom settings. There are some approaches to teaching that are similar to physical computing and aim at complementing computer science classrooms with embedded systems (e.g. [1–5]). In such projects, the advantage of haptic perceptibility is judged to be positive. While in those concepts the imitation of existing embedded systems is focused, physical computing adds the inclusion of aspects of art and design and thus offers new creative potentials for informatics lessons. Physical computing is happening when students use programmable hardware to creatively design and craft interactive objects. "Interactive Objects […] perceive their environment with sensors, which in turn deliver data to be processed by the microcontroller. According to the configuration of the systems these data are processed and passed on to the actuators. In this way, interactive objects communicate with their environment. They are created with crafts, art and design material. They fulfill a particular purpose, which may be purely artistic." [6]. In English-speaking contexts, more and more scientific publications devote themselves to the topic. Paulo Blikstein for instance has investigated physical computing kits for their underlying design principles [7]. Despite its potentials for constructionist learning environments for computer science, physical computing in education is mainly used by non-computer scientists, e.g. in arts, physics or biology classes. In the ongoing research it is intended to analyze what effects physical computing has on students in computer science classes by investigating the impacts on students' motivation, creativity, constructionist learning, learning success, growth in competences and their understanding of computer science and computing systems. A recent study among students in secondary schools in Berlin and Brandenburg has delivered some interesting insights concerning their understanding of informatics devices and their perception of computer science classes. The results of this study are presented and discussed in this paper and conclusions drawn towards the implementation of physical computing activities in the computer science classroom. In a first school experiment with ninth-graders physical computing was conducted in the context of computer science with "My Interactive Garden" (MyIG) [8] and the approach of 'informatic pottery making'. The school experiment gave a first impression of students' acceptance of the approach and the added value of physical computing to the class [9]. The approach and experiences are briefly described in this paper.

2 The Survey

In order to investigate the effects of physical computing activities on learners and how learners are influenced by physical computing activities, as a first step a questionnaire was developed to investigate students' understanding of informatics devices and their perception of computer science classes. This study was conducted with students in secondary schools in Berlin and Brandenburg who had no physical computing experience with the aim of getting an overall impression of the current situation and of the applicability of the questionnaire.

2.1 Objectives of the Survey

The survey presented here aimed at receiving students' assessment of their informatics classes concerning creativity and constructionist learning, as both are meaningful and promising for computer science education with regard to deeper learning and better results [10–12]. However, they are rarely visibly applied in the classroom. Another aim of the study was to investigate students' perception of informatics devices in everyday life. Nowadays products of computer science pervade students' everyday lives. There are sensors and actuators everywhere around us, embedded in computing systems that ease and enrich our lives in many ways, but in informatics classes very often the personal computer is the dominant medium of discussion. In this respect it was also an objective of the study to find out about students' experience with robotics, embedded systems or physical computing activities.

2.2 Cluster Sampling: Participants

As it is impossible to ask every student for his or her opinion, the method of choice was cluster sampling. Participants were chosen from different schools and school types, grades and federal states to allow for comparisons between different groups of students and to raise heterogeneity of the participants chosen. The survey was conducted with 115 students from four different schools in the Berlin/Brandenburg area. 113 out of 115 students returned the questionnaire. 62 of the participants were male, 51 female. Students from grades seven, nine, eleven and twelve as representatives for their age groups – students in lower and upper secondary education – were given the questionnaires. In sum, the questionnaires of 80 students in lower and 33 students in upper secondary education were evaluated. The lower number of upper secondary students is explained by two reasons. First, in upper secondary education informatics classes are not obligatory for students. Second, upper secondary at "Gymnasium" only consist of eleventh and twelfth grade. That results in smaller classes and fewer students. In lower secondary education 22 participants were currently in seventh grade and 58 in ninth grade. Among the ninth-graders 34 students attended "Oberschule" (general education, grades seven to ten, degree confirms VET maturity and depending on the marks qualification for high school) while all the other participants attended "Gymnasium" (grades seven to twelve, degree confirms eligibility to study at a university). In upper secondary 19 students took a "Grundkurs" (basic course) and 14 students were enrolled in a "Leistungskurs" (advanced course).

2.3 Survey Method: Questionnaire

Participants in studies perceive questionnaires as more anonymous than personal interviews. This results in more reliable data, since people are more likely to answer honestly when they know their replies cannot be traced back to them [13]. This is particularly important in school settings, where students might fear that given answers could influence their marks. It was therefore decided, not to interview students personally but to hand them self-administered questionnaires to be filled in anonymously. They did this in class so that all participants filled in their questionnaires under similar conditions. Teachers were given detailed instructions with the questionnaire about how to perform the session.

The first part of the questionnaire contained questions concerning the person and thus collects the independent variables: gender, age, federal state, school type, city, school, grade, attended school courses and experience with robotics or embedded systems. The second part of the question investigates on students' perception of their computer science classes and the third part of the questionnaire aims at informatics perception in everyday life.

To investigate students' perception of their computer science classes it was decided to use a four-point Likert scale (Table 1). Questions for the Likert scale part were designed based on characteristics of constructionist and creative learning environments, which both share many characteristics with intrinsic motivation [10, 12, 14].

Table 1. Questions to find out about students' perceptions of informatics classes

	Strongly agree	Agree	Disagree	Strongly disagree	Cannot tell
Informatics classes allow me to independently gain new ideas, solutions or insights.	☐	☐	☐	☐	☐
In informatics classes I can experiment a lot.	☐	☐	☐	☐	☐
I use knowledge from informatics lessons outside school.	☐	☐	☐	☐	☐
Informatics classes are fun for me.	☐	☐	☐	☐	☐
In informatics classes, I can be creative.	☐	☐	☐	☐	☐
Topics of informatics lessons are interesting for me.	☐	☐	☐	☐	☐
In informatics classes, we can create larger products together.	☐	☐	☐	☐	☐
I have presented products of informatics lessons to my friends or family.	☐	☐	☐	☐	☐
In informatics lessons we create similar things as artists and designers.	☐	☐	☐	☐	☐
In informatics classes I can implement my own ideas.	☐	☐	☐	☐	☐
In informatics classes we work on many different projects / products.	☐	☐	☐	☐	☐
In informatics lessons I can invent new things.	☐	☐	☐	☐	☐
I understand the subject matter in informatics education.	☐	☐	☐	☐	☐
I can try many things in informatics education.	☐	☐	☐	☐	☐

The indicators for constructionist learning environments give an impression on how students perceive their lessons. The results do not show if students actually are learning in constructionist learning environments, but rather if the settings of their lessons promote constructionist learning. The following test items were used to measure in how far current informatics classrooms promote constructionist learning in the perception of students:

1. I use knowledge from informatics lessons outside school. (relevance, usefulness)
2. In informatics classes we work on many different projects / products. (student's field of interest)
3. Topics of informatics lessons are interesting for me. (student's field of interest)
4. In informatics classes, we can create larger products together. (collaboration in construction)
5. In informatics classes I can implement my own ideas. (self-determined learning)
6. I have presented products of informatics lessons to my friends or family. (create meaningful products to show around)

Those test items were weighted according to their importance. Questions 1, 3, 5 and 6 are each assigned three points, as they build constructionist pillars. Item two is an indirect indicator for settings where students' interests are covered, as many different projects should cover many different fields of interest. This item is therefore assigned two points. Similarly item 4 is treated: as according to the constructionist theory people learn through making (constructing) things, it is a key feature of constructionist learning environments to offer students space for creating. In collaboration this works a lot better than e.g. in competitions. This item is therefore valued one point.

The indicators for creativity do not measure if students are creative, but if they feel that in their informatics classes they can be creative. Along with a number of test items a control question was placed to find out if students have the same understanding of creativity as the conductors of the survey. Some of the test items are indicators for creativity and constructionist learning, they will therefore be listed again. The following test items were used to measure how creative current informatics classrooms are in the perception of students:

1. Informatics classes allow me to independently gain new ideas, solutions or insights. (self-determined learning, different approaches to finding solutions)
2. In informatics classes I can experiment a lot. (experimental methods)
3. I use knowledge from informatics lessons outside school. (relevance, usefulness)
4. In informatics classes, I can be creative. (control question)
5. In informatics lessons we create similar things as artists and designers. (creative products)
6. In informatics classes I can implement my own ideas. (self-determined learning, different approaches to finding solutions)
7. In informatics lessons I can invent new things. (creative products)
8. I can try many things in informatics education. (risk taking, choices)

Again, the test items are weighted according to their importance. Items 1, 2, 6 and 8 are clear indicators of a setting that invites learners to be creative, with item 6 being used as a confirmation question for item 1. They are valued three points each. Items 5 and 7 are very similar to each other again and thus treated as test and confirmation questions, each valued two points. Item 3 is not directly connected to creative learning, but shows that knowledge and competences gained during informatics lessons are relevant for the learner's private life; it is valued one point. Item 4 contains the control question and will be evaluated separately as well as comparatively.

To investigate students' perception of computing systems and embedded systems in everyday life, they were given little tasks, e.g.:

A) List five everyday objects that have informatics inside
B) What would you like to invent with the help of informatics? Give as many examples as you come up with.

To find out about students' interest in different project proposals they were asked to grade different ideas with school marks (Table 2).

Value the following project proposals with the school grades 1 to 6 according to how much you would like to participate in the project (1: very gladly, 6: extremely reluctantly; each grade should be assigned exactly once):

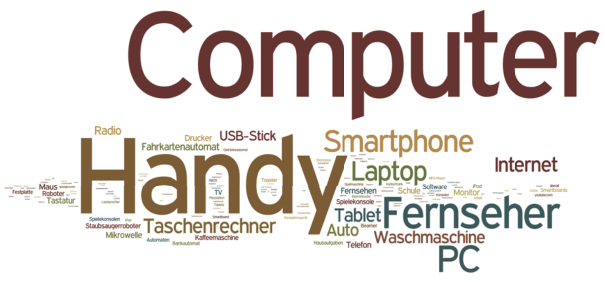
Table 2. Project ideas to be graded by students

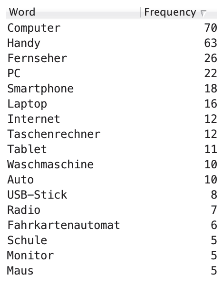
	1	2	3	4	5	6
Solving mathematical problems	☐	☐	☐	☐	☐	☐
Simulation of a slot machine	☐	☐	☐	☐	☐	☐
Creating a mobile app	☐	☐	☐	☐	☐	☐
Controlling a robot vehicle	☐	☐	☐	☐	☐	☐
Creating interactive clothing	☐	☐	☐	☐	☐	☐
Creating an interactive mood lamp	☐	☐	☐	☐	☐	☐

2.4 Findings and Conclusions

In the first part of a larger study, the general aim was to find out about students' perception of their environment concerning computing systems and embedded systems as informatics devices and about their perception of their informatics classes, which are most likely not rich of embedded systems, robotics and especially physical computing activities concerning creativity and constructionist learning.

Informatics Devices. Experience has suggested that, despite the ubiquity of such devices, most students do not think of embedded systems when thinking about informatics devices in their everyday life and instead rather list "obvious" computing devices such as personal computers, mobile computers, tablets, smart phones and iPods. The survey asked students to list everyday objects that have informatics inside.





Word	Frequency
Computer	70
Handy	63
Fernseher	26
PC	22
Smartphone	18
Laptop	16
Internet	12
Taschenrechner	12
Tablet	11
Waschmaschine	10
Auto	10
USB-Stick	8
Radio	7
Fahrkartenautomat	6
Schule	5
Monitor	5
Maus	5

Fig. 1. Tag cloud and word frequency generated from students' answers

A word analysis of the proposals made by students shows that indeed most of them do not come up with embedded systems as informatics devices. Very often students are not aware that informatics is very present in many areas of their everyday life apart from personal computers and smart phones: cars for instance were only named by ten students out of 113 – that is 9% only; which is particularly alarming as this together with washing machines is the most frequent occurrence of embedded computing devices (Fig. 1). This is accompanied by the fact that computer science students often perceive this subject as covering abstract and escapist themes that are only suitable for a special clientele of talented informatics students [15].

Robotics Experience. Based on experiences from classroom visits and conversation with teachers and teacher students it was assumed that only few students have ever had the chance to participate in robotics, embedded systems or physical computing projects at school. It was further assumed that if at all, such projects were mostly robotics projects taken in afternoon clubs or at home as a hobby. When asked about their experience with robotics and embedded systems, 22 students (19%) replied positively. All of them were engaged in building robots; none had named any embedded systems or even physical computing activity. Those who stated a particular robotics kit, all named LEGO Mindstorms. As expected, the large majority of students had never programmed any embedded device or robot. It is also shown, that all of the students who have had any experience, gained it in their leisure time, either as a hobby or in afternoon clubs at school, club houses, etc.

Constructionist Learning. When it comes to programming, students sometimes complain that programs they develop in school were useless, irrelevant for their lives and not meeting their expectations with respect to what they had imagined 'learning to program' would mean. It was therefore assumed that the majority of students taking informatics classes would not perceive their classrooms as settings where constructionist learning takes place. In the survey students were asked to evaluate test items on a four-point Likert scale. One of these items was the statement "I have presented products of informatics lessons to my friends or family." Only 45 students (40%) answered in the affirmative by ticking either 'agree' (29) or 'strongly agree' (16); more than half of the students (64 students, 57%) answered in the negative by ticking 'disagree' (31) or 'strongly disagree' (33).

Creativity. It was assumed that creativity is not a very dominant feature of students' perception of informatics classes. Classroom visits often give the impression of very teacher centered approaches, old-fashioned methods despite using modern tools and a lack of student participation in finding problems to solve. When asked directly, 46 of 113 students (41%) stated that they can be creative in the computer science classroom. Interestingly, this is in contrast to the results shown by creativity indicators. Only nine students (8%) have classified their informatics lessons as creativity promoting according to those criteria.

Interests. It was assumed that students are more interested in activities that involve real-world objects than virtual objects and that mathematical problems are the least interesting subject on the list. The survey has shown, that on average 42% of the students were interested in creating robots, but when analyzing the data of female participants, it were only 29% as opposed to 53% of the males. Similar results were brought up by the suggestion to design interactive clothes, only that this time 65% of the female students were in favor for such a project as opposed to 18% of the boys. As expected, solving mathematical problems was the least interesting project for most students: 36% graded it with on of the marks 5 or 6.

Summarizing, the results of the survey confirm the expectations based on the authors' experience: only few students experience creativity-rich lessons in constructionist learning environments. Despite the popularity of such activities, none of the students who participated in the survey ever had the chance to participate in computer science projects that involve physical computing, embedded systems design or robotics activities in a regular school lesson. Of course with 113 students those findings should not be generalized and overestimated, but they should stimulate teachers to reflect if they can improve their lessons. According to the constructionist learning theory, learning is most effective when learners construct knowledge and develop competences from their own initiative and for a personally relevant purpose [10]. Resnick added: "What's important is that they are actively engaged in creating something that is meaningful to themselves or to others around them" [11]. In constructionist settings, learners will thus develop meaningful products they present to and discuss with friends and family. As shown by Romeike [12], creativity has positive effects in many respects: students who undertook a creativity-rich lesson series have shown higher motivation, interest, better results and a better understanding of informatics. In order to provide students with the opportunity to recognize and understand embedded systems they can find in their daily environments, it seems reasonable to address these as subjects in computer science classes. After a long period of time when software development has dominated the design-oriented part of informatics education, it is now time to link the virtual and the real world.

3 Physical Computing as Informatic Pottery Making

For the reasons mentioned above, it is desirable to eliminate the identified deficiencies. One way to address the problem is a more in-depth study of physical computing in order to situate new contents within the field of computer science education. In contrast to attempts that mainly deal with rebuilding or imitating existing embedded systems, physical computing emphasizes a greater involvement of aspects of art and design, which opens up a wider range of opportunities to become creative. Physical computing further allows students to develop concrete, tangible products of the real world, which arise from the learners' imagination. This way, constructionist learning is raised to a level that enables students to gain haptical experience and thereby concretizes the virtual. Vahrenhold [16] mentioned that computer science education lacks a "Going to Paris Effect". While for students and their parents the aim of learning the French language to be able to communicate on a journey to Paris is obvious, such an aim is missing for informatics students beyond improvements in computer use. Under the aspect of *Informatic Pottery Making* we see it as a result of computer science lessons, that children – similar to making a vase in pottery class – may bring home from school digital, interactive artifacts they themselves have created and programmed in computer science class and that can be investigated, shown around and admired in a constructionist sense. Based on this understanding, computer science will become personally relevant for students [6].

Implementing physical computing activities in the computer science classroom means making decision concerning hardware, programming environment and the teaching context. Students' interests differ greatly depending on age, gender and even school form. It is therefore necessary to find a context that allows both, boys and girls, to follow their interests. With "My Interactive Garden" (MyIG) such a context is provided to students. MyIG includes a constructionist learning environment, which allows informatics students to craft, design, program and build their own interactive objects. It entails a construction kit based on Arduino that includes preassembled sensors and actuators and a shield that allows to plug in (Fig. 2). The aim of learning with MyIG is to collaboratively create an exhibition of interactive objects as they could be found in a futuristic interactive garden. Such objects can be anything from magical flowers over noisy scarecrows to interactive party lights.

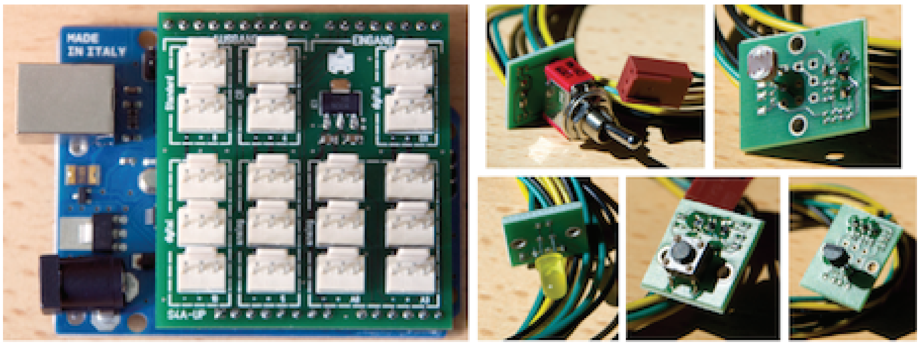


Fig. 2. Arduino with MyIG shield and pluggable sensors and actuators

This framework allows for multiple and manifold projects and triggers students' creativity. With reference to [2] the following strategies are recommended:

Focus on themes, not just challenges. If provided with themes instead of concrete tasks, students can find their own projects to work on, follow their own interests and find their own problems to solve.

Combine arts and computer scientific modeling. When an artistic component is added, as described with the approach of informatics pottery making, the subject will become personally relevant to students and call for creativity which leads to more diverse and interesting projects and problems to solve.

Encourage storytelling. When encouraged to tell a story about their creation, students will link their objects to the real world and reflect on influences their invention may have on the society. They will also tend to design their interactive objects according to what they want them to do instead of thinking about what they are able to make them do, which in the end may lead to deeper learning and understanding.

Organize exhibitions rather than competitions. Collaborative work is a very valuable means of constructionist learning. As has been mentioned earlier, a key idea is that learners are engaged in creating artifacts that are personally relevant for them and for others around them, which can be discussed and investigated with others. In collaborative work there is a need to discuss the single parts of a bigger project with the contributors, thus it matches perfectly to constructionist learning.

3.1 Experience

MyIG was piloted in a school experiment with students of a ninth grade and gave a first impression of students' acceptance of informatic pottery making, balance between informatics and crafting activities and the added value of physical computing. Data was collected with questionnaires and by observing the students. There were first tendencies observed in this pilot project, suggesting that physical computing may help students in expanding their understanding of computing systems. The students liked the pottery making approach and the amount of crafting influenced the amount of programming positively: the more complex the students' interactive objects became, the more complex were their programs [9]. These data are not statistically valid, since only a very small number of students were involved in the project. It was a first trial to test the approach and figure out difficulties. Nevertheless there was a lot to be learned from the students' way of dealing with physical computing.

Competences. Many skills and competences can be gained with physical computing; some are more obvious than others. While it is very clear that programming concepts and control structures such as decisions and loops, variables, comparisons or arithmetic operations will be needed to create objects that can blink, move and make sounds reacting on influences from their environment, several additional topics become relevant for students. One student for instance investigated a temperature sensor and stumbled about the difference of data and information: he read values and noticed that those were not matching any temperature scale he knew. Further research led him to the conclusion that the values he read were "raw data" that needed to be interpreted. Students also learned about sensors and actuators, about the difference and use of analog and digital data when controlling actuators, about the use of exchanging messages between programs, about communication in work-sharing projects and many more. Altogether they gained diverse competences on their personal levels that went beyond algorithmic thinking.

Extendibility. Physical computing projects are never complete. This does not mean, that they will never reach a sufficing level, but that they allow for iterative work. Students always found possibilities to improve either the design or functionality of their interactive objects and installations. One of the students for instance had built a sunshade that automatically opened when it was bright enough. When he had finished, he added an additional feature to save power: a button was included that needed to be pushed in order to activate or deactivate the mechanism.

Motivation and Creativity. Students are very ambitions when working on their own projects. They had many creative ideas, even brought crafting materials to school. Their projects became very complex and they rarely discarded any ideas. They praised each other for their achievements and in the end were really proud about the interactive garden they had made (Fig. 3).

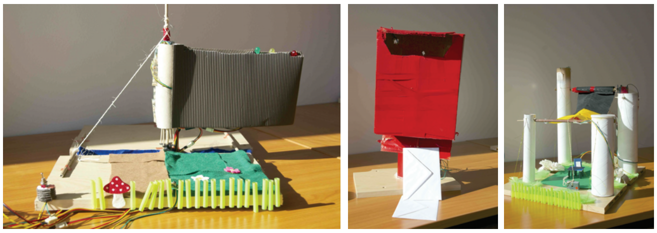


Fig. 3. Students' projects "Automated Gate", "Clever Letterbox" and "Smart Sunshade"

4 Discussion

Computer science teaching has always been faced with the challenge of providing time stable ideas and concepts but to motivate them by picking current issues as central themes and using modern tools. Physical computing offers the opportunity to transfer the potential of creative design possible with software development into products of the real world. Considering the motivational value a self-created product offers, such as by potters, it becomes quickly clear that this can also be used in computer science education. The combination of informatics with art and design has the potential to appeal to less computer-savvy students, too.

Our Experiences show that students have fun with physical computing. They expressed this verbally, but it was also visible in the lessons. They often wanted to stay longer to continue their work. Students also asked where they could buy physical computing construction kits for their homes, as they only had the chance to work on their interactive objects once a week.

Until now, physical computing is regarded as an interesting and exciting phenomenon by many teachers, but for most of them it has not been suitable for classroom use due to the technical complexity of breadboard and soldering activities. With MyIG and similar construction kits (e.g. TinkerKit, Hummingbird) teachers are provided with ideas and solutions to this problem. A detailed analysis to extract and define topics relevant for physical computing and to define fields of competencies that can be gained with physical computing is yet to come and will be in focus for the ongoing research.

5 References

1. Strecker, K.: Wie viel Programmierkompetenz braucht der Mensch? LOG IN. 169/170. 40–47 (2011).
2. Rusk, N., Resnick, M., Berg, R., Pezalla-Granlund, M.: New Pathways into Robotics: Strategies for Broadening Participation. Journal of Science Education and Technology 17. 59–69 (2008).
3. Baumann, R.: Eingebettete Systeme verstehen. Teil 1: Kreatives Experimentieren mit Arduino. LOG IN. 171, 33–45 (2011).
4. Baumann, R.: Eingebettete Systeme verstehen. Teil 2: Arduino zwischen analoger und digitaler Welt. LOG IN. 174, 37–48 (2012).
5. Pelz, L., Arnhold, W.: Die Waschmaschine - Embedded Computing im Alltag. Workshop: 12. GI-Tagung der Fachgruppe "Informatik-Bildung in Berlin und Brandenburg". Berlin (2013).
6. Przybylla, M., Romeike, R.: Physical Computing im Informatikunterricht. In: Breier, N., Stechert, P., and Wilke, T. (eds.) Informatik erweitert Horizonte. 137–146. Lecture Notes in Informatics, Kiel (2013).
7. Blikstein, P.: Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future. Interaction Design and Children 2013. 173-182. ACM, New York (2013).
8. Przybylla, M., Romeike, R.: My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education. Proceedings of Constructionism 2012. 395-404. Athens (2012).
9. Przybylla, M., Romeike, R.: Physical Computing mit „My Interactive Garden“. Department of Computer Science, CAU, Kiel (2013).
10. Papert, S., Harel, I.: Situating Constructionism. In: Papert, S. and Harel, I. (eds.) Constructionism. Ablex Publishing Corporation, Norwood (1991).
11. Resnick, M.: Distributed Constructionism. International Conference on Learning Sciences. 280-284. ACM, New York (1996).
12. Romeike, R.: Kreativität im Informatikunterricht. Dissertation Thesis, Potsdam (2008).
13. Bortz, J., Döring, N.: Forschungsmethoden und Evaluation: für Human- und Sozialwissenschaftler. Springer-Verlag, Berlin, Heidelberg, New York (2003).
14. Ryan, R.M., Deci, E.L.: Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. Contemproary Educational Psychology. 25, 54–67 (2000).
15. Knobelsdorf, M.: Biographische Lern- und Bildungsprozesse im Handlungskontext der Computernutzung. Dissertation Thesis, Berlin (2011).
16. Vahrenhold, J.: On the Importance of Being Earnest: Challenges in Computer Science Education. Proceedings of WiPSCE 2012. ACM, New York (2012).

Competence Measurement for Informatics

Didactics of Information Security as a Key Competence

Thomas Schiller[1], Peter Micheuz[2]

[1] BG / BRG Ramsauerstraße and
Pedagogical University of Upper Austria, 4020 Linz
th-schiller@gmx.net

[2] Alpen-Adria University Klagenfurt, 9020 Klagenfurt
peter.micheuz@aau.at

Abstract. This paper deals with teaching and learning information security at school level. It starts with current descriptions of this broad term, its implications on society in general and on its curricular implementation. A brief overview shows how it fits into many guidelines, reference models and existing curricula for digital competence and computer science. As an indispensable part of informatics information security must be regarded as a key competence pupils and students have to acquire, preferably in a formal and ensured way at school. After presenting concrete approaches of teaching information security within existing frameworks and reference models, a focus is laid on the current situation in Austria.

Keywords: teaching, learning, information security, key competences, computer science education

1 Introduction

"Information security is all about protecting and preserving information. It's all about protecting and preserving the confidentiality, integrity, authenticity, availability, and reliability of information" [15]. It "encompasses the study of the concepts, techniques, technical measures, and administrative measures used to protect information assets from deliberate or inadvertent unauthorized acquisition, damage, disclosure, manipulation, modification, loss, or use" [17].

These definitions make clear that information security has to be regarded and treated as a multidisciplinary area of study and activity. It is concerned with understanding, developing and implementing security mechanisms with respect to technical, organizational, human-oriented and legal issues. It plays an outstanding role in all systems in which information is created, processed, stored, transmitted and deleted.

All items listed in these definitions are very important and relevant for teachers and students, especially in times of heavy usage of smartphones and social networks. Do I have my (posted) data under control? Is it really my friend behind a possibly faked digital identity? There are many other questions to deal with, for example, cybermobbing as a possible consequence of taking pictures of persons causing inconvenient and serious situations because of the spatial distance between offender and victim.

Information security, especially seen as a subject matter to be provided in schools, covers a wide range of challenges from social aspects as illustrated above to technical questions such as understanding techniques like encryption and verification mechanisms.

In this paper the term information security holds also for safety issues. It is not intended here to elaborate on linguistic issues which are often language specific. In German, for instance, there exists only one expression for security and safety, namely “Sicherheit”. As information is - broadly speaking - individually interpreted data, we permissively also include data protection in our considerations. In contrast to information security, data protection deals mainly with technical aspects, not to confuse it with privacy which is person centered.

2 Information Security Embedded in Frameworks

Looking at exemplary didactic approaches and its characteristic guiding principles for teaching, information security seems to fit in nearly all theoretical considerations and concepts of computer science education, and moreover, affects other disciplines at school level as well.

Baumann’s guidelines for computer science education are problem solving, principles of computer science systems and foundation and limits of information science knowledge processing [1, p. 66]. From his perspective, computer science systems “represent knowledge of different type and origin, process this knowledge representation in form of data and programs and to users in a suitable form." [1, p. 63]. Further guidelines, e.g. Hubwieser’s information-based approach encompassing presenting information, processing and transport of information and interpretation of information representation [14, p. 81], can be seen as the common basis and theoretical rationale for dealing with information security.

In 2001 an expert group of the German society for computer science compressed many existing guidelines in form of "recommendations for a comprehensive approach to computer science education in schools for general education” [11]. This seminal concept consists of four guidelines dealing with information, active principles of computer science systems, problem solving with computer science systems and working with models [18, p. 56- 57]. Based on these guidelines, system and application competence, dealing with structures, functions, limitations, safety and effects of (networked) computer systems [10, p. 8], information security is embedded well in a theoretical framework of computer science education.

Theoretical considerations are only one side of the coin, its practical implications in implementing information security and its aspects in current curricula, frameworks and not least in didactic settings and concrete lesson plans and tasks is the other one. This chapter will give some answer regarding the approaches how this important subject matter is represented in existing frameworks. We will see that information security, seen in broader perspective including data protection, covers many aspects of informatics and ICT. If it should be an obligatory part of general education to understand also the technical basics as encryption and authentication behind complex user interfaces, is up to discussion and depends on particular curricula.

Not least, “teaching Information security is raising awareness, creating attitude and anchoring in behaviour” [12].

Competence is a buzzword dominating the educational field in Europe, gaining much attention especially in the field of vocational education within the European Qualifications Framework, where “it is seen as the most advanced element of the framework descriptors and is defined as the proven ability to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development and is described in terms of responsibility and autonomy” [21].

In recent European policy recommendations there is a different definition of competence. In the Key Competences Recommendation, competence is defined as a combination of knowledge, skills and attitudes appropriate to the context [21].

In this paper, competence is understood as a set of knowledge, attitudes and skills. Let us look exemplarily at three major frameworks to get a concrete and holistic view on the field.

2.1 DIGCOMP – The Digital Competence Framework

The current Digital Competence Framework [4], initiated by the European Commission, consists of five main competence areas: Information, Communication, Content-Creation, Safety and Problem-Solving. Safety, which is equivalent to Security, is one explicit competence area and encompasses the following subcategories, denoted here as competences.

Table 1. “Safety” in the Digital Competence Framework [4]

Competence areas	Competences	Cross-references
Safety	Protecting devices	Browsing, searching, filtering information Solving technical problems
	Protecting data and digital identity	Browsing, searching, filtering information Managing digital identity
	Protecting health	Interacting through technologies Netiquette
	Protecting the environment	Innovating and creatively using technology

It is not surprising that security issues are highly connected with other areas which are made explicit in the competence matrix above.

Computing in schools is a wide field, and hopefully am still existing confusion even among teachers should be soon a thing of the past as computing in schools can easily be divided into three main areas: Educational Technology, Information Technology and Computer Science. Expressed pointedly, it is all about the fundamental difference between using IT to learn (technology enhanced learning), learning IT to use (application of digital media) on the one hand and understanding and developing software (systems) on the other hand (computer science or informatics).

Information security is an essential part of all three above-mentioned areas. Until now we have skipped computer science which indispensably has to come into play to guarantee a basic understanding of information security on a technical level [3].

2.2 CSTA K-12 Standards for Computer Science

The prominent and seminal CSTA K–12 Standards for Computer Science can be regarded as a worldwide recognized initiative to structure and implement computer science at school level, serving as guidelines for informatics education throughout all grades in primary and secondary education. These guidelines encompass the strands

- Computational Thinking,
- Computing Practice & Programming,
- Computers and Communication Devices,
- Collaboration,
- Community, Global and Ethical Impacts.

Not surprisingly, these strands have at a first glimpse very little in common with the competence areas of the EU’s DIGCOMP framework. On second glance, there are some accordances especially in the areas of problem solving and content creation. Digital competence and computer science cannot be seen as disjoint areas. There is a considerable mutual interdependence in which digital competence includes parts of computer science and vice versa. Regarding the significance of information security in these two approaches, it is obvious that the European approach attaches much more importance to this issue. This is not only due the broad area of digital competence, but apparently also to the European tradition of scrutinizing technology and having reservations about its drawbacks.

This notwithstanding, within the detailed description of the strand Community, Global and Ethical Impacts some Information security aspects get explicitly visible “Principles of personal privacy, network security, software licenses, and copyrights must be taught at an appropriate level in order to prepare students to become responsible citizens in the modern world.” [3]. In the “scaffolding charts” with 170 “competences” describing the five strands more precisely, the term “security” is part of the strand Community, Global, and Ethical Impacts within the subcategories Responsible use, impacts of technology, information accuracy, ethics, laws, and security, equity. Less than 10 competences are associated in a narrow sense with information security, as for instance “Identify the impact of technology, e.g. (social networking, cyberbullying, mobile and web technologies, cybersecurity, and virtualization) on personal life and society (allocated for the age group of 10-12 years), and “Describe security and privacy issues that relate to computer networks” for the age group of about 16 years old students.

It is true that this seminal paper lacks the listing of cross-references – this work has still to be done – but the paper consists of about 20 elaborated and concrete lesson plans which illustrate many, but not all of the 170 competences. Interestingly (but understandably) no tasks deal with security issues. This might be due to the fact that security issues are seen more likely to be a knowledge area than a field of activities. The next chapter on ECDL confirms this assertion.

2.3 ECDL Security Modul

The widespread European Computer Driving License (ECDL) and its international representation (ICDL), founded almost twenty years ago and having a remarkable impact on IT in school education in some countries, has recently undergone a considerable reform in structure and content. With the upcoming and dramatic shift from standalone computers to connected digital devices with fascinating opportunities to communicate on the one hand, but a variety of threats on the other, the ECDL foundation developed within the standard modules such as presentation, databases or image editing also the module IT Security.

The ECDL/ICDL does not claim to be a high level certificate providing sophisticated and deep knowledge of a field. However, it offers candidates an internationally recognized certificate about certain end-user computer skills. It is explicitly addressing skills and a qualification as shown in the following table. It is not about (deeper) competences. At first sight the skillsets and the underlying syllabus meet very exacting requirements and connote completeness and coherence. But that is immanent to most curricula and teaching plans. The certification process consists of multiple choice questions and lacks the testing of practical knowledge. It is likely that in the course of the preparation for this module often a mere teaching, learning and training to the test takes place, resulting necessarily in superficial knowledge.

Table 2. ECDL IT-Security Modul [7]

Category	**Skillset**
Security Concepts	Data Threats, Value of Information, Personal Security, File Security,
Malware	Definition and Function, Types, Protection
Network Security	Networks, Network Connections, Wireless Security, Access Control
Secure Web Use	Web Browsing, Social Networking, Storage and Compression
Communications	E-Mail, Instant Messaging
Secure Data Management	Securing and Backing up, Secure Destruction

3 Further Initiatives

The European Parliament defined eight key competences that should be acquired by learners at the end of their compulsory education [21]. These key competences include mastery of the native and one foreign language, learning to learn and on fourth place digital competence. Digital Competence is defined as the critical use of Information Society Technology (IST) including the responsible use of interactive media, accompanied by basic skills in Information and Communication Technology (ICT).

But long before, in Norway for instance, educative and safety-related measures since 2000 have been taken [23], resulting in institutionalized initiatives to raise awareness about information security. Besides the (intensive) media presence helping to reach the whole population, the direct integration of teachers helps directly to educate the children better targeted to modern technology and its (safety) issues. (comp. [23])

"Knowledge about information security must be strengthened. The National Curriculum for Knowledge Promotion in Primary and Secondary Education and Training has ICT as one of its five basic skills, and ICT is part of the professional competency goals. Information security should therefore be included as a natural part of ICT use in the educational framework. This places demands on basic education and competency development for teachers and on teaching materials". [19]

Similar developments could be observed also in many countries throughout Europe. Saferinternet.at (Saferinternet.eu), co-founded by the European Commission [22] is an interesting project resulting in brochures for students, teachers and parents and organizing projects for the "Safer Internet Day" which is also carried out in Austria.

4 The Austrian Case and Implementation Challenges

The general part of the computer science curriculum in Austria implies a deeper insight into social contexts and implications of information technology (such as work and leisure, as well as consequences for security and legal consciousness) [16, p. 1]. One of the listed teaching topics is "understand key measures and legal principles related to data security, privacy and copyright, as well as learn about the impact of technology on individuals and society" [16, p. 2, translated].

In general, teachers of all subjects should deal with this topic, but especially computer science teachers should handle it due to their system competences and (software) technical background.

The core problem is that troubles due to lack of awareness about information security are starting much earlier, long before compulsory computer science lessons at secondary level begin at the age-group of 15. Digital devices are nowadays reality for pupils at secondary level I and even earlier. Increasing and easy access to the Internet and using social networks at any time, sharing photos and videos as well as unreflected use of data, resulting in severe issues as cybermobbing and sexting.

Therefore, Austrian institutions provide different facilities to raise the awareness about information security, accompanied by support with teaching materials and guides including completed online courses. Workshops with external presenters, even from the police departments ("Click&Check"- Workshops [20]), for students, teachers and parents can be booked. Due to inviting external experts, pupils find it more interesting and the workshops can be placed after regular lessons which do not disturb the scheduled teaching.

According to external recommendations as the Digital Agenda [6] and an ongoing conversion from input oriented curricula to student-centred competence oriented learning plans, similar frameworks have been developed for general education at lower and upper secondary level, covering all age-groups, beginning with primary education.

Table 3. Main Structure of Austrian reference models [8]

Digital Competences at Primary and Lower Secondary Level	Informatics Education at Upper Secondary Level
Information Technology, Human and Society Responsibility in using IT, Data protection and Data Security	
Informatics Systems	
Software Applications	Applied Informatics
Informatics Concepts	Practical Informatics

All three reference models for primary, lower and higher secondary level education contain the common main category "Information Technology, Human and Society" including sub categories such as Responsibility in using IT, Data protection and Data Security, and competences referring aspects of Information security in form of so called descriptors with assessable "I can ..." statements. The framework for lower secondary education contains 72 descriptors with 13 referring directly to security issues. One of them, the competence "I can backup data and know about the risk of loss of data" can be found in the technical category Informatics systems within the competence area "Design and Use of Personal Informatics Systems", which indicates the inevitable and inherent cross-reference of information security issues.

Providing schools in a top down way with thoroughly elaborated frameworks, reference models, curricula and syllabi, defining something like a common core standard, is a necessary endeavour. It is, however, the first step on a long way to bring written words into life and into schools, and into the heads of teachers and students. To implement these standards, work is needed in three important areas: Teacher preparation, commitment to these standards and curriculum materials development.

As the issue of materials is concerned, recently a big step forward has been accomplished in form of a collection of worked out examples that integrate aspects of information security. Currently, good and meaningful lesson plans and examples about security issues represent still the minority because the topic seems to be more knowledge than activity based. That could be related to the fact that activity based assessments and tasks are rather expensive. As mentioned and reasoned above, among twenty lesson plans worked out in the seminal CSTA position paper [3], there is none dealing with information security. Most likely, this would not have happened if these K-12 Computer Science Standards were released in 2013. Beyond hot topics as "the state as big brother in times of NSA" there are plenty of interesting starting points as cryptography, encryption and understanding the principles of malware.

Looking at the Austrian strategy to implement Digital Competences at lower secondary level, [5] provides schools with a growing collection of ready to use tasks. For reasons mentioned above, there exist explicitly not as many tasks as in other areas as the production of digital media.

The construction of good lesson plans and meaningful tasks and assignments is demanding, especially with respect to such a complex topic as information security, which unfolds along the dimensions age-appropriateness and content-orientation encompassing the whole range of sociological, human, legal and technical issues.

Due to the raising importance of information security it is necessary that existing projects like Saferinternet.at are to be continued at least in form of maintenance of providing schools with excellent teaching and learning materials and taking part in actions like the Saferinternet Day or the European Cyber Security Month (Oct. 2013, [9]).

We need both, the integration of information security aspects into teacher education and in special Informatics lessons, but also a broader professional handling on information security issues integrated in other disciplines. Moreover, through teaching information security in a formal way at schools, raising the awareness on this topic could be more easily supported through external institutions.

5 Conclusion

Recent worldwide initiatives indicate that information security can be regarded as a fundamental idea and key competence, and therefore should be taught from primary education on. Children in this age-group are attracted by topics like secret writing and information hiding as proposed by initiatives as CS Unplugged [2]. Thus, security issues can be the driver of computational thinking already at an early stage.

Whereas the approach in primary education is presumably a playful one, the window of opportunity to foster awareness and to deepen knowledge about security issues is wide open at lower secondary education. A vast majority of students in this age-group of 10-14 years is personally deeply involved, as digital devices play a big role in their lifes. Legal and technical issues should be an obligatory part of education, preferably in a discipline in its own right. Due to increasing self-reflection, the students should be prepared to take responsibility as digital citizens, and their developing brain should be ready to grasp even basic concepts of data and information security issues, resulting in a basic understanding of processes, terminology and in mastering of associated software tools as well.

Building on a solid foundation provided already at lower secondary level – but being aware that we are still far away from this ideal situation –, security issues should play a continuous role in upper secondary education at high school level. In this age group computer science as a discipline comes into play, with Information security as an indispensable and meaningful part of it. Data security (measures), a deeper understanding of the internet, cryptology and the comprehension of threat scenarios caused by malware, hacking, modelling and simulating a local cyberwar within the school intranet are meaningful topics that can be treated on different complexity levels, even going far beyond a "key competence".

In the design of content standards and guidelines with respect to information security, it is important to ensure the distinction between digital literacy and training IT skills on the one hand, and the teaching of computer science (Informatics) on the other hand. Information security in itself is a very broad field encompassing nearly all aspects, levels and layers of information technology. Very few experts in the field have a deep technical knowledge where information security is to a certain extent a game of cat-and-mouse when cybercriminality comes into play.

Schools cannot educate all students to be prospective specialized security experts, but they are accountable for preparing pupils and students to be digital citizens with appropriate knowledge and skills in digital security and safety issues.

References

1. Baumann, R. (2003). Grundlagen und Bildungsziele der Informatik in der Schule. (Foundations and educational goals of computer science at school) In: Reiter, A. (publisher) et al: Schulinformatik in Österreich. Erfahrungen und Beispiele aus dem Unterricht (computer science in Austrian schools. Experiences and examples from class), Ueberreuter, p. 57- 70. In [10]
2. Computer Science Unplugged: http://csunplugged.org/
3. CSTA: Computational Thinking in K–12 Education Teacher Resources, 2nd edition. http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf
   CSTA Standards Task Force: K–12 Computer Science Standards, revised 2011; http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf
4. DigComp: A Framework for Developing and Understanding Digital Competence in Europe, European Commission, Joint Research Centre, Institute for Prospective Technological Studies, 2013 (http://ftp.jrc.es/EURdoc/JRC83167.pdf)
5. DigiKomp: http://www.digikomp.at/ (16/11/13)
6. Digital Agenda: http://ec.europa.eu/information_society/digital-agenda/documents/digital-agendacommunication-de.pdf (2011-03-31)
7. ECDL IT-Security Modul: http://www.ecdl.org/programmes/index.jsp?p=2928&n=2944
8. From Digital Competence to Informatics Education - Structuring a Wide Field, in Digitale Schule Österreich, OCG, Wien
9. European Cyber Security Month – Austria. http://cybersecuritymonth.eu/ecsm-countries/austria (16/11/13)
10. Fuchs, K. & Landerer, C. (2005). Das mühsame Ringen um ein Kompetenzmodell (The struggle over a competency model). In: CD Austria 12/2005, p. 6- 9. infobild-web.pdf (24/7/13)
11. Gesellschaft für Informatik (2001). Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen (Recommendations for a comprehensive approach to computer science education in general schools), Beilage LOG IN 2. In [10]

Using ICT in Informatics Education and Selected Subjects in Primary Education for Developing of Pupils´ Abilities

Gunčaga Ján, Karasová Mária

Faculty of Education, Catholic university in Ružomberok, Ružomberok, Slovakia
{jan.guncaga, maria.karasova}@ku.sk

Abstract. In our contribution we would like to present informatics education in primary school in Slovakia. Different kinds of educational software plays important role in primary education. We use software for painting, counting and other subjects. Especially, we show examples in some school subjects, which help to develop interdisciplinary connections. We present some results from our research about using ICT in these subjects in primary level. We discuss about positive and negative aspects of using this ICT in appropriate level for pupils' age in primary education.

Keywords: ICT. Primary education. Informatics education. Interdisciplinary relations.

1 Introduction

Modern ICT is now a normal part of the educational process, and over time reveal new opportunities for their effective use, and problems associated with their implementation. The situation in Slovakia is slightly different in comparison with countries such as USA, United Kingdom, and so on. We still feel the shortcomings at the actual implementation of ICT, practice shows not only positives but also negatives. ICTs are an excellent means to streamline the educational process, to make it exceptional, make it more appealing to pupils and less intensive for the teacher training. But the teacher must know how to use them so that it was not only motivational but also reasonable.

Many researches in Europe are oriented to impact of ICT in primary education. For example the study of EACEA bring statement, that "teachers find that ICT supports in equal measure a range of learning and teaching styles, whether didactic or constructivist, in passive activities (exercises, practice) and in more active learning (self-directed learning, collaborative work)" [10].

In the Slovak Republic informatics education appeared since 2008. Informatics in education has an important role in the lower as well as upper secondary education. Over time, it has proved the necessarity to introduce such an object within primary education. In addition, this course helps students develop thinking, can help them to eliminate their weaknesses at the beginnig, bad habits and other problems resulting from the use of ICT that pupils bring with them to the school environment. Students come into daily contact with ICT and therefore it is necessary to educate them on proper access to them. This problem occurs increasingly in children of preschool age, that's why in the state educational program we can find recommendations to address this issue as early as kindergarten.

1.1 Characteristics of the subject informatics education in

Because in this subject it is necessary not only to teach students skills in working with ICT to acquire digital competences, it is important to acquire competences of media and information, assimilate name of this subject in primary education as informatics education. Name tells us that it is essential for younger school age students to be raised, not just educated.

Informatics education has its goals and mission. According to State Educational Program – ISCED 1 [8] is mainly to help pupils to make them understand the basic concepts, processes and techniques that are used when working with information. This builds so called informatics culture, which is supposed to educate to effective use of ICT. Informatics education in primary education thus builds a foundation for computer science in lower secondary education.

The mission of Computer Science and Information Technology education is intertwined. Both items should lead students to understand basic concepts as well as the procedures and techniques used when working with data and information flow in computer systems. It has not only lead to efficient use of ICT but also to respect the legal and ethical principles of their use [9]. Computer Education is compulsory from 2nd year of primary school with the time appropriation one lesson per week. Several school heads in Slovakia it put it to the school curriculum from the first year.

Appendix of the State Educational Program – Informatics education offers a wide range of uses of diverse educational software, because topics are built quite freely and they are repeated every year with increasing difficulty.

The thematic areas relate to the following areas [9]:

- *Information around us* (set is designed to work with text, graphics, audio, video, information, entertainment, etc..).
- *Communication through ICT* (set focuses on the communication possibilities of ICT, internet work, highlight security risks and pitfalls when working with the Internet, choice and selection of information).
- *Procedures, problem solving, algorithmic thinking* (this set focus on specific troubleshooting procedures through ICT, the basics of algorithmic thinking and concepts of algorithm and program, programming).
- *Principles of operation of ICT* (set is devoted to the description and understanding of ICT, working with hardware and software, etc..).
- *Information society* (here the students should have become familiar with examples of the use of ICT in everyday life and realize the need for a critical and reflective

attitude towards available information, also we have to lead students to responsible use of media).

The base topics of the informatics education curricula contains introduction to the work with computer, the base skills and abilities in manipulation with hardware and sotware (mainly graphics editors, text documents, working with files and folders, developing of the algorithmical thinking, behaviour and security in the work with internet, to create and to present different kind of presentations, etc.).
Pupils need to acquire informational and media literacy, which belong to the package competencies in primary education. In the world of information they need to learn not only skills and work effectively with ICT but primarily to select information and work with them. Especially younger school age pupils can not process large amounts of information at the appropriate level, therefore there is so important teacher's personality. The influence of media to pupils is possible to see in changes of pupils' behaviour. „Changes in the process of communication, domination of the use of the media in free time over games in the open cause essential changes in the budget of free time, and they influence in a significant manner the behaviour of children.“ [11] This space offers a particularly informatics education, which in addition to the acquisition of competences mentioned, offers a wide range of developing cross-curricular activities.

1.2 Developing of the Cross-curricular relationships

As we said informatics education can significantly contribute to developing cross-curricular activities, because within the lessons, we can not only involve the processing of other curriculum subjects, but also deepen the knowledge, habits and value. This can be seen even on the basis of the objectives of this subject. "The aim of informatic on the Primary school is familiarization with the computer and its possible use in everyday life. Through the application appropriate to the age of pupils to acquire basic skills in using the computer. In the context of Cross-curricular relationships the pupils using a variety of applications practice their basic subject matter of mathematics, Slovak and foreign language, acquire knowledge with using educational programs of natural science and geography and develop their creativity and aesthetic sensitivity in different graphic editors. We need to emphasis the understanding the options that we can use in everyday activities, not to manage application control." [9]

In this subject we can contribute to the fact that we the combine knowledge into a coherent picture. With ICT, pupils encounter in other subjects, so we have a number of options to meet these objectives. It is necessary to choose appropriate and effective methods, which verifies effectiveness teacher can confirm in practice and assess the results, since the methods are chosen effectively.

"Cross-curricular relationships bring unconventional teaching methods that develop the creative abilities of the pupil, his logical thinking, independent learning and prepares students to deal with life situations. Focus more on cross-curricular relationships is therefore beneficial to the knowledge and skills that students acquire are comprehensive." [7] It helps students to think in context, link specific issues under various themes. At the same time more student meets with the application in practice and, thus, the subject matter is not so unknown, unrelated to reality.

1.3 The need for using the ICT in primary education

The curriculum defines also competencies in the using of information and communication technologies. The pupil must [2]:

- know the using of ICT in everyday life,
- understand, that using of ICT need to have critical and considered approach to available informations,
- responsible to use interactive media – understand the possible risks they bring,
- through the ICT realize partial tasks and outputs in the frame of project and cooperative learning.

We want to point out the factors associated with the use of ICT in educational process and also the disadvantages and problems that are associated with their implementation.

According to Oldknow & Taylor [6] we can identify at least three reasons for promoting the integration of ICT in primary education:

- *Desirability:* In terms of students, the use of ICT may stimulate their motivation and curiosity; encourage them to develop their problem-solving strategies. In terms of teachers, the use of ICT may improve their efficiency, release more time to address students individually, stimulate re-thinking their approach to teaching and understanding.
- *Inevitability:* Many fields of publishing have moved from printing to electronic form. This applies to conference proceedings, reference works such as encyclopaedias, small-circulation textbooks, special journals, etc. At the same time, pupils are faced with lots of information especially in the online environment and they need to acquire skills in working with them and also develop critical thinking skills.
- *Public policy:* In Slovak National Curriculum ISCED 1 is defined that Informatics education as a subject belongs to the group Mathematics and Working with Information. And also to work with ICT, teachers should use it within any subject according to available options the school provides and skills that teachers themselves have when working with them.

2 Research on the use of ICT by teachers of primary education

Nowadays we can observe also based on some research as the most common use of ICT in educational process not only the on the first stage of primary school. Some teachers only use PowerPoint in which they prepare the presentation used in the exposure part of the lesson or students can work with a computer and their action is only unorganized surfing. We recognize that not only based on research conducted in Slovakia (see in Karasová [3]; Kubiatko - Haláková [4]; Fančovičová – Prokop [1], and others), as well as thanks to practical experience. Many teachers do not use the potential of ICT, which is hidden in them, because they do not have much experience or fear their skills were inadequate , or they consider it difficult.

We conducted research, which focused on the use of ICT by teachers in primary education. Research was conducted in the months of May-June 2013. In the research we used selected quantitative and qualitative methods. The aim of the questionnaire in qualitative research was to see the use of ICT by teachers (use of different kinds of teaching techniques and software) across different subjects and stages of the learning process. The research was conducted on a sample of 70 teachers of primary education, particularly in the area of central Slovakia. In this paper we will shows the partial results which was devoted to the teaching with the most used didactical technologies and educational software by teachers during the lessons. The research was oriented only to using ICT in selected school subjects and we would like to find the frequency of using selected kinds of didactical technologies and educational software.

2.1 Use of ICT in different subjects

We focused on how much and if at all teachers use ICT in different subjects. Since informatics education in Slovakia is the subject that with the use of ICT is closely linked, it is understandable that teachers are using ICT. In the results we present group of subjects (mathematics (MAT), Slovak language (SJ), foreign language (CJ), music education (HV) technical skills (PRAC.V.), geography education (VLAST) and science education (PRÍR)), while comparing with informatics education in primary education.

We used cluster analysis with the help of Chic software. The following figure represents prior similarity tree.

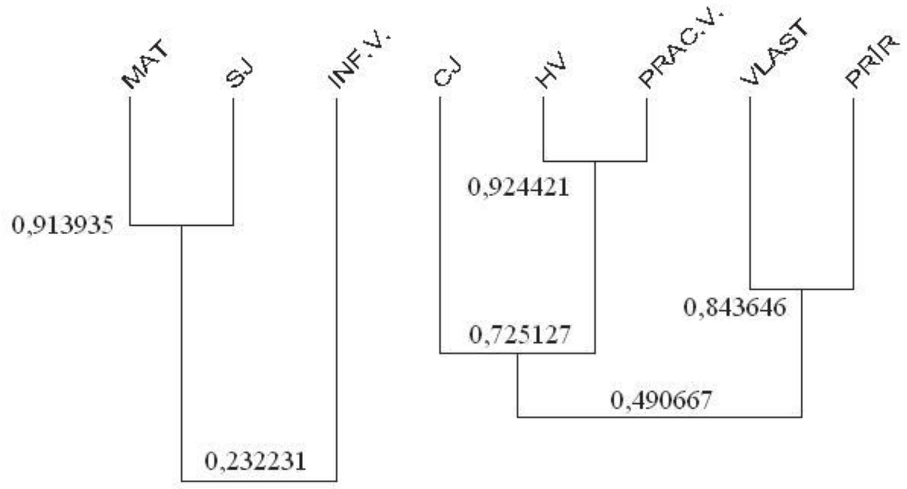


Fig. 1. Similarity tree of the factors – different schools subjects and Infomatics education

Figure 1 shows that the frequency of use of ICT is very similar in the subjects of mathematics and Slovak language (mother tongue). The great similarity is between the geography and nature science. An interesting fact was the discovery of the similarity between music and technical skills subject. In the computer science education virtually all teachers should use ICT, making it difficult to compare the use of ICT for informatics education with other subjects.

2.2 Use of different types of teaching techniques in teaching

Order of the different types of didactic techniques in the following table we got to the ranking average degrees scale where for each type of didactic techniques could returne response rates ranging from 1 to 7.

Table 1. The order of occurrence of the use of different kinds of type of didactic techniques in teaching

Order	Average	Type of didactic techniques
1	6,30	PC/Notebook
2	5,36	Radio/mp3
3	4,79	Dataprojector
4	4,39	DVD player
5	4,01	Interactive whiteboard
6	2,19	Overhead projector
7	1,75	Robotic toys
8	1,65	Voting devices

The table shows that the most commonly used type of teaching techniques in the context of ICT is computer or laptop. The least used among teachers is a voting machine. Near the top were placed devices that teachers master best while having an affordable price. At last rungs devices that are complicate to operate and more expensive, beside the overhead projector, which is already an outdated device, so its use is gradually disappearing. Teachers use teaching techniques depends greatly on their skills and experience to work with it.

2.3 Using various educational software in education

We focused on the frequency of use of programs in the teaching of any subject. We present the results of a group of educational software. We used the package od software, which is most often used in Slovakia in primary education. Alik - cheerful Mathematics (Alik), Children's Corner 1-5 (DetskýKútik), Hot Potatoes (Hotpot), programs to work with interactive whiteboard (InterwriteWorkspace, ACTIVstudio) , graphic editors (Paintbrush and Tux Paint - Paint), PowerPoint (PP), children's programming languages (Baltie and Imagine) and Internet (educational portals for teachers in Slovakia: zborovna.sk, modernyucitel.net).

The following figure represents the similarity tree of mentioned factors.

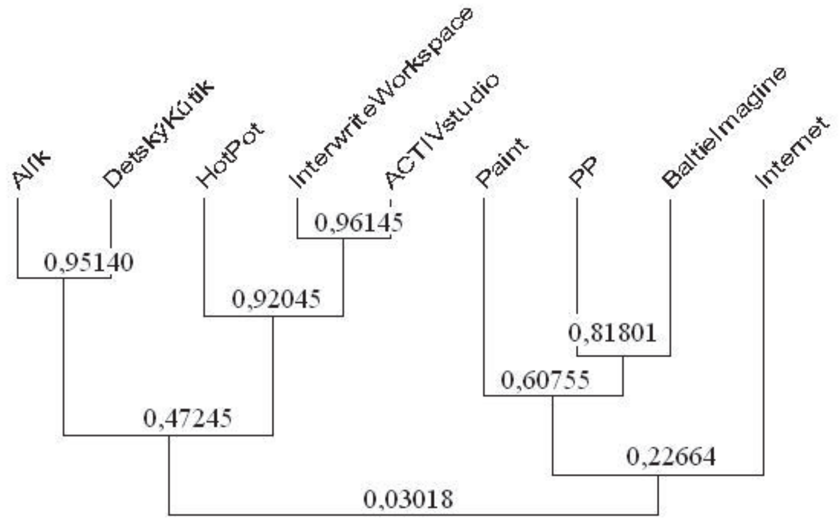


Fig. 2. Similarity tree of the factors – different types of educational software

We have discovered the similarities in using the following types of software:

- Alík – Children´s corner,
- Interwrite Workspace – ACTIV studio, connection to Hot Potatoes.

Alik is educational software, which is aimed at practicing of arithmetic for pupils aged 6-9 years. Children's Corner is a set of programs, which is divided into 5 parts. Each section focuses on practicing knowledge of several subjects: Mathematics, traffic education, geography and Slovak language (mother tongue) for the primary stage. Interwrite Workspace and Active Studio software is designed to work with the interactive whiteboard. Hot Potatoes is a program for creating quizzes and tests that teachers mainly used for diagnostic. According Majherová [5] Imagine and Baltie are children programming languages.
Context resulting from the similarity of programs or key of competences of students, that these programs develop.

2.4 Use of ICT in different phases of educational process

We studied this aspect from the point of view of these phases of educational process: motivational (M), expositional (E), fixating (F), applicational (A) and diagnostical (D).

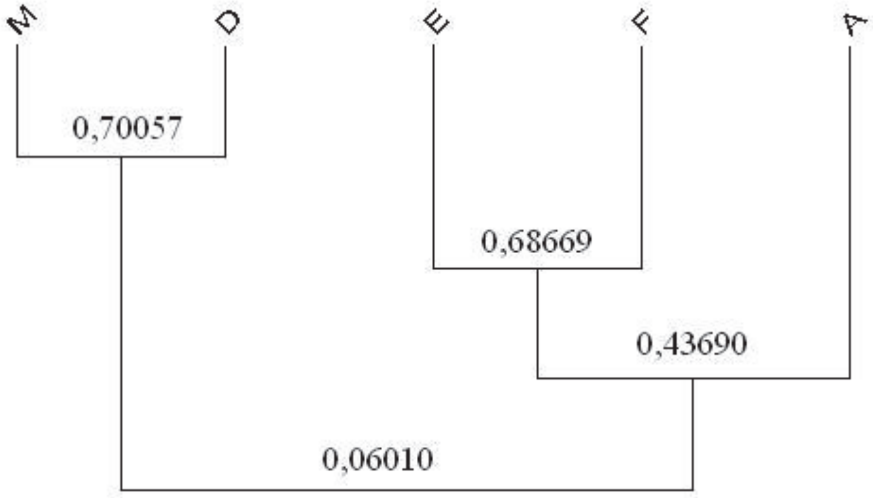


Fig. 3. Similarity tree of the factors – different phases of the educational process

We have discovered the similarities in the following pairs: motivational and diagnostic, exposure and fixation. Most teachers use technology in the exposure and the fixating phase, in motivational and diagnostic phase are much less used.

3 Conclusions

Teachers most often use ICT in education in informatics education, geography education and natural sciences education. Teachers in primary education use less ICT in mathematics, Slovak language, music education and technical education. We observed a shift in the use of ICT in mathematics and Slovak language to geography education and natural sciences education.

In terms of the types of teaching techniques there is a need for continuous training of teachers in the use of various types of teaching techniques to gain the appropriate experience and skills. It turns out that there is still a certain group of teachers who are afraid to use ICT.

Frequent software utilities are PowerPoint presentations, as well as electronic materials available on various portals, which are accessible to teachers and created by other teachers. PowerPoint presentations are too static and teachers don't use more interactive components. Still there is less frequency of use of educational software, while these softwares have the possibility of feedback and meet the requirements placed on them.

As teachers mainly use PowerPoint presentations and freely available Internet materials and ICT is used in the exposure and fixation phase and much less in other stages of the learning process. We consider it particularly important to strengthen the motivational phase, because different options of dynamic software can help to motivate students to solve various problems. Facilitate the modeling of real-life situations, which is focused on the national educational program ISCED 1.

At universities that prepare teachers for education lacks implementation of these changes in relevant bachelor's and master's programs for primary education teachers. International experiences, which may be helpful for Slovak universities, are valuable.

Practicing teachers would need continuing education focused more on methodological aspect of the use of ICT in their subjects with appropriate methodology adapted to the subject. The use of ICT in terms of pupil, brings specific psycho-hygienic problems that teachers should take into account in their practice. Then it is possible to achieve a balanced and effective use of ICT in the educational process.

References

1. Fančovičová, J., Prokop, P.: Postoje žiakov vybraných základných škôl k informačno-komunikačným technológiám. In: E-pedagogium. vol. 1, pp. 16 – 27. Univerzita Palackého, Olomouc (2006), http://www.upol.cz/fileadmin/user_upload/PdF/e-pedagogium/e-ped_2-2006.pdf
2. Jablonský, T.: Cooperative Learning in School Education. FALL, Kraków (2006)
3. Karasová, M.: Využívanie informačných a komunikačných technológií v súčasnej katechéze. VERBUM, Ružomberok (2012)
4. Kubiatko, M., Haláková, Z.: Používanie IKT vo vyučovaní biológie. In: Biologie – Chemie – Zeměpis. vol. 2, pp. 72 – 74. (2007), http://www.kubiatko.eu/clanky_pdf/pouzivanie_ikt_vo_vyucovani_biologie.pdf
5. Majherová, J.: Use of program Baltie 4 C# in teaching of informatics. In: Bednarczyk, H., Sałata, E. (eds.) DIDMATTECH, pp. 322-326, Technical University of Radom, Radom (2010)
6. Oldknow, A., Taylor, R.: Teaching Mathematics using Information and Communications Technology. Continuum, London – New York (2003)
7. Pálinkásová, I.: Využitie IKT v medzipredmetových vzťahoch. (2007), www.cenast.sk/files/documents/2007/523/palinkasova.pdf
8. Štátny vzdelávací program. ISCED 1 – primárne vzdelávanie. Bratislava (2008), www.statpedu.sk/files/documents/svp/.../isced1/isced1_spu_uprava.pdf
9. Štátny vzdelávací program. Informatická výchova – príloha ISCED 1 – primárne vzdelávanie. Bratislava (2008), www.statpedu.sk/files/.../svp/.../isced1/.../informaticka_vychova_isced1.pdf
10. Balanskat, A.: Study of the impact of technology in primary schools. Synthesis Report. EACEA, Brussel (2009), eacea.ec.europa.eu/...impact...primary_school/02_sy...
11. Juszczyk, S.: Media influence on children and adolescents. In: The New Educational Review. vol. 3, pp. 93 – 110. (2004)

Emerging Technologies and Tools for Informatics

An early evaluation of the HiPPY tool usage: the France-IOI case study

Mahdi Miled[1], Christophe Reffay[2], Mona Laroussi[3]

[1]École Normale Supérieure de Cachan, France
mahdi.miled@ens-cachan.fr

[2]Université de Franche-Comté, France
christophe.reffay@univ-fcomte.fr

[3]Institut National des Sciences Appliquées et de Technologie, Tunisia
mona.laroussi@insat.rnu.tn

Abstract. We present in this paper early results of a large-scale experiment about the use of a prototype called HiPPY (an epistemic HyPermedia to learn PYthon language). This tool was integrated in the France-IOI platform to study solving strategies mainly in high school in France. Conceptually, it is based on a fine grain learning resources called "epistemes", on a dynamic navigation system and on an individualized epistemic diagnosis. In addition of traces collected from the interaction with the system, we conducted a questionnaire dedicated to learners to judge the satisfaction and the utility of the overall system. We identified eventual strategies in relation with the solving process. We propose further to combine computed traces and declarative ones mainly got from a questionnaire to verify and validate several hypothesis taking into account privacy issues. According to the questionnaire, the emerging profile is widely related to beginners in programming.

Keywords: adaptive hypermedia, traces, trajectories, learning programming, Python.

1 Introduction, context and research questions

Appeared in autumn 2012, a new elective course dedicated to pupils in high schools in France also called ISN (*"Informatique et Sciences du Numérique",* i.e.: "Informatics and Digital Sciences") brought several changes to promote a renewal in teaching informatics in France [15]. Supported by the French association training International Olympiads in Informatics (IOI) challengers, the France-IOI platform offers a hundred exercises covering the practical part of the ISN topics. These exercises are available in seven programming languages. Python seems to be a good candidate to be retained as a language taught in this introductory course [3] [10]. We defined a graph of fine grain programming concepts (named *epistemes*) for Python in [13] to support this set of exercises. Distant users solve their exercises and alternatively navigate through the HiPPY epistemic graph to find the needed concepts to solve an exercise. Our research questions are essentially related to (1) identifying solving problems trends from data analysis, (2) assessing the integration of the HiPPY prototype into the France-IOI platform and (3) assessing its usability and its utility from a questionnaire. We present in the first part the HiPPY tool, its conceptual foundations, its main features and its use as a part of the France-IOI platform. We describe, in the second one, the experimental stage including collecting data associated with interaction traces and a questionnaire. We also give first results got from the experiment. We conclude with research perspectives and future directions.

2 The HiPPY prototype

A lot of research dealt with adaptive hypermedia regarding techniques, methods frameworks and environments [1]. Several environments were also devoted to learning programming [7] [2]. Various other environments allow evaluating learners and making correlations between usage and performance sometimes in following learners approaches to analyse usage [4]. Kumar system in C++ [8] and Logic-ITA [11] are case illustrations. Other situations like [9] seek to determine what types of knowledge are exploited to solve a given problem. As an adaptive hypermedia, the epistemic HyPermedia to learn PYthon Language (HiPPY) has three major components: a graph of epistemes [12], a dynamic navigation and an individualized epistemic diagnosis. A preliminary form of HiPPY was already introduced and detailed in [13] mainly about its graph of epistemes that represents a set of fine grain concepts. The general dynamic navigation tool in the HiPPY prototype offers two approaches:

- A **task-oriented** approach**:** At the exercise level, the provided graph contains only helpful epistemes to solve the current exercise or task.
- A **generic** approach: At the chapter level, the student accesses the graph enclosing epistemes of a given topic, i.e.: useful epistemes to solve all exercises of a given chapter. At the course level, the whole graph is also available.

For both of these approaches, the obtained graph presents epistemes as nodes in various forms according to the user familiarity with it, in order to visually differentiate: *unvisited and non validated* epistemes, those *visited but not yet validated*, and *validated* ones (but not necessarily visited). Currently, an episteme is *validated* once a related exercise has been successfully submitted. To navigate the graph (see **Fig. 1**), a simple click allows clear view and displays only the dependencies of the selected episteme. A double-click opens the content of the episteme. Students may also open epistemes thanks to links to ancestors (pre-requisite) directly in the text content.
As a result, the visualization of this graph is intended to give the user an individualized epistemic diagnosis, especially useful for users who diverge from the order of the suggested sequence of exercises (see part 3.2).

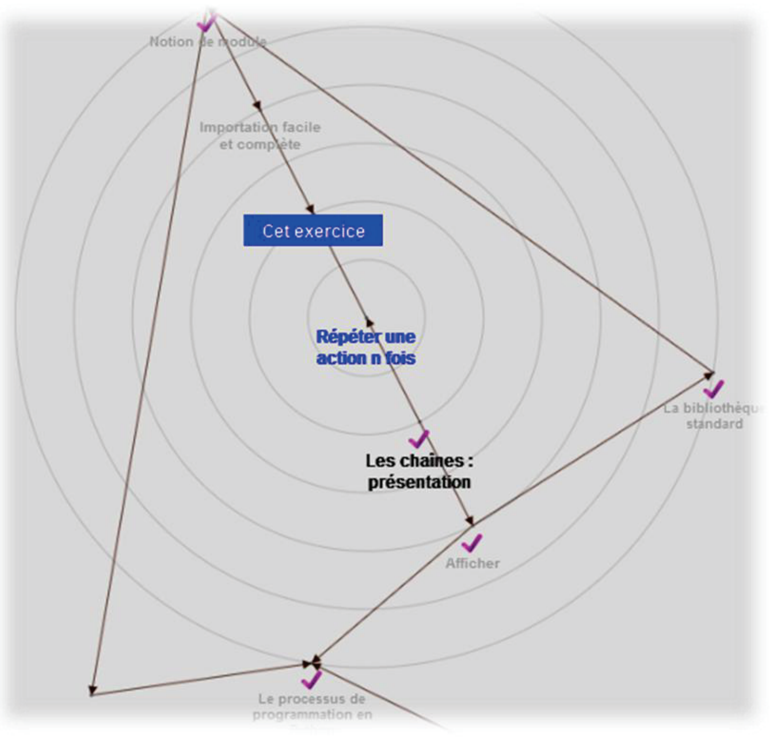

Notion de module
Importation facile et complète
Cet exercice
Répéter une action n fois
Les chaînes : présentation
La bibliothèqu standard
Afficher
Le processus de programmation en

Fig. 1. A task-oriented sub-graph of epistemes in the HiPPY tool

3 Experiment

A pre-stage (from April 2013 to June 2013) was conducted to test traces collecting process and interaction with the system. The reported experiment really started in September 2013 that coincided with the new school year. We explain in this section collecting, modeling and analyzing traces processes. We expose afterwards early results both from traces analysis and from the questionnaire. We finally attempt to combine the results obtained from both of these data collection and analysis.

3.1 Collecting and modelling traces

Thanks to primary traces collected in SQL tables, we were able to compose them with different techniques of selection, filtering or composition to elaborate more useful and significant traces also called modelled traces [5]. These traces are intended to support the 3-level trajectory defined as follows:

- The **macro-level trajectory** (**Fig. 2**) is the ordered sequence of successfully solved exercises for a specific user in a given period and for a given set of exercises. The white exercises are unopened exercises, those that have been opened (but not yet solved) contain small slashes and those that contain dots are solved. The solid arrows denote the sequential order suggested by the platform. In this example, the exercises **E_1**, **E_2**, **E_3** are part of a particular topic (or chapter involving a set of epistemes). **E_3**, **E_k** and **E_t** obey to the same objective. This objective can partially cover several topics. Macro- trajectory of learner **L_1** would be the sequence (**E_1**, **E_k** ,**E_t** and **E_n**).

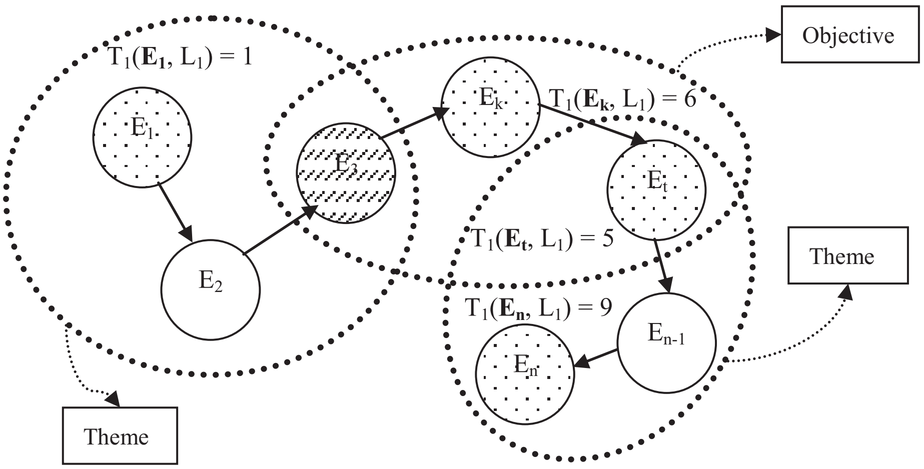
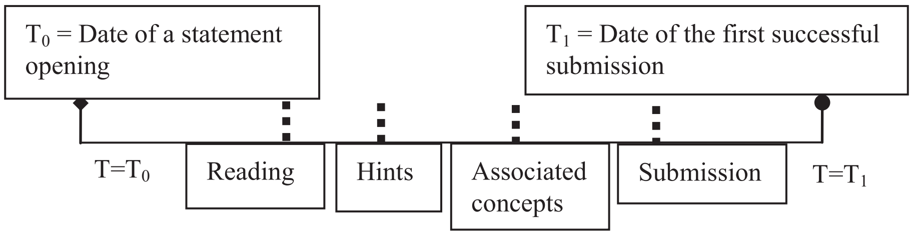


Fig. 2. Ordered set of exercises, themes, objectives and example of a macro-trajectory for the learner L1

- The **micro-level trajectory** (**Fig. 3**) is the ordered sequence of events (reading a task or an exercise, associated concepts, submission etc.) for a specific user between the (first) visit of the exercise description (i.e.: task definition associated with the *read* event) and the first submission of a correct solution.



T_0 = Date of a statement opening
T_1 = Date of the first successful submission
$T=T_0$
Reading
Hints
Associated concepts
Submission
$T=T_1$

Fig. 3. Micro-trajectory structure

- The **epistemic-level trajectory** (**Fig. 4**) which belongs to a task (or an exercise) concerns the list of epistemes used at a wide sense according the different events: epiGraphClick (for a navigation purpose), epiGraphDoubleclick (opening from the epistemic graph), epiSeen (opening episteme contents) and epiClick (opening from prerequisites list).

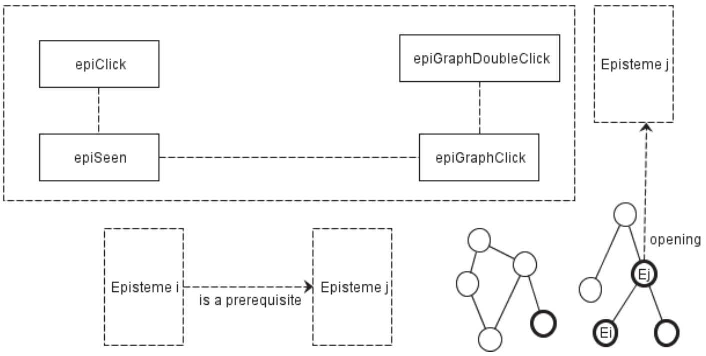


Fig. 4. Epistemic trajectory components

This epistemic trajectory matches the "Associated concepts" tab. It was necessarily incorporated when we first integrated the "Associated concepts" tab in the France-IOI platform.

3.2 Revisiting previous statistics of France-IOI platform

Some traces of the France-IOI platform (before inclusion of the graph of epistemes) already allowed us to analyze some trends in the macro- trajectories. Global statistics indicate that about 865 registered users since September 2012 and who chose Python language, 721 validated at least one exercise, 438 have validated at least six and 220 have validated at least sixteen (out of 106). The first two chapters (as a thematic list of exercises) include sixteen exercises. We can notice that the 220 having completed at least sixteen, 75 have strictly followed the prescribed order without avoiding any exercise (34 %). 66 jumped or inverted between 1 and 4 exercise (30%). The remaining 79 skipped or reversed five or more (36%). These initial data show a very sequential presentation giving no indication of the dependencies between the concepts. Despite this fact, two-thirds of users of the France- IOI platform having completed at least 16 exercises diverged from the suggested sequence. We think they could (1) directly benefit from associated concepts related to exercises (made in advance) and (2) better perceive the concepts they would not validated.

3.3 Traces results and discussion

We present here some results from traces analysis and questionnaire answers. We could observe that the first effective results from the experiment which began in September 2013 confirm the partial ones retrieved during the pre-experiment.

Early traces analysis.
The trace analysis included two stages: a pre-experiment which started in April 4th 2013 (high school course) and an experiment which started in September 1st 2013

(collected traces are up to October 18th). We'll mainly present results about micro and epistemic trajectories.

Pre-experiment results.
Number of learners who used the "Related concepts" tab reached 200 users till May 17th 2013 (it was about 32 users until April 12th, 80 until April 24th). Let us see first epistemic trajectory indications. The **Fig. 5** is an example of that trajectory. The first column is the user ID (21372), the second is the episteme ID, the third is the time stamp, the fourth is the event associated with the episteme being used at the indicated time-stamp. As for the fifth, it distinguishes between task oriented and generic approach. The last column is the task (or exercise) ID. 62 epistemes were used (out of 69). The episteme "print" ("Afficher") is the most used episteme. From April 4th to April 12th 2013, out of 567 instances (of epistemic trajectory) 530 were task-oriented, 37 were Non-oriented task. Out of these 567 instances, 299 were related to epiGraphClick, 136 to epiSeen, 78 to epiClick and 54 to the epiGraphDoubleClick event.

21372	14	05/04/2013 09:20	epiClick	Task	1979
21372	**14**	05/04/2013 09:20	epiSeen	Task	1979
21372	20	05/04/2013 09:22	epiClick	Task	1979
21372	**20**	05/04/2013 09:22	epiSeen	Task	1979
21372	63	05/04/2013 09:22	epiClick	Task	1979
21372	**63**	05/04/2013 09:22	epiSeen	Task	1979
21372	45	05/04/2013 09:23	epiClick	Task	1979
21372	**45**	05/04/2013 09:23	epiSeen	Task	1979
21372	7	05/04/2013 09:23	epiGraphClick	Task	1979

Fig. 5. An excerpt of an epistemic trajectory

We tried also to get a snapshot to measure durations into the different tabs. According to the **Fig. 6**, we have three different users ID (21134, 21867 and 21909) for different tasks (1877, 1878, 1880 and 1881).

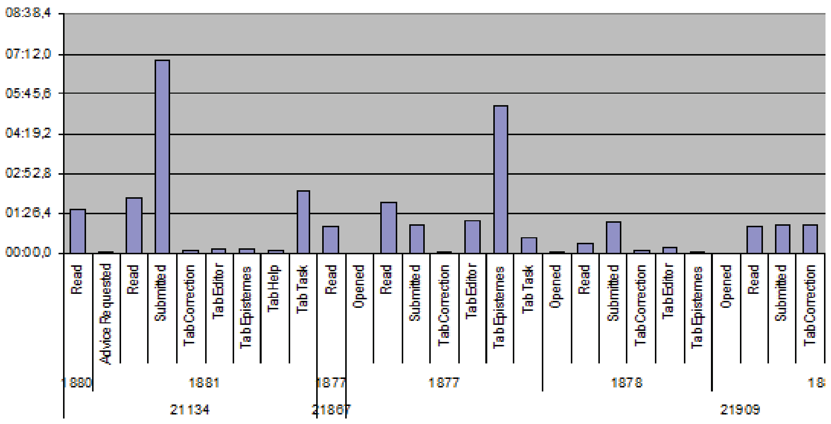


Fig. 6. Elapsed time on different tabs for three users

These tasks are the first in the France-IOI order. It may explain why the duration is relatively low. The maximum duration (about 7 minutes) in this excerpt is for the state "Submitted" that is actually included in the TabEditor. Observing time stays into the different tabs may be insufficient, that's why we decided to analyze tabs order in the purpose to recognize relevant patterns linked to the TabEpistemes. Here is a result (**Fig. 7**) of a limited analysis. Relevant patterns are the following: (*, TabEpistemes, TabCorrection) and (*, TabCorrection, TabEpistemes). This obviously shows two potential uses or strategies involving TabEpistemes:

- as a **resolution support** (before a submission): this is way to get more information about necessary concepts to the task. This type of use is related to the pattern (*, TabEpistemes, TabCorrection);
- and as a **consolidation support** (after a submission): this is especially done to verify which concepts were used after a successful submission. This corresponds more to (*, TabCorrection, TabEpistemes).

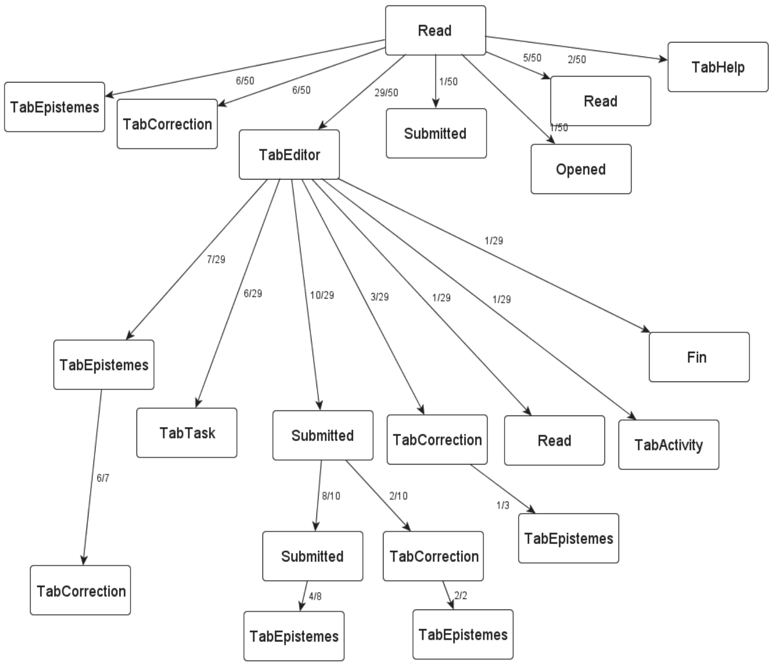


Fig. 7. Recurrent patterns in micro-trajectories

Even though we got 7/29 versus 3/29 patterns, we cannot conclude so far in terms of proportion to say whether it is more used as a resolution support or as a consolidation support.

Some experiment results.
Total distinct users who used at least one episteme are 858 (between September 1st and October 18th 2013). Total registered users during the same period were 2762. 2105 selected Python language. 1018 users solved at least 10 exercises (or tasks). In the experiment, 8901 instances (90.94%) were task oriented approach, 887 were belonging to the generic approach, confirming that the graph of epistemes was widely used in a task context. The most visited episteme is: « *Afficher* » (print, id=8). This is also a confirmation that we saw in the pre-experiment stage. In the experiment, 66 epistemes out of 69 were used. The results were confirmed with a more significative value in the experiment (out of 9788 instances): 5549 Epigraphclick (56.69%), 1181 Epiclick, 2085 Episeen and 973 Epigraphdoubleclick.

3.4 Questionnaire results and discussion

A questionnaire was given to users to evaluate and give directions to improve the current HiPPY prototype. This questionnaire was also useful to validate some hypothesis about the users' profiles (beginner, intermediate or advanced), getting information about the practice of programming not necessary on Python language, and identifying the proportion of users belonging to ISN courses. Technically, the questionnaire was a google document where each user can answer online. We put the condition that a user must have solved at least 10 problems. Here are some of the questions asked.

Identifying users demographic profile and programming profile
The users were asked to give their gender, age (optional) and last class attended or got diploma. We also asked if the users belong to an ISN group. Information about the programming profile were requested: "Which is your programming level before you join the France-IOI platform?" the different possibilities were: "I have already written more than 10000 lines of code in one language, I have made some tries in at least two programming languages, I have good notions but no practice" and, "I discover programming for the first time".

Evaluating usability, effectiveness and utility
Here are respectively questions to inquire usability satisfaction and effectiveness of the *Individualized Epistemic Diagnosis* (IED): "How do you find the navigation system? How do you find the effectiveness of the IED?" The user could choose from four alternatives (unsatisfactory, somewhat satisfactory, satisfactory and very satisfactory). As for utility of the "associated concepts" tab, the user ought to select an answer from "not useful, sometimes useful, useful" and "very useful".

Explaining some users' actions/preferences
In this section, the first question was "Which is the element that you appreciate the most in the "Associated concepts" tab". Possible answers were: Animation and nodes centering, displaying of assimilated concepts, reminders about a concept or prerequi-

site concepts displaying for an exercise. The second question was "Why do you consult "Associated concepts" tab?" Possible reasons were: to verify which concept is linked to another, to see assimilated concepts, to see exercise prerequisites, to see reminders about a concept or other reason.

Some questionnaire results

We collected 121 responses (from November 5th until December 4th 2013). Flat data give 25% female users. 52% coded less than 10 code lines, 21% between 10 and 100. 41% are discovering programming for the first time and 22% have good knowledge but not practice. 42% have begun programming in 2013. 38% of total users belong to an ISN group. All These elements denote a dominating beginner profile in programming. Concerning dynamic navigation satisfaction (**Fig. 8**), 50% are satisfied (including very satisfied). This percentage is higher when it concerns male users not belonging to ISN groups (n=58) the satisfactory rate reaches 58% (among 10% very satisfied).

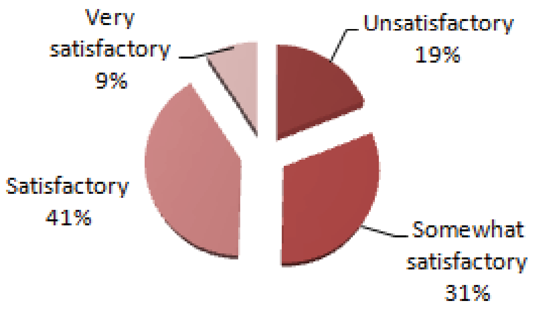


Fig. 8. Evaluating the dynamic navigation satisfaction (n=121)

As for the individualized epistemic diagnosis (**Fig. 9**), 64% find it satisfying (including very satisfying). For male users not belonging to ISN groups (n=58), the individualized epistemic diagnosis is very satisfactory (9%), satisfactory (62%), somewhat satisfactory (22%) and unsatisfactory (7%). Out of 121 users, 33% find the HiPPY tool useful, 11% very useful, 35% sometimes useful and 21% not useful, but at the opposite with other results, here users belonging to ISN groups find it more useful than others, (42% useful and 12% very useful). Results concerning utility show us that 44% find it useful, so we decide to investigate more and try to make a kind of a cross validation of these declarative results.

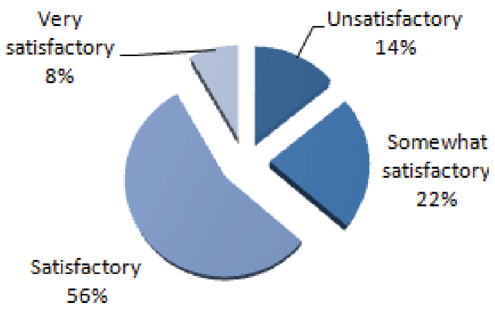


Fig. 9. Evaluating the individualized epistemic diagnosis effectiveness (n=121)

Combining both partial traces and questionnaire results.

Assuming that if a user find useful the "associated concepts" tab, we tried to verify he had more chance to get more access to this tab. It is partially true because, it could be an explanation of a harder task which demands more reminders about the concepts studied to solve the exercise. We defined a new ratio-metric called "Epistemes Tab Usage Rate" (ETUR) which concerns all tasks without distinction for each user as follows:

$$ETUR = \frac{\text{Usage Epistemes Tab Frequency (UETF)}}{\text{Usage of All Tabs Frequency (UATF)}}$$

We obtained a graph (**Fig. 10**) mapping the declarative results (about the utility of the Associated concepts tab) and the effective ones (ETUR ratio-metric).

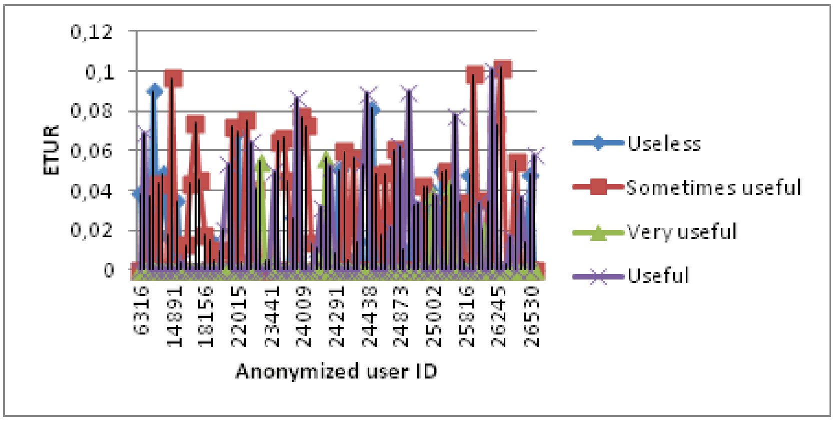


Fig. 10. Relation between declarative utility and effective usage

The horizontal axis represents anonymized users, the vertical axis concerns the associated ETUR values. What we expected was to get highest values of ETUR with very useful markers. It wasn't the case here. Highest values are clearly for "useful" marker and "sometimes useful" one. Other contradictory results give ETUR high values (0.09) for useless marker. Here are the different values (Table 1) of the Usage Epistemes Tab Frequency (UETF), the Usage of All Tabs Frequency (UATF) and the Epistemes Tab Usage Rate (ETUR).

Table 1. Average, deviation, maximum and minimum values of UETF, UATF and ETUR

	UETF	UATF	ETUR
Average	13.5	299.5	0.048028
Deviation	2.12132	129.400541	0.013668
Maximum	210	3218	0.101796
Minimum	1	181	0.003023

Even though these results give some contradictory views, it would be interesting to investigate more to see for example specific durations. This aspect wasn't taken into account here. We could also see if these ETUR values are more significant with micro-trajectories and declarative utility rate combination.

4 Conclusion

We presented results concerning the integration of an epistemic HyPermedia to learn PYthon language on the France-IOI platform. This HiPPY tool works in conjunction with the France-IOI exercises list. It relies on a graph of epistemes, on a dynamic navigation and on an individualized epistemic diagnosis. Usage analysis results through collected traces (such as task oriented approach, graph navigation events) in the pre-experiment were validated and confirmed in the experiment. Thanks to certain frequent patterns, we identified two potential uses or strategies involving the TabEpistemes: as a resolution support and as a consolidation support. We also retrieved useful information from the questionnaire. The dominant profile of users is related to beginners in programming. 64% of the users (n=121) found the individualized epistemic diagnosis for instance satisfactory and very satisfactory. We tried to combine partial traces results and declarative results about the utility. The ETUR values were discordant with what we expected to get. Future perspectives will include a more specific characterization of strategies, using data mining techniques to discover from effective traces and declarative data more elaborated models in relation with clustering, and association rules.

Acknowledgments.
We would like to thank the France-IOI association for allowing us to share and federate learning resources.

References

1. Brusilovsky, P. (1999). Adaptive and intelligent technologies for web-based eduction. *KI*, *13*(4), 19‑25.
2. Brusilovsky, P., & Sosnovsky, S. (2005). Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK. *Journal on Educational Resources in Computing (JERIC)*, *5*(3), 6.
3. https://www.coursera.org/course/interactivepython. An introduction to Interactive Programming in Python | Coursera. Retrieved January 28th 2014.
4. Delozanne, E., Le Calvez, F., Merceron, A., & Labat, J.-M. (2007). Design Patterns en EIAH: vers un langage de Patterns pour l'évaluation des apprenants. *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, *14*.
5. Djouad, T., Mille, A., Reffay, C., & Benmohammed, M. (2010). A New Approach Based on Modelled Traces to Compute Collaborative and Individual Indicators Human Interaction (p. 53‑54). IEEE.
6. Dominguez, A. K., Yacef, K., & Curran, J. R. (2010). Data Mining for Individualised Hints in e-Learning. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.174.6422&rep=rep1&type=pdf
7. Hsiao, I.-H., Brusilovsky, P., & Sosnovsky, S. (2008). Web-based parameterized questions for object-oriented programming. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (Vol. 2008, p. 3728‑3735).
8. Kumar, A. (2005). Usage Analysis in Tutors for C++ Programming. In *Proceedings ofÇUsage Analysis in Learning Systems Èworkshop* (p. 57‑64).
9. Labat, J.-M., Pastré, P., Parage, P., Futtersack, M., Richard, J.-F., & Sander, E. (2007). Analyser les stratégies de résolution de problèmes en situation naturelle grâce à un simulateur: le cas des régleurs de plasturgie. In *Actes de la conférence EIAH 2007*.
10. http://mechanicalmooc.org/. A gentle Introduction to Python – Mechanical MOOC. Retrieved January 28th 2014.
11. Merceron, A., & Yacef, K. (2004). Mining student data captured from a web-based tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research*, *15*(4), 319‑346.
12. Ortiz, P. (2012). Hypermédia adaptatif à épistèmes pour l'apprentissage des langages de programmation. *RJC EIAH'2012*, 99.
13. Reffay, C., Miled, M., Ortiz, P., & Février, L. (2013). An Epistemic Hypermedia to Learn Python as a Resource for an Introductory Course for Algorithmics in France. In Local Proceedings of the 6th International Conference on Informatics in Schools; Situation, Evolution and Perspectives. ISSEP 2013, Oldenburg, Germany. (p. 111). Retrieved from http://opus.kobv.de/ubp/volltexte/2013/6368/pdf/cid06.pdf#page=111
14. Romero, C., & Ventura, S. (2010). Educational data mining: a review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *40*(6), 601‑618.
15. Tort, F., & Drot-Delange, B. (2013). Informatics in the french secondary curricula: recent moves and perspectives. In *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages* (p. 31‑42). Springer.

A Formula is an Orange Juice Squeezer - Understanding Spreadsheet Calculation Through Metaphors

Michael Weigend

Institut für Didaktik der Mathematik und der Informatik,
Fliednerstr. 21, 48149 Münster, Germany
michael.weigend@uni-muenster.de

Abstract. Manuals and class room activities related to spreadsheet calculation are often procedural (learning by doing). However, deeper understanding requires (declarative) knowledge of fundamental informatics concepts, including relative and absolute reference, formula (expression), format versus value, input versus output data. This paper discusses how to use metaphors from everyday life to elaborate spreadsheet related concepts and presents some findings from two pencil-and-paper exercises.

Keywords. Spreadsheet calculation, informatics education, co-operative learning, metaphors, concept maps

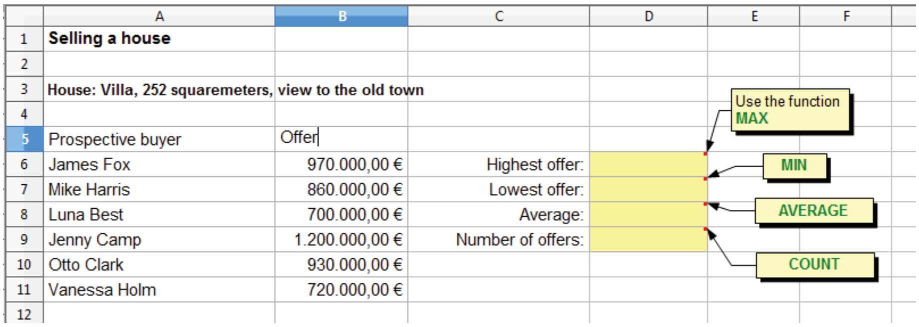
1 Procedural and Declarative Knowledge About Spreadsheet Calculation

Spreadsheet calculation (first implementation 1962 on an IBM 1130) is one of the first and most successful digital tools. Today, the ability to use software like Microsoft Excel or OpenOffice Calc for numerical calculation and modeling is regarded to be a basic informatics competence. Module 4 and AM4 of the European Computer Driver's License (ECDL) are dedicated to spreadsheet calculation (www.ecdl.com). Some educators regard spreadsheet calculation as a "quick alternative" to a programming language for introducing some informatics concepts [1]. In math and science education spreadsheet calculation is often used as a tool supporting constructivist learning [1].

The CSTA curriculum mentions spreadsheet calculation just as an example of a productivity technology tool (beside word processing and presentation), which is relevant in the strand "collaboration" [2]. The German "Educational Standards for Computer Science in Lower Secondary Education" [3] suggest using spreadsheet calculation as an example of a software system that students can analyze adopting computer science concepts like data processing and object oriented thinking. For examples, tables, cells, rows and columns can be seen as objects with attributes and methods.

Often people learn spreadsheet calculation in a procedural way ("learning by doing") based on media, a collaborative social environment and self-directed experimenting with the system. In a media-based learning style the students often follow step-by-step instructions in videos or text books. Such tutorials explain what to do in order to reach a certain effect.

Exercises may be completely embedded (via comments) in spreadsheets like in figure 1. The learner edits a half baked spreadsheet, which is completely self-explanatory.



	A	B	C	D	E	F
1	**Selling a house**					
2						
3	**House: Villa, 252 squaremeters, view to the old town**					
4						
5	Prospective buyer	Offer				
6	James Fox	970.000,00 €	Highest offer:			
7	Mike Harris	860.000,00 €	Lowest offer:			
8	Luna Best	700.000,00 €	Average:			
9	Jenny Camp	1.200.000,00 €	Number of offers:			
10	Otto Clark	930.000,00 €				
11	Vanessa Holm	720.000,00 €				
12						

Fig. 1. Spreadsheet with embedded instructions

Spreadsheets like this can be found in the award-winning website Easy4me (`http://www.easy4me.info/open-office/modul-4/`). This type of activity is even more procedure-oriented than conventional step-by-step instructions, since there is less verbalization.

Modern user-friendly spreadsheet systems like OpenOffice Calc support learning by doing. The user just starts and finds out the system functionality by exploring and experimenting. Such systems adopt affordances [4] and multiple representations to enable humans to use it "intuitively" without (much) specific technical knowledge. For example a dialog box for formatting numbers contains labeled checkboxes and entry fields for specifying the number of decimals, and additionally displayed examples of different formats. The user may choose an example and sees the abstract specification or vice versa.

Spreadsheets may contain undetected errors. Brown and Gould [5] studied spreadsheets made by experienced users. They found that 44% of the investigated spreadsheets were not correct despite the programmers' reports to be quite confident that they were error free. Kruck (2006) found evidence that certain practices help to increase the accuracy of spreadsheets (cited in [6], p. 135). Successful aids include

- "Use an organized layout to isolate data and computation areas."
- "Use descriptive labels."
- "Repeat the input data near the output."
- "Relative and absolute cell addressing should be used where appropriate."
- "Formulas should contain only cell references."

To be able to understand and follow these rules, students need explicit declarative knowledge about the underlying computational concepts of spreadsheet calculation.

Let me now shortly mention eight concepts that are elaborated in two class room activities which I am going to present later in this contribution.

- Cell. A spreadsheet is a two-dimensional grid consisting of cells. Each cell can be an input and output device for data. It may contain explicit data or a formula. A cell is identified by a cell reference like `A2`, or `$A$2`.
- Formula. A formula is an expression made of explicit data (numbers, strings), cell references, operators and function calls. Since formulas are evaluated immediately by the system they can be considered as a mechanism that calculates output data from input data.
- Copying. A spreadsheet is mostly constructed by copying. The user defines just a few basic formulas explicitly – say the first row of a table – and then copies them to many more cells. Formulas with relative references change when copied. Thus copying in spreadsheet calculation is different from the everyday concept of copying (creating identical instances).
- An absolute reference like `$A$2` addresses one unique cell of the spreadsheet. This implies two important facets of meaning: It never changes when copied and it identifies unambiguously one object.
- A relative cell reference like `A2` refers to a cell at a position relative to the cell, where it is stored.
- Format. The format of data is an attribute of the displaying cell. Numbers look different depending on the data format that has been defined. For example the strings `10%`, `0.1` and `0.9951234` may represent the same numerical object.
- Output data are produced by formulas. Therefore cells with formulas are meant to be output devices.
- Input data are meant to be processed by formulas. Some cells are used as input devices. The user is supposed to put data into these cells.
- Access control. A cell can be protected from changes by "locking". This way it is restricted to the function of an output device. In fact the protected object is the formula stored in the cell.

Some of these items may be considered as threshold concepts, "opening up a new and previously inaccessible way of thinking about something" [7]. For example it is possible to construct correct spreadsheets without understanding the principle of relative cell referencing. Some users accept the change of a formula during copying just as "part of the magic" of a user friendly system. Like an autocorrecting text processor, the spreadsheet system "guesses" or "figures out" from previous behavior the user's intentions and does the right thing. This thinking might work to a certain degree, since spreadsheets often have a similar structure (table with a header). But ignorance of the difference between relative and absolute references is a threshold, preventing the user to design and understand tables with formulas that contain an absolute reference to one unique cell with a global constant.

Explicit declarative knowledge – beyond partly unconscious procedural knowledge ("implicit knowledge") – is necessary to be able to avoid and detect mistakes, to learn new features, to create new table designs and to communicate with others about spreadsheet design. Such knowledge can be represented by regular text (like in this section), but it can also be explicated and visualized by concept maps and metaphors.

2 Concept Mapping and Metaphors

Concept mapping is based on Ausubels learning theory and was developed in the early 1970ies by Novak [8]. In this section I discuss similarities and differences between concept mapping and the class room activity used in this study. The idea of concept mapping is to construct individual "organized knowledge". In concept mapping a person designs a directed graph consisting of rectangular blocks (vertices) representing concepts and arrows between them. The arrows are annotated with a label indicating a relation between the concepts. The combination of two concepts and a relation between them is called a proposition or semantic unit. It is recommended to use a focus question that addresses in some way the knowledge domain you want to map. It might be a concrete problem you want to solve. The focus question represents a reason for constructing the concept map. Consider this focus question: "How do you use formulas and locks in spreadsheet calculation?" Figure 2 depicts a corresponding concept map, which was used in one of the activities. This structure represents a coherent system of propositions like "A lock protects a formula" or "A formula yields output datal".

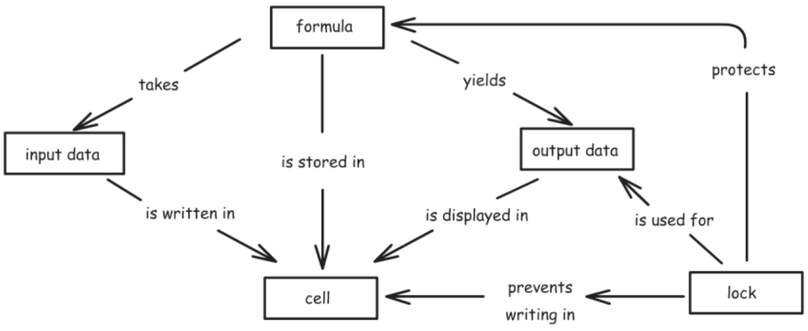


Fig. 2. Concept map used in activity LOCKS (see figure 3)

There are many ways to use concept mapping in classroom activities.

- The students create a concept map to a given focus question by themselves.
- The students get a list of concepts and construct a concept map by adding relations. A variant of this approach is to use card with concept-labels which can be arranged on the desk top (structure-laying technique [9]).
- The students finalize an incomplete concept map and add missing elements.

The general idea is to evoke individual elaboration and – in a collaborative environment – give a reason to discuss declarative knowledge, hopefully leading to a better understanding. Additionally concept mapping can be used for "diagnosis" since they indicate students' misconceptions. Based on the analysis of six meta-studies (investigating altogether 287 studies and describing 332 effects) John Hattie [10] concludes that concept mapping has a visible positive effect on learning (effect size d=0.57), especially when it is performed at the end of a learning sequence. The students need familiarity with the new field of knowledge.

The activities of this study follow the general idea of concept mapping as a way to elaborate and reflect declarative knowledge, but they do not imply the active construction of relations between concepts.

A precondition for successful concept mapping is that the creator and reader comprehend the individual concepts (boxes) within the structure (see [11]). The structure itself may explain a few semantic aspects of the concepts via relations, but this is not enough. For example the concept of copying a formula containing relative and absolute cell references is very specific and is not completely explained in the example concept map. This leads to another problem: The labels which identify concepts must be understood. A certain word or phrase – used as label – may refer to different concepts.

To improve the expressive power, a concepts map may contain "specific examples of events or objects that help to clarify the meaning of a given concept" [8, p.2]. These are put in ovals instead of rectangles to point out that these units are not labels for concepts but just alternative representations of concepts already included in the graph. The class room activities of this study focus on this feature of concepts maps. The students had to assign images from everyday life to concepts or propositions. The depicted objects were supposed to be a conceptual metaphors related to certain facets of spreadsheet calculation. Ingredients for cookie dough for example are a metaphor for input data, being processed by the cook. A cookie taken from the oven can be seen as a metaphor for output data, the result of some processing.

According to Lakoff and Nunes [11, 12] a conceptual metaphor is a transfer of knowledge from a familiar source domain to a still unfamiliar target domain. Conceptual metaphors are "vehicles" carrying the learners to new knowledge beyond their intellectual horizons. Using metaphors like the number line has a long tradition in math teaching. In informatics metaphors like stack and queue are used for data structures. Software development often starts with a project metaphor, describing the fundamental idea of a new software system in a holistic way. These examples illustrate that the usage of metaphors is a part of computer science knowledge.

Additionally to facilitating thinking, metaphors add color to computer science. Images are rich and they can evoke emotions and personal involvement. They may be designed carefully for different age groups, genders and subcultures. Elaborating metaphorical illustrations implies constructing additional meaning beyond the formalisms. For example, connecting an image depicting cookies taken from the oven to the concept of *output data,* which has been produced by a formula, highlights semantic facets like these:

- Output is something that is consumed by the user.
- Output data has a higher value than the input data.

- The quality of the output data depends on the quality of the input data and the formula processing the input.

3 Two Classroom Activities

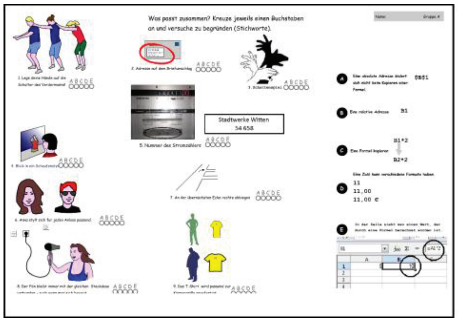
In this study, students from a computer science class (grade 9, age 15-16) at a comprehensive school in Germany were asked to do two pencil-and-paper exercises, which I call BASICS and LOCKS.

BASICS focused on basic concepts: absolute and relative reference, formula, value and format of numbers and copying. LOCKS was about protecting tables. A cell in table may serve either as input or as output device. The user of a spreadsheet is supposed to write numbers in certain cells (input). Other cells contain formulas that process the input and yield output which is displayed in the cell. Open access is a source of possible errors since the user may destroy a formula accidently by overwriting it. To lock a cell means to make it an output device and to protect the formula.

These two activities took place at different days within a sequence of hands-on lessons on spreadsheet calculation. Each activity consisted of three sessions: USE, EXPLAIN, EVALUATE.

The USE session focused on procedural knowledge. The students were asked to describe how to perform operations like changing the format of a cell display or protecting cells by picking words from a list and filling them in a cloze text. In other tasks they had to specify how cells look like, when a user has performed certain operations like choosing a format or copying a formula from one cell to another. The tasks included screenshots from the OpenOffice Calc user interface. It was *not* required to *explain* any concepts.

The EXPLAIN sessions focused on declarative knowledge. The participants were asked to explain concepts by associating images to them. The EXPLAIN session of BASICS was based on the first working sheet in figure 3. On the left hand side there were captioned images of nine objects or scenes from everyday life (like for example three persons walking in a row), On the right hand side there were descriptions of five concepts relevant in spreadsheet calculation. Each description consisted of a short text and an example. It can be assumed that the students were familiar with these concepts and had adopted them in several projects. The participants had to decide for each situation on the left hands side, to which of the concepts it would correspond most.


Was passt zusammen? Kreuze jeweils einen Buchstaben an und versuche zu begründen (Stichworte).
A B C D E
Stadtwerke Witten 54 658
5. Nummer des Stromzählers
B1
B1
B1*2
B2*2
11
11,00
11,00 €

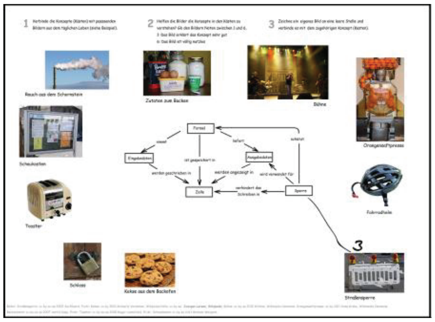
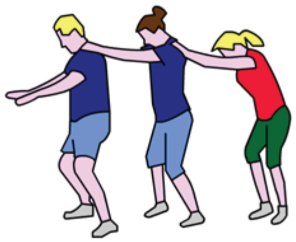


Fig. 3. Working sheets for EXPLAIN sessions

The EXPLAIN session of the second activity LOCKS was slightly different (see figure 3, right hand side). The working sheet depicts a concept map surrounded by captioned images, representing possible metaphors for the concepts in the boxes of the middle. The students were allowed to cooperate but had to fill their working sheet individually. In fact, a lot of discussion took place during this phase. Furthermore, the students were encouraged to design an image visualizing a concept on their own.

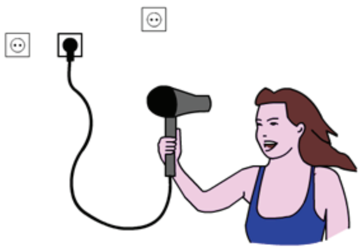
The students performed USE and EXPLAIN in different orders. Half of them started with USE and the others started with the EXPLAIN session. They had (almost) no opportunity to communicate when they solved the tasks in the two USE sessions.

The third part EVALUATE was a classroom discussion about the concepts and metaphors, which took place in the next lesson, a couple of days later. The students saw a presentation. Each slide depicted one of the images and one of the concepts for which it was supposed to be a metaphor. Then everybody got the opportunity to suggest additional interpretations. After a short discussion a written statement was shown, which explained in what way this image could be regarded as a metaphor. The students got evaluation cards and gave a mark from A to F to each metaphor indicating how far they would accept the reasoning in the explaining statement (plausibility). They gave another mark from A to F for the usability of this illustration for understanding the associated concept. Figure 4 shows two translated examples of these explaining slides.



Place your hands on the shoulders of the person in front of you.

Explanation: The expression "person in front of you" is a relative reference, since each person has a different person in front (or none).



The hair dryer is always connected to the same outlet – even when you move.

Explanation: The outlet is an absolute reference. It exists independently from the hair dryer and does not change.

Fig. 4. Two slides with explanations of metaphors that were to be judged by students

4 Findings

In BASICS the students recognized an average of only 1.7 (out of 9) of the intended metaphors ("hits") and 5.1 (out of 9) in LOCKS. The difference may be due to the opportunity for collaboration and/or more obvious metaphors in LOCKS. Of course, there is more than one reasonable interpretation for each image. For example a student reported that he considered the line of people (image 4, left hand side) to be a metaphor for copying a formula, since everybody copies the behavior of the first person in the line. Still, the number of hits reflects the ability to find metaphors to some extent.

In BASICS, students, who started with USE, found significantly more intended metaphors in the EXPLAIN part (one-sided exact Fisher's test) than those who started with EXPLAIN. This was not the case in the LOCKS activity.

Table 1 comprises some data on the students' selections and judgments of conceptual metaphors during the EXPLAIN sessions of BASICS. They are described in the style suggested by Lakoff and Nunez [11] by expressions *A* is *B*. *A* is an informatics concept related to spreadsheet calculation (target domain) and *B* is an object from everyday life (source domain).

For each depicted object *B* from everyday life students had to choose a corresponding spreadsheet-concept *A*. Such a selection defines a metaphor *A is B*. The second column of table 2 displays the percentage of participants of BASICS, who selected the intended metaphor. The third column displays, which unintended concept *B'* was associated most to *A*. The last column shows the students' judgments about usefulness and – in parentheses – plausibility of the explanation given during the EVALUATE session.

Intended metaphor	Recognized by	Most selected not-intended target-concept	Usability (plausibility) 1=A, 6=F

Absolute reference is writing an address on a letter.	47%	Formula (20%)	2.5 (2.3)
Absolute reference is a connection to an outlet.	44 %	Copying (19%)	2.6 (2.8)
Absolute reference is an ID number.	20%	Relative reference (37%)	2.7 (2.6)
A format is a look	44 %	Relative reference (22%)	2.8 (2.9)
Copying a formula is customizing T-shirt production	24%	Formula (29%)	3.0 (2.9)
Relative reference is touching a person	0%	Absolute reference (43%)	3.2 (3.1)
Relative reference is navigation	13%	Copying (40%)	3.4 (3.3)
Displaying the result of a formula is decorating a shop window	23%	Relative reference (29%)	3.9 (3.7)
Displaying the result of a formula is shadow play	12%	Relative reference (31%)	4.5 (4.1)

Table 1. Recognition and evaluation of spreadsheet-related metaphors (BASICS)

None of the metaphors were really convincing to the students – at least at first sight. The best accepted and recognized images were those visualizing absolute reference and format of numbers. But less than half of the students recognized these during the EXPLAIN session. Although most students proved in the USE session that they were able to adopt relative referencing properly, they did not pick the images that were intended to illustrate this concept. Complete failures were metaphors for evaluating formulas. The students neither selected the images as appropriate illustrations nor did they accept the explanation given in the EVALUATE session. Here is an example of an explanation of the unsuccessful metaphor "displaying the result of a formula is shadow play", which got a "D minus"both for usability and plausibility:
"The hands are a formula, the shadow is a value and the cell is a projection screen. The hands produce an image that is displayed on the screen. "

On the other hand the figures seem to show some learning effects: Metaphors like "relative reference is touching a person" (see figure 4) and "copying is customizing T-shirt production" were not selected at first sight during the EXPLAIN session but were accepted later during EVALUATE. Obviously many students accepted these metaphors only in combination with verbal discussions and explanations.

The second activity LOCKS (23 participants) evinced a few very successful metaphors, including these:

- Output data is smoke from a chimney (96%)
- Access control (lock)[1] is locking with a padlock (96%).

[1] In English spreadsheet user interfaces the metaphorical term "lock" is used for access control (blocking write-operations), while in German-speaking cultures the metaphor "barrier" (German: "Sperre") is applied.

- access control (lock) is protection by a bicycle helmet (74%).
- output data is cookies from an oven (74%).
- input data is ingredients for baking (57%).
- A cell is a stage (39%).

The numbers in parentheses are the percentages of those, who chose the metaphor by drawing a line between an image and a concept box. Rather unsuccessful were all metaphors illustrating the idea of a formula:

- A formula is an orange juice squeezer (26%).
- A formula is a toaster (9%).

Both metaphors support the idea that a formula processes input data and delivers output data as a result. This idea was explicated in the given concept map.

In LOCKS-EXPLAIN the students were asked to draw one additional image that illustrates one of the concepts. Most of them (10 persons) tried to visualize access control (lock) by connecting it to real life objects, including these: A PIN number on a cell phone (2), a prison (2), a knight's shield, a closed door with a "No admittance"-sign, access control to a computer by password check, blocking friends in Facebook, a sealed letter, a strong door protecting against burglary.

Another 6 persons created a picture illustrating output data (pizza take away, gases from the exhaust of a car or a motor cycle, nut-cookies from the oven) 3 persons tried to visualize a spreadsheet cell and one person illustrated a formula just by giving a concrete example. Nobody dared to visualize a formula.

5 Educational perspectives – how to use metaphorical images in the classroom

Let me sketch two class room activities (collaborative games) using metaphorical images.

Collaborative puzzle. Three or four people play together. On the table there is a laminated sheet of paper as game board. It depicts an incomplete concept map. Missing labels for relations (arrows) are tagged by rectangles with question marks. Additionally there are circles with question marks close to concepts (boxes) as placeholders for metaphors. The players use a deck of cards with metaphorical images and labels for relations. The cards are shuffled and distributed – face down – to the players. In turn, each player places a card on an appropriate location on the board and covers a question mark. The she or he explains the move. When all question marks are covered, the players compare their result with the solution.

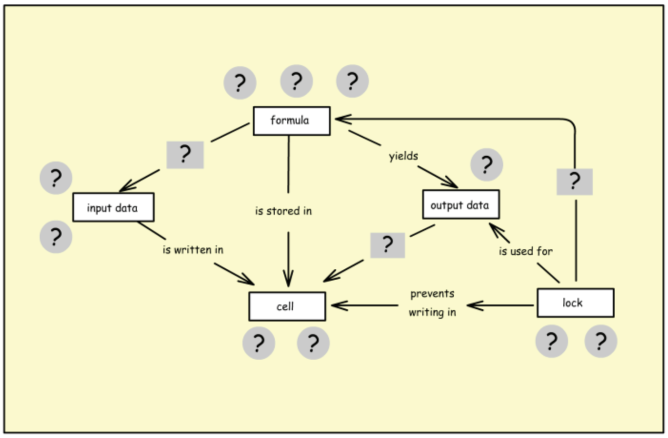


Fig. 5. Game board for a collaborative puzzle.

Concept Rummy. Three or four players share a deck of cards. The deck consists of ten series consisting of four cards each. The first card of a series (concept-card) shows a description of a spreadsheet concept illustrated by a screen shot. The other three cards are corresponding metaphorical images depicting things from different domains. The cards are shuffled and put on a stack, face down. In turn each player takes the top card and tries to place it. Concept-cards can be put immediately on the table. Images must be placed close to a suitable concept-card. Each move has to be explained ("This image corresponds to this concept, because …"). When a player cannot place a card, she or he just keeps it and waits for the next turn.

The idea of these activities is to evoke active elaboration and discussion. This way, implicit semantic facets of concepts are verbalized and explicated. The students use the relevant words and construct meaning. According to Hattie [10] there exists evidence that collaborative activity in small groups (opposed to individual learning) increases the efficiency of learning (effect size d=0.56). In order to avoid a "split-attention effect" [13] and reduce cognitive load the images should be located close to the corresponding concepts. This was not the case in the EXPLAIN sessions but it can easily be done using cards.

Many variations of such games are possible. Even better if students create images and games by themselves. In Wikimedia Commons, teachers and students can find millions of images that are published under Creative Commons Copyright and can be used legally for creating new metaphorical illustrations. But it must be kept in mind that creating media is difficult and takes time, which may not always be available. What is important is to establish a culture of explaining things. Really deep understanding is required when it comes to building unusual connections between objects from knowledge domains which are far away to each other – like an orange juice squeezer and a spreadsheet formula.

6 References

1. Baker, J., Sugden, S.J.: Spreadsheets in Education – The First 25 Years. In: Spreadsheets in Education (eJSiE): Vol. 1: Iss. 1, Article 2. Available at: http://epublications.bond.edu.au/ejsie/vol1/iss1/2. (2003).
2. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cuniff, D., ... & Verno, A.: CSTA K-12 Computer Science Standards, available at http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf (2011).
3. Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I; Empfehlungen der Gesellschaft für Informatik eV. LOG IN Verlag, available at https://www.gi.de/fileadmin/redaktion/empfehlungen/Bildungsstandards_2008.pdf (2008).
4. Chemero, A.: An Outline of a Theory of Affordances. In: ECOLOGICAL PSYCHOLOGY, Lawrence Erlbaum Associates, Inc. 15(2), pp. 181–195 (2003).
5. Brown, P.S., Gould, J.D.: An experimental study of people creating spreadsheets. In: ACM Transactions on Office Information Systems, 3 pp. 258-272 (1987).
6. Powell, S. G., Baker, K. R., & Lawson, B.: A critical review of the literature on spreadsheet errors. *Decision Support Systems*, *46*(1), 128-138 (2008).
7. Meyer, J.H.F., Land, R.: Threshold Concepts and Troublesome Knowledge 1 – Linkages to Ways of Thinking and Practicing. In: *Improving Student Learning – Ten Years On.* C.Rust (Ed.), OCSLD, Oxford (2003).
8. Novak, J. D. & A. J. Cañas, The Theory Underlying Concept Maps and How to Construct Them, Technical Report IHMC CmapTools 2006-01, Florida Institute for Human and Machine Cognition, available at: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf (2006)
9. Scheele, B., Groeben, N.: Dialog-Konsens-Methoden zur Rekonstruktion Subjektiver Theorien: die Heidelberger Struktur-Lege-Methode (SLT), konsensuale Ziel-Mittel-Argumentation und kommunikative Flussdiagramm-Beschreibung von Handlungen. Tübingen: Francke (1988).
10. Hattie, John: Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement, London: Routledge (2009).
11. Lakoff, G., Núñez, R.: The Metaphorical Structure of Mathematics: Sketching Out Cognitive Foundations for a Mind-Based Mathematics. In L. English (Ed.), *Mathematical Reasoning: Analogies, Metaphors, and Images* (pp. 21-89). Hillsdale, NJ: Erlbaum (1997).
12. Lakoff, G., Núñez, R. E.: Where mathematics comes from: How the embodied mind brings mathematics into being. Basic books (2000).
13. Sweller, J.: Visualisation and instructional design. In *Proceedings of the International Workshop on Dynamic Visualizations and Learning*, pp. 1501-1510 (2002).

Teacher Education in Informatics

Eliciting Computing Science Teachers' PCK using the Content Representation Format

Experiences and Future Directions

Erik Barendsen[1], Valentina Dagienė[2], Mara Saeli[3], and Carsten Schulte[4]

[1] Radboud University Nijmegen, Institute for Computing and Information Sciences, and Open Universiteit, The Netherlands, `e.barendsen@cs.ru.nl`
[2] Vilnius University, Faculty of Mathematics and Informatics, Lithuania, `valentina.dagiene@mii.vu.lt`
[3] Milton Keynes, United Kingdom, `marasaeli@yahoo.it`
[4] Freie Universität Berlin, Institut für Informatik, Germany, `schulte@inf.fu-berlin.de`

Abstract. This paper focuses on the Content Representation (CoRe) format as an instrument to elicit teachers' Pedagogical Content Knowledge (PCK) in the area of computing education. We discuss several methodological variations in the use of the instrument, originating from previous studies by the authors in which CoRe was applied. As a new example, we report on a recent case study, portraying the PCK of Lithuanian teachers with respect to programming. The variations in the teachers' articulated PCK appear to reflect their respective backgrounds, as well as various known beliefs on the nature and purpose of programming education. We reflect upon the effects of the methodological variations in the above case studies. Group sessions tend to result in more substantial PCK data than individual elicitations. However, the individual data of the Lithuanian case study, obtained in very limited time, appeared surprisingly rich. Our observations give rise to new research questions.

1 Introduction

The concept of *pedagogical content knowledge* (PCK) was introduced by Shulman (1986) as "the knowledge of teachers to help others learn", including "the ways of representing and formulating the subject that makes it comprehensible to others". PCK is considered to be topic-specific and thus closely related to teachers' *content knowledge* (CK). The exact nature of the relation between PCK, CK and general pedagogical knowledge is subject to debate (*PCK Summit*, 2012).

We adopt the model by Magnusson, Krajcik, and Borko (1999) and distinguish four aspects of PCK with respect to a certain topic: (a) knowledge about learning goals and objectives connected to the topic, (b) knowledge about students' understanding of the topic, (c) knowledge about instructional strategies for teaching the topic, and (d) knowledge about ways to assess students' understanding of the topic.

Insight in teachers' PCK of a given topic, as well as the way such PCK develops, is interesting in itself but also crucial for teacher education. However, eliciting PCK from teachers is difficult. Indeed, PCK is personal and tends to be tacit, and underlying reasons for applying a certain instructional strategy are seldom explicitly articulated or shared between colleagues (Loughran, Mulhall, & Berry, 2004). Several methods have been proposed, including interviews (e.g., Henze, Van Driel, & Verloop, 2008), Pedagogical experience Repertoires, PaP-eRs, (Loughran et al., 2004), classroom observations (e.g., Gess-Newsome et al., 2011) and reflective journals (e.g., Wongsopawiro, 2012).

Thus far, PCK has mainly been investigated in the context of science education. Only few attempts have been made to capture PCK of teachers of computing subjects such as informatics or IT. However, the PCK approach has shown to be fruitful to explore professional knowledge of informatics teachers (Saeli, 2012).

In this paper we will discuss various ways to apply the Content Representation (CoRe) format introduced by Loughran et al. (2004). After introducing the CoRe instrument, we will summarize the authors' experiences from previous activities. Then we focus on a new case study, investigating the PCK of Lithuanian teachers on the subject of programming. Since this paper focuses on methodological issues, we present only part of the results. We conclude with a discussion of the applied techniques, comparing the effects of the methodological variations, and an outlook on future research.

2 The Content Representation Format

2.1 The Instrument

The Content Representation (CoRe) format is an instrument to investigate teachers' PCK of a specific topic (Loughran et al., 2004). It captures the key ideas connected to the topic, and sets out the teachers' knowledge about each idea through 8 questions, see Table 1. The questions cover the four PCK aspects by Magnusson et al. (1999) mentioned in the previous section: questions 1, 2 and 3 refer to aspect (a), 4 and 5 to aspect (b), 6 and 7 to aspect (c), and 8 to aspect (d). Loughran et al. (2004) originally introduced the CoRe format as an interview tool. In the following subsections we will describle the application of the instrument in various settings in the context of computing education.

2.2 Example: Focus Groups on Programming

Focus groups as a means to collect PCK data was applied by Saeli (2012). Because the goal of the study was to portray a common ('shared') PCK of Programming, only experienced teachers with Computer Science background were invited to take part to the research. Six different focus groups were organized in 4 European countries: Lithuania, Italy, Belgium and the Netherlands. A total of 31 teachers took part to the focus groups, varying from 4 to 7 teachers. Teachers were told the sessions would concern the teaching of programming, and consisted of two parts for a total of up to 2 hours. Each session was video-taped and written notes collected. During each part teachers first worked individually and then shared their answers with their peers and discussed their views/opinions in group.

Table 1. CoRe questions

0. What are important ideas/concepts ('Big Ideas') concerning this topic?

For each 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. Which knowledge about students' thinking influences your teaching of 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?

During the first part of the focus group session, teachers were asked to list what in their opinion are the core topics of programming. After discussing their results, the group was asked to choose two or three of these topics. At this point, teachers were presented a CoRe form and were asked to answer the questions using their experience. During these sessions teachers had the chance to reflect on their own practice and become acquainted with others' approaches. Seven Big Ideas were discussed in the different sessions, namely: control structures, data structures, arrays, problem-solving skills, decomposition, parameters and algorithms. The results of this study were compared with the literature. While some of the results turned out to confirm what the literature already suggested as possible teaching approaches, others appeared to be new.

It appeared that teachers were motivated to take part in the study as it was an opportunity to share views/ideas with peers. In most cases, the subject was taught by only one teacher per school, limiting the informal contact between teachers. From the research point of view, the focus-group approach, in contrast with individual interviews, gives teachers the opportunity to deepen their personal reflection as a result of group discussion. Organizing the focus groups turned out to be tedious due to varying teaching schedules, but teachers seemed to recollect more of their own experiences and view while discussing with their peers. On the other side, the collected data gave little insight into individual teachers' PCK.

A similar focus group setting has been used in the context of an in-service teacher education programme in Nijmegen (the Netherlands). The participants were divided into groups of 4–5 teachers. The big ideas obtained from the international study were presented to the groups, and each group chose one idea. The group members discussed the idea according to the CoRe questions. One of the group members was asked to summarize the discussion on a paper with a CoRe form. Two of the authors monitored the group activity, which turned out to be very lively. The discussions were forced to end after one hour. The participants reported that they enjoyed the session since this had been one of the few occassions to discuss teaching in an in-depth way, without being drawn to general pedagogical matters. The completed sheets showed far less information than the researchers had observed during the session. This suggests that additional data collection (such as audio recordings) is necessary in the case of groupwise elicitation, complementing the written summaries.

2.3 Example: Electronic Collection of PCK data

CoRes have also been used to investigate Dutch teachers' PCK, using the results described in the previous example as standard for comparison (Saeli, Perrenet, Jochems, & Zwaneveld, 2012). Aiming at reaching as many teachers as possible, the CoRe instrument has been adapted to be used as an online questionnaire.

Teachers were first prompted with a closed question concerning the aforementioned core concepts of programming obtained from the focus groups. The teachers were asked to choose one. The following part of the questionnaire intended to get an idea about the teachers' content knowledge. The remainder of the questionnaire consisted of eight open ended questions based on the CoRe format. These were presented with the same wording as the questions asked during the focus group sessions.

A total of 92 teachers took part to the questionnaire, with 69 valid responses. Teachers were categorized according to their disciplinary backgrounds, as in the Netherlands CS teachers have mostly a degree in a subject other than Computer Science. Dutch teachers' PCK was compared with respect to both CK and to content representation.

The results with respect to CK were low to average. The results on the content representation component showed poor answers without much elaboration, especially on the question relative to extracurricular knowledge and teaching methods. This could indicate that teachers need more support by, e.g., teaching materials and examples. This result was also reflected in an international study (Saeli, Perrenet, Jochems, & Zwaneveld, 2011). The answers were much richer on issues 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 (Perrenet, Van Diepen, & Zwaneveld, 2011) in the process of revising the whole subject.

The use of CoRe as an electronic instrument gives the chance to gather more participants, as teachers can comfortably answer the questions at their place of choice and at their own pace. Also, teachers were given the opportunity to complete the questionnaire in installments, to allow maximum flexibility. However, some answers might have been more complete if given in an interview.

2.4 Example: CoRe in Pre-service Teacher Education

Another use of the instrument is as a part of pre-service teacher education. To foster reflection, group discussion, and to get feedback on the students' learning progress (formative assessment). Within the Bachelor degree of CS teacher education at Freie Universität Berlin, teacher students design, implement, teach and refine a teaching unit. The teaching is done by inviting high school classes to the university, into the teaching and learning lab. The instrument is used immediately before a teaching experience, afterwards, and after a cycle of two or three repetitions of teaching and refining the teaching module. In order to assess and foster a discussion on the most central topic of the teaching unit, the participants formulate the topic (Big Idea) for themselves (question 0). Results of group discussions are also collected as CoRes and presented to the whole course.

Buchholz, Saeli, and Schulte (2013) use the result to monitor progress. The questions are divided into a teaching nexus (questions 1, 2, and 3) and a learning nexus (4, 5, 8). Questions 6 and 7 were not assigned to be either on teaching or learning side. The distinction between teaching nexus and learning nexus draws upon work from Trigwell, Prosser, and Ginns (2005), who distinguish a teacher centered, and a student centered view in teacher professionalism, together with the idea that students progress from a teacher centered to a student centered viewpoint.

Results in the Buchholz et al. (2013) study show a greater progression in the questions associated to teacher nexus than in the questions related to the learning nexus. This might be due to the focus of the university course, in which the materials were updated during the cycles, but not much reflected on specific learning issues of high school students. In the first three questions, the answers mostly took a teacher-oriented point of view. Also in the learning nexus the students' initial teacher-centered focus is visible. E.g., only very few answers to Question 4 point at specific learning obstacles.

It appears that this use of the CoRe instrument integrates well into the pre-service teacher education. It serves as a formative assessment of the learning progression of the students (on a group and an individual level). We have seen that it can be used with rather strict time constraints; so it does not take much more than around 15 minutes. However, repeatedly answering the same questions tends to get tedious for the participants. In addition, if participants define the topic (question 0) on their own it becomes somewhat unclear if the answers are really belonging to the same topic.

3 Case Study: Portraying Lithuanian Teachers' General PCK on Programming

3.1 Context of the study

For many years, Informatics has been a subject in Lithuanian secondary education. It was a compulsory subject from 1986 to 2004, and has been replaced by Information Technologies (IT) in 2005. IT is a compulsory subject in grades 5, 6, 9 and 10. There are optional modules on informatics (viz., programming) in grades 9–12. In small schools, however, it can be hard to get enough students to justify teaching these optional modules.

The participants in this study were attending a workshop for CS and IT teachers from rural areas of Lithuania. All teachers had at least 5 years of teaching experience.

Investigating these teachers' PCK is interesting because the change from informatics to IT has had substantial impact on teacher qualifications. The older informatics teachers (educated before 2005) have a strong background in informatics as well in mathematics (as the subjects were commonly combined in education programmes). The younger IT teachers do not necessarily have a background in computer science. Moreover, the focus of in-service training has shifted from development of technical skills to the pedagogical aspects of integrating IT into education. Substantial attention is paid to develop computational thinking and new skills to be required in the information age (e.g, handling large amounts of information, algorithmic thinking, structured thinking, and problem solving).

3.2 Aim of the Study

The study aims at exploring the teachers' general PCK with respect to programming. In this paper we focus on the first aspect of PCK, knowledge about learning objectives. We are also interested in the relation between this PCK and the teaching experience of the participants. On a methodological level, we aim at exploring the nature and quantity of the information collected using a simple CoRe questionnaire without any content specific guidance.

3.3 Method

The participants were given a questionnaire with the 8 CoRe questions in Lithuanian, to be filled in on paper. No 'big ideas' were given or asked for: 'programming' was stated as subject. The actual questions spanned 1.5 sheet of paper, so the space for writing down the answers was limited. The respondents were given 30 minutes to complete the questionnaire. We also asked the participants to fill in their teaching experience (in years) and whether or not they were teaching programming as a subject.

The completed forms were scanned and translated into English to prepare them for analysis. The translation was verified by two native Lithuanian speakers with computer science expertise, adding explanations to implicit references to local teaching customs where necessary.

The teachers' answers were first subjected to an exploratory inductive qualitative analysis, adding descriptive codes to the respective answers. The initial coding was reviewed by the authors and codes were combined until a comprehensive and compact coding scheme was reached. This final scheme was then applied to the data. The resulting coding was used for a frequency analysis.

The participants were divided into four groups according to their experience: less than 10 years (group A), 10–14 years (group B), 15–19 years (group C), and 20 years or more (group D). Using the coding and grouping, we looked for patterns within groups and differences between groups. The frequencies and patterns were used to spot interesting issues for a further (in-depth) review of the answers.

3.4 Results

A total number of 34 participants completed the questionnaire. All completed forms were considered suitable for translation and analysis. On average the teachers had more than 10 years of experience. The experience group sizes were as follows. Group A: 10 teachers (4 of which were teaching programming), Group B: 11 (10), Group C: 7 (6), Group D: 6 (5).

In this more methodologically oriented paper we discuss only questions 1 and 2. The full results will be included in a forthcoming article.

Question 1: What do you intend the students to learn? Statements concerning 'programs' and 'algorithms' were dominant. We distinguish some levels inspired by Bloom's (1956) hierarchy. The codes are displayed in Table 2.

Table 2. Codes for Question 1

code	*description*
programs-know	simple reference to programming languages or constructions
programs-understand	same, with 'understanding' as explicit goal
programs-apply	apply programming constructions to produce elementary, simple pieces of code
programs-analyze	analyze programs
programs-synthesize	create programs for a complete task
algorithms-know	simple reference to algorithms
algorithms-understand	same, with 'understanding' as explicit goal
algorithms-synthesize	creating algorithms, algorithmization (applying and synthesizing could not be distinguished effectively based on the short answers)
problem solving	explicit reference to solving problems or tasks
logical thinking	literal reference; plain 'logic' was coded in category 'other'
analytical thinking	literal reference
attitude	aspects related to students' attitudes, such as critical thinking, persistance, etc.
other	other learning goals or topics

The frequencies can be found in Table 3. We mention the total number of occurrences. While we are aware of the small sample sizes, we also display the relative frequencies within each group (i.e., number of occurrences divided by group size), in order to assess the global distribution of the codes. The Bloom level 'evaluate' was not found in the answers. We summarize the findings of the frequency analysis and subsequent qualitative analysis, illustrated by key examples taken from the data.

To role of programming to solve problems was recognized throughout. Experienced teachers tend to stress higher order skills.

"To analyse a task, to create an algorithm to solve a task [. . .]".

Beginning teachers focus more on simple applications.

"To create the very elementary programs, to understand the principles of programming."

Attitude development was mentioned especially by experienced teachers.

"patience [. . .] accuracy"
"motivation to finish the work; self-assessment;"

Moreover, the responses of experienced teachers contained more skills and a greater variety.

Table 3. Frequencies for Question 1

code	*total*	*relative A*	*relative B*	*relative C*	*relative D*
programs-know	3	0.1	0.1	0.1	0.0
programs-understand	4	0.1	0.0	0.4	0.0
programs-apply	5	0.3	0.2	0.0	0.0
programs-analyze	1	0.0	0.0	0.0	0.2
programs-synthesize	6	0.1	0.2	0.3	0.2
algorithms-know	1	0.1	0.0	0.0	0.0
algorithms-understand	2	0.1	0.0	0.1	0.0
algorithms-synthesize	11	0.0	0.5	0.6	0.2
problem solving	10	0.2	0.3	0.4	0.3
logical thinking	9	0.2	0.4	0.1	0.3
analytical thinking	2	0.0	0.0	0.1	0.2
attitude	11	0.0	0.2	0.7	0.7
other	14	0.1	0.5	0.6	0.7

"Ability to structure the task (problem); foresee the sequence of actions; define possible data/result types. Learn to analyze the created program, looking for mistakes."

The teachers take a general point of view and hardly mention any technical details. Only once some programming concepts were mentioned (*"loop, array, record function"*), whereas the responses contain plenty of process issues (*"algorithmization"*).

Question 2: Why is it important for the students to learn this? We used the codes in Table 4 to analyze the answers. The observed frequencies are displayed in Table 5. Again, respondents with more experience provided more substantial and more varied answers.

Table 4. Codes for Question 2

code	*description*
helps to develop structured thinking	logical or analytical thinking
helps attitude development	critical thinking, confidence, positive attitude towards programming, etc.
develops problem solving skills	–
prepares for future studies	without explicit mentioning of IT
prepares for future IT studies	–
prepares for future profession	without explicit mentioning of IT
prepares for future IT profession	–
prepares for future personal life	for applications in real-life situations
develops awareness	responses related to views on IT and programming
other	–

Table 5. Frequencies for Question 2

code	*total*	*relative A*	*relative B*	*relative C*	*relative D*
helps to develop structured thinking	14	0.5	0.0	0.6	0.8
helps attitude development	6	0.2	0.1	0.4	0.0
develops problem solving skills	1	0.0	0.1	0.0	0.0
prepares for future studies	1	0.0	0.1	0.0	0.0
prepares for future IT studies	6	0.1	0.3	0.0	0.3
prepares for future profession	3	0.0	0.1	0.3	0.0
prepares for future IT profession	9	0.1	0.5	0.0	0.3
prepares for future personal life	3	0.0	0.2	0.1	0.0
develops awareness	4	0.3	0.0	0.1	0.0
other	11	0.2	0.2	0.3	0.8

Many teachers stressed the role of programming to develop thinking skills. It is interesting that logical thinking was considered as a necessary *learning goal* or prerequisite in the context of programming by beginning teachers (Question 1), whereas more experienced teachers regard it more as a *result* of learning to program (Question 2).

References to preparations for future activities of the students (22 occurrences in total) were mostly made by teachers in group B (13 occurrences, relative total frequency 1.2). Relevance for a professional career is dominant.

"Professions where coding is needed are becoming more popular. Their [students] choice is based on enrollment, further studies, and demand of IT specialists."

3.5 Discussion

Part of the apparent differences between beginning and experienced teachers, such as the varying roles of 'logical thinking', are likely to be reflections of their respective backgrounds with informatics or IT teacher education. The results show several interesting differences, e.g., with respect to the role of attitudes and the level of programming tasks, which are worthwhile exploring further.

The various opinions concerning the relationship between programming and problem solving correspond to well-established positions in the scientific community. For example, Feurzeig, Papert, and Lawler (2011) argue that transfer of problem solving skills, including rigourous thinking, can be a reason to introduce programming in schools. Other authors stress the importance of problem solving skills and creativity in the process of programming (e.g., Romeike, 2008).

4 General Conclusion and Discussion

In this paper, we have examined several variants in using CoRe, an instrument consisting of eight questions to uncover and document a teachers' pedagogical content knowledge. The variants and their effects are summarized in Table 6.

Table 6. CoRe usage comparison

aspect	*Focus groups*	*Electronic*	*Teacher Education*	*Lithuanian case study*
data	group	individual	individual and group	individual
scope	given topic (programming) with participants defining Big ideas	given topic and given Big idea (loop within programming	given topic without big ideas (programming)	
medium	verbal and written	written (electronic)	verbal and written	written (pen & paper)
context	reflection in context of a topic/big idea	reflection in context of a topic/big idea	reflection with focus on one specific teaching experience	reflection in context of a topic/big idea
time	1-2 hours	individual	15 min to 60 min	15-30 min
intention	individual diagnostics & reflection on practice with peer discussion	research, characterizing PCK	individual diagnostics & reflection on practice with peer discussion	research, characterizing PCK

Taking the general experience into account that instant, individual, unguided usage of the CoRe instrument as a questionnaire tends to give poor data (e.g., Loughran et al., 2004), the data obtained from the Lithuanian case study appeared surprisingly rich. The limitations in time and space make the results even more remarkable. We intend to explore the other PCK aspects using the teachers' collected data and elaborate on the findings in a future paper.

Nevertheless, data obtained from guided group sessions results in more substantial and more varied data, as the focus group sessions show. Our examples show, however, that filling in a CoRe form only partially reflects the repertoire of the participants. Therefore, recording and analyzing the group activity is necessary. Combining individual work an groupwise discussion seems a fruitful way to collect both individual data and peer-elicited elaboration.

The question remains which aspects of PCK are visible in the Lithuanian study, given the above limitations and the fact that the data lacks conceptual technical details. As the starting point was 'programming' in general, in contrast with elicited or given conceptual 'big ideas', it seems likely that the CoRe questions also elicited, at least partly, the teachers' general curricular beliefs (Van Driel, Bulte, & Verloop, 2008). This connection is worthwhile exploring.

At least in the cases where group activities were involved, there are clear signs that the CoRe instrument stimulated reflection by teachers on their ideas and practice, which in turn will help developing their personal knowledge and thus their PCK (Clarke & Hollingsworth, 2002; Wongsopawiro, 2012). From this perspective, it is useful in teacher education.

Acknowledgements. This research was supported by the Baltic-German University Liaison Office through the German Academic Exchange Service (DAAD) with funds from the Foreign Office of the Federal Republic of Germany. The authors are endebted to Dr. Tatjana Jevsikova and Malte Buchholz for their help in processing the questionnaires and the data.

References

Bloom, B. S. (1956). *Taxonomy of educational objectives: The classification of educational goals*. Harlow, Essex, England: Longman Group.

Buchholz, M., Saeli, M., & Schulte, C. (2013). Pck and reflection in computer science teacher education. In *Proceedings of the 8th workshop in primary and secondary computing education.* New York, NY, USA: ACM to appear.

Clarke, D., & Hollingsworth, H. (2002). Elaborating a model of teacher professional growth. *Teaching and teacher education*, *18*(8), 947–967.

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, *19*(5), 487–501.

Gess-Newsome, J., Cardenas, S., Austin, B., Carlson, J., Gardner, A., Stuhlsatz, M., … Wilson, C. (2011). *Impact of educative materials and transformative professional development on teachers' PCK, practice, and student achievement.* (Paper set presented at the Annual Meeting of the National Association for Research in Science Teaching, Orlando, FL, April 6)

Henze, I., Van Driel, J. H., & Verloop, N. (2008). Development of experienced science teachers pedagogical content knowledge of models of the solar system and the universe. *International Journal of Science Education*, *30*(10), 1321–1342.

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. G. Lederman (Eds.), *Examining pedagogical content knowledge* (pp. 95–132). Dordrecht: Kluwer.

PCK Summit. (2012). Working conference held in Colorado Springs on October 20–26, 2012. (http://pcksummit.bscs.org)

Perrenet, J., Van Diepen, N., & Zwaneveld, B. (2011). Which way with informatics in high schools in the Netherlands? The Dutch dilemma. *Informatics in Education*, *10*(1), 123–148.

Romeike, R. (2008). What's my challenge? The forgotten part of problem solving in computer science education. In *Informatics education – supporting computational thinking* (pp. 122–133). Springer.

Saeli, M. (2012). *Teaching programming for secondary school: a pedagogical content knowledge based approach*. Unpublished doctoral dissertation, Eindhoven University of Technology, The Netherlands.

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

Saeli, M., Perrenet, J., Jochems, W. M. G., & Zwaneveld, B. (2012). Programming: Teachers and pedagogical content knowledge in The Netherlands. *Informatics in Education*, *11*, 81–114.

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

Trigwell, K., Prosser, M., & Ginns, P. (2005, November). Phenomenographic pedagogy and a revised approaches to teaching inventory. *Higher Education Research and Development*, *24*(4), 349–360.

Van Driel, J. H., Bulte, A. M. W., & Verloop, N. (2008). Using the curriculum emphasis concept to investigate teachers' curricular beliefs in the context of educational reform. *Journal of Curriculum Studies*, *40*(1), 107–122.

Wongsopawiro, D. (2012). *Examining science teachers' pedagogical content knowledge in the context of a professional development program.* Unpublished doctoral dissertation, Leiden University, The Netherlands.

Curriculum Issues

Towards Combining Conceptual Lesson Patterns with Austrian K12 Computer Science Standard Curriculum in the Context of Pedagogical Content Knowledge

Bernhard Standl and Wilfried Grossmann

University of Vienna
Faculty of Computer Science
Währinger Strae 29, 1090 Vienna
{bernhard.standl,wilfried.grossmann}@univie.ac.at

Abstract. The teacher's knowledge of teaching can be allocated at the intersection pedagogy and content knowledge and was suggested by Shulman as *Pedagogical Content Knowledge*. Such knowledge supports motivation of students for the subject and teaching and learning in an inspiring way. Beyond different approaches for defining teaching content for computer science lessons at K12, we chose the Austrian standards for K12 of the *DigiKomp* curriculum. Based on successful lesson scenarios for computer science, which were described as a pattern network, we combine these patterns with the Austrian standards and propose a structure how the patterns can be used for application of pedagogical-content knowledge in Computer Science teaching. It can be seen as a first step towards way to describe lessons in a modular representation.

Keywords: conceptual patterns, pedagogical content knowledge, digital competencies, lesson planning

1 Introduction

In this paper we introduce an approach how student-centered conceptual patterns as pedagogical value base can be combined with computer science content given by the Austrian curriculum for digital competencies as described in [1] and *http://digikomp.at*. Considering pedagogy and content as parts of the teacher's knowledge in the context of Shulman's [2] pedagogical-content knowledge (PCK), we use our pedagogical patterns and as content the Austrian K12 computer science standard curriculum. We see that both, the patterns and the competencies, can have a positive potential for the teachers individual lesson planning if the lesson scenarios are described as abstract patterns in combination with the given standards. In this discussion paper we first introduce the pattern approach where the pedagogical theory of the student-centered approach as introduced by Carl Rogers in [3] is captured uniformly as a network of 24 patterns. In the second section we describe the Austrian standards for digital competencies as computer science related content part. In the third section we show how pedagogy and content can be combined for practice in the context of pedagogical content knowledge. In the last section we give an example how an instantiation in practice could be carried out for the topic ”Algorithms and Introduction into Scratch”.

2 Patterns

If learning scenarios are described uniformly as abstract patterns, relations, workflows and interdependent entities, a view beyond detailed lessons is possible. One advantage of such a representation is, that during the application it is more practicable for the teacher if the scenarios are not given in detail but rather described as a pattern framework to provide the freedom for an individual application. The structure of the pattern approach as applied in this work is based on the pattern language as developed by Derntl in [4] respectively C. Alexander's approach in [5, 6] and was modified for computer science K12 education in [7]. We suggest to use this pattern network, which contains 24 patterns from general ones as 'Project Style' to specific ones as 'Group Work'. The characteristic of the network is, that the patterns unfold their strengths if there are combined with each other.

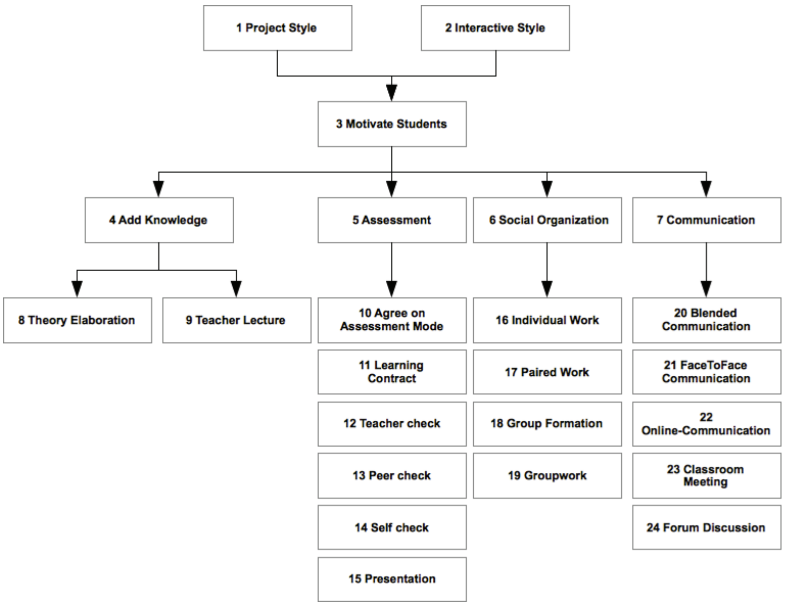
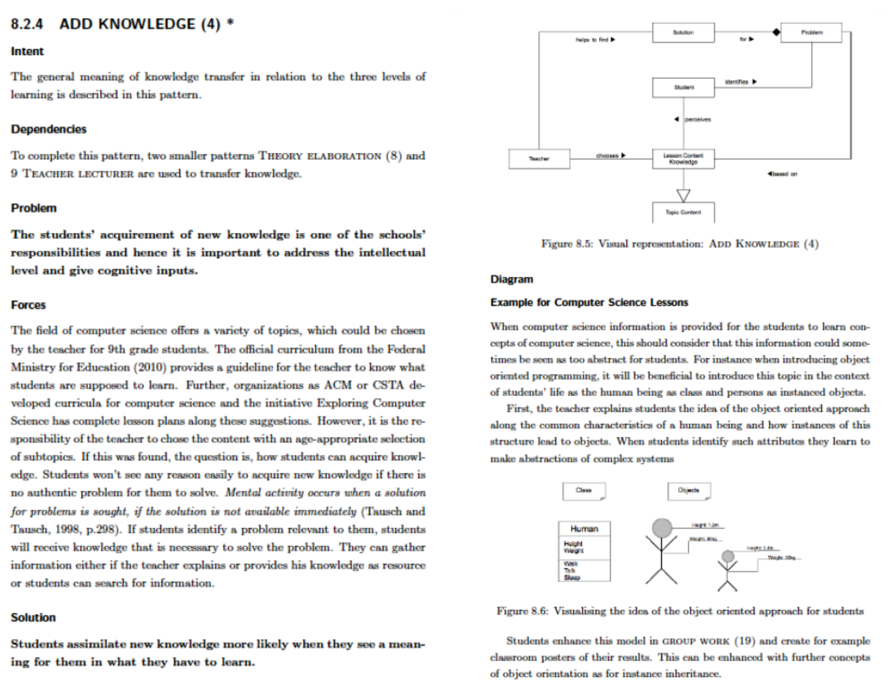


Fig. 1. The pattern network includes 24 patterns.

As it is depicted in the pattern network above, the relation from one single pattern to other patterns is actually the most important part of the pattern system. Alexander explicitly mentions that solely adding patterns to each other without interrelating them does not lead to a meaningful pattern language. This means, that if the teacher plans to carry out 'Group Work' in the context of a 'Project Style' lesson he has to consider both patterns and possible related further patterns as for example 'Social organization'. Even if our patterns describe rather pedagogical issues of lessons, they were developed in the context of computer science lessons at secondary level and hence are especially valid in such settings.


Solution
Problem
helps to find ▶
for ▶
Student
identifies ▶
◀ perceives
Teacher
chooses ▶
Lesson Content Knowledge
◀based on
Topic Content
Class
Objects
Human
Height
Weight
Walk
Talk
Sleep

8.2.4 ADD KNOWLEDGE (4) *

Intent

The general meaning of knowledge transfer in relation to the three levels of learning is described in this pattern.

Dependencies

To complete this pattern, two smaller patterns THEORY ELABORATION (8) and 9 TEACHER LECTURER are used to transfer knowledge.

Problem

The students' acquirement of new knowledge is one of the schools' responsibilities and hence it is important to address the intellectual level and give cognitive inputs.

Forces

The field of computer science offers a variety of topics, which could be chosen by the teacher for 9th grade students. The official curriculum from the Federal Ministry for Education (2010) provides a guideline for the teacher to know what students are supposed to learn. Further, organizations as ACM or CSTA developed curricula for computer science and the initiative Exploring Computer Science has complete lesson plans along these suggestions. However, it is the responsibility of the teacher to chose the content with an age-appropriate selection of subtopics. If this was found, the question is, how students can acquire knowledge. Students won't see any reason easily to acquire new knowledge if there is no authentic problem for them to solve. *Mental activity occurs when a solution for problems is sought, if the solution is not available immediately* (Tausch and Tausch, 1998, p.298). If students identify a problem relevant to them, students will receive knowledge that is necessary to solve the problem. They can gather information either if the teacher explains or provides his knowledge as resource or students can search for information.

Solution

Students assimilate new knowledge more likely when they see a meaning for them in what they have to learn.

Figure 8.5: Visual representation: ADD KNOWLEDGE (4)

Diagram

Example for Computer Science Lessons

When computer science information is provided for the students to learn concepts of computer science, this should consider that this information could sometimes be seen as too abstract for students. For instance when introducing object oriented programming, it will be beneficial to introduce this topic in the context of students' life as the human being as class and persons as instanced objects.

First, the teacher explains students the idea of the object oriented approach along the common characteristics of a human being and how instances of this structure lead to objects. When students identify such attributes they learn to make abstractions of complex systems

Figure 8.6: Visualising the idea of the object oriented approach for students

Students enhance this model in GROUP WORK (19) and create for example classroom posters of their results. This can be enhanced with further concepts of object orientation as for instance inheritance.

Fig. 2. Example of a pattern: Add Knowledge

Considering the example pattern above, patterns explain models for a solution of a recurring problem in practice in order to make the solution available for an individual application and reuse in similar situations. Each pattern is structured in the same way and includes the parts Name, Intent, Dependencies, Problem, Forces, Solution, Diagram, and Example in Practice. The diagram is visualized as UML diagram where we use *Classes* and *Associations* as structural elements and associations visually to describe the characteristics of the pattern with possible connections to other patterns.

3 Austrian Standard for Digital Competencies

Computer science education in Austria at the secondary level is based on an unified approach for competencies at the lower secondary level (11-14 years) and upper secondary level (15-18 years). Details may be found in [1] and *http://digikomp.at*. The model combines a content dimension with an activity dimension. Content of the reference model is grouped into four main topics: Information Technology, Humans and Society, Information Systems, Applied Computer Science, and Computer Science Concepts. At the lower secondary level main emphasis is on practice and ICT and skills, whereas at the secondary level the approach focuses more on computer science concepts. This distinction is of major importance for Applied Computer Science and Computer Science Concepts. The activity dimension is oriented on Bloom's taxonomy and covers lower cognitive skills like Knowing, Applying and Reflecting as well as Higher cognitive skills like Understanding, Designing and Evaluating.

4 PCK as Pattern-Competencies Knowledge

Shulman's definition in [2] of the specific teacher's knowledge as 'pedagogical content knowledge' was, to place such knowledge at the intersection between subject knowledge of experts and pedagogical knowledge of learning and teaching sciences. He described the teacher's knowledge as the ability to make subject content for students available in a way that students can easier understand and learn it. In more detail PCK considers a number of knowledge components, which have been elaborated theoretically and empirically for teaching science in [8] [9]. Most important are the following components: orientation towards teaching, subject and curriculum knowledge, knowledge of students understanding, assessment knowledge, and instructional strategies. Based on this ideas we think that for teaching Computer Science it is important to describe both parts - pedagogy and content in a general way to give the teacher the required flexibility for lesson planning. Therefore we use the previously described pattern network and combine these patterns with the K12 standard competence curriculum in Austria. The relation of the patterns and PCK can be briefly summarized as follows: The knowledge component "orientation towards teaching" is the background for all teaching. Patterns 1, 2, and 3 help to transform knowledge about students understanding into practice. Patterns 4, 8, and 9 are essential for application of knowledge about the curriculum. Patterns 10 - 15 list different alternatives for assessment, and patterns 6, 7, and 16 - 23 support the development of instructional strategies. Using the patterns, the teacher can develop his individual pedagogical content knowledge by combining our pedagogical patterns for student-centered teaching with the contents as defined by standard curriculum. Hence, our approach can be seen as a base for teachers to support the development of the specific knowledge as Shulman intended it. To create an individual lesson while going conform with the pedagogical approach of student-centered lessons and the topics as given by the curriculum the teacher uses the modularity of each system, the patterns and the curriculum. In the next section we describe, how a teacher can instantiate such scenario in practice for the introduction into programming at 9th grade.

5 Example Application

The following example shows how the combination of patterns and the standards can work in application of PCK in teaching Computer Science.

1. **Learning Objectives.** First it has to be decided what students should learn. In this example we want for example introduce basic principles of algorithms and programming.

2. **Context in Curriculum.** Next, the corresponding section in the digital competencies curriculum has to be chosen.
   4 Practical Computer Science
   4.2 Algorithms, Data and Programming
   In our example the section in practical computer science is appropriate for the intentions.
3. **Example.** On *http://digikomp.at* examples for the curriculum are provided where a adequate can be chosen from. For our intentions we found an example for *Introduction into Scratch*. This can build the framework for the indented lesson plan.
4. **Patterns.** Next the pedagogical approach for teaching the topic has to be chosen where the patterns become relevant. Considering the pattern network as proposed above, a pattern language as to be built for introducing basic concepts of algorithms and programming language. We suggest to start with a teacher focused phase with emphasis on knowledge transfer, followed by a project phase where students can learn self-directed the achieved knowledge. The diagram bellow shows how content can be combined with an appropriate combination of the patterns.

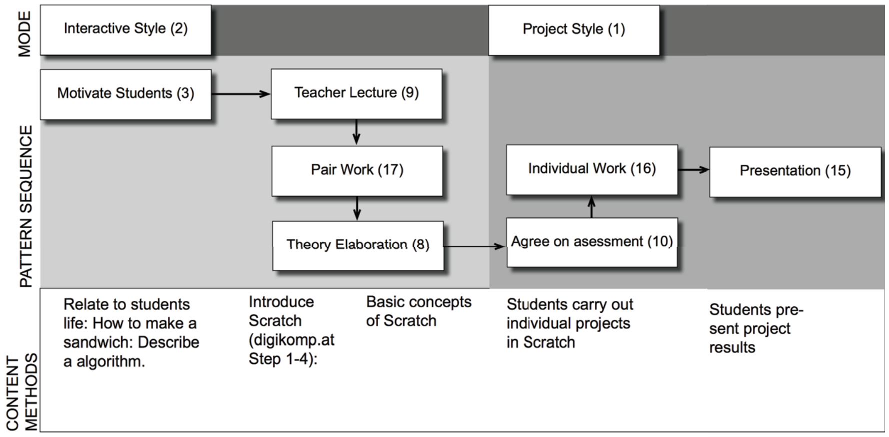

MODE
PATTERN SEQUENCE
CONTENT METHODS
Interactive Style (2)
Project Style (1)
Motivate Students (3)
Teacher Lecture (9)
Pair Work (17)
Theory Elaboration (8)
Agree on asessment (10)
Individual Work (16)
Presentation (15)
Relate to students life: How to make a sandwich: Describe a algorithm.
Introduce Scratch (digikomp.at Step 1-4):
Basic concepts of Scratch
Students carry out individual projects in Scratch
Students present project results

Fig. 3. Representation of the patterns and content.

5. **Methods.** We see further methodical concepts as further layer to be considered beside pedagogy and content. As the examples on *http://digikomp.at* provide methodical considerations for the examples, this layer can be extracted right from there.
6. **Practice.** In order to document the planned approach in practice, a lesson plan can be created in order to capture the intended plan.

This example demonstrates how our proposed approach can be carried out in practice in the context of pedagogical content knowledge with patterns as pedagogical value base and content as provided by the national standards for digital competences.

6 Conclusions and Further Work

In this discussion paper we presented our approach of combining a pedagogical pattern network with computer science content provided by the Austrian standards for K12 computer science education. The pattern network was developed for computer science lessons and provides a value base for student-centered computer science lessons. We described how these patterns can be seen in the context of pedagogical content knowledge as pedagogical part. The approach was applied in classroom and showed that pattern are useful to increase reflection-in-action and reflection-on-action. Moreover a preliminary test showed that efficacy of teaching is increased [7]. According to [8] this are important advantages of PCK. Future work includes the combination of the patterns further parts of the Austrian computer science standards with examples for practice. As next step this method has to be tested in practice and researched with case studies not only aimed at proving the practicability and impact of our approach but also in the optimization of the pedagogical pattern network. In order to make our approach more reusable we plan to further develop a web application for an individual lesson planning and export to a learning plattform as described in [10].

References

1. Micheuz, P.: From Digital Competence to Informatics Education. Structuring a WideField. In Micheuz, P., Brandhofer, A., Sabitzer, B., Ebner, M., eds.: Digitale Schule in Österreich. Österreichische Computer Gesellschaft (2013) 372–380
2. Shulman, L.S.: Knowledge and Teaching: Foundations of the New Reform. Harvard Educational Review **57** (1987) 1–22
3. Rogers, C.: Freedom to Learn for the 80's. Charles E. Merrill Publishing Company, Columbus, Ohio (1983)
4. Derntl, M.: Patterns for person centered e-learning. PhD thesis, University of Vienna (2006)
5. Alexander, C.: The Timeless Way of Building. University Press, Oxford (1979)
6. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. University Press, Oxford (1977)
7. Standl, B.: Conceptual Modeling and Innovative Implementation of Person-centered Computer Science Education at Secondary School Level. PhD thesis, University of Vienna (2014)
8. Park, S., Oliver, J.S.: Revisiting the Conceptualisation of Pedagogical Content Knowledge (PCK): PCK as a Conceptual Tool to Understand Teachers as Professionals. Research in Science Education **38**(3) (June 2007) 261–284
9. Veal, W., MaKinster, J.: Pedagogical Content Knowledge Taxonomies. Electronic Journal of Science Education **3**(4) (1999)
10. Standl, B.: A Web-Application for building Common Cartridge Learning Objects. In: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2013, Chesapeake, VA, AACE (2013) 1461–1466

New trend in Russian informatics curricula: integration of math and informatics

Sergei Pozdniakov[1] and Svetlana Gaisina[2]

[1] Saint Petersburg State Electrotechnical University (ETU),
pozdnkov@gmail.com
[2] Academy of post-degree pedagogical education, Saint Petersburg,
g.selania@gmail.com

Abstract. Since the beginning of 1970s the course of informatics in USSR and then in Russia has been in development as an independent subject. Now it has a tendency to widen the covering of theoretical informatics and discrete math. The article analyses perspectives of the integration processes of mathematics and informatics and ways of smooth transition to the new curricula.

1 Introduction. History of school informatics in the USSR and Russia

Below, we present a historical review that demonstrates the relationship between mathematics and informatics in different periods of formation of the school informatics course first in USSR and then in Russia. Note, that the informatics course in Russia and USSR has never been a part of the mathematics course, although such proposals appeared from time to time. Now the question about interconnections of mathematics and informatics became urgent again, because according to the new standards that will come into force in 2015, the informatics and mathematics courses are included into the same educational area: “Mathematics and informatics”. At the same time, the subject “Informatics and ICT” is substituted with the subject “Informatics”, while ICT becomes a part of metasubject competences, that should be formed as the result of studying of all subjects.

1st stage. 50-70’s years of the XX-th century. It was before the emergence of the school informatics. First computers were installed in some specialized mathematical schools and their students began studying programming. Integration with mathematics was presented as the application of computing schemes to the solution of mathematical tasks.

2nd stage. 80’s years of the XX century. An informatics were introduced in school curricula. The first textbook of informatics (author Yershov) appear. This period was named period of “paper” informatics: schools have not computers and students wrote programs in special algorithmic language in writing-books. Integration with mathematics during this period was present at the level of representation of numbers, use of formulas, logical functions, the block schemes of algorithms. Subject was taught, as a rule, by the mathematics teacher.

3rd stage. 90’s years of the XX century. This stage was characterized by the emergence of personal computers and computer classes at schools. The study of algorithms rapidly extended at schools curricula. Interests of pupils started to move from mathematics to programming. So, the infomatics as a school subject, started to compete with the mathematics, becoming the main passion for many pupils and the main application of their forces. For those who was interested in depth study of informatics, the literature about the creation of algorithms both with proving their correctness appeared. Note, that, during this period, a dynamic geometry software and a number of programs to manipulate plots of functions were created, so it became possible to apply ICT for the mathematical education, but computer classes were still used only to teach informatics, so this was practiced only in rare schools.

4th stage. Beginning of the XXI century. The diversity of software leads to a partition of the informatics into two courses that together formed an educational area named “Informatics and ICT”. The first course was the informatics itself, and the second course was the technology course that replaced labour lessons and included ICT in particular. In ICT lessons, mostly the office applications were studied. This period is characterized by the huge drop of the interest to programming, that for a long time had been widely connected with the concept of “informatics”. Programming was gradually moving to the extracurricular activities. At the same time, one could mark a tendency in informatics courses to widen the coverage of theoretical informatics such as the information theory. Extracurricular courses stated to study cryptography.

5th stage. Late 10’s of the XXI century. Fast development of the Internet influenced the course of informatics. The Renaissance of the interest to programming was observed because of the web programming. A prototype of the Unified State Examination (USE) in mathematics, informatics, and popular internet competitions appeared. The last were used as a way to direct work of teachers to areas in which the course of informatics was going to be extended.

The current stage is characterized by a stabilization of ideas about a proportion between programming and information technologies in the school course of informatics. The Unified State Examination became an obligatory and a defining factor influencing the contents of textbooks on informatics. The new Concept of mathematical education in Russia was discussed, the main idea of its first versions was to integrate mathematics and informatics. In the final version of the Concept this integration is reflected in less extent, but the decision to associate the mathematics and informatics courses is anyway made for the high school (10–17 years old). The experiment with the integrated course has already began, and in 2015 it is going to be taught in all schools of Russia.

In this article some problems of integration of mathematics and informatics courses will be discussed and possible solutions of the arising problems will be proposed.

2 Changes in curricula: the new federal standards

The new federal standard changes the structure of the basic educational program [1, 2]. It unites the mathematics and informatics subjects in one subject domain “Mathematics and informatics” and defines general requirements to educational results such as the development of logical and mathematical thinking, an acquaintance with an idea of mathematical models; mastering mathematical reasonings; an ability to apply mathematical knowledge to the solution of various tasks and to estimate the obtained results; building skills to solve educational tasks; development of mathematical intuition; developing a picture of information processes in practical situations [3].

Table 1 presents a comparison list of skills that are tested by the Unified State Examination (USE in what follows) in mathematics and informatics. The integration of courses supposes that the building of these skills may become more effective if both topics are learned together.

In the field of informatics, the Federal standard is aimed at study the its fundamentals, it treats the informatics as a sciences about methods of analysis and assessment of the information, that allow for an effective decision making [4]. The important type of the educational activity is the application, analysis, and transformation of information models of real objects and processes by means of the information and communication technologies (ICT), including the study of other school disciplines. The conceptual idea of the new course, as before, is a mastering of the modeling as a method of learning objects, phenomena and processes of the real world. The new standard amplifies the orientation of the informatics course onto formation and development of algorithmic thinking, skills of algorithms construction and programming. The ICT competence was defined by the standard of 2004 as the "educational result, formed by the course of informatics and ICT". The new standard defines this competence as metasubject educational results formed and developed throughout all the process of training at high school.

Table 2 presents an analysis of USE tasks that demonstrates how many different informatics tasks demand diffeferent mathematical areas.

In the new standard the course of informatics is considered as a continuous course which includes a preliminary course at elementary school (6–10 years old), training in the middle and high school, and also a profile training in informatics in the senior classes (15–17 years old). By the end of elementary school (10 years) pupils get ICT competence, sufficient for further training. The purposes of school informatics are: the comprehensive development of pupils personality, obtaining knowledge, building of necessary skills, development of cognitive interests and creativity, development of personality traits valuable for each person and society as a whole. The course of informatics in the middle-school relies on the experience of everyday application of the ICT formed at elementary school and gives theoretical judgment, interpretation and synthesis of this experience. Together with mathematics, physics, chemistry, biology the course of informatics lays the foundation of the natural-science outlook.

The new standard introduced the two methodological sections in the mathematics curricula: logic and sets. In the existing standard of 2004, a topic "Elements of Combinatorics, Statistics and Probability Theory" was included in the profiling level, but in new standard it is offered to be studied at the basic level.

3 The unified state examination (USE) as a method of reorientation of teachers onto new standards: analysis of mathematics and informatics variants

Russian Federation recently accepted the Bologna convention and as a result introduced the all-Russian system of the knowledge quality assessment of pupils in the form of the unified state examination (USE). It defined the significance of theoretical knowledge for practical activities. So, for example, tasks on the subject “Elements of Combinatorics, Statistics and Probability Theory” mentioned above were included in materials for monitoring and measuring (MMM) and constituted 3% of the total number of mathematical tasks. In MMM of “Informatics and ICT” this subject was not selected as an independent section, but a student should know it because it was used as a tool to solve informatics tasks. About 40% of tasks in informatics and ICT check skills in representation of tabular and graphic data, a sequential and a simultaneous choice of several elements from a finite set, an ability to solve combinatorial problems, to determine a probability and statistical frequency of an event. The tasks in MMM on mathematics and informatics suppose the knowledge of numerical characteristics of data series, properties of binomial coefficients and the Pascal’s triangle, number of permutations, combinations, arrangements, and they also suppose an ability to apply this knowledge to the solution of practical problems.

The topics which traditionally belonged to the course of mathematics such as “Simulation”, “Numeration systems”, “Logic” in MMM were attributed to the informatics and ICT, where they were divided into independent sections “Simulation and Computer Experiment”, “Numeration systems”, “Logic and Algorithms”, “Processing of Numerical Information”, and they composed 30% of the total number of tasks. The structure of MMM underwent both qualitative, and quantitative redistributions (the Figure 1).

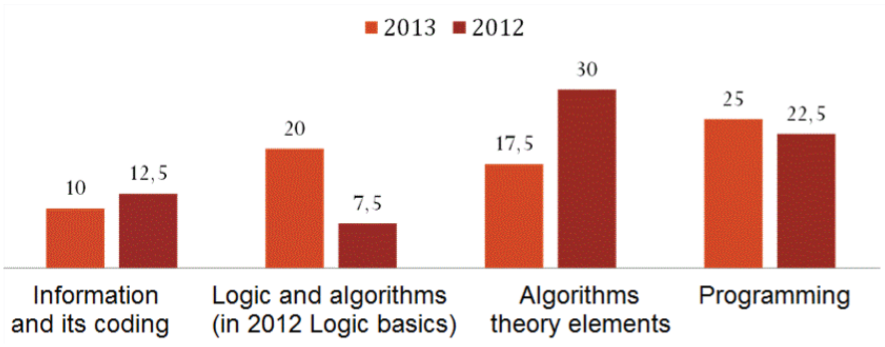


Fig. 1. Distribution of sections weights. In 2014th there were no changes in the distribution of jobs in comparison with 2013th

Table 1. Checked abilities on Unified State Examination

Mathematics	Informatics
– to be able to use the acquired knowledge and abilities in practical activities and everyday life; – to be able to execute computation and conversions; – to be able to solve the equations and inequalities; – to be able to execute actions with functions; – to be able to execute actions with geometrical figures, coordinates and vectors; – to be able to build and research mathematical models.	– simulation of objects, systems and processes; – interpretation of results of simulation – determination of the quantitative parameters of information processes – to accomplish information search and selection – to create and use data storage structures – to work with widespread automated information systems – to use the computer for sound processing

Based on the analysis of the demo version of the Unified State Examination of 2014 it is possible to see that only 2 tasks from 32 tasks (less than 10%) on informatics do not require computations.

Table 2. Comparing of Unified State Examinations jobs in mathematics and informatics

Units of the mathematics codifier	Number of tasks in mathematics according to the specification	Number of tasks in informatics which require knowledge in corresponding section in math
Algebra	8	30
Equations and inequalities	5	6
Functions	2	6
Elements of combinatorics, statistics and theory probabilities	1	14

All these changes characterize the strengthening of integration of mathematics and informatics.

4 Current problems of integration of mathematics and informatics and direction of their solution

Since the implementation of the new standards and the training programs which integrate mathematics and informatics has begun just recently, it is necessary to make an analysis of problems which may arise and propose ways for their solution.

Problem 1. One of the important problems is the insufficiency of ideas of the discrete mathematics and theoretical informatics in the traditional course of mathematics. There is a need to expand the mathematics course by adding new sections of the discrete mathematics. How to do it without exceeding the volume of mathematics within the curricula?

Problem 2. The first problem is closely connected with the problem of how to save mathematical culture while modifying the curricula. The mathematical culture is traditionally based on the contents verified and optimized during many years. The changes of the subject contents should not deplete the means of formation of the intelligent mechanisms of the trainee.

Problem 3. The third problem is the continuity problem. How should changes be done to switch to the new contents smoothly? Whether it is possible to use a traditional material to create the new representations relevant for the modern reality?

Problem 4. How to take into account the process of informatization of the society that influences the changes in nature of intellectual activities, without sacrificing the development of trainees' intelligent mechanisms?

Solutions proposed for Problem 1. The changing of the curricula in the modern conditions can start in the supplementary education. Supplementary education in new Russian standards is considered as a mandatory part of the school training. As an example of such an approach in Russia we present the competition "Construct, Test, Explore" [5–7]. This project is organized by the Computer Tools in Education Journal. Every year three special educational laboratories are created on the basis of this project. These laboratories propose an active approach to master new fundamental ideas of informatics and discrete mathematics. Certain research goals are set for the students, then they are provided with the educational software that estimates the progress in reaching of the goals based on a set of criterion for their partial solutions. The software allows students to track the improvements of their solutions, and also to compare the solutions with solutions of other participants. Within a week, school students make experiments. After that the solutions are sent to the jury to determine winners. Teachers and pupils then receive the methodical materials explaining theoretical aspects of problems which were provided by the competition. The examples of educational modules in discrete mathematics and informatics created for the research on the basis of the "Construct, Test, Explore" project are provided below [5]. One should note that modules usually have interdisciplinary nature.

1. Is it possible to compute without consuming energy? (Billiard computer by Fredkin and Toffoli)
2. Post's lattice of boolean clones and their bases
3. Boolean schemata for pattern recognition
4. Shannon entropy in communication

5. Fermat's principle in the search for the shortest path
6. Laws of interaction in an ensemble of particles
7. Optimization of routes in a graphs (a transport network)
8. Functional sorting
9. Calculations optimization
10. Gears and continued fractions: clock-calendar
11. Getting acquainted with the knot theory
12. Transformation groups and decomposition of complex transformations
13. NP-complete problems (variations of a traveling salesman problem)
14. Euclid algorithm and its generalizations (pouring fluids)
15. Finite automaton and the Turing Machine
16. Pendulums and the control of complex oscillations
17. Algorithms for programming devices

The other example is the "Beaver" competition [8] which along with the Unified State Examination reflects the tendencies of the modification of the informatics course. We have important results about the subjective complexity of the tasks for students when they collide with new ideas of informatics [9].

Solutions proposed for Problem 2. Note that the traditional basis for the formation of mathematical concepts is based on the external environment, that significantly changed during recent years. An important role in the student's habitat is now played by computers and the Internet, in other words, by the virtual reality which influence is comparable with the influence of natural events. So, it is necessary to explain the laws of a noosphere along with an explanation of the laws of the nature: how surrounding artificial subjects are arranged and how they work. It is a source of new interpretations of that general concepts which are already present in the mathematics; so, we should increase a "weight" of the general concepts which have clear virtual presentations (for example, geometrical conversions, polynomials, long numbers, logic expressions, formal grammars).

It is necessary to consider the following psychological aspect: in a classical educational environment the students themselves were the executors of algorithms, the conceptualization of many concepts was carried out through elementary algorithms for actions, i.e., skills [10]. The next step of the interaction of the person with the new environment is the usage of external tools to fulfill different intentions. Therefore it is important to partition algorithms to methods of conceptualization of knowledge and objects of study in mathematics. In the first case a human uses herself as a performer. This is not always justified, because the "data representation" of a human brain differs from the data representation in the case of computers, and so the same algorithms turn into different actions of the person and the machine.

Therefore it is important to separate the study of data structures from the study of algorithms in the new integrated subject. Such a decision will change an approach to study some math objects. For example it will lead to the teaching of long integers and finite state automata, syntax trees as formulas representations and so on.

Solutions proposed for Problem 3. The support of the continuity of math courses can be achieved by new interpretations of the traditional mathematical curricula [11]. For example ideas of multiplication and factorization of integers naturally receive interpretations in terms of complexity of algorithms, and lead to basic cryptography ideas.

The study of operation with polynomials and other symbolic (algebraic) object can be based on the question of "how these operations are executed inside the computer". The ideas of the formulas description through syntax trees naturally arise, and then an analysis and interpretation of operations with formulas through operations on trees become possible. The separation of the formulas syntax from their semantic aspects is important not only for informatics (discrete mathematics) but also for the "classical mathematics". For example, it is known that both for school students and for many teachers there is a problem to distinguish concepts of an algebraic expression and a function because both are described by formulas. The study of computer representation of formulas can help teachers to solve pedagogical problems connected with different types of historically formed elementary function notation, which not fall under the general template $f(x)$ (an exponential function, a logarithm on arbitrary base, a power function, a root).

Solutions proposed for Problem 4. The solution of the 4th problems can be based on the usage of the computer features that expand human opportunities to form new links inside one subject field and between various subjects. This is equivalent to the extension of possibilities of "understanding". So, the usage of verification tools, such as, for example, implemented in the environments of dynamic geometry, allows to expand a set of means for human to justify statements concerning properties of various mathematical objects [12].

5 Conclusion

1. Integration of mathematics and informatics makes it necessary to change the content of school mathematics to reflect ideas of the discrete mathematics and theoretical informatics.
2. The smooth changing of the curricula can be achieved by the introduction of new interpretations of studied concepts and the changings of accents in studying of traditional objects.
3. It is important to separate the study of data structures from algorithms in the new integrated subject. Such a decision will change an approach to study some math objects. "How computers would do it" is the thesis for the organization of practical activities.
4. The conceptualization of mathematical concepts in the rich information environment can be reached by the extension of a number of links between mathematics and informatics. This will compensate the reduction of conceptualization basis of mathematical concepts through the building of a wide set of skills.

References

1. "Federal Educational Standard of Russia (FES)" ["Federalnyj gosudarstvennyj obrazovatelnyj standart osnovnogo obshhego obrazovaniya"] available at: `http://standart.edu.ru/catalog.aspx?CatalogId=2588`
2. "FES: Secondary (full) general education" ["FGOS: Srednee (polnoe) obshhee obrazovanie"] available at: `http://standart.edu.ru/catalog.aspx?CatalogId=4099`

3. “Draft of curricula in mathematics” [“Primernaya programma osnovnogo obshhego obrazovaniya po matematike”] available at: `http://standart.edu.ru/catalog.aspx?CatalogId=2629`
4. “Draft of curricula in informatics” [“Primernaya programmya po informatike”] available at: `http://standart.edu.ru/catalog.aspx?CatalogId=8421`
5. Posov I., Pozdniakov S. (2013), “Implementation of Virtual Laboratories for a Scientific Distance Game-Competition for Schoolchildren”, The 2013 International Conference on Advanced ICT (Information and Communication Technology) for Education, September 20-22, (ICAICTE 2013), Hainan, China, Atlantis Press, pp. 495-499.
6. Pozdnyakov S., Posov I, Akimushkin V., Maytarattanakon A. (2013), “The bridge from science to school”, X World Conference on Computers in Education July 2-5, Toruń, Poland, vol.3, pp.131-132
7. Pozdniakov S. , Posov I., Pukhov A., Tsvetkova I. (2012), “Science Popularization by Organizing Training Activities Within the Electronic Game Laboratories”, International Journal of Digital Literacy and Digital Competence (IJDLDC), Volume 3: 2 Issues, pp 17-31.
8. Cartelli A., Dagiene V., Futschek G. (2010), “Bebras Contest and Digital Competence Assessment: Analysis of Frameworks”, IJDLDC 1(1), pp. 24-39.
9. Yagunova E., Ryzhova N.. “The use of on-line competition protocols to evaluate the task complexity and improve the validity of measuring procedure” [“Ispolzovanie protokolov on-lajn konkursov dlya ocenki slozhnosti zadach i povysheniya validnosti izmeritelnoj procedury”], Computer Tools in Education Journal, N 6, 2013, pp. 33-44.
10. Shapiro S. I. (1973). “From algorithms to judgement”, [Shapiro s.i. Ot algoritmov — k suzhdeniyam], M., Sovetskoe radio, 288 p.
11. Bogdanov M. , Pozdnyakov S. , Puhov A. (2009). “Multiplicity of the knowledge representation forms as a base of using a computer for the studying of the discrete mathematics”, PEDAGOGIKA 96, 136-142, ISSN 1392-0340.
12. Pozdniakov S. (2012). “Domain specific language approach to technology-enhanced learning”, The 12th International Congress on Mathematical Education, July 8-15, COEX, Seoul, Korea, ICME-12 proceedings p. 3676-3685.
13. Alyoshina N. A. and other (1990). “Logic and computer. Simulation of reasonings and validation of programs” [“Logika i kompyuter. Modelirovanie rassuzhdenij i proverka pravilnosti programm”], M., “Nauka”.
14. Velihov E.P. (1988), “Simple and dificult ideas in programming” [“Prostoe i slozhnoe v programmirovanii”], M., “Nauka”, ISBN: 5-02-006595-1.

Curriculum Integration Ideas for Improving the Computational Thinking Skills of Learners through Programming via Scratch

Filiz Kalelioğlu[1], Yasemin Gülbahar[2], Sümeyra Akçay[3], Dilek Doğan[4]

[1] Başkent University, Department of Computer Education and Instructional Technologies, Ankara, Turkey
`filizk@baskent.edu.tr`
[2] Ankara University, Department of Informatics, Ankara, Turkey
`gulbahar@ankara.edu.tr`
[3] Başkent University Private Ayşeabla School, Ankara, Turkey
`sumeyrabildi@gmail.com`
[4] Ankara University, Department of Informatics, Ankara, Turkey
`surbahanli@ankara.edu.tr`

Abstract. How and what would you teach if you had only one course to help students explore the essence of computation and perhaps inspire a few of them to think computationally? Generally, students learn all the information and communication tools provided to them, but they are never expected to write any computer programs. A new ICT curriculum, which started being implemented last year in Turkey, gives students a chance to learn not just using computers, but to think like computers. This paper provides implementation suggestions for the integration of Scratch into the existing ICT curriculum, based on prior research on the phenomenon. Hence Scratch, as a computational thinking tool, will be discussed in terms of its possible contribution to students' computational thinking skills.

Keywords: Scratch, computational thinking skills, ICT curriculum

1 Definition, Components and Reasoning of "Computational Thinking"

Today's learners are expected to possess 21st century skills, which contain problem solving and critical thinking skills. These skills are important for effective reasoning, and for analysing and adapting existing knowledge into new contexts, setting up connections between information and arguments, making proper judgments, interpreting information and for drawing conclusions based on the best analysis. They also have to reflect critically on what is learnt for the purpose of decision making and solving problems effectively. Moreover, these abilities are not just for computer scientists [25], they are needed in all disciplines as mentioned by [8]: "The ability to see a problem and then to solve it is important, not just in science and the military; lawyers need a similar ability when putting together and arguing a case. Authors need it when conceiving a subject and then putting it into words."

However, even graduates are having difficulties with solutions to real life problems or applying their theoretical knowledge and putting it into practice, which is reflecting or connecting related information into proper situations. As stated by [24], all these skills are related to the human mind, since cognitive capacity to solve any problem is limited by the knowledge stored in the mind. On the other hand, knowledge is derived from thinking processes based on a variation from simple to complex. Hence, the application of critical thinking processes via existing knowledge, and computers for solving complex technological problems, can be called "Computational Thinking (CT)". In other words as stated by [25] "Computational thinking involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science". Similarly, [1] considered computational thinking as "...the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms".

In their article, [6] defined computational thinking as an approach to solve problems through computers and also as a problem solving methodology that can be automated and transferred and applied across subjects. The researchers also underlined that through computational thinking "Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyse data, and to create real and virtual artefacts". Parallel with this idea, [11] defined some elements which form the basis of curricula that aim to support its learning. The researchers mentioned abstractions and pattern generalisations, systematic processing of information, symbol systems and representations, algorithmic notions of flow of control, iterative, recursive, and parallel thinking, and some other elements as important assets for the development of ICT skills.

To summarise, we know that the human mind is the most powerful problem-solving tool and the ability to extend the power of human thought with computers and other digital tools has become an essential part of the 21st century skills set. This is why we should continue researching how, when, and where computers and other digital tools can support us in solving problems [6]. Hence, this study discusses the implementation of a visual programming tool within the related curriculum in order to equip learners with computational thinking skills at grades 5-6.

2 "Information and Communication Technologies and Software" Course in Turkey

In 2012, the computer literacy course curriculum, as well as the course name, changed in Turkey. A standards-based curriculum approach was preferred and a framework was established, based on the international standards of ICT proposed by ISTE and NAACE. It is composed of four dimensions: (1) Digital literacy, (2) Communication, Knowledge Sharing and Self-Expression via ICT, (3) Research, Knowledge Construction and Collaboration, and (4) Problem Solving, Programming and Development of Authentic Materials. The official name of the course is specified as "Information and Communication Technologies and Software" [19, 20, 23]. Three learning levels were defined for students: basic, intermediate and advanced, with each level composed of two stages. In the "Problem Solving, Programming and Development of Authentic Materials" standard, learners are expected to possess skills about Problem-Solving Approaches, Algorithm and Strategy Development, Programming and finally Software Project Development, Implementation and Dissemination. With this framework, teachers are free to choose the tool and software to teach computing and programming skills to children [12]. As a first attempt, Scratch program has been translated into Turkish and inserted into the main portal of the National Ministry of Education, or "Education and ICT Network (http://www.eba.gov.tr/ara?q=scratch)". Moreover, sample tutorials are also presented in the network for different examples of the use of Scratch.

Since our implementation was at the introductory level, we planned a 5 week course and our objective was teaching "Problem-Solving Approaches" together with some programming skills. Hence, our learning outcomes were designed as shown in Table 1.

Table 1. Intended Learning Outcomes to Teach Program Solving Approaches at 5th Grade

	Standard 4. Problem Solving, Programming and Development of Authentic Materials					
Levels → **Standards ↓**	**Basic I: Comprehension of ICT**	**Basic II: Access to information and evaluation**	**Inter-mediate I: Managing information**	**Inter-mediate II: Information conversion**	**Advanced I: Information generation**	**Advanced II: Share information**
Problem-Solving Approaches	1.1. Defines the concepts of algorithm, strategy and problem-solving. 1.2. Awareness of the problems encountered in the use of ICT. 1.3. Refers to the importance of problem solving.	2.1. Comments on solvability of a problem in the process of problem solving. 2.2. Determines the required variables and processes to solve problems. 2.3. Realises the relationship of the concepts of algorithms and strategies.	3.1. Lists different problem-solving approaches. 3.2. Debugging to make necessary corrections to run the program correctly.	4.1. Suggests a different solution for solving the problem 4.2. Creates flowchart for displaying the solution of a problem 4.3. Creates animated scenes according to prepared by flow.	5.1. Creates the specified steps to solving the problem. 5.2. Reaches the most effective solution to questioning the validity of developed steps for problem solving.	6.1. Offers creating solutions for the identified problems and approach 6.2. Shares program code and executable file in social media.

3 A Computational Thinking Tool for Children: Scratch

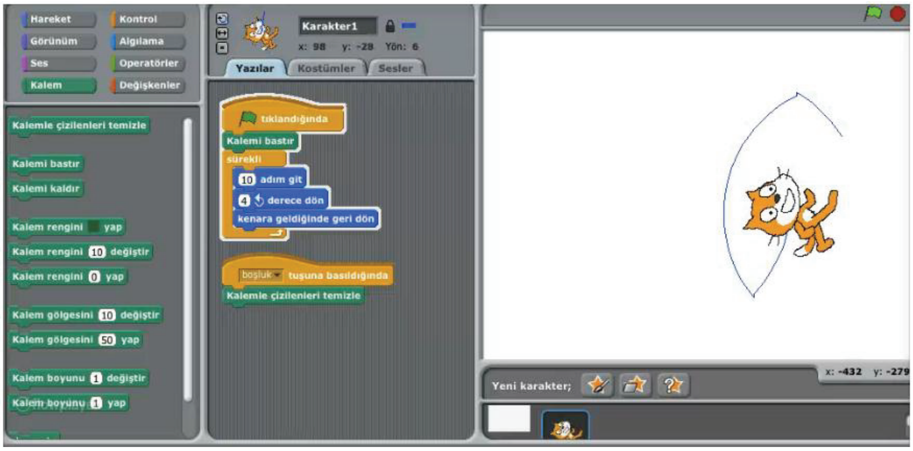
According to [18], in programming, learners need to acquire three information types: Syntactic, Conceptual and Strategic/Conditional Knowledge. Syntactic Knowledge includes knowing the syntax of a programming language, and the ability to explain syntactic differences. Conceptual Knowledge includes using syntax rules to write programs. Strategic/Conditional Knowledge includes the using of syntactic and conceptual knowledge effectively to design code and test programs to solve problems. In the traditional teaching of programming, generally students don't know how common programming structures work, nor do they have any concept about errors [4].

Visual programming software, like Scratch, Alice etc., can facilitate the teaching of programming to children without the need for memorisation, since the curriculum aims to teach problem solving and computational thinking skills to learners. Scratch is a free, open-source software, offering support for 61 different languages. Although Scratch is a programming language for 8-16 year old children, younger children can also work with the help of their parents or older siblings [9]. Children can program their stories, games and animations, and they can also share their artefacts and programs with people from all over the world [14]. It has been designed and developed by the Lifelong Kindergarten group at the MIT Media Lab since 2003 [21]. According to 2014 data from Scratch, there are 2,692,964 registered users and 4,727,653 projects shared [22].

Scratch helps people learn to think creatively, reason systematically, and work collaboratively with 21st century skills. It ensures users make their projects personally engaging, motivating and meaningful because of its easy use to import or create any kind of media, such as images, sounds, and music [17]. Scratch provides opportunities for students [15], some of these are interactivity and the incentive for multi-sensory, active, experimental learning environment; focus on design of visual statement instead of memorising specific syntactic learning of programming languages; conditions to create rich useful coding tools, condition of tools to organise a variety of linked, dynamic representations, numerical, textual, and audio-visual; supporting of problem solving; providing immediate feedback on student programming actions for self-correction; motivation and representation of different activities such as black-box activities, and collaborative learning activities.

As suggested by [16], Building-block programming, Programmable manipulation of rich media, Deep shareability, Integration with the physical world and Support for multiple languages are the core features to be considered in the design of learning environments.

With the building-block programming feature, users can drag and drop graphical block structures so they can put together the parts of the program easily without memorising scripting. Thus, syntax errors, which are the biggest problem for beginners, are eliminated in the teaching of a programming language. Different stacks of blocks execute in parallel, such as procedures or functions. There are various block categories available on Scratch, such as motion, looks, sound, pen, data, events, control, sensing, operators and other blocks (Fig. 1 shows an example).



```
tıklandığında
Kalemi bastır
sürekli
    10 adım git
    4 derece dön
    kenara geldiğinde geri dön

boşluk tuşuna basıldığında
Kalemle çizilenleri temizle
```

Fig. 1. A Sample Screen from the Pen Block

According to [7], there are different Scratch block types such as trigger, reporter, loop, conditional, and predicate, command and operator blocks. Trigger blocks allow runscripts when the green flag is clicked, a specified key is pressed, a message broadcast from another sprite, backdrop switches to a certain background, or selected attributes such as loudness, timer etc. Loop Blocks repeat blocks and run statements inside of them. Conditional blocks allow executing if/else statements or comparing commands. Predicate blocks return true or false. Operator blocks perform mathematical and text operations.

Through programmable manipulation of media rich features, Scratch allows users to manipulate images, animations, movies and sound with a visual interface and therefore without the initial boring activities of numbers, strings or simple graphics associated with traditional programming. The feature of deep shareability supports the sharing of components, techniques, ideas and other projects on many types of devices such as desktop computers, laptops, tablets, and mobile phones. Through integration with the physical world feature, it is possible to control the behaviour of Scratch creations via switches, sliders, distance sensors, motion detectors and sound sensors. With the support for multiple languages, Scratch ensures a multilingual environment to support the collaboration and sharing of users from different countries.

In addition to these properties, programming concepts and skills are supported in Scratch so that students can learn important computational skills and concepts in a virtual environment via 2D or 3D characters and objects which have been designed by users. Students do not need to know programming syntax in the process of writing computer programs. The success of teaching and learning programming can be achieved by real life problems [10]. Scratch provides an easy interface for users to create their own environments and their own code without the need for memorising. Moreover, in their study, [2] emphasised that algorithms, introduction to informatics, information technology and applications, computer graphics and editing them, information technology and security, different programming languages, creating database and animation and creating games should be taught from the 5th grade of primary school, with individuals to be brought into a manufacturer of information technology. The purpose and targeted age groups of Scratch makes it suitable for programming courses for the 5th grade of primary schools.

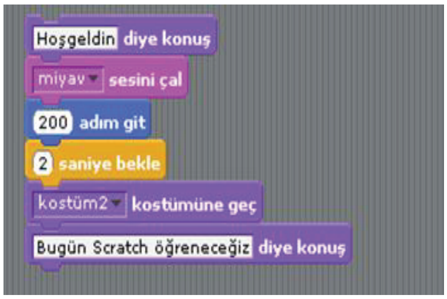
Based on these facts, scratch could be seen as an effective computational thinking tool, since a computational thinking tool must meet some conditions within in a curriculum. According to the computational thinking tool checklist proposed by [28], CT tool has low threshold, high ceiling, scaffolds flow, enables transfer, supports equity and systemic and sustainable. To illustrate this point more elaborately several points can be considered: (a) a student can easily program a working and playable game with this tool, (b) the curriculum has to support development of these skills with these tools, (c) curriculum has to support transferring between them, (d) the tool has to support equity across gender and ethnicity boundaries, and lastly (e) a CT tool and curriculum can be used by all students and teachers.

4 Ideas for Curriculum Integration

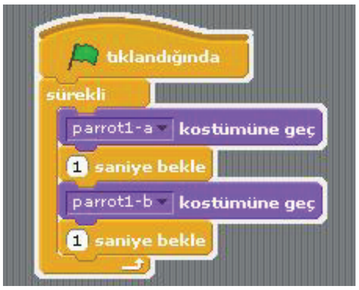
A five week instructional programme to teach problem solving skills to 5th grade primary school students via Scratch was designed by [13]. In this instructional programme, the students were taught topics such as an introduction to Scratch programming, installation of a Scratch platform, introduction to User Interfaces and the writing of programs such as Hello World, Parrot, Aquarium programs and Maze project. In total, students have five hours to write these programs at school.

In the first week, students are introduced to the basic terminology and concepts of problem solving and programming. They practice with the Scratch program and learn about the interface and logic of the software. Hence, it is aimed that the learning outcomes of Basic I Comprehension of ICT level are achieved (1.1, 1.2 and 1.3 in Table 1).

In the second week, with the Hello World program (Fig. 2), students added a background picture to their scenes. Then they assigned a meow sound command to the cat character, they moved the cat and finally they made the cat give messages such as “Welcome”, and “Today we will learn how to scratch”. In the Parrot program (Fig. 3), students learned how to start the program, iterate the block and change the cat character. They made the parrot character fly by changing the costumes of the character. Therefore, it is intended that the learning outcomes of Basic II: Access to information and evaluation level are achieved (2.1, 2.2, 2.3 in Table 1).



```
Hoşgeldin diye konuş
miyav sesini çal
200 adım git
2 saniye bekle
kostüm2 kostümüne geç
Bugün Scratch öğreneceğiz diye konuş
```



```
tıklandığında
sürekli
    parrot1-a kostümüne geç
    1 saniye bekle
    parrot1-b kostümüne geç
    1 saniye bekle
```

Fig. 2. Flow of the Hello World Program

Fig. 3. Flow of the Parrot Program

In the third week, with the Aquarium program (Fig. 4), students learned how to manage and control more than one character, and practiced how to iterate the blocks. They learned to how to turn the character around if the character was on the edge, how to point towards the mouse cursor, how to change the colour effect, and how to play sounds. Consequently, with this lesson, it is planned that the learning outcomes of Intermediate I: Managing information level are accomplished (3.1 and 3.2 in Table 1).

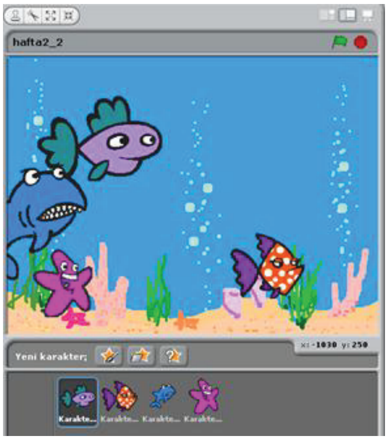
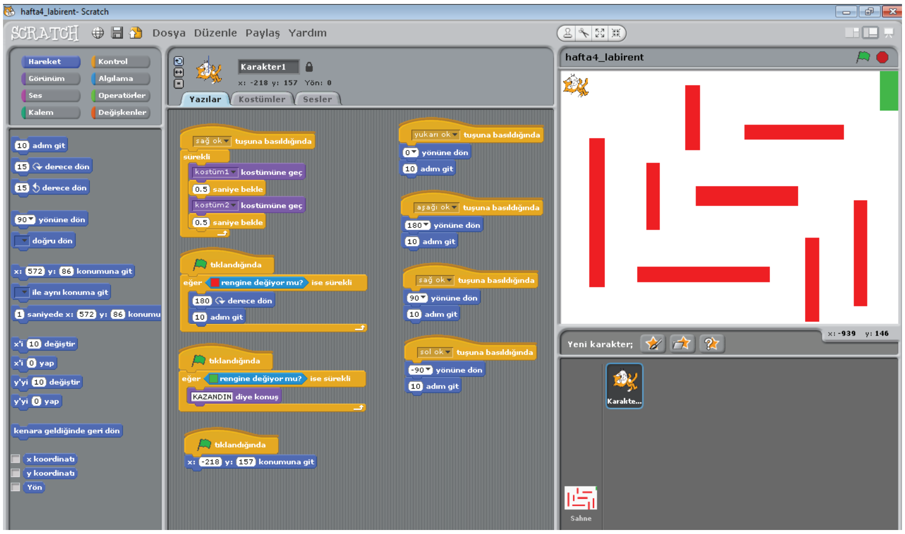


Fig. 4. Aquarium Program

For the 4th and 5th weeks, students spend these two weeks on the Maze project (Fig. 5) by adding variables for calculating time spent in the game, life of the character and by adding sounds to the scene. Moreover, students learned how to use if conditions, how to use and activate arrow keys from the keyboard and how to change the X and Y positions of the characters. Consequently, in this lesson, different learning outcomes at different levels are accomplished (2.2, 3.1, 3.2, 4.1, 4.2 and 4.3 in Table 1).



```
[sağ ok] tuşuna basıldığında
sürekli
    [kostüm1] kostümüne geç
    (0.5) saniye bekle
    [kostüm2] kostümüne geç
    (0.5) saniye bekle

tıklandığında
eğer <[ ] rengine değiyor mu?> ise sürekli
    (180) derece dön
    (10) adım git

tıklandığında
eğer <[ ] rengine değiyor mu?> ise sürekli
    [KAZANDIN] diye konuş

tıklandığında
x: (-218) y: (157) konumuna git

[yukarı ok] tuşuna basıldığında
(0) yönüne dön
(10) adım git

[aşağı ok] tuşuna basıldığında
(180) yönüne dön
(10) adım git

[sağ ok] tuşuna basıldığında
(90) yönüne dön
(10) adım git

[sol ok] tuşuna basıldığında
(-90) yönüne dön
(10) adım git
```

Fig. 5. The Maze Project

5 Perceptions about the Implementation: Interview with ICT Teacher

After the implementation of the instructional programme, a structured interview was conducted with the ICT teacher. She thought that Scratch was pretty good as an interface for graphical programming language for students. She added that the basis of the algorithm could be given while teaching programming in Scratch. Especially for beginners, teaching the fundamental concepts of programming is so important to gaining this high level skill [10, 26]. Conceptual information of programming and problem-solving skills are also required as well as programming syntax for students [26]. On the question about what kind of applications students should practice in Scratch, she suggested dialogues to begin with, then move on to design the scene, determine the character / characters, make characters talk and move and lastly, to create a game using conditions and variables. She explained that there were learning issues while teaching the conditions topic. As [10] stated, the teaching of programming is directly related to the student's problem solving skills. Due to the difficulty of gaining programming skills for beginners, student successes in programming courses is generally quite low.

In answer to the question, which competencies that students gain; she stated that students learn the rules and follow a logical sequence within the framework of blocks, learn to correct mistakes and reach quicker solutions while writing a program, and thereby learn to solve problems.

On students' reflections about Scratch, she said that students take great pleasure from using the Scratch program. On the whole, most male students love the idea of writing a gaming program, whereas the girls usually face more difficulties. On suggestions about teaching programming more successfully, she said that students and teachers should take programming very seriously and should organise events and publicity. Children should be encouraged. A Turkish national resource should be created and IT teachers should share how they teach programming to their classes. A repository about programming could be created under the EBA site.

6 Discussion & Conclusion

At the basic level of the ICT curriculum, concepts, different problem-solving approaches, identifying variables and processes, using the compiler, commenting on problem solving are important. In teaching programming languages, generally the sequencing rules of any programming language is transferred to the students, from simple to complex, starting with the algorithmic way of thinking. However, teaching programming is a quite difficult and complicated process.

Generally students try to solve problems without a real understanding of it and they don't think to establish the correct analogies between their old knowledge and the new problems, so their solutions are usually unrelated and incorrect [4]. Most of time students can find programming to be a difficult and complex cognitive task [3]. Most of the time teachers and students focus on the syntactic details of a programming language [4] and the teaching of syntax may take up considerable time for teachers and students. In a traditional approach, the teaching of programming is not personalised and teachers cannot provide appropriate teaching strategies for each student [4]. Students can create their own scenarios; see the syntax error and program with different solutions at their own learning speed with Scratch. Teachers can use it in different ways on different subjects in the curriculum to involve them.

When the intended learning outcomes of teaching problem solving approaches for the 5th grade were reflected into the lesson with sample program ideas for curriculum integration, it is revealed that the basic and intermediate levels were achieved, while the learning outcomes at advanced levels were not. Due to time limitations and distribution of the other subjects in the curricula, such a result is not surprising. In fact, for an entry level of such a course, those applications could be seen as sufficient; it could be decided that the learning outcomes at higher levels can be achieved in the upper classes as this curricula suggests.

The more important point here is that whether or not students gain computational thinking skills with these intended learning outcomes. Writing programs is not enough to develop computational thinking skills, but it is important to support computational competencies [11]. Hence, they stated the characteristics believed to form the basis of computational thinking skills. Moreover, it is believed that these characteristics are used to assess the development of computational thinking skills within the teaching and learning processes. These characteristics are abstractions and pattern generalisations (including models and simulations), systematic processing of information, symbol systems and representations, algorithmic notions of flow of control, structured problem decomposition (modularising), iterative, recursive, and parallel thinking, conditional logic, efficiency and performance constraints and debugging and systematic error detection. Similar characteristics that form the basis of computational thinking skills are analysing and logically organising data, data modelling, data abstractions, and simulations, formulating problems such that computers may assist, identifying, testing, and implementing possible solutions, automating solutions via algorithmic thinking and generalising and applying this process to other problems (http://en.wikipedia.org/wiki/Computational_thinking).

When the created applications of the students were considered, it can be said that students tried to achieve some cognitive tasks for developing computational thinking. Because, students processed information systematically, they used symbolic systems and representations, algorithmic notions of flow of control, practiced iterative, recursive, and parallel thinking. And lastly, it can be said that they tried to use conditional logic, debugging with their programs, to identify, test, and implement possible solutions and to generalise solutions to other problems. In brief, it is thought that practicing in Scratch has possibilities to contribute to students' computational thinking skills. On the other hand, from a more scientific perspective, to present evidence about possible contribution to students' computational skills, developed skills should be measured with a valid and reliable instrument. Alternatively, in a study of [27], researchers proposed an assessment framework in order to present the development of computational skills of the Scratchers. These approaches are project portfolio analysis, artifact-based interviews and design scenarios. For the first approach, project portfolio analysis, a user analysis tool such as a Scrape tool was used to analyse the portfolio of the students' projects uploaded to the online community. This analysis provided a visual presentation of the used and unused blocks in every project. With the artifact-based interviews, researchers can interview with the students about their project and project creation process. For the last approach – design scenarios, students were engaged in several independent activities such as students were tried to explain what the selected project does and, how it could be extended, fix a bug and remix it by adding a something new. Each approach has strengths and limitations in itself. Therefore, no single assessment approach is sufficient enough to prove the computational skills of the students and a mix of assessment approaches should be used to show evidence of these skills.

References

1. Aho, A. V.: Computation and Computational Thinking. The Computer Journal. 55(7), 832-835 (2012)
2. Akpınar, Y., Altun, A.: Bilgi Toplumu Okullarında Programlama Eğitimi Gereksinimi. Elementary Education Online. 13 (1), 1-4 (2014)
3. Aktunc, Ö.: A teaching Methodology for Introductory Programming Courses using Alice. International Journal of Modern Engineering Research (IJMER). 3(1), 350-353 (2013)
4. Anabela, G., Mendes, A.J.: Learning to Program – Difficulties and Solutions. Proceedings of International Conference on Engineering Education (ICEE). September 3-7, Coimbra, Portugal (2007)

5. Barr, D., Harrison, J., Conery, L.: Computational Thinking: A Digital Age Skill for everyone. Learning & Leading with Technology. 38(6), 20-23 (2011)
6. Barr, V., Stephenson, C.: Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?. ACM Inroads. 2(1), 48-54 (2011)
7. Beug, A.: Teaching Introductory Programming Concepts: A Comparison of Scratch and Arduino. (Master's Thesis). The Faculty of California Polytechnic State University, San Luis Obispo (2012)
8. Day, C.: Computational Thinking Is Becoming One of the Three Rs. Journal of Computing in Science and Engineering. 13(1), 88 (2011)
9. Ebrahimi, A., Geranzeli, S., Shokouhi, T.: Programming for Children; "Alice and Scratch Analysis". Conference on emerging Trends of Computer Information and Technology (ICETCIT). Singapore, November 6-7 (2013)
10. Genç, Z., Karakuş, S.: Tasarımla Öğrenme: Eğitsel Bilgisayar Oyunları Tasarımında Scratch Kullanımı. International Computer and Instructional Technologies Symposium ICITS 2011, Fırat University, Elazığ, Turkey (2011)
11. Grover, S., Pea, R.: Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43 (2013)
12. Gülbahar, Y., Ilkhan, M., Kilis, S., Arslan, O.: Informatics Education in Turkey: National ICT Curriculum and Teacher Training at Elementary Level. Informatics in Schools: Local Proceedings of the 6th International Conference ISSEP 2013 – Selected Papers, 77-87 (2013)
13. Kalelioğlu, F., Gülbahar, Y.: The Effect of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. Informatics in Education, 13(1), 33-50 (2014)
14. Karabak, D., Güneş, A.: Ortaokul Birinci Sınıf Öğrencileri için Yazılım Geliştirme Alanında Müfredat Önerisi. Journal of Research in Education and Teaching. 2(3), 175-181(2013)
15. Kordaki M.: Diverse Categories of Programming Learning Activities Could Be Performed within Scratch. Procedia -Social and Behavioral Sciences. 46, 1162-66 (2012)
16. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M.: Scratch: A Sneak Preview. Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan, 104-109 (2004)
17. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The Scratch Programming Language and Environment. ACM Transactions on Computing Education (TOCE). 10 (4) (2010)
18. McGill, T. J., Volet, S. E.: A Conceptual Framework for Analyzing Students' Knowledge of Programming. Journal of Research on Computing in Education. 29(3), 276-197 (1997)
19. NAACE: Draft Naace Curriculum Framework Information and Communication Technology (ICT) Key Stage 3 (2012)
20. NAACE: ICT Framework: A Structured Approach to ICT in Curriculum and Assessment (Revised Framework) (2007)
21. Scratch: Scratch Statistics-Imagine, Program, Share. `http://scratch.mit.edu/` (last checked 2/4/2014)
22. Scratch Statistics: Scratch Statistics-Imagine, Program, Share. `http://scratch.mit.edu/statistics/` (last checked 2/4/2014)
23. The International Society for Technology in Education (ISTE): National Educational Technology Standards for Students. `http://www.iste.org/standards/nets-for-students` (last checked 1/31/2013)

24. Voskoglou, M. G., Buckley, S.: Problem Solving and Computers in a Learning Environment. Egyptian Computer Science Journal (ECS) . 36(4), 28-46 (2012)
25. Wing, J. M.: Computational Thinking. Communications of the ACM, 49(3), 33-35 (2006)
26. Yurdugül, H., Gültekin, K.: Çokluortamın Bilgisayar Programlama Başarısı Üzerine Etkisi. Proceedings of 9th International Educational Technology Conference (IETC 2009). Hacettepe University, Ankara, Turkey, May 6-7-8, 449-457 (2009)
27. Brennan, K., & Resnick, M. Using artifact-based interviews to study the development of computational thinking in interactive media design. Paper presented at annual American Educational Research Association meeting , Vancouver, BC, Canada, April 13-17, 1-25 (2012)
28. Repenning, A., Webb, D., & Ioannidou, A. Scalable Game Design and the Development of a Checklist for Getting Computational Thinking into Public Schools. http://www.cs.colorado.edu/~ralex/papers/PDF/SIGCSE10-repenning.pdf (last checked 4/8/2014)

Poster Presentations

Effects Of Programming Course On Middle School Students' Reflective Thinking Skills Towards Problem Solving

Kadir Burak Olgun[1], Gonca Kızılkaya Cumaoğlu[1], Sevinç Gülseçen [2]

[1]Yeditepe University, İstanbul, Turkey
{*kadir.olgun*, gonca.kizilkaya}@yeditepe.edu.tr
[2]Istanbul University, İstanbul, Turkey
gulsecen@istanbul.edu.tr

Algorithm is a method to solve problem by following significant instructions entirely [2]. This method is one of the important steps of programming. According to [3], there are three steps of correct programming. First one is problem analysis, second one is to generate an algorithm and the last one is to implement this algorithm. [5] suggest that the main point of the programming is to develop problem solving strategies. In respect of the study of [1], 7 years old students, who are programmers, have more reflective thinking skills than non-programmers. In the light of these researches, the aim of this study is to analyze the effect of programming course on middle school students' reflective thinking skill towards problem solving.

In this research, Scratch, improved by Lifelong Kindergarten Group in Massachusetts Institute of Technology [7] and a graphical programming language that simplify programming [4], was used for teaching programming on experimental group (N=50) for 4 weeks. Control group (N=50) have only have the regular computer course. After the treatment, reflective thinking skill scale, developed by [6], was applied on 100 sixth grade students. The results of the study and the implications of the future researchs will be discussed.

References

1. Clements, D. H., Gullo, D. F.: Effects of computer programming on young children's cognition, Journal of Educational Psychology, 76(6), 10-51 (1984)
2. Futschek, G.: Algorithmic Thinking: The Key for Understanding Computer Science, In Lecture Notes in Computer Science, 4226, 159-168 (2006)
3. Garner, S.: Learning resources and tools to aid novices learn programming, Informing Science & Information Technology Education Joint Conference (INSITE), 213-222 (2003)
4. Genç, Z., Karakuş, S.: Tasarımla Öğrenme : Eğitsel Bilgisayar Oyunları Tasarımında Scratch Kullanımı, 5th International Computer & Instructional Technologies Symposium, 22-24 September 2011 Elazığ (2011)
5. Kazimoglu, C., Kiernan, M., Bacon, L., Mackinnon, L.: A serious game for developing computational thinking and learning introductory computer programming. Procedia-Social and Behavioral Sciences, 47, 1991-1999 (2012)
6. Kızılkaya, G., Aşkar, P.: The Development of A Reflective Thinking Skill Scale Towards Problem Solving, Education and Science, 34(154), 82-92 (2009)

7. Resnick, M., Maloney, J., Hernandez, M. A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Sılver, J., Silverman, B., Kafai, Y.: Scratch: Programming For All, Communications Of The Acm, 52(11), 60-67 (2009)

The Effect of Collaborative Game Design on Critical Thinking, Problem Solving and Algorithm Development Skills

Mehmet Fatih ERKOÇ[1], Sevinç GÜLSEÇEN[2]

[1]Yildiz Technical University, İstanbul, Turkey
mferkoc@yildiz.edu.tr
[2]Istanbul University, İstanbul, Turkey
gulsecen@istanbul.edu.tr

1 Introduction

Since the mid-1990s, computer games have captured the attention of training professionals, educators, and researchers. As a result of this situation, a considerable amount of study has been published on the value of educational use of the games and its potential effects on learning. [1] claims that games can produce engagement and enjoyment which offer a powerful format for educational environments in learning. New generation of students should be considered not only as consumers of technology but also as producer of 21st Century technologies. This proposal, have focused on examine the effect of game design with Microsoft KODU, which is an innovative tool for game making, on critical thinking, problem solving, and algorithm development skills of 6th grade students.

2 Literature Review

[4]'s in their study, they claimed that students' motivation, as measured by attention gained, perceived relevance, confidence instilled, and satisfaction, was improved by playing educational games. It can be mentioned that providing game for educational use can encourage critical, pro-social problem solving to support the development of the 21st century skills [6]. To be a successful employee and an effective member of society in the 21st century, today's students must have a range of skills such as creativity, critical thinking, problem solving etc. Many researchers have investigated the potential effects of building children's and young people's own games on learning new knowledge and skills [6]. [5] claimed that creating game is a complex task requiring creative skills such as problem finding, problem solving, evaluation and communication. However, according to [2], game building activities can be considered as an effective way for improving most relevant skills including collaboration, communication, ICT literacy, creativity, critical thinking, problem solving and productivity, required to be successful in the Knowledge Society. Additionally, [3] mentioned that, as a possible future learning method, learning by making digital games approach can prepare students for the challenges of the 21st Century. When

examining the literature, game building activities motivate and encourage the students for learning and constructing the new knowledge and 21st Century skills. On the other hand, it provides new opportunities to teachers for creating satisfactory classes.

3 Method

This is a quasi-experimental study with pretest and posttest design, will be carried out during the fall semesters of the 2014-2015 academic years. Both quantitative and qualitative analyses will be used to examine the collected data through Cornell Critical Thinking Skills Test, Problem Solving Test, Algorithm Developing Achievement tests developed by researchers and the semi-structured interviews. All of the three groups will take these tests before begin implementing. In addition to face-to-face Microsoft KODU education, twenty five students who will be randomly assigned to first treatment group will divide 5 groups, and each group will make their own game collaboratively. Other twenty five students who will be assigned second treatment group, they will make their game individually. Last group will be assigned as control group. And the last group will be assigned as control group. At the end of the study all of the groups will take the tests again. SPSS package program will use for analyzing collected data.

References

1. Boyle, T.: Design for multimedia learning. Prentice Hall (1997)
2. Dagnino , F., Earp, J., Ott, M: Investigating the" MAGICAL" Effects of Game Building on the Development of 21st Century Skills. 5th International Conference of Education, Research and Innovations (pp. 19-21). Madrid,SPAIN: IATED (2012).
3. Earp, J., Dagnino, F., Kiili, K., Kiili, C., Tuomi, P., Whitton, N.: Learner Collaboration in Digital Game Making: An Emerging Trend. Learning & Teaching with Media & Technology (439-447). Genoa, ITALY: Association for Teacher Education in Europe (2013).
4. Klein, J. D.: Freitag, E. Effects of using an instructional game on motivation and performance. The Journal of Educational Research, 84(5), 303-308 (1991).
5. Robertson, J.: Making games in the classroom: Benefits and gender concerns. Computers & Education, 59, 385–398 (2012).
6. Thomas, M. K., Ge, X., Greene, B. A. Fostering 21st century skill development by engaging students in authentic game design projects in a high school computer programming class. Journal of Educational Computing Research, 44(4), 391-408 (2011).

Contributions of Universities to Children’s Informatics Education in Turkey

Şebnem Özdemir

Istanbul University, Department of Informatics, Istanbul, Turkey
sebnemozde@gmail.com

Education is a fundamental force for economic improvement, social wellbeing and construction of society. At that point universities have important roles as educational establishments and centers of social and technological changes with their educational curriculums, labs, research centers and different type of certificate and support programs. Besides universities have special missions such as researching the society’s regional and global issues, needs and being ready for describing and analyzing new generation, who have different habits, learning style. That generation has an innate predisposition to using new technologies. Although their predisposition, they need guidance for increasing their potential and being young people with well-developed algorithmic thinking [1] [2] [12]. Many universities organize science, art and etc. events for children not only for guidance to a new generation, but also need of being information society. The concept of “children university or children science center” was established to create a stable and a continual platform for promoting interest in science among the young generation [3].

The idea of children university based on developing and shaping the future member of society by triggering their academic curiosity [4] [6] [7] [13]. The first children university was founded in 2002 at the University of Tübingen in Germany [3]. In Turkey, the first children's university was founded 2009 in Ankara [4]. Even though there are 195 universities in Turkey [5] [8], only 16 of them have children university or children science center.

The goal of this study is to analyze the contributions of universities in Turkey to children’s informatics education via children university and children science center. With that goal, their summer, winter and weekend programs were examined and analyzed in order to find out the courses that are related to informatics education.

There were over 150 different courses in those children universities and children’s science centers, but only 20 of them were related to informatics education, such as “planet of informatics, dynamic math, programming with Small Basic, programming with Kodu and etc.” There was no common educational purpose and time planning in those courses. For example, programming with Small Basic in Istanbul University’s children’s university takes three weeks, but “game programming” in Ankara University’s children’s university takes only 5 days, even though they have the same educational purpose and common expectations. On the other hand, according to Sadonichy (2011) informatics mostly contains mathematical fundamentals of informatics and information technologies [9]. But the contents of the courses were mostly focusing information technology and mostly ignoring mathematical fundamentals of informatics.

Because of Turkey's Information Society's policy, information course was added to curriculum of primary school as "computer course" in 1998 and changed as information technologies course in 2007 [10][11]. In spite of that policy and changes in the curriculum, universities, as the important educational establishments, have very inadequate number of children universities or children science centers. When the number of universities is considered, it can be expressed that their contribution to children's informatics education is not sufficient. There should be generally common design and planning for informatics education in all children universities and children science centers. The interoperability issues should be handled such as training duration and learning outcomes.

References

1. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman B., Kafai, Y.: Scratch: Programming for All. Communucation of the ACM, 52, 11 (2009)
2. Gülseçen, S., Özdemir, S., Gezer, M., Akadal, E., Özen, Z.: The Good Reader of Digital World, Digital Natives: Are They Good Writer of that World? 6th World Conference on Educational Sciences, Malta (2014) (in pubslishing process)
3. Maslen, G.: EUROPE: New Network of Children's Universities (2008). http://www.universityworldnews.com/article.php?story=20080731155222429 (last checked 20.05 2014)
4. Ankara Üniversitesi Çocuk Bilim Merkezi: Amaçlar. (2009) http://cocukuniversitesi.ankara.edu.tr/?page_id=547. [retrieved 10.05.2014]
5. YÖK: Üniversitelerimiz (2014): http://www.yok.gov.tr/web/guest/universitelerimiz (last checked 07.04.2014)
6. Aydın Çocuk Üniversitesi: Hakkımızda. (2013) http://cocukuniversitesi.aydin.edu.tr/index.asp?id=2 (last checked 04.05.2014)
7. İstanbul Üniversitesi Çocuk Üniversitesi: Kuruluş ve Amaçlar. (2010) [Online] Available: http://cocukuniversitesi.istanbul.edu.tr/?p=6068 (last checked 11.05.2014).
8. Günay, D., Günay, A.: 1933'den Günümüze Türk Yükseköğretiminde Niceliksel Gelişmeler. Journal of Higher Education and Science, vol 1 pp. 001-022 (2011)
9. Sadonichy, V. A.: Informatics and Teaching It at School. Programming and Computer Software, 6, 273-278 (2011)
10. MEB Talim ve Terbiye Kurulu Başkanlığı: Ortaokul ve İmam hatip ortaokulu Bilişim Teknolojileri ve Yazılım Dersi (5,6,7 ve 8. Sınıflar) Öğretim Programı. MEB, Ankara (2012)
11. MEB Talim ve Terbiye Kurulu Başkanlığı: İlköğretim Seçmeli Bilgisayar (1-8. Sınıflar) Dersi Öğretim Programı: Ankara: MEB (2006)
12. Gein, A. G.: Informatics in Schools: Problems of Content. Programming and Computer Software,37, 284-287 (2011)
13. Children's Universty of Manchester: About the Children's University. (2012) http://www.childrensuniversity.manchester.ac.uk/about/ (last checked 03.05.2014).

Analysis of the Quality of Teaching Computer Science and ICT

Gaisina Svetlana Valer'evna

State budget institution of additional professional education (training) specialists St. Petersburg Academy of Postgraduate Pedagogical Education, Russia, St. Petersburg
s_v@bk.ru

The results of studies of quality of teaching computer science at the primary school in St. Petersburg, assesses the qualifications of teachers of computer science.

Teacher - the main subject of management of cognitive activity of students, so the quality of teaching is primarily associated with his professionalism. Everything is important here - skill level, attitude in the activities and understanding of the occurring changes, a positive attitude to seek a new and of course the availability of necessary professionally significant personality traits teacher.

The study was conducted in St. Petersburg in 2011-2012 academic years. Pedagogical community of teacher's informatics amounted to 1,881 persons at the beginning of the study. In the study, were surveyed 554 teachers who teach computer science and ICT in the schools of St. Petersburg, it is - 30% of the total number of teachers in this category. Consequently, this sample can be considered representative.

The survey showed that all teachers have a university degree, (99%, 546 teachers) or incomplete higher education. Higher pedagogical education have 49% of respondents, higher technical -268 teachers (48 %) are enrolled in graduate school - 16 teachers (3 %). The average teaching experience of work as a teacher of computer science is 10.8 years.

During the last 5 years, 505 teachers (91%) have undergone training at professional development. Improve their skills and teachers informatics through participation in conferences, scientific workshops and festivals at various levels (43%, 237 teachers). Content analysis of the results showed that in pedagogical environment there is demand for allowances, lesson plans using innovative educational technologies, innovative teaching aids. The results of analysis of the quality of teaching computer science and ICT based on these data are presented in Table 1.

Table 1. Quality of teaching computer science and ICT (sample - 465 teachers)

Key characteristics	**Data of the survey**	**Index**
Professional competence of teachers in the field of system knowledge (specialization in informatics and ICT)	Basic education (number of teachers) higher professional (science teacher) -123 ; Higher Technical - 268; secondary technical 8	98%
Professional competence of teachers in the system of pedagogical knowledge	Basic higher pedagogical education (number of teachers) science teacher -137; teacher of other disciplines – 131	49%
Professional experience of teachers	Number of teachers who have experience as a teacher of computer science for over 5 years – 410	74,13%
Increasing scientific and innovative potential teachers	retraining - 3; training in the system of advanced training of teachers – 505	91%
Values and motivation of teachers to ensure a high quality of teaching (orientation in activities at socially significant problems, the desire to provide personal development of each pupil; aspirations for self-realization, improvement of own activities)	participation in conferences, scientific workshops and festivals at various levels – 237 organization Involvement of Pupils in scientific conferences, competitions and festivals at various levels – 282	94%

Content analysis of the results showed that in pedagogical environment there is demand for allowances, lesson plans using innovative educational technologies, innovative teaching aids.

During lessons are not used materials of only of one educational and methodical complex, 56% of teachers use in the teaching of Informatics additional textbooks. In addition to the selected educational complex teachers use approved and recommended by the Ministry of Education textbooks and teaching aids of other authors, and also textbooks on logic and Programming. According to teachers, these materials are not enough, and teachers Informatics create their own electronic educational resources for all stages of learning activities.

Training refresher courses, participation in a conferences and seminars, proves that the teachers is interested in increasing their professional competence. Informatics teachers have a positive attitude to the introduction and dissemination of scientific achievements, innovative development and effective's experience. Teachers are aware of the role of proper education in modern dynamically changing of life of society.

References

1. Bordovskih, GA., Nesterov, AA., Trapitsyn, SY: Quality control educational process. St.Petersburg.: Publ RSPU. AIGertsena. 359 (2000)

Data Management: More Than a Matter of CS

Andreas Grillenberger and Ralf Romeike

Friedrich–Alexander–Universität Erlangen–Nürnberg (FAU)
Department of Computer Science, Computing Education Research Group
{andreas.grillenberger,ralf.romeike}@fau.de

1 Big Data and Data Management

Data management is one of the most exciting and challenging topics in computer science. Nowadays, handling large amounts of varying data in short processing times, which is Big Data, is intensively discussed not only in CS; it has also a tremendous influence on daily life and society, as handling data and dealing with the new chances and threats increases in relevance for everyone. However, data management is only a marginal topic in current education. Hence, we will outline the importance of this topic for school by presenting three aspects mainly affected by the current developments in this field: the paradigm change in data management, the influences of this field on everyday life and using it as a tool.

2 A new Paradigm in Data Management

Data management concerns with storing, managing and analyzing data, often using database management systems (DBMS). As today the requirements on such systems are clearly changing, new types of DBMS, summarized as NoSQL databases ("not only SQL" [1]), evolved. Hence, common concepts of DBMS are challenged, e. g. the relevance of redundancy: In RDBMS, preventing redundancies is a main aim, while NoSQL databases use redundancy intentionally for accelerating access to distributed stored data. The decision whether to handle or to prevent redundancy has to be made from case to case. Therefore, various publications state an ongoing paradigm change in data management (cf. [4]): in the future a great variety of DBMS can be expected.

3 Data Management in Daily Life

Today, people not only use but also generate and manage large amounts of data every day. These data need to be handled in a proper way, depending on their value. While private data need to be protected from unauthorized access, for example by encryption, in other cases protecting the device on which the data is stored is sufficient. Other requirements are coming from the large amount of data-driven devices and applications everyone uses: For example, the task of synchronizing data between devices includes several important aspects of data management like handling redundancy and inconsistencies. Therefore, the user is often confronted with phenomena like duplicated contacts, accidentally reverted changes or other synchronization conflicts that have to be resolved by hand. Summarizing, data management comprises various decisions everybody needs to make: where to store data, how to protect them, how to deal with threats for data privacy, and so on. As these tasks are strongly related to everyday life, data management becomes increasingly relevant for everyone.

4 Using Data Management as a Tool

The amount of publicly available data ("Open Data") increases continuously. By analyzing such data, various information can be derived that are relevant for understanding decisions, political topics, and so on. These methods for analyzing data are not only relevant in CS and in daily life, but also in other fields and sciences. Not only the data, but also more and more easy-to-use tools for analyzing and visualizing them are made available. In addition, there are indicators that people want to understand such analyses, e. g. not only weather forecasts are provided as smartphone apps but also the satellite images used for deriving these forecasts. Today, when moving to a foreign city, one cannot only read others views on the possible neighborhood, but can in detail analyze data, provided for example by the city administration: e. g. New York City offers data on the calls to the service number 311 at their open data portal. When aggregating these by borough and mapping them, it is easy to discover boroughs with for example bad street conditions or high noise. As in the digital life more and more decisions base on the results of data analysis, this is an important method and competency. This relevance is not only limited to CS: as data analysis becomes increasingly important, even a new profession is rising—the "data scientist".

5 Conclusions

Within the last years, data management clearly changed: not only does handling data follow a new paradigm, also its influences on everyday life become visible. In addition, it is used as a tool in other fields. These three aspects entail numerous new requirements for CS education (cf. [2]). By emphasizing data management at school, a main aim of general education can be reached: fostering knowledge, skills and competencies strongly related to everyone's daily life (cf. [3]).

References

1. Edlich, S., Friedland, A., Hampe, J., Brauer, B., Brückner, M.: NoSQL [in German]. Hanser, Carl Gmbh + Co. (2011)
2. Grillenberger, A., Romeike, R.: Big Data - Challenges for Computer Science Education. In: Proceedings of ISSEP 2014 (2014)
3. Grillenberger, A., Romeike, R.: Teaching Data Management: Key Competencies and Opportunities [in print]. In: Proceedings of KEYCIT 2014 (2014)
4. Mayer-Schönberger, V., Cukier, K.: Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt (2013)

Boardcasting: Web-Development Mobile Teaching and Learning

Ilia Gossoudarev

Russian Herzen State Pedagogical University, Russia
gossoudarev@herzen.spb.ru

1 The Mobile Learning Environment

The learning process in a university is focused on students' personal development and interaction, guided by the teachers – lecturers, professors etc. The up-to-date way to provide a proper environment for all the participants of the process where they could interact and develop is to create what we could call a mobile environment. This environment has to store educational information, be accessible anywhere regardless of time or geolocation, and allow sharing stuff with other students and teachers.

Nowadays teachers may choose from a great variety of tools to create such an environment. Most of them have been using blogs and sites recently. But in this case the interaction is not really full-duplex. A teacher provides information on their site or blog and students comment on the posts, or vice versa. But true collaboration means that both students and teachers equally donate to the final result of the research project. This in effect implements the initial proposal by Tim Berners Lee – the Web coedited by all.

And this is what is achieved with the aid of cloud tools. We use a cloud document like Google Spreadsheet, a wiki page or site, a screencast or a video tutorial and some specific subject-related tools, depending on what we teach. For instance, let's consider number systems – base 10, base 2 etc. We can construct a mobile environment out of a reliable educational wikisite, like http://tibasicdev.wikidot.com/binandhex, a screencast, a Google spreadsheet with some formulas and instructions.

A cloud assignment is a wikipage or a cloud document which is shareable with learners. A teacher may create a document with public access but not editable, then students copy it to their environments (Google Drive, dropbox etc) and create their own versions based upon this primary document, to share it with the instructor eventially. Basing upon [2] study of dialogue in distance education and the works of [1] we can implement the idea of a dialogue as a method of collaborative code development involving the teacher and the students.

2 Boardcasting

A boardcast (https://en.wikipedia.org/wiki/Boardcast) is a cloud document which is being edited online in real time mode. It may or may not allow for the audience to edit or comment on it. I teach web programing on Javascript using an online cloud code editor created by myself, kodaktor.info. It can host “boards” like http://kodaktor.info/cc42acc with assignments and instructions (while http://kodaktor.info/cc42acc_1 contains a solution to it). Boards can be re-edited by an educator and created by everyone. You can find more information about it at http://kodaktor.info or watch a screencast at http://youtu.be/Csu80pZHidY

In Russia students often tend to search for optional sources of education information and that’s why academic mobility is so important for them. Mobile learning environment based on cloud tools offers great opportunities for such type of learning. It makes learning independent from a specific place / location, provides ways of sharing and collaboration, is suitable for mobile / portable devices and also creates a platform for continuous / lifelong learning because it follows and individual throughout his progress in space and time.

References

1. Moore, M. G.: Theory of Transactional Distance. In D. Keegan (Ed.), Theoretical principles of distance education, 1, 22-38. New York: Routledge (1993)
2. Shearer, R. L.: Transactional Distance and Dialogue: An Exploratory Study to Refine the Theoretical Construct of Dialogue in Online Learning (Unpublished Dissertation). The Pennsylvania State University, University Park (2009)

Which One Leads The Quality in Education: Technology by Mechanicity or Methodology by Humanity

Cem Turan

İstanbul University, Department of Informatics, İstanbul, Turkey
turancem@windowslive.com

1 Summary

In general, technology means truthiness, high speed and big domains wherever it is used. It seems fine and that is why to be loved and accepted easily by the crowds. More speed, tones of information may be a reason for making more profit but is every area of the life suitable for gaining speed and to be digitized as much as possible? Theoretically, the public decided to use the technology thus it would make our lives easier and free our time in order to spend for social relationships. Unfortunately, the marks related with the results of using uncontrolled technology shows us which some expectations about the using of technology are not True-Positive: We hoped but it would not.

Especially, in last few decades, technology dramatically occupied education and redesigned the methods of teaching. Technology has been still leading a part of education strategies globally without spending time enough for thinking on density of uncontrolled and socially unplanned technology to shape young brains.

Of course, I do not mean to say not to use technology at all, as a representative of computer science which is one of the pioneers of technology concept but try to point possible harmfulness of unprepared usages.

2 Perception of Informatics in the Human

Education is not only a series of efforts on transporting knowledge from one person to others mechanically but also it uses some human-specific behaviors such as mimics, feelings, level of voices etc. Teaching is not only an occupation which makes students memorize whatever they should, statically. As a result of this reality, teachers should be the main part of education processes and should not assign this role to the products of technology [2].

We took a poll about the perception of using technology in classroom and preferred ways of learning among various age and social groups. Majority of answers confirm our estimations about the right roles of technology in education [4]. Almost all attenders of the survey do not share with the thought of "having a robot-teacher" and believe that education is an "emotion based activity". Pure technology usage in education may cause losing some brain capabilities.

3 General Evaluation for Schools

Technology is a tool in teaching in order to offer more understandability and expanded sense for students. As appliers, teachers should be ready for using this tools correctly and effectively but the researches shows us most of the teachers are not ready and / or not interested in knowing true usage of them. In universities, colleges and other schools have been becoming cemeteries of unused hi-tech toys. Some of soft materials on hardware such as multimedia contents or software were designed as if they were used in ordinary environment [1].

A new, innovational, technology-aided but human-aimed approach is needed for non-robotized, aware of emotion (EQ), social responsibility, the aim of education which is not only "what" but also "how", especially "why", identify himself / herself as a part of the nature for healthy, happy and "live" next generations. They must not memorize but really learn with great sensibility for using in the future for the civilization.

4 Discussion: Eduinformatics

Education needs particular sections from specialized informatics forms such as psychoinformatics, neuroinformatics and socioinformatics because the target of education is the brain which has more complexity than we expect and it is not a machine. It uses different memory areas according to the kind of information, environmental conditions, feelings etc. [3]

5 Results

Technology is for mankind and should be directed by the specialists in education-aimed informaticians in so-called "eduinformatics", not vendors or marketing stories of IT sector. Because every product of education systems is a person who should be prepared for the future as a perfect "human" not "machine or automaton". Modernization does not mean having technology but means using technology correctly. We have proofs enough to point the risk.

References

1. Leidner D.E., Jarvenpaa Sirkka L.: The Use of Information Technology to Enhance Management School Education: A theorical View. MIS Quarterly. 19(3) (1995)
2. Harms C.M., Niederhauser D.S., Davis N.E., Roblyer M.D., Gilbert S.B.: Educating Educators for Virtual Schooling: Communicating Roles and Responsibilities. Electronic Journal of Communication, 16(2). Communication Institute for Online Scholarship (2006)
3. Salomon G.: Distributed Cognitions: Psychological and Educational Considerations, Cambridge University Press (1993)

4. Culbertson C., Daugherty M., Merrill C.: Effects of Modular Technology Education on Junior High Students' Achievement Scores, Journal of Technology Education, 16(1), Council of Technology Teacher Education and the International Technology Education Association (2004)

Workshops

Educational Standards for Digital Competence at Lower Secondary Level

Peter Micheuz

Alpen-Adria University Klagenfurt, Institute for Informatics Didactics
Universitätsstraße 65-67, 9020 Klagenfurt, Austria
peter.micheuz@aau.at

Until now, very few countries have been successful in implementing Informatics or Computing at primary and lower secondary level. The spectrum from digital literacy to informatics, particularly as a discipline in its own right, has not really achieved a break-through and seems to be underrepresented for these age-groups in many countries.

The goal of this workshop is not only to discuss the anamnesis and diagnosis of this fragmented field, but to discuss and suggest viable forms of therapy in form of setting educational standards, referring to the holistic European approach in form of the buzzword "Digital Competence", derived from the Digital Agenda.

However, an overview of worldwide endeavors gives hope that Informatics (or Computing) within the broader approach of Digital Competence will play a more significant role in lower secondary education in a foreseeable future. Widely accepted definitions related to Informatics (computing), information and communication technologies, digital literacy and technology enhanced learning, and the acceptance of existing frameworks, competence models, curricula and teaching aids should support this process.

Recently, an increasing number of position papers, frameworks and country reports explicating the wide field of Informatics at schools have been published. These activities should remedy the unacceptable situation of big distortions of computing in education even within countries, regions and schools. Incoherence from country to country, state to state and even from school to school, is not the exception but the norm. Informatics education (standards) varies widely and its picture especially at lower secondary education shows distortions and inconsistencies referring to

- different perceptions of the term Informatics which often serves for every activity with computers,
- formal Informatics education between obligation and freedom of choice within autonomous decisions of schools and regions,
- an antagonistic view on approaches to develop students' digital competence and basic Informatics education, integrated across the disciplines or as a discipline in its own right,
- different structures of reference frameworks in many countries, and
- Different preconditions, cultural backgrounds and requirements worldwide.

Making visible good practices and implementation strategies in some countries and comparing successful approaches are rewarding tasks for this workshop.

Discussing, defining and agreeing common educational standards on a transcontinental level for the age-group of 14 to 15 years old pupils/students in a readable, assessable and acceptable form should keep the participants of this workshop active beyond the limited time at the workshop.

A similar workshop took place at the IFIP conference KEYCIT in Potsdam at the beginning of July 2014. Its tentative results will be a good starting point for a further discussion on this issue.

After a compact and comparing overview of worldwide approaches provided in a comprehensive way by the organizer of the workshop, additional contributions from the participants - who preferably should be aware of the situations in their countries - are appreciated.

Robotic Programming for Teaching Programming Languages

Orçun Madran

Hacettepe University, Department of Information and Document Management, Ankara, Turkey
orcunmadran@gmail.com

Computer programming is a problem solving and production process where different skills are executed simultaneously. Gaining the skills necessary for computer programming is possible in various schools and grades via either compulsory courses or voluntarily courses. Among the possible factors affecting the success in those tranings, students attitudes and perceptions toward the programming, adequacy in computer (or information technologies) literacy and consistency of the selected programming language and goals of the training. One or more of these factors may both affect the success of students in programming and impede the application of those skills in other field. For example, independent from the programming language selected, programming can provide problem solving skills like logical thinking and algorithm construction, and analytic thinking abilities.

In this workshop, in order to provide programming skills, to make learning programming language easy, to raise motivation and increase the success of students, we will focus on robotic programming techniques.

Workshop language is Turkish. 20 people maximum can attend this workshop. Attendees may use their notebooks and group 4-5 people together to do some basic examples and sample applications.

Total workshop time is approximately 2 hours (1 hour theoretical, 1 hour practice).

Informatics Education at the Crossroads: Round Table on the Dutch Case

Nataša Grgurina[1], Erik Barendsen[2]

[1]University of Groningen
PO Box 800, 9700 AV Groningen The Netherlands
n.grgurina@rug.nl
[2]Radboud University Nijmegen and Open Universiteit
PO Box 9010, 6500 GL Nijmegen The Netherlands
e.barendsen@cs.ru.nl

There is a strong consensus among IT and computing professionals from industry, research and development, and education, that computing education of the general public is essential for society. In many countries, however, computing education is inadequate and out of date. This alarming situation has been described in various reports [2, 5]).

In the Netherlands, the present situation is precarious as well [7]. Informatics has been introduced as an elective subject in the 10th and higher grades of pre-vocational and senior general secondary education in 1998 [4] and has not received much attention since. A recent report by The Royal Netherlands Academy of Arts and Sciences [1] summarizes the problems with current informatics education and suggests a redesign of the informatics curriculum and introduction of a new compulsory digital literacy subjects in lower grades of secondary education.

This report has led to two initiatives. First, a research by the Netherlands Institute for Curriculum Development among the practicing Informatics teachers about their enacted curriculum and their ideas about a desirable Informatics curriculum and its implementation. This research resulted in a report for the ministry of education containing a number of recommendations for the future development of the Informatics education in higher grades of secondary education [6]. The second initiative is the workshop Computing in Secondary Education at the Lorentz Center in September 2014. During this week long workshop, leading national and international experts discuss the curriculum and (re)design issues concerning Informatics education in general and in the Netherlands in particular.

In the ISSEP workshop we will report on the outcomes of the two initiatives, ask the participant for feedback and discuss the national and international implications.

References

1. Digitale geletterdheid in het voortgezet onderwijsAmsterdam: Koninklijke Nederlandse Akademie van Wetenschappen (2012)
2. Furber, S. (Ed.). Shut down or restart? The way forward for computing in UK schools. London: The Royal Society (2012)

3. Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., et al.: Informatics education: Europe cannot afford to miss the boat Informatics Europe & ACM Europe (2013)
4. Grgurina, N., & Tolboom, J.: The first decade of informatics in dutch high schools. Informatics in Education, *7*(1), 55-74 (2008)
5. Kaczmarczyk, L., & Dopplick, R.: Rebooting then Pathway to Success; Preparing Students for Computing Workforce Needs in the United States. New York: Association for Computing Machinery, Education Policy Committee (2014).
6. Tolboom, J., Krüger, J., Grgurina, N.: Informatica in de bovenbouw havo/vwo, Naaraantrekkelijk en actueel onderwijs in informatica, SLO, Enschede (2014)
7. van Diepen, N., Perrenet, J., & Zwaneveld, B.: Which way with informatics in high schools in the Netherlands? the dutch dilemma. Informatics in Education, 10(1), 123-148 (2011)

Discovering Python

Michael Weigend

Institut für Didaktik der Mathematik und der Informatik,
Fliednerstr. 21, 48149 Münster, Germany
`michael.weigend@uni-muenster.de`

Python is a powerful object-oriented programming language that is free and open, runs on every platform and is easy to learn. Python programs are short and comprehensible. Many schools and universities teach Python in introductory and advanced programming courses.

This workshop consists of presentations, discussions and hands-on programming activities. It starts with an introduction into Python's main language features:

- Meaningful layout of source code
- Duck typing
- Comprehensible expressions (like `0<x<10`)
- Type hierarchy
- Functions with different types of parameters
- Object oriented programming: classes and instances
- Polymorphism
- Modules
- Preventing and finding errors: assertions and tests

The first hands-on activity focuses on exploring basic Python concepts and modeling with lists, sets and dictionaries.

The second presentation discusses example projects that might be interesting for computer science education at schools. Since Python programs are small, it is easier to implement software that is meaningful and related to relevant contexts: editors, image processors, web services, simulations. Python is one of the programming languages, which run on the Raspberry Pi (RPi). The RPi is a very small and inexpensive computer, which is used a lot in science related projects at schools and universities. It is rather easy to connect sensors (like digital thermometers) to the RPi's general input/output device (GPIO) and process the data in a Python grogram.

In the second programming activity, the participants develop a text editor with a graphical user interface and special features, starting with a given prototype and adopting agile methodology (Extreme Programming). The idea is to create a few stories (in these case features, estimate the development time and implement some of the stories in iterations. The advantage of this approach is that at the end of each iteration there is a program that actual works. Thus, in the classroom students can work and learn with different speeds and still have finished their projects, when time is up.

Author Index

Ankara University Press
http://basimevi.ankara.edu.tr
ISBN: 978-605-136-171-0
e-ISBN: 978-605-136-172-7