Daniela Bezáková, Ivan Kalaš (Eds.)

ISSEP 2011

Proceedings of Selected Papers

5th International Conference on
Informatics in Schools: Situation, Evolution and Perspectives

Bratislava, Slovakia, October 26 – 29, 2011

Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

ISSEP 2011

Informatics in Schools: Situation, Evolution and Perspectives

Proceedings of 5th International Conference ISSEP 2011 – Informatics in Schools: Contributing to 21st Century Education. 26 – 29 October, 2011
Bratislava, Slovakia

Conference organized by

Association of the Infovek Project
and
Faculty of Mathematics, Physics and Informatics
Comenius University,
Bratislava, Slovakia

Edited by: Daniela Bezáková and Ivan Kalaš
Production: Daniela Bezáková and Andrea Hrušecká
Graphic design: Martina Kabátová

All papers have been reviewed by the International Review Committee

Published by	Library and Publishing Centre, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava
First published	2011
Number of pages	56
Printed in Slovakia by	Bratia Sabovci, s..r.o., Pribinova 8183/170, 960 07 Zvolen
ISBN	**978-80-89186-90-7**

ISSEP 2011
Informatics in Schools: Situation, Evolution and Perspectives

Conference website

www.issep2011.org

Conference Chair

Ivan Kalaš, Slovakia

International Programme Committee

Valentina Dagiene, Lithuania
David Ginat, Israel
Gerald Futschek, Austria
Juraj Hromkovič, Switzerland
Ivan Kalaš (chair), Slovakia
Peter Micheuz, Austria
Roland Mittermeir (co-chair), Austria
Sigrid Schubert, Germany
Maciej M. Sysło, Poland
Tom Verhoeff, The Netherlands

Printed with the kind support of the Austrian Ministry for Education, Arts and Culture

bm:uk Bundesministerium für Unterricht, Kunst und Kultur

ISSEP 2011

Informatics in Schools: Situation, Evolution and Perspectives

Local Organizing Committee

Ivan Kalaš
Daniela Bezáková
Andrea Hrušecká
Roman Hrušecký
Martina Kabátová
Katarína Kalašová
Zuzana Kubincová
Roman Riška
Mário Varga
Miroslav Wagner

Conference organized under the patronage of the City Mayor of Bratislava Milan Ft§ļ nik

Sponsors and Co-organizers of the Conference

Contents

Workshops

Foreword

We often hear people talking about how young Informatics or Computer Science is and how quickly it develops. I believe that it is exactly this conference – already 5th in the row – which shows something else: if there is anything that quickly develops in the field of Informatics, it is its didactics or pedagogy[1]. That is the reason why I was tempted to title my foreword *Transitions* (inspired by the Proceedings of Constructionism 2010[2]) or even more ambitiously: *Reaching Maturity*.

Finally I resisted those temptations because in our field of Informatics education we don't have so far sound and generally accepted criteria to verify what really is correct and mature in the pedagogy of Informatics. Nevertheless, what I can observe with no uncertainty is the following. Although the ISSEP meetings occur rather often – with an exotic frequency of approximately 18 months – every following one is interesting, inspiring, surprising, too far from boring... In a year and a half so much happens in the Informatics education! What exactly do I mean close to opening the 5th conference? Based on the papers accepted for Bratislava meeting I want to mention four different aspects.

1. Obviously, we are systematically improving our thinking about the learning goals and objectives of Informatics in school, about how to implement them and evaluate – we have learned a lot about pedagogical reflection within our subject.
2. The interest in educational research in our field is ever growing and so is the number of the doctoral students and smaller or bigger research projects. We are less guessing and more observing and analyzing! We are adopting new research designs. In particular, I am referring to increasing number of remarkable projects that apply new research strategy named design based research, see e.g. (Cobb et al., 2003)[3], (diSessa, 1991)[4] or (diSessa and Cobb, 2004)[5]. We are referring to a strategy (close to action research) in which researchers – often supported by practitioners from the real settings – tend to create more exact theories of learning, through designing, creating, implementing, observing and iteratively redesigning theoretically justified pedagogical interventions into real classes.
3. We increasingly ponder about how school Informatics can better meet the needs of so called *21st century learners*, with new competencies requested, new priorities and values, new forms of organizing of the teaching-learning processes, new relations between learners and the teacher, with new roles of the teacher... To put it simple, we are thinking about the role of Informatics in the *modern school*.
4. We are more and more thinking about when the school Informatics (whatever it is called) should start. Based on the conference papers we can conclude that several countries have started the implementation of Informatics into the lower secondary or even primary education... or at least they are seriously thinking to do so. To certain extent we had expected such transition and thus together with the Zurich 4th ISSEP Programme Committee we agreed to redefine the ISSEP acronym from *Informatics in Secondary*

---

[1] with all the teensy-weensy divergences these words may have in different languages

[2] Clayson, J. E., Kalaš, I.: Constructionist approaches to creative learning, thinking and education: Lessons for the 21st century. Proc. of International Conference Constructionism 2010, Paris. Published by Comenius University, Bratislava. ISBN 978-80-89186-65-5

[3] Cobb, P., Confrey, J., diSessa, A., Lehrer, R. and Schaube, L.: Design Experiments in Education Research. The Educational Researcher, 32 (1), pp. 9-13, 2003

[4] diSessa, A. A.: Local sciences: Viewing the design of human-computer systems as cognitive science. In J. M. Carroll (Ed.): Designing Interaction: Psychology at the Human-Computer Interface. NY: Cambridge University Press, pp. 162-202, 1991

[5] diSessa, A. A., Cobb, P.: Ontological innovation and the role of theory in design experiments. Journal of the Learning Sciences, 13(1), pp. 77-103, 2004

Schools: Evolution and Perspectives to *Informatics in Schools: Situation, Evolution and Perspectives*, so that we open the door to all submissions which study Informatics in other than (upper) secondary schools.

The last of these aspects or observations fills me with satisfaction and excitement: Finally, we can think about school Informatics as a compact process going through all stages of education. Doubtless to say, its curriculum and policy of implementation will keep changing frequently. However, an opportunity for developing a complex Informatics education strategy has emerged in some countries. This is a real challenge!

Eight months ago in our Call for papers for the 5th ISSEP conference we suggested that the conference will be a... *scientific probe into Informatics as a formative part of the general education – in different forms and at different stages of school, i.e. in primary and secondary education, in pre-service and in-service education of the Informatics teachers, as well as in many supporting activities with clear links to Informatics.*

We also declared that the conference will... *reflect upon educational goals and objectives of Informatics as a subject, its curricula and various teaching/learning paradigms, programming, programming languages and pedagogy of programming, teaching/learning materials, various forms of assessment, evaluation and testing, traditional and innovative educational research designs, issues of safety and threats, Informatics' contribution to new education and the development of the 21st century competencies, competitions, displays, projects and other supporting activities, new forms and formats of interactions, social, cultural and ethical issues emerging from and within Informatics education, class management, Informatics and special education needs, Informatics and integrating digital technologies in other subjects, Informatics and the development of complex digital literacy and other related issues.*

The answer and result to our call is what the readers have today in their hands – as a result of rather demanding and challenging reviewing and editing process. We received unexpectedly high number of submissions. Out of them the team of 39 reviewers accepted 20 for the LNCS volume of *Informatics in Schools* published by Springer[6] and 31 for this Proceeding published by Comenius University. In total we accepted approximately 70% of all submissions. Besides that, we also accepted 7 workshop proposals, which we believe will attract participant's interest. Brief annotations of the workshops are included in this Proceeding.

Finally, I want to thank all authors and presenters of the keynotes, all papers and workshops. I want to thank all who have contributed in any way to the quality of ISSEP 2011, including the paper reviewers who altogether produced more than 200 reviews. Due to their effort we have managed to prepare high quality contents of the conference.

I also want to thank all members of the Programme Committee and Organizing Committee who helped to make this event possible, all co-organizers and sponsors for supporting the conference, our Faculty and University top management for unhesitating support and encouragement.

I am happy for this conference and I am already looking forward for the next one in 2013, for the next event labelled by increasingly recognized logo of ISSEP. If I managed to learn a lot while preparing and implementing ISSEP 2011, I hope I will learn even more at ISSEP 2013.

Ivan Kalaš

[6] Kalaš, I., Mittermeir, R. T. (Eds.): Informatics in Schools. Contributing to 21st Century Education. LNCS 7013, Springer, 2011. 235 p. ISBN 978-3-642-24721-7

On Competence-Orientation and Learning Informatics

Peter K. Antonitsch

Alpen-Adria University Klagenfurt
Institute of Informatics Systems, Informatics Didactics
Universitätsstr. 65 – 67
Klagenfurt/AUSTRIA
Peter.Antonitsch@uni-klu.ac.at

Abstract. “Competence” has become a catchphrase of modern pedagogy and didactics. This article relates competence-orientation to known concepts like outcomes-based education or self-organized learning, and points at tools like competence-matrices or task-checklists designed to support competence-oriented learning. Furthermore, it sketches an attempt of the author to introduce these tools into Informatics lessons at upper secondary level, describes first experiences and outlines questions that still have to be answered.

Key Words: Competence-orientation, personalization, programming

1 Competence-Orientation and Personalization

Competence denotes a combination of knowledge and the ability to apply that knowledge in order to solve problems or to react in an appropriate way [1]. Competence-based curricula do not (only) point at what the learners *should learn* in school but tell what learners *must be able to do* at the end of a certain learning process. For instance, the forthcoming curricula for learning “Informatics” at Austrian engineering schools (upper secondary level, age 15 to 19) read like: “The students can install and configure operating systems.” (from the area of competence labeled “Basics of Electronic Data Processing”) or: “The students can develop algorithms and describe the necessary steps of computation in a systematic way.” (from the area of competence labeled “Programming”).

Competence-based curricula resemble the framework of outcomes-based education. This is “an approach to planning, delivering and evaluating instruction that requires [...] teachers and students to focus their attention and efforts on the desired results of education—results that are expressed in terms of individual student learning.” [2]. Competence-orientation supplements competence-based curricula with a methodical framework resembling the concepts of self organized learning (see [3], for instance).: Learners should be enabled to follow an individual learning path, thus transforming predefined common and compulsory learning goals into individual learning goals. [4]. This is called personalized learning or – short – personalization.

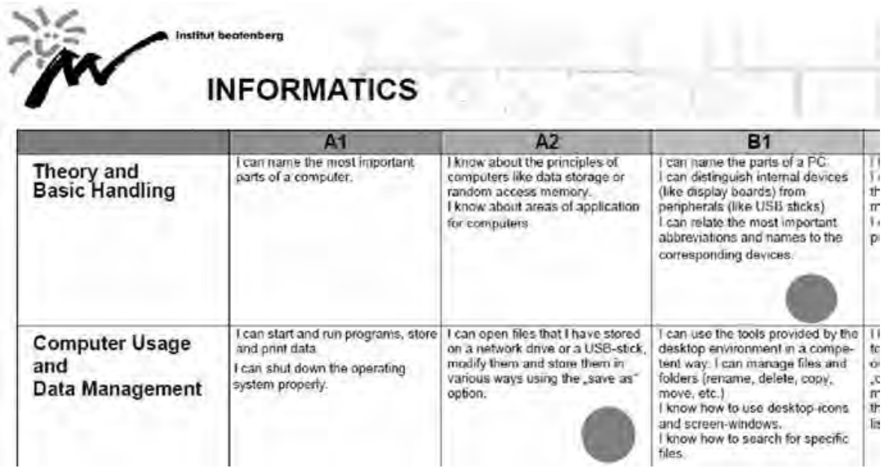
Obviously, competence-orientation and personalization of learning processes go together

2 Competence-Orientation in Action

2.1 A general framework

When looking for ways how to apply competence-orientation to learning processes we can learn from schools that have tried alternative concepts for quite a time and provide tools that are needed to support competence-oriented learning ([5], [6]).

With competence-orientation, learners are supposed to plan their own learning process. Therefore, they need information about the learning objectives, about ways to reach their goals and about means to test whether they have succeeded or not, or rather: to what extent they have succeeded to reach the goals. While advance organizers provide a survey of the learning goals, a *competence-matrix* gives more detailed information by pairing goals/competences and different levels of competence - acquisition: Even if a learner can not reach the highest level of a specific competence, he/she should be enabled to acquire that competence to a certain extent:


Institut beatenberg

INFORMATICS

	A1	A2	B1
Theory and Basic Handling	I can name the most important parts of a computer.	I know about the principles of computers like data storage or random access memory. I know about areas of application for computers	I can name the parts of a PC. I can distinguish internal devices (like display boards) from peripherals (like USB sticks) I can relate the most important abbreviations and names to the corresponding devices.
Computer Usage and Data Management	I can start and run programs, store and print data. I can shut down the operating system properly.	I can open files that I have stored on a network drive or a USB-stick, modify them and store them in various ways using the „save as" option.	I can use the tools provided by the desktop environment in a competent way. I can manage files and folders (rename, delete, copy, move, etc.) I know how to use desktop-icons and screen-windows. I know how to search for specific files.

Fig. 1. Example competence-matrix (snippet) for the subject matter Informatics (age 11 to 14): The rows name topics of interest/learning objectives, while the columns specify different levels of competence for a certain topic. Each "cell" contains a description what I (!) must be able to do to prove that (level of) competence (source: [7]). The circles visualize a learner's individual competence profile.

Furthermore, a competence matrix can be used by the learner to mark the levels of the listed competences he or she has already reached. This results in an individual competence profile helping to decide on further learning steps to improve the learner's competences.

Learning tasks are the primary means to acquire competences. Combining a *checklist* of *learning tasks* with the *competence-matrix* guides the learners to choose tasks appropriate to reach a certain level of competence and to test whether they succeeded as well.

A task shall be called *learning task*, if it points at a gap between an actual situation and a different situation desired by the learner. If, furthermore, a learning task gives an idea how to fill that gap, it becomes a task learners are willing to accept and able to solve. Solving a new task (i.e. “filling the gap”) due to prior knowledge or skills is commonly called “learning” [8]. Therefore, to support systematic acquisition of competence, learning tasks have to provide a situation that motivates to apply knowledge and/or skills [9].

2.2 A Personal Approach: The Tool, Sub-Competences and Tasks

The author decided to switch to competence-orientation in autumn 2010 with two first-year Informatics courses at an Austrian engineering school. Each course consisted of 17 learners at the age from 15 to 16, and was based on the old, topic-oriented curriculum[1], which lists the learning objectives “basic knowledge about informatics systems” (focusing on hardware and operating systems,) “standard-software” (focusing on text-processing and spreadsheets), and “principles of programming” (focusing on solving problems with simple algorithms and coding of simple algorithms).

To keep things simple at first, the author restricted competence-orientation to “principles of programming” and defined the corresponding learning objective: “The learners can solve meaningful tasks by utilizing control- and data-structures that are available within the specific programming environment in use.”, which instantly raised two questions: “Which programming environment is the best learning environment?”, and, as learning to program is considered “notoriously difficult” [10]: “How can that big learning objective be split into competences of different levels?“

Prior experiences with microworld-environments [11, 12], made “Greenfoot” the author’s favourite software[2]. But due to what might be called “school-culture” the software that had to be used was C# Express, which, in its pure form, is no learning environment providing “situations that motivate to apply knowledge or skills” for programming novices. But C# allows for project-templates that can be used to prepare programming-scenarios with a predefined set of methods inside the “big C#-world”. This feature was used by the author to create a turtle-like scenario called “graphic-robot” with basic movement-commands like »forward(distance)« or »turn(angle)« to start programming with a visual representation of the coded solution. A second

[1] The new, competence-oriented curriculum that was mendioned before will come into effect in autumn 2011.

[2] With Greenfoot, the immediate feedback due to the visible “reaction” of the programmed objects triggers personalized learning processes and the option to use different “scenarios” helps to provide tasls that are meaningful inside that certain scenario. Furthermore, the need to write textual code from the beginning necessitates to build mental models of the used “programming structures” [12].

scenario provided simplified graphical user interfaces for input and output to pay tribute to the "simple algorithms". The third scenario was demanded by the learners, who, when programming the "graphic robot", said those tasks were "nice" but that it would be even more fun programming a "real robot". This gave birth to the idea to use a Java-environment to program Lego-Mindstorms robots at the end of the course.

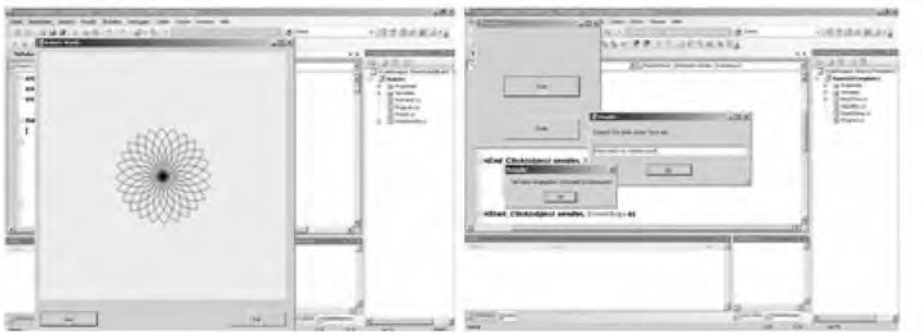


Fig. 2. C# learning environments ("scenarios): "Graphic robot" (left) and "Simple GUI" (right)

The choice of these three learning environments influenced the definition of six competences, dealing with:

coding programs to navigate the graphic robot inside the "graphic-robot scenario":

– I can navigate the graphic robot by means of loops and methods.
– I can navigate the graphic robot by means of variables, loops and methods.
– I can navigate the graphic robot by means of branching, of methods with and without return values, of variables and loops.

writing, reading and understanding code of programs that input, process and output data inside the "graphical user interface-scenario".

– I can input and output data, code simple calculations with command-sequences and branches and use flowcharts as another representation of programs.
– I can structure programs by using loops or branches to fulfill a condition and by splitting the code into several methods.

utilizing classes and objects inside the "graphical user interface-scenario" or the "Mindstorms robot-scenario".

– I can explain the terms "class" and "object", create "object-variables" and I am well aware when I use methods of objects or classes. Furthermore, I can utilize means of help to find out which methods are provided by a specific class.

Each of these competences has four competence levels that were presented to the learners in a table containing links to corresponding tasks as well (see Fig. 3).

To prevent a learner from falling too far behind when he or she chooses to be satisfied by reaching the lowest competence-level D, competences referring to the same scenario partly build upon each other: For instance, tasks corresponding to competence-level D of the second competence resemble tasks corresponding to competence-level C or B of the first sub competence. Therefore, when learners do not go beyond a low competence-level of an "early" competence, they will most likely get stuck at the following competence and are supposed to go back (or seek the help of their classmates).

…to train and check the 2nd competence:
„I can direct the »graphic robot« by means of variables, loops and methods."

level of competence			
D	**C**	**B**	**A**
I can understand and adapt code that directs the graphic robot by means of loops and methods (repetition of the 1st competence).	I can use loops and methods to make the graphic robot move (repetition of the 1st competence).	I can use loops, methods and position variables to make the graphic robot move.	I can program a sequence of graphic robot movements by means of loops, methods and variables.
I can solve tasks 1 to 4 of task-sheet 2.	I can solve tasks 5 to 8 of task-sheet 2.	I can solve tasks 9 to 12 of task-sheet 2.	I can solve tasks 13 to 16 of task-sheet 2.

Task 6

a) Use the method `private void semicircle(int angle)` to program a method `private void U(int angle) {...}` that gets the graphic robot to draw a „U".

b) Direct the graphic robot so that it draws a "U-blossom" (i.e. a blossom the petals of which resemble the letter „U"). <u>*Hint:*</u> *Use the method* `private void U(int angle)` *and a loop.*

Fig. 3. Above: The 2nd competence with its four competence-levels labeled D to A, A denoting the highest (sub-) competence-level. The last row contains links to tasks on the corresponding 2nd task-sheet.
Below: An example-task from task-sheet 2.

This "iterative structure" inside a certain scenario is augmented by another structure spanning the three scenarios:

When solving tasks inside the "graphic-robot scenario", the learner acquires competences to deal with loops, with methods, with variables and with branches.

Switching to the "GUI-scenario" restarts this learning process: The learner acquires (or applies) (competences to deal with loops, with methods, with variables and with branches, extended by the possibility to input and output data and by the learner's awareness of classes and objects.

At last, when programming Mindstorms-robots, the learner acquires (or applies) competences to deal with loops, with methods, with variables and with branches, and with classes and objects that are omnipresent inside the Java-Mindstorms programming environment.

Within this "spiral structure", basic competences can be acquired throughout the entire learning process. It allows learners to "join in" in a later round of competence acquisition if he "stepped out" in one of the rounds before. For instance, this was very valuable for those learners, who had no idea what the words "loop" or "branches" stand for before they started programming the graphic robot, but could use these structures of control when they continued programming within the "GUI-scenario".

2.3 A Personal Approach: Assessment

The decision for four competence-levels had a practical reason. As the learner's competence has to be transformed into a single grade for the term's report (and Austria's school-system provides five grades, 1 = "very good", 5 = "not enough"), the competence levels were designed to correspond to the four "positive" grades 1 to 4: If a learner acquires competences of what level ever, he or she of course has earned a positive grade. Furthermore, if a learner does never solve tasks beyond competence-level C, he or she can self-estimate the final grade, which, most likely, will be "3". This possibility of self-estimation was meant to help the learners defining their individual learning goals.

Final grading was on one hand based on the teacher's monitoring of the learners work in class On the other hand, the learners were encouraged to hand in their solutions to the tasks to document the acquisition of a certain competence level. Furthermore, at the end of the practising period for a certain competence, the learners had to pass short written exams consisting of eight tasks, two from each competence-level. The "examination-tasks" were rather similar to the task-sheet-tasks. In these short exams, again, the learners had to self-estimate their knowledge and skills by reading all of the tasks first, choosing the competence-level they considered to be appropriate, and finally solving only the two tasks of the chosen competence level.

Of course, false self-estimation within such a competence-oriented examination scheme might cause confusion: What to do, if a learner chooses a high competence-level, say: B, and fails to solve (parts of) the tasks? Grading the performance with "2" would not be fair, but how to decide and to argue whether the grade should be "3", "4" or "5"?

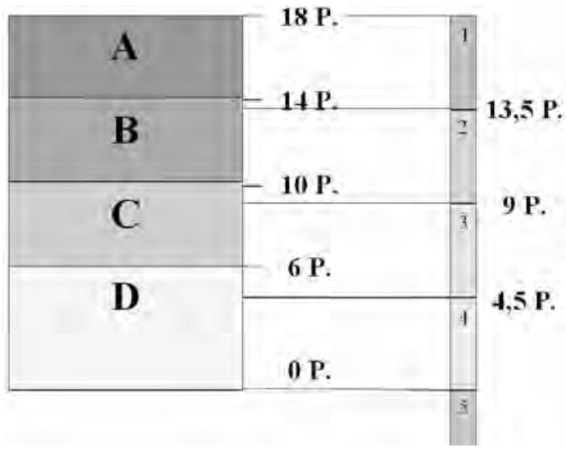


Fig. 4. Competence-level and distribution of points with the competence-oriented examination scheme. The numbers at the centre give the maximum amount of points for the two tasks of a certain competence-level. To the right, the corresponding grades (from "1" to "5") are displayed.

To avoid this from the beginning, each task was valued with a certain amount of points: Tasks at competence-level D valued 3 points each, those at competence level

C 5 points each, at competence level B 7 points each, and, finally, those at competence level A valued 9 points each. Thus, partial solutions could be taken into account as well. Fig. 4 shows how the total of points reached by a learner corresponds to the grades.

It should be noticed with this grading scheme, that only 0 points qualify for a negative grade (which did not prevent some of the learners to earn a negative grade now and then), and that choosing a certain competence-level can result in a better grade than expected. For instance, when choosing competence-level C, the learner estimates the grade "3" to be appropriate. But, if the solutions to both of the tasks are correct, the grade will be "2".

This strategy of mapping points to grades should meet self-underestimation of the learners, which proved to be as much a problem as overestimation. Therefore, only three of the exams were competence-oriented, while two exams had a traditional form, where the exam-sheets contained five tasks across all competence levels. In these exams, the learners were free to choose any task and any number of tasks to solve, but had to reach at least 8 points (out of 56) to earn a positive grade.

2.4 A Personal Approach: Some Reflections and (First) Experiences

Although things start to change, in most Austrian state-schools it is still common to teach and learn in "traditional mode" rather than to learn and teach in "competence-oriented and personalized mode": The teacher knows the learning goals and directs the learning paths of the learners, who are used to follow (more or less). Therefore, the described pattern to move towards competence-oriented learning was a new field of experience for the teacher/author and for the learners as well.

Being driven by the teacher's attitude of "Stop musing about competence-orientation, try it.", the described "competence-oriented pattern" was not planned down to the last detail from the beginning. For the teacher, it was a process of incremental adaption and/or step-by-step development of exam-sheets, task-sheets, competence-tables and working environments, guided by the knowledge about tried-out pedagogical tools. On one hand, this kind of workflow allowed to consider the actual learning situations in class, on the other hand it made it impossible to provide a complete competence-matrix right at the start of the courses. The learners had to work with consecutive one-dimensional tables for single competences instead.

Therefore, the described pattern of tools didn't create a "pure" competence-oriented learning environment but rather defined a "hybrid" approach:

- Although the learners could choose from a wide range of tasks,
- although the learners were given information about the levels of competence that might be acquired by solving the provided tasks,
- and although, therefore, the learners could choose a personalized learning path when attempting to acquire a (certain level of) competence,
- it was the teacher and the teacher alone, who had a general view of the learning goals from the start.

In other words: The teacher still directed the learning paths of the learners at a bigger scale.

In spite of being imperfect from the viewpoint of competence-orientation, this setting met the needs of the learners. For them, competence-orientation in the subject-matter Informatics during their first year at an engineering school was a twofold challenge: According to their foregoing experiences, Informatics consisted of text processing, working with presentation software and doing some calculations with spreadsheet programs. Only few of the learners had already done some coding (with HTML) or some programming (with Scratch). Furthermore, they were used to very detailed task-descriptions like "Change the format of the first paragraph to right alignment and select double line spacing." Thus, most of the learners were new to programming AND to working autonomously, that is deciding for a task and developing a plan to solve a task by themselves.

The narrow borders of the "hybrid" competence-oriented traditional setting provided familiarity for most of the learners, who simply focused on the actual competence and started to solve the tasks, either alone or in pairs. This, in turn, allowed the teacher to coach the individual learning-processes by giving individual feedback to (almost) all of the learners within one learning unit, by paying special attention to those who had problems to advance on their learning-path, and by advising the learners to profit from each others competences.

Above all, introducing aspects of competence-orientation seems to have resulted in more learning-specific interactions within the learning group, but having little effect on the learners' learning habits: For example, in spite of different levels of learners' speed and motivation, all of them started by solving task number one and proceeded by following the task-order provided by the task-sheet. Even those learners, who did well with self-estimation and were rather fast at writing programs, hardly skipped tasks, even if the tasks were quite similar to those solved just a minute ago. Some of these learners, who spent some time for programming at home, managed to solve all tasks at all competence-levels for a certain competence, the others usually got stuck at competence-level C or B, simply because they ran out of time. On the other hand, those learners, who worked on their competences but worked rather slow, did not want to skip tasks, either, even if they were not able to solve a specific task. Furthermore, a few learners "misunderstood" the freedom of choice and dawdled away after having solved one (and in most cases: the easiest) task. To come across this attitude, the teacher defined "milestones" like: "The tasks corresponding to competence-level C have to be solved until next week's lesson", which provided the impetus to start working for some of the few.

Competence-orientation, as it was meant by the teacher, took effect only with the written exams, where the learners had to choose a level of competence. Due to the learners' prior experience with similar tasks at a specific level of competence and/or the restriction to only two tasks led to better grades than with written exams in "traditional mode"[3]. Astonishing enough, the learners preferred the latter kind of exams, though. Obviously, learning habits that have been trained for eight years are hard to overcome.

---

[3] Having become used to the competence-oriented mode of written exams, most of the learnes had the "strategy" to choose competence level „C". In the learners' own words, competence-level "C" sounded "easier" than level "B" or "A", but still offered the chance to earn a "2".

3 Further Work

Most resources about competence-orientation discuss learning environments for learners at lower secondary level (ages 11 to 14), where competence-orientation concerns all subject matters and general competences are given prominence within the competence matrices. The hybrid approach sketched above was a first step to introduce competence orientation into Informatics at the learners' age of 15, providing a valuable stock of experience, competence-definitions and tasks to build on.

Starting with autumn 2011, the new curriculum will take effect. Thus, next years' approach to competence-orientation should overcome (at least some of) the shortcomings of this first attempt by expanding the concept of competence-orientation to the (other) topics named in the curriculum, "Informatics basics" and "standard-software"[4], and by providing competence-matrices at the beginning of the learning-process. With the latter, much care has to be taken to choose proper words within the description of the competences, ensuring that learners can understand what they are about to learn. Furthermore, it seems necessary to find answers to questions like: "Can self-estimation of a learner's competence become a self-fulfilling prophecy?" and to rethink the "culture of tasks":

– Is it possible to find a classification of tasks, which helps to decide whether a task is easy or hard FOR LEARNERS, depending on their prior knowledge and skills?
– How can tasks (and the definition of the corresponding competence) be redesigned to foster cooperative learning?

All of that calls for teachers' cooperation and leads to another question: How can, at a larger scale, teachers be motivated to develop a culture of competence-orientation, especially with a subject-matter like Informatics with no centralized final exams?

References

1. Schwedes H.: Wie Kinder Lernen. http://gaebler.info/schwedes/lernen.pdf (2005)
2. Killen, R.: Outcomes-based education: Principles and possibilities. Unpublished manuscript, University of Newcastle, Faculty of Education (2000), http://drjj.uitm.edu.my/DRJJ/CONFERENCE/UPSI/OBEKillen.pdf
3. http://bildungsserver.berlin-brandenburg.de/sol_allgemein.html
4. Müller A.: Dem Wissen auf der Spur. Institut Beatenberg (2003), http://www.institut-beatenberg.ch/xs_daten/Materialien/Artikel/artikel_lernjobs.pdf
5. Institut Beatenberg, http://www.institut-beatenberg.ch/
6. Max Brauer Schule Hamburg, http://www.maxbrauerschule.de/mbs/downloads/2008_neue_mbs_bsp.pdf

[4] As these topics form a major part of the European Computer Driving Licence, it seems reasonable to use (and alter) competence-definitions that can be found in the ECDL syllabus.

7. http://www.institut-beatenberg.ch/xs_daten/Materialien/kompetenzraster.pdf
8. Girmes R.: [Sich] Aufgaben stellen. Professionalisierung von Bildung und Unterricht. Kallmeyer, Seelze (2004)
9. Lersch R.: Kompetenzfördernd unterrichten. 22 Schritte von der Theorie zur Praxis. In: PÄDAGOGIK Heft 12/2007. http://www.uni-marburg.de/zfl/ueber_uns/artikel/artikel-lernsch-2007-KompetenzfArdernder_Unterricht
10. Dehnadi S., Bornat R.: The Camel Has two Humps (2006) http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf
11. Antonitsch P.: Standard Software as Microworld? In: Mittermeir R. (ed.): From Computer Literacy to Informatics Fundamentals. Lecture Notes in Computer Science Vol. 3422, Springer, Berlin-Heidelberg (2005)
12. Antonitsch P.: Erfahrungen zur Individualisierung im Programmierunterricht. In: G. Brandhofer et al. (eds.): 25 Jahre Schulinformatik in Österreich. Österreichische Computer Gesellschaft, Wien (2010)

All links have been accessed on Apr. 26th, 2011

The Art of Programming in a Technical Institute after the Italian Secondary School Reform

Alberto Barbero[1], G. Barbara Demo[2]

[1] Istituto Tecnico Superiore "G.Vallauri"
via San Michele 68, Fossano (CN) – Italy
barbero@vallauri.edu
[2] Dip. Informatica, University of Torino
c.so Svizzera 185 - 10149 Torino – Italy
barbara@di.unito.it

Abstract. The Italian Secondary School Reform became effective with the 2010-2011 school year. It establishes guidelines for the different subjects but leaves large autonomy and responsibility for fixing concrete curricula to teachers. This autonomy is crucial for Informatics because it does not have uniform teaching models as do Mathematics and the other classical disciplines. During the discussions about the Reform, Computer Science faculties and researchers from Italian universities developed the "Manifesto of Informatics in Secondary Schools" which was issued in May, 2010. Hints from the Manifesto along with teachers' actual experiences are meant to help solidify Reform guidelines and define appropriate school curricula. As a contribution to these discussions, we present here the activities under development in one technical secondary school for first year students, i.e. those about fourteen years old. Integrated with the planning of the second year, 2011/2012, this experience is proposed both as a reference activity for technical secondary schools and as a component of the Informatics course introduced by the Reform in the Applied Science secondary schools in Italy.

Keywords: Information Science, Manifesto for Informatics in Secondary Schools, Scratch, Programming, Computational Thinking.

1 Introduction

2010-2011 has been the first year after the reform for all types of Italian Secondary Schools. During the reform development, discussions concerning Informatics in schools have been frequent. Newspaper articles appeared claiming that "it is a bizarre idea to have Informatics as a separate discipline with dedicated hours in secondary schools because, although computers are all around us, we do not think that everybody must learn how to program them in some formal language. Indeed, we do not need to be an engineer in order to know how to drive a car".

Computer Science Engineers with Information Science faculties and researchers contributed to the discussions about how to reform Informatics in schools. Their common answer is the "Manifesto for Informatics in secondary school", outlined in the second section of this paper. Its original version is still published in the Italian language only[1]. The Manifesto points out that in today's society, computer science is considered from three quite different points of view. One is the *pragmatic* or operational perspective that considers Informatics to be a set of hardware and software systems. The second is the *technological* perspective shared by people that conceive Informatics as a set of technologies to be used to implement software systems and applicative packages. The third is the *cultural* perspective, viewing Informatics as a scientific discipline founding computer technology.

Meant as a contribution to the reform, discussions concerning the Manifesto pointed out that in the Italian schools, Informatics is almost only present in the last years of technical schools as learning technologies to implement software. In the large majority of other schools, Informatics is seen as using Office-type suites or specialized software such as the well known GeoGebra or other such systems. Thus, according to the Manifesto community, in most Italian schools the operational and the technological aspects of Informatics are present while activities contemplating Informatics as a scientific discipline are only seen quite occasionally.

With the Reform, computer science related subjects are present in both technical schools and in Applied Sciences secondary schools starting in the first year. Also, in the classical, scientific and pedagogical secondary schools, teachers must cover "Computer science elements" during Mathematics lectures. Thus there is now the need to have new approaches of teaching computer science.

For every subject, the Reform establishes guidelines for each school year in every school type, leaving to teachers and individual schools the job of defining their teaching methods. This provides an autonomy and responsibility level higher than previous education curricula allowed. This autonomy is most critical for Computer science related disciplines because there is no tradition concerning the contents and teaching methodologies teachers may refer to or feel obligated to use. Moreover, for historical reasons, Informatics can be taught by teachers having quite different backgrounds: teachers can be electrical or mechanical engineers, physicists, mathematicians or have other specializations and consequently, in several cases, they have only a pragmatic experience with Informatics.

Fortunately, taking advantage of the greater autonomy provided by the Reform, Informatics teachers in a number of secondary schools, some together with a university computer science department, reconsidered their approaches to teaching Informatics during the 2010/2011 school year, and aimed to make their teaching activities comply with the Manifesto's intent. Ongoing school experiences are being monitored jointly by secondary school and university computer scientists and will be reviewed for possible best practices for use by other schools.

One of these experiences is addressed in the third section of this paper. It concerns current year work in a first class of a technical school, i.e. with students about fourteen years old, and some of the activities planned for the next 2011/2012 year. Before the reform, in secondary technical schools, programming was generally considered too difficult for 14-15 years old students and only introduced from the third year, using Visual Basic, C, C++ and Java languages. In the experience here presented, teachers decided to achieve first year guideline competencies through problem solving and programming. Indeed, from their previous teaching experience they consider it to be most important to avoid losing the attention of students entering a technical school. Thus, they choose programming activities as most motivating for these students. A critical point was then finding a suitable programming language and related Integrated Development Environment (IDE), not only to introduce coding but also to introduce students to algorithms and programming principles. The *Scratch* language and its IDE developed by Resnick's group at MIT [2] were chosen for the reasons given in the third section with the outline of the introductory activities.

The fourth section addresses Scratch experiences as of the end of the current year and plans for the next (second) year. The experience around the game "Guess what number is between m and n" allows the class to study the binary search algorithm and introduces students to the properties of complexity, generality and correctness related to algorithms and programs. In the second year, the use of Arduino with a Scratch interface is planned. Also, different interfaces and programming languages are being considered based on the experiences that teachers are exchanging among themselves so as to improve activities for next year's classes.

2 The Manifesto for Informatics in Secondary Schools

In Italy there are three main national associations of computer scientists: the National Consortium for Informatics Inter-universities (CINI), the Group of Informatics Engineers (GII) and the Group of Researchers in Information Science from Italian Universities (GRIN). When the Education Ministry was developing the Secondary School Reform these associations contributed to the discussions in different ways. One result was the "Manifesto for Informatics in Secondary Schools" issued at the beginning of May, 2010. The Reform was effective for schools in September, 2010.

The Manifesto points out that, in Italian schools, Informatics is almost always present as a learning technology to implement some software in technical schools such as Office and Open Office, GeoGebra or other similar software for use in learning Mathematics or other subjects.

The Manifesto notes that "Informatics is becoming the kernel of our modern world both because it is needed for the normal development of our everyday duties and because its development shapes and directs the advancement of our whole society. Nowadays, in all areas of human activities we can find the influences of digital discoveries and achievements. Indeed, the computer is no more used for the traditional scientific calculus only, but it is also used in all areas of industrial production, medicine, publishing and communication to name only some of its applications. Two billion people have at least one contact on the net each day. We have around us products full of hundreds of millions of billions (no typos here) of transistors– elementary hardware components supporting information technology – in our cars, in domestic appliances, inside the gas pumps, in our videogames, and they are half of the financial value of the products. Hundreds of billions of software instructions, expressions of human intelligences, give life to these components and, through them, to all processes peculiar to our modern society.

Computer Science or Informatics has three distinct meanings; related, but quite different. A person can have a:

1. a *pragmatic* conception of Informatics and see it merely as a set of hardware and software systems;

2. a *technological* conception and perceive Informatics as a set of technological tools to be used to build devices and to implement system and applicative packages;
3. a *cultural* conception and perceive Informatics as a scientific discipline, thus making possible information and technology science.

Most people hold the first view of Informatics: knowing computer science means knowing which digital devices to buy and how to use software packages. For technicians, knowing Informatics means knowing how to develop software systems".

The Informatics discipline in schools is a largely debated question. Likely there are different answers depending on the point of view one looks at computer science: the *pragmatic, technological* and *cultural* conceptions all inspire different ways of dealing with Informatics in schools. Different types of secondary schools can plausibly have different aims in making their students competent in Informatics and consequently they may decide to stress one of the above aspects over the others. For the conception of Informatics as a science we should mainly address its epistemological aspects and focus on its connections with Mathematics, in particular Logics, Philosophy and History. This is particularly relevant to Italian classical, scientific and pedagogical secondary schools. Nevertheless, it should be an aspect of digital literacy possessed in some form by everyone at the end of any secondary school. Unfortunately, this aspect is missing in (almost) all of our secondary schools.

3 Scratch activities in the first year of Reform

It should be discussed what "digital natives" means in 2011 and whether Italian fourteen or fifteen years old students are digital natives or not [3]. In our schools, due to the very different levels of familiarity with computers and computer science, the guidelines for the first year in technical schools includes the European Computer Driving License (ECDL) syllabus. Most of the schools offer to their students the ECDL certification to ensure a basic common level of Informatics competencies [4].

As we said in the introductory section, teachers have the autonomy of defining a complete curriculum for their class. At the Vallauri Technical Institute of Fossano (Cuneo), Italy, teachers have decided to move to the first year part of the competencies that were previously part of the third year guidelines in order to introduce students to algorithms and programming. They aimed at gaining the interest of first year students by finding computer science motivating activities and at the same time allowing students to acquire skills established in the guidelines and also needed to pass the ECDL certification. They decided to look for a programming language different from those normally used in the last years of secondary schools and in universities which are too difficult and require too much time to obtain motivating results.

Scratch is a visual programming language developed by Mitchell Resnick and his Lifelong Kindergarten Group of the M.I.T. MediaLab in Boston. Scratch and its visual environment suited the curriculum teachers were thinking because of the following reasons:

- Scratch is easier to use for young people beginning a technical secondary school with no prior experience of programming. For example, it forces syntactically correct programming;
- it was specifically created to introduce basic concepts of problem solving and programming to very young students
- it is motivating for young students because it offers an attractive environment
- it is sharable among all secondary school types and is flexible enough so that each school can adapt it to its own needs while still maintaining a common ground of experience;
- it is suitable for teachers and older students, even those in non-technical schools such as classical or scientific secondary schools (called "liceo" in Italy);
- it forces structured programming; and
- it provides an open environment.

At a first look, Scratch appears to be a play-tool, however under closer examination, it proves to cultivate logic and reasoning.

From the outset it is possible to use variables, lists of values, primitives for selection and iteration. In Scratch, programmers can easily implement animations, simultaneously execute different processes, make them interact and use events. Programming is done using a visual block editor. The blocks have jigsaw puzzle shapes in order to direct which language components can be put together as shown in figure 1. Each block has a primitive instruction written on it. Blocks have different colors and shapes depending on their function. Blocks are connected by clicking on the icon and dragging the chosen block to its proper position, forming a command.

Scratch is open source and freely downloadable (http://scratch.mit.edu). It is stable and powerful. Every day, students find some new function to use. On the Scratch site, students can find free manuals, examples, forums, instructional videos, and more than 1,623,937 projects under the Creative Commons license (accessed 3rd March 2011). They can also upload their own project to share with other users all over the world.

In teachers' current plans, formalisms and more difficult reasoning projects are being left to future years. As an example, properties of complexity and generality of algorithms are only introduced in year one as are properties for loops (such as preconditions, post-conditions and invariants). A structured development of algorithms to solve problems is necessary when using Scratch because of the shape of the puzzled statements used in the language.

For the sake of space here, we have summarized only three activities which had important roles during their first year of experience: the "cat – think of a number" game which was worked on during both Informatics and Mathematics classes; a "calculator" for basic arithmetic operations; and the last activity assigned during this 2010/2011 school year, the "high-low, guess-a-number" game.

Programming activities started with the "think-of-a-number" game that students sometimes play in school. We had the students develop a program in which a cat asks the user sitting in front of the screen to do actions such as: "think of a number between 1 and 9", "now add 1", "multiply the result by 3", "subtract the number you thought of at the beginning", "again subtract 2". At this point the cat says "Tell me the number you finished with". Once the user gives his or her answer the cat provides the original number that the user thought! In figure 1 we show the commands the cat gives during the game. They are simply a sequence of "think of" instructions followed by the number of seconds before the next command is given. In addition, input and output communication commands are used. This experience lead to discussions and activities with Mathematics teachers who analyzed the game as an exercise of one (or two) variable(s) equation(s).

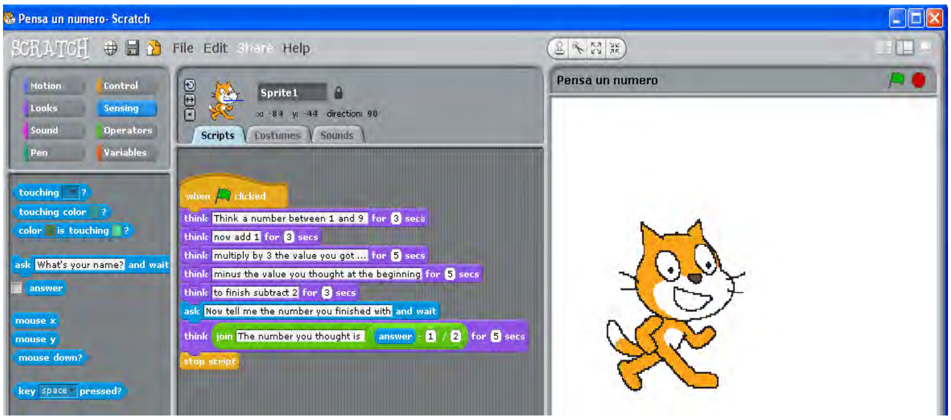


Fig. 1. Think a number game.

After the students discussed the mathematical aspects of the game, each group of students was asked to invent their own version. Then, after the different games were coded, they were integrated into a unique program in which, at the beginning, a random choice defined for each run which game the cat proposed to the user.

The second activity was to build a calculator for basic arithmetic operations, as shown in figure 2. Working on first and second projects students discovered simple variables, input/output management and events.

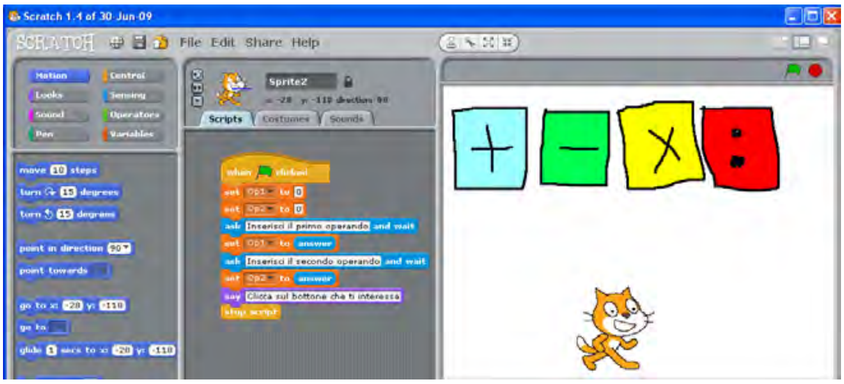


Fig. 2. Calculator.

Teachers supervised the programming experiences and asked students to write reports showing the different solutions they have developed. During the process, students performed tasks meaningful and enjoyable to them while achieving most of the competences required by their grade guidelines. They are also better prepared to pass ECDL- like tests that, as noted above, are often offered at the end of the second year in technical secondary schools.

4 Revisiting Scratch experience without and with Arduino

The final activity, the “high-low-guess-a-number” game involves a cat and a dog. The dog has to find out by trial and error the number the cat has thought (within a given range).

Fig. 3. High-low-guess-a-number.

The objective for the students is to try to minimize the number of trials. Obviously, the best strategy would be to implement a binary search type algorithm. Each group of students works on its own game version. The students then compare different solutions to the same problem and learn to understand that algorithms and programs have a property called “time complexity”. While programming this game, partly shown in figure 3, we introduce the iteration via the repeat until primitive, the possibility of communications via messages between the cat and the dog, and the binary search algorithm.

Teachers are currently defining the plan for next year’s activities. It will consist of refining the experiences carried out during the first year and defining new activities for the second year students. For the second year, both App Inventor for Android, part of Google Labs [5], and the Arduino board are being considered for use [6]. By using App Inventor, students write applications for their smart phones. This is extremely motivating for them. Arduino is an open-source platform which is easy to enrich with sensors and actuators suitable for the interactive environments in which we want to use it. Our interest in Arduino is because we aim to provide a concrete example of how a program can interact with the real world. A contribution specifical to the relationships between Informatics and Robotics in Secondary Schools is in [7].

We are considering programming Arduino using the S4A (Scratch for Arduino) environment developed by Citilab in Barcelona (Spain). S4A is a Scratch modification supporting simple programming of Arduino [8]. Our current plans are to also introduce the language "Processing" for Arduino during the third year because it will not only make students work with a third programming language but it will also change from a visual to a textual programming type within an already known framework. Processing is a C-like language used to write programs using Arduino to exchange data or similar.

The planning of the complete curriculum for the second year was finished by the end of May, 2011. It also contained the design and programming of web sites with an emphasis on making students understand the difference between form and content.

5 Conclusions

We have presented a preliminary contribution for introducing Informatics in secondary schools during the 2010/2011 first year of the Secondary School Reform in Italy. By beginning these new experiences in technical schools in which computer science is already present, we try to establish educational objectives and activities which can be used to introduce Informatics to all types of secondary schools, even those where Informatics has not been previously present. This experience is proposed to be a curriculum for vocational secondary schools as well and, with science history integrations, for Liceo.

When asked which courses they are having during first two years for a Computer Science degree, (too) many students name the languages they are using to code programs and, for example, say "Java" rather than the actual name Programming Principles of this course. The aim of our computer science activities in secondary schools is to improve our students' understanding of abstract properties of programming beyond that of prior students. Obviously our aim in secondary schools is not to educate all students to become good programmers but rather to introduce all future e-citizen to the computational way of thinking that many researchers consider essential for our next generations. Jeannette Wing, the President's Professor of Computer Science and head of the Computer Science Department at Carnegie Mellon, during her presentation in the Computer Science Distinguished Lecture Series at Carnegie Mellon in Qatar, said: "Computational thinking is a fundamental skill used by everyone in the world, and should be incorporated into educational programs along with reading, writing and arithmetic to grow every child's analytical ability. Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science. To flourish in today's world, computational thinking has to be a fundamental part of the way people think and understand the world" [9].

Our problem solving and programming activities have been inspired by many researchers that propose experiences motivating students and introducing them to basic topics of computer science. We name here the most relevant of these inspiring colleagues: Mark Guzdial and Barbara Ericson for their Context and Media Computation proposals for a motivating yet systematic introduction to CS of freshmen in all subjects but Informatics at Georgia Tech [10]. Our next year's work also considers adopting suggestions from Context Computation planning activities for the third year in our secondary school.

The books by Mariano Tomatis on "Mathematics and Informatics for crimes" are also worthy of attention in planning our next activities because this approach is gaining large attention in presentations to Informatics and Mathematics students at several universities and secondary schools [11].

References

1. Manifesto sull'Informatica nella riforma della scuola superiore, http://www.grin-informatica.it
2. Resnick, M. et all., Scratch: Programming for All, CACM, Vol. 52, No. 11, pp 60-67, November 2009
3. Prensky, M.: Digital Natives, Digital Immigrants, On the Horizon, MCB University Press, Vol. 9 No. 5, October 2001
4. European Computer Driving License – ECDL, www.ecdl.org/
5. App Inventor for Android, http://appinventor.googlelabs.com/about/
6. Arduino Open hardware Project, www.arduino.cc
7. Barbero, A., Demo, G. B., Vaschetto, F., A Contribution to the Discussion on Informatics and Robotics in Secondary Schools, in *Proc. RiE* 2nd International Robotics in Education Conference*, 15-16 September 2011*, (to appear).
8. S4A - Scratch for Arduino, http://seaside.citilab.eu/scratch/arduino
9. Wing, J., Computational thinking is a fundamental skill, Presentation in the Computer Science Distinguished Lecture Series at Carnegie Mellon in Qatar, April 2011 http://www.cmu.edu/news/blog/2011/Spring/computational-thinking.shtml
10. Guzdial, M., Ericson, B. Introduction to Computing and Programming in Python, A Multimedia Approach, 2/E, Pearson , (2010)
11. Tomatis, M., "Mathematics and Informatics for crimes", Seminar at the Department Computer Science, University of Torino, www.marianotomatis.it/index.php?lang=en

ICT in the Czech and Slovak National Curriculum

Jan Berki[1],

[1] Department of Applied Mathematics, Faculty of Science, Humanities and Education,
Technical University of Liberec, Studenstka 2
46312 Liberec, Czech Republic
jan.berki@tul.cz

Abstract. One source of the revision of the curriculum is the international experience. This article describes a probe based on a conceptual analysis of the national curriculum documents of two countries with similar conditions – the Czech Republic and Slovakia. The comparison focuses on three parts: a comparison of the definition of the concept of basic education and its content, the areas of education focused on computer science and the amount of integration of ICT into other educational areas. The interview with three teachers of these educational areas was focused on the vision of changes in concept of the curricula at the national level.

Keywords: National curriculum, computer science, information and communication technology, conceptual analysis.

1 Introduction

Informatics is ranked among newer – yet well-established disciplines. However, as a field of education, it has built its nest only recently, in terms of basic education. Therefore, any examination or assessment of the curriculum should be, on principle, based on comparison with foreign concepts. Evaluation should be a natural component of scheduled, cyclical revisions of curricular documents. The aim of this study is to evaluate identical features and differences between the national informatics curricula introduced in the Czech Republic and in Slovakia.

We are connected with the Slovak Republic in various ways, and our common history is an extremely important and strong link between the two countries. Due to the fact that we used to be two states in one, it is obvious that the Slovak Republic is very similar to our country, both from the geophysical and the (geo)political point of view. In general, there are not many differences in terms of the initial situation in the educational systems. Both countries have chosen a very similar approach to educational reforms. What is more, understanding Slovak in our environment is broader than understanding other (although international) languages. Therefore, choosing Slovakia for comparison suits the purpose very well.

2 Definition of terms and content of study

To qualify the content of the study, a concept hierarchy defined by Punch [1] is used:

- Area: Information and communication technologies (ICT) in the basic school curriculum
- Theme: ICT in the Czech and Slovak national curriculum
- Aim: To compare the ICT areas in the Czech and Slovak national curricula (to describe identical and different features)
- General research questions: What are the concepts of the ICT educational area? What is the same in terms of ICT implementation concepts in the Czech and Slovak curricula? What is different?
- Specific research questions: What is the main objective of the ICT educational area? What sub- aims have been defined within this context? Are information and communication technologies integrated in other educational areas systematically? Which grades and how many ICT lessons are involved? How is the concept accepted and assessed? What is the nearest future of the concept?

Within both compared systems, two levels are included in the basic education – primary education in the sense of level 1 in the International Standard Classification of Education (ISCED 1); and lower-secondary in the sense of ISCED 2. [2] A pupil means an subject of education. Basic education is usually provided by basic schools. For the purposes of this study, the basic school means a fully organized basic school with all grades, i.e. with 1st through 9th grades. In either country, the lower-secondary level in basic school corresponds to the lower-secondary level (ISCED 2) in schools called "gymnázium".

There is disambiguity in terms of definitions connected with informatics, information and communication technologies, information education, educational informatics, and in regard to their correlations. Viewing the fact that this is one of the aspects subjected to the research, it is necessary to avoid any definitions in this chapter as this would lead to anticipation of conclusions, which may influence the fair-mindedness of the analysing. Let's not forget that, for example, in English language there are some other definitions as well, such as computer science, information science, information technology etc.

The curricular documents subjected to the examination are the documents which are used by relevant organizational bodies within the government (i.e. the legislative, as well as executive bodies) in order to specify the educational content of the basic education. Although the analysis is mainly focused on the educational area of ICT, the context of other parts of the document cannot be omitted. Implementation of the concepts in basic schools is not the aim of this study but a comprehensive comparison of e.g. Framework Educational Program and School Educational Program may be interesting. Roughly, this has already been done at the level of content analysis [3]. The curricular documents define mandatory framework, on which schools educational programmes have to base. Current version of Czech national curriculum has been since 2007 and Slovak since 2008. This means the last classes are still taught according to the old programmes in this school year.

3 Methodology

In compliance with Punch [4], having completed chapters dealing with questions, what we examine and why, we will describe how the declared aim can be achieved. The specific research questions have been divided in two categories, while the criterion was whether the question stands for a description or an attitude.

Thus, the solution of the first part of the study follows from the aim to analyse the text document. The conceptual content analysis [5] was narrowed to searching for defined features in the sample of unique terms. The analysis was conducted on two levels. Terms include both words (e. g. model, electronic, digital), but also topics/phrases (ICT). Part of speech or grammatical case didn't decide into account. Such words were accepted as equal. Although the code led to a record level of word-frequency, not the relationship between characters is not possible according to Haman [9] possible to ignore the context of maintaining the validity of the data. Irrelevant information was left out. The research data contained the Framework Education Programme (FEP) for basic education [6] and the National Education Programme (SEP) for the primary school [7] and for the lower-secondary school [8]. Irrelevant data sets were put aside.

The main objective of the study was to compare the results from the conceptual analyses of particular curricular documents. The following comparison criteria had been set:

- Name of the subject,
- Number of mentions, accent upon the ICT in the preamble,
- Degree of integration of ICT in the other educational areas, or the number of mentions about ICT in the subject matter, outputs or in the preamble of other educational areas
- Number of ICT lessons and placement in grades,
- Theme blocks – curriculum x outputs,
- Definition of the content, specification of the aim of the educational area,
- Conception /orientation within the primary and the lower-secondary level.

In general, they were completed with the following:

- Total number of lessons,
- Cross-curricular themes.

The part which is focused on questions related to possible (scheduled) shifts within the content of the curricular documents actually follows from interpretation of opinions and attitudes towards current wording of relevant persons. The main research method used within this part of the study was an interview. Prior to the interview, based on the research questions, themes have been specified. Interpretation of the interview then provides respondent's attitude. According to Chráska [10], this type of interview is called "unstructured interview" as neither specific, particular questions nor their order had been set. However, the prepared themes, according to Švaříček [11], may be a significant feature of a semi-structured interview. This kind of interview cannot be conducted as neutral, as the interviewer communicates and interacts with the respondent. Whenever possible, a form of a natural interview was applied, within relaxed atmosphere, without hurrying, not to reduce the asymmetry level of the speech flow. From this reason, the interviews were not recorded (it was not natural, in terms of the situation). The content of the interview was recorded after the interview had been completed, which Chráska [10], from the psychological point of view, considers as more suitable. The records are not transcriptions but interpretations only.

Themes used for interviewing (regardless the order):

- Examples of best practice from abroad (examples)
- Priorities within primary education
- Degree of satisfaction with status quo
- Positive features in terms of status quo
- Suggested changes/shifts for the "domestic" curriculum
- Suggested changes for the "neighbour's" curriculum (to their knowledge)

It is anticipated that the respondents' attitude towards the curriculum is rather positive. Interviews were conducted with three teachers. Each of them is representative of another part of education system. One is a man, Slovak, acting at university. Second is a woman, Czech, acting at research institute. And the last is a woman, Czech, teaching at basic school. All of them are middle-aged people. All respondents work with curricular documents in their practice.

4 Comparison

The following sub-chapters correspond to the declared parts of the comparison. First, the general concept of the basic education is compared. The second sub-chapter is focused on the ICT educational area and on differences in the conception. At the conclusion, assessment on prospective changes in the curriculum is provided.

4.1 Differences in the concepts

The first – and very obvious – difference is the concept of the document alone. The Czech version has been developed as a single document which comprises the characteristics of both educational levels (ISCED 1, 2) and complex characteristics of the educational areas and fields of education. In the Slovak version, the levels are separate and they are described in separate documents. In the Czech environment, the ISCED 1 is divided in the 1st and 2nd period. In the Czech curriculum, the educational areas are divided into fields of education, while their distribution into specific school subjects depends on the school. The Slovak concept determines obligatory subjects in the educational areas as well. Detailed characteristics of the subjects can be found in the Attachment. The educational areas are defined by means of the characteristics (the description and the aims), theme blocks or units (in the Slovak version – the description and the aims, definitions, properties and relationships, procedures and methods), anticipated outputs and the subject matter. The Slovak theme blocks are defined by means of an educational standard (both the content and the performance). Educational standards have been developing in the Czech environment as well. However, they do not exist for the ICT (yet).

The headstones of the school reform are the so-called key competences. They are similar in both concepts (refer to Table 1).

Table 1: Key competences

Czech key competences	Slovak key competences	
	ISCED 1	ISCED 2
* Learning competencies * Problem-solving competencies * Social and personal competencies * Civil competencies * Working competencies * Communication competencies	* Learning to learn * Problem solving * Social-and-communicative competences * Personal, social and citizenship competences * ICT competences * Perceive and comprehend the culture, being able to express yourself through instruments of culture * In the area of mathematical and scientific thinking	* Competence for life-long learning * Problem solving * Social-and- communicative competences * Social and personal * Citizenship competence * Work-related competence * ICT competences * Perceive and comprehend the culture, being able to express yourself through instruments of culture * Apply the base of mathematical thinking and general ability to learn in the field of science and technology * towards initiative and go-aheadness

In regard to the content of the article, it is worth noticing that within the Slovak concept, the ICT-related competences are the key competences. In addition to the ability to use the selected ICT, communication through electronic media and (related-to-the-age) ability to search for information on the internet, this area is connected with the problem area of danger of virtual world. The topic of e-safety is a "trendy" topic in the Czech Republic as well. However, algorithmic thinking is missing in the Czech curriculum.

In regard to the cross-curricular themes, the curricular documents do not differ much (refer to Table 2). Attention should be paid to integration of media-related education.

Table 2: Cross-curricular themes

Czech	Slovak
* Personal and social education * Democratic citizenship * Education towards thinking in European and global contexts * Multicultural education * Environmental education * Media education	* Personal and social education * Traffic education * Project work and presentation competences * Safety and protection of health and life * Multicultural education * Environmental education * Media education

The last comparison (Table 3) is focused on school subjects and disciplines whose titles follow from the names of the subjects previously included in the school curriculum. Analysis of (ICT-)words-frequency was realised on all of these fields.

Table 3: Educational fields

Czech	Slovak
Language and communication (Czech language and literature, Foreign language) Mathematics and its application Information and communication technologies Humans and their world[1] Humans and society[2] (History, Civil education) Humans and nature[2] (Physics, Chemistry, Natural sciences, Geography) Arts and culture (Music, Fine arts) Humans and health (Education to health, Physical education) Humans and the world of work	Language and communication (Slovak language and literature, Foreign language) Mathematics and information processing (Mathematics, Educational informatics[1], Informatics[2]) Nature and society[1] (Science and homeland study) Humans and nature[2] (Physics, Chemistry, Biology) Humans and society[2] (History, Geography, Social studies) Humans and values (Ethics / Religion) Humans and the world of work (Manual training) Arts and culture (Music, Fine arts, Education through art) Health and physical movement (Physical education[1], Physical education and sports[2])

[1] primary only (ISCED 1) [2] lower-secondary only (ISCED 2)

4.2 Differences in the educational area

Let us start with the differences described in the first part of the comparative study. In Slovakia, besides the educational area, the ICT is defined as the key competence as well. In the Czech environment (refer to Table 3), the area is named "Information and communication technologies" and it is a separate area. Within the Slovak curriculum, information processing is grouped with mathematics, and it is divided in two subjects. Educational informatics is taught at the primary level (ISCED 1) and Informatics is scheduled for the lower-secondary level (ISCED 2). The allocation of lessons is shown in Table 4, including disposable lessons which can be added by the school to a particular educational area. The basic difference is the even distribution of the lessons among the grades within the Slovak concept, contrary to the one-off allocation in one grade in the Czech environment (refer to Table 4). It is obvious that this approach does not allow a cyclical development of competences in this area. We can deduce that the intention was to develop skills for mastering the technologies. The application level should have been added within the other subjects. The fact that this aim has not been fulfilled in the Czech Republic is shown in the theme report issued by the Czech School Inspectorate. It is declared that the main cause of the failure is the unsatisfactory level of ICT competences in teachers.

Table 4: Allocation of lessons

<table>
<tr><th>Areas / Subjects</th><th>Grade 1</th><th>Grade 2</th><th>Grade 3</th><th>Grade 4</th><th>Grade 5</th><th>Grade 6</th><th>Grade 7</th><th>Grade 8</th><th>Grade 9</th></tr>
<tr><td>Information and communication technologies</td><td colspan="5">1</td><td colspan="4">1</td></tr>
<tr><td>Educational informatics</td><td></td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Informatics</td><td></td><td></td><td></td><td></td><td>0,5</td><td>0,5</td><td>0,5</td><td>0,5</td><td>0,5</td></tr>
<tr><td>Disposable lessons</td><td colspan="5">14</td><td colspan="4">24</td></tr>
<tr><td>Optional lessons</td><td>5</td><td>5</td><td>5</td><td>5</td><td>6</td><td>6</td><td>6</td><td>6</td><td>6</td></tr>
<tr><td>Total allocation</td><td colspan="5">118</td><td colspan="4">122</td></tr>
<tr><td>Minimum</td><td>18</td><td>18</td><td>22</td><td>22</td><td>22</td><td>28</td><td>28</td><td>30</td><td>30</td></tr>
<tr><td>Maximum</td><td>22</td><td>22</td><td>26</td><td>26</td><td>26</td><td>30</td><td>30</td><td>32</td><td>32</td></tr>
<tr><td>Total</td><td>22</td><td>23</td><td>25</td><td>26</td><td>27</td><td>29</td><td>30</td><td>30</td><td>30</td></tr>
</table>

note: The grey colour means the Slovak concept.

It is obvious – and the names of the subjects clearly indicate that the orientation and the content of the subjects is different. The aim of the Slovak curriculum in the area of informatics is to build competences in terms of working with data, in fact, within the discipline of the same name. On the contrary, the aim of the Czech curriculum indicates the emphasise upon work with technologies. Let us compare the theme blocks (refer to Table 5)

Table 5: ICT themes

Czech	Slovak
Basics in using computers Searching for information and communication Information processing and usage	Information around us Communication through ICT Procedures, problem solving, algorithmic thinking Principles of ICT functioning Information society

On the Czech side, two themes are missing: first, the algorithmization or, more precisely, algorithmic thinking (which should be understood as the base of the Informatics), and the information society. This social context is completely omitted within the Czech concept. In part, this phenomenon is present within the context of social networks but neither the (expected) analysis of the changing society nor the relationships created on the basis of ICT is included. Obviously, this theme may be classified as the theme not related to any specific subject, or the theme slightly included in the area of civics, or social studies. The Slovak subjects at the primary level are focused on development of basic competences for work with data in the electronic form. The ability to use computer resources in everyday life is emphasized in both of the national documents.

The usage of common keywords through the entire concept is the interesting context. The Table 6 shows the basic keywords. Also a related form of the word was a part of the occurrence. The analysis did not contain the chapter, which directly define the content of computer courses.

Table 6: Number of words

	Czech (86 pages)	Slovak (37 pages)
Information	41	25
Communication	77	56
Technology	5	1

ICT was mentioned 4 times in the Czech curriculum and 12 times in the Slovak. And according to this fact is evident that the area of Informatics Study plays the more fundamental role in the Slovak concept. The other terms have been occurring: the model (27, 11) – usually in the meaning of Scenario, the computer software in various versions (6, 2) electronic (7, 4), the algorithm (6, 1), etc. The most common keywords with the area of ICT in the Czech document are in the area of Language and Language Communication (communication, information), Mathematics and its Applications (model), Arts and Culture (communication), and this is the same area in the Slovak language as well. However, this is the Art, which contain the most of the direct links to electronic versions of documents in the Czech educational field as well as in the Slovak version.

4.3 Differences in future development

The information in this chapter were obtained thanks to the interviews with three respondents about their opinions of national curricular document in their countries. The concept of the Great Britain was identified as an example of good international practice in the Czech and Slovak republic as well. New Zealand was marked as the potential model from one respondent. The ability of learning was supposed as the basic aim and it is the preparation for work and social life, too. The main change in the Slovak concept should be more rigorous curricula. On the other hand, the Czech respondents consider greater flexibility as an advantage. A range of teaching Informatics at the primary school and the teaching of algorithm and programming (also there is no difference between them at this first degree of schools) are the most important positive phenomenon of Slovak cirruculum.

It would be good to start with changing of the name of subject in the Czech curriculum. It is possible dividing the subject of ICT to two parts – one of them is focuses on digital technology and the second one is focuses on Computer Science. A major problem is a failed identification what are the computer science skills. ICT standards should be included in FEP. A greater integration of ICT into the other subjects is not possible because of the low level of ICT skills non-informatical subjects.

4.1 European comparison

European Schoolnet issues reports focused on ICT in education. Overview of seven national curricula [12] has included the Czech Republic, Finland, Lithuania, Norway, Portugal, Slovakia and Switzerland. We can find ICT competences like key competences in Norway and Switzerland too (excepting Slovakia).

There is specified ICT knowledge in these same countries at primary level. Most of the described countries, excepting Portugal, define ICT skills in their curricula. And Norwegian, Lithuanian and Swiss documents define ICT attitudes. Situation of secondary level is different in curricula. There are defined as ICT knowledge as ICT skills in all of these countries. Portuguese and Czech concepts don't defined ICT attitudes neither for secondary level.

5 Conclusions

It may be preferred the opinion that more changes are needed in the Czech concept. The content of the course must be define more precisely and must be more connected with computer science, respectively with algorithm. Increasing the prestige and the pressure on the integration of ICT into other subjects within the meaning of application would be including ICT skills among the key competencies. The second step should be more explicit declaration in the context of educational content – expected outcomes and curriculum in the FEP. The integration of passages of computer education would require also an increase in the time allocation and distribution of this subject to more years to develop a systematic means to work with.

This article was created with financial support of the project SGS-FP-TUL 20/2011.

6 References

[1] Punch, K. F. *Základy kvantitativního šetření*. 1st edition. Praha: Portál, 2008. 152 pgs. ISBN 978-80-7367-381-9. (EN version – Punch, K. F. Survey research: The basics. SAGE, 2003. 136 pgs. ISBN 9780761947059.)

[2] Průcha, J. *Srovnávací pedagogika*. 1st edition. Praha: Portál, 2006. 264 pgs. ISBN 80-7367-155-7. (Analogous in EN – Basic Education and the review of the ISCED [online]. Draft Version October 2009, UNESCO Institute for Statistics. p. 3. [cit. 2011-07-13] Available from <http://www.uis.unesco.org/Education/Documents/ISCED_RM_Basic_Ed_proposal_EN.pdf>.)

[3] Berki, J. a kol. *Sonda do obsahu ICT předmětů*. Unpublished yet.

[4] Punch, K. F. *Úspěšný návrh výzkumu*. 1st edition. Praha: Portál, 2008. 232 pgs. ISBN 978-80-7367-468-7. (EN version – Punch, K. F. Developing Effective Research Proposals. SAGE, 2007. 176 pgs. ISBN 9781412921268.)

[5] DVOŘÁKOVÁ, I. *Obsahová analýza / formální obsahová analýza / kvantitativní obsahová analýza*. AntropoWebzin, 2010, 2. [online]. Antropologie, Západočeská univerzita v Plzni, c 2010. [cit. 2010-09-27]. Available from <http://antropologie.zcu.cz/media//webzin/webzin_2_2010/AntropoWebzin-2-2010_1_.pdf>. ISSN 1801-8807. (Analogous in EN – Methods of Conceptual Analysis [online]. © 1993–2011, Colorado State University. [cit. 2011-07-13] Available from <http://writing.colostate.edu/guides/research/content/pop3a.cfm>.)

[6] *Rámcový vzdělávací program*. Praha: Výzkumný ústav pedagogický v Praze, 2007. (EN version – Framework Education Programme for Basic Education. Prague: VÚP, 2007. Available from http://www.msmt.cz/vzdelavani/framework-education-programme-for-basic-education.)

[7] *Štátny vzdelávací program pre 1. stupeň ZŠ v SR* [online]. Bratislava: Štátný pedagogický ústav, 2008. [cit. 2011-03-10]. Available from <http://www.minedu.sk/index.php?lang=sk&rootId=2319>.

[8] *Štátny vzdelávací program pre 2. stupeň ZŠ v SR* [online]. Bratislava: Štátny pedagogický ústav, 2008. [cit. 2011-03-10]. Available from <http://www.minedu.sk/index.php?lang=sk&rootId=2319>.

[9] Haman, A., Jílek, J. *Obsahová analýza beletrie pomocí obsahových indikátorů* In Materiály, studie, informace 17. Praha: Státní knihovna ČSR, 1986.

[10] Chráska, M. *Metody pedagogického výzkumu – Základy kvantitativního výzkumu*. 1st edition. Praha: Grada, 2007. 265 pgs. ISBN 978-80-247-1369-4. (Analogous in EN – see below.)

[11] Švaříček, R., Šeďová, K. a kol. *Kvalitativní výzkum v pedagogických vědách*. 1st edition. Praha: Portál, 2007. 384 pgs. ISBN 978-80-7367-313-0. (Analogous in EN – ESDS Qualidata teaching resource: exploring diverse interview types [online]. © 2003–2011, Universities of Essex and Manchester. [cit. 2011-07-13] Available from <http://www.esds.ac.uk/qualidata/support/interviews/semi.asp>. – or – Semi-structured interview [online]. Wikipedia, the free encyclopedia, last modified on 2011-05-12. [cit. 2011-07-13] Available from <http://en.wikipedia.org/wiki/Semi-structured_interview>.)

[12] Balanskat, A. (ed.) Review of National Curricula and Assessing Digital Competence for Students and Teachers: Findings from 7 Countries [online]. Brussels: European Schoolnet, 2010. 116 pgs. [cit. 2011-07-13] Available from <http://cms.eun.org/shared/data/pdf/curricula_review_final_reduced.pdf >.

Informatics for Primary Education. The case of Russian mathematical school

Elena Bulin-Sokolova *(Center for Information Technologies and Learning Environments, Moscow),*

Alexei Semenov*, and* **Valery Vardanyan** *(both – Computing Center of Russian Academy of Sciences, Moscow)*

What do we understand by Informatics? A sketch

Before giving an answer to the question let us say that we are going to present only one possible view on the school subject by name informatics. The goal of our paper is not to discuss this, but mostly to elaborate what our group in Moscow understands by this term. We do not pretend that we have the only ingenious understanding of it.

The traditional and the contemporary mathematics

The ancient mathematics developed starting with numerical quantities (integers and rationals) and geometrical objects. Then it moved to the continuum – real numbers and to continuous processes. In XIX century abstract algebra was invented, but primarily as a tool to describe and analyze space transformations and equations' solutions. A radical change started in the second part of XIX century: beyond proving and calculating we started to develop a general understanding what is proven theorem and what is computable function. This understanding was completed in its most important components right in time: when emerging computers started to do complex computations and human intellectual work.

So, for us informatics is the science of mathematical study of human and computer formal reasoning and algorithmic behavior. It is called also theoretical or fundamental informatics, mathematics of computation, or theoretical computer science as well as discrete mathematics or finite mathematics. It is not a part of hardware or software engineering but lies in the fundamental basis for them as well as for other fields of technology and science.

Recreational math, puzzles, parables and wisdom

We think that informatics as described above is the place where the school education can be enriched by involving into it treasures of human culture, represented by 'recreational' puzzles (started with puzzles of Alcuin - *Flaccus Albinus* - river-crossing of wolf, sheep, and cabbage etc.) parables, and paradoxes.

The school mathematics and informatics

Why do we teach mathematics in school? This is our way to develop reasoning abilities of students, to provide a part of general understanding of the world, and to give a toolkit for reasoning (and development of reasoning) in other subjects.

We believe that the similar is true for informatics in school. It can (to a large extent) contribute to development of students' reasoning, to help in general understanding of the world (cf. Cybernetics). We think that the algorithmic, process reasoning can contribute more to effective human behavior than other types of mathematical reasoning. We think also that the development of the ability of algorithmic thinking should go in the process of problem solving.

We should admit that informatics in school contribute little today to reasoning in other subjects beyond this general understanding. Nevertheless we believe that following concepts (and terms), as well as their scientific and every-day meaning, should be studied in informatics and used in other subjects: system and its components, interaction, signal, signal transmission, control, feedback. In their study formal automata theory as well as visual programming can be used as a source of problem-solving activities (see below).

Connection with ICT in school

In Russia informatics as a school subject appeared in the mid-1980-s by a decision of the top-level authorities of the country. But the name of the subject of that time was “Basics of Informatics and Computer Technology”. So, informatics was distinguished from computer technology. Today the name is “Informatics and ICT”. Again, the difference is assumed. Moreover, today:

1. ICT learning and mastering is dissolving and being made effective and motivating in other subjects.

2. The outcomes of the Informatics study are outlined by the Informatics exams (after 9-th grade, and final, after 11-th grade) as mathematical informatics problem-solving activity.

So, school informatics is not study of ICT.

How ICT help in learning informatics

To understand this it is good to consider how ICT help in other school subjects. We think that there are two major functions of ICT in learning as well as in other fields of human activity:

1. Tools (instruments) for information processing

2. Information sources and access to them

In our vision of school informatics the second function is not so much relevant for the goals of learning informatics. The first is more important. Again, in the first function general tools as LMS or text editor are used, but subject specific tools are important for learning informatics. These are tools that visualize and materialize objects and processes of informatics and provide visual and spatial environments to construct them.

For primary education this is extremely important. Also important is that informatics environments for mathematical constructing and reasoning can compensate pure (still) mental and pen-paper objects of other parts of mathematics. Sometimes environments for informatics, arithmetic, and geometry are joined in one environment as happened in the miraculous Logo.

How informatics helps in learning ICT

Let us repeat: informatics is the fundamental science of human reasoning and formalized behavior as well of information processing happening not in human brains and human adjacent environments (like in pen-and-paper calculations) but in the modern and (predictable) future digital technologies.

So, informatics can (and cannot) help in using computers in using ICT as (roughly speaking) physics can help in ironing and chemistry in cooking. In fact both learning informatics and using ICT immerse humans into a new reality of “being digital” as well as physics and using machinery gives them feeling of “being material and technological”.

Algorithm design and programming

We believe that algorithm design (and algorithm execution "by hand") is an important part of informatics. Solving problems of algorithm design can be facilitated by using virtual (computer) environments of the design and execution of algorithms. Naturally, the activity of algorithm design in such environment can (and usually does) follow a formal discipline, syntactic rules and restrictions, etc. This means that the student does practical programming and should learn these discipline, rules, and restrictions. But the goal is the algorithm design, as less rules and restrictions should be learned as better, correlation with existing programming "industrial" languages is not relevant, etc.

Motivation coming from the real effect (nice graphical design, unexpected result of numerical experiment, winning a game against opponent or computer) can be very important, motivation by the fact of using "adult" programming language is not so important (with some exception of very small high school population that is going to enter relevant universities or job places).

An important phenomenon of debugging should be considered very carefully. From one-hand side it is a very productive way to correct and further develop your reasoning. From another hand side it should not be used as a trial-and-error blind method to obtain needed result without serious thinking and proving.

Math and info in primary

Everything is visual and palpable

The one most important feature of our approach to math and informatics in primary school is the emphasis of seeing and understanding of all objects and processes, their properties, and operations. Let us make this clear. In a word problem about travellers with speed of 5 or 50 kmph via distance 10 or 100 km a student does no see the distances and speeds. When we ask a student to construct a string following a given instruction she or he sees all objects and actions.

In arithmetic we spend more time on counting. The experience of exact counting of few thousands of seeds (peas, or beans for example) provides basis for understanding by inventing of: commutative and associative laws, division of labor, etc. Trying to make 100 (or any other number) as a sum of lengths of some of 7 rods of different length is a good addition exercise as well as introduction to NP-completeness.

Objects

Primitive objects (atoms) are beads. Beads can have shape and color, they can be symbols of alphabets, digits, etc.

More complex objects are string and bags (multi-sets) of objects (recursively).

In multi-digit addition and multiplication we deal with visual objects (strings of symbols, not quantities) but in a very narrow, restricted and not so practically important way. Here we work with our objects in more logical and broader way.

Language and logic. Rules of the game

Understanding (written, mathematical) text is an important activity in math and info course. The initial vocabulary is not big: bead, string, bag, be the same, operations on strings and bags, ‘all’, ‘there is’, operators of structural programming.

Semantics is given (for a long initial period of learning) in graphical way by examples, not through verbal explanations.

The first logical connectives are ‘all’ and ‘some’ and their synonyms. Examples: ‘there is a red bead in any string here’ (on this page). One more connective is ‘no’, ‘not’: ‘there is no round bead in this string’. The connectives can be used in a “dual” way: ‘all statements here are true’.

The ‘name – value’ correspondence (“variables”) is introduced early.

We consider the obtaining of clear and “formal” common understanding of restricted “math-info” vocabulary as a very important result and condition of study. ‘Rules of the game” should be understood and identical for teachers and students.

Algorithms

Algorithm design is based on structural programming operations. Visual environments for design and execution are used. One of the most effective is Robot in the maze – the concept that was developed in Bratislava, Cornell, and Moscow independently but in a surprisingly similar ways.

Logo phenomenon

Logo can be considered as a unique example of environment were different objectives of learning informatics are well balanced in a visual setting.

We developed our version of Logo LogoFirst (IconLogo as Seymour Papert called it) to learn with before reading and writing and to learn verbal literacy along with graphical and algorithmic literacies.

Interaction and games

Games are used widely. Strategic games of two players with complete information constitute an important class of games. NIM – is an example.

Programmable Robot control in the real environment (on the floor) is a primary school introduction into several topics, including feedback and event-driven programing. LEGO version of it gives a flexible construction kit as well as a visual programming environment.

Complexity

Understanding of complexity is one of the goals in learning informatics. We mentioned above an NP-complete problem (the Knapsack packing). Even more visual is considering the following problems: 1, find a given face among 50 on the page. 2. Find two identical faces on the page of 40 faces.

Implementation

The described approach is used in hundreds of Russian schools for 25 years. Last year it was integrated into the Federal standards for primary education in Russia.

Genesis of Mathematical Curves by Turtle Geometry

CSINK, László – FARKAS, Károly

Óbuda University, H-1034 Budapest, Bécsi u. 96/b
csink.laszlo@nik.uni-obuda.hu farkas.karoly@nik.uni-obuda.hu

Abstract.

Keywords: turtle geometry, syntonicity, epistemology, math teaching, Papert, P· lya

1 Introduction

In our earlier work [1] we presented a new way of generation of a couple of curves using turtle geometry. In this paper we discuss the epistemological issues regarding our approach, as well as present some additional novel algorithms we have used in teaching informatics in a humanities secondary school. In turtle geometry we focus on the vital elements of Logo using its epistemological values with special emphasis on syntonicity. When working with turtle geometry, we find the use of intrinsic commands most effective for teaching mathematics and geometry on secondary school level. We use the concept of syntonicity in Papert's sense; we build on Logo's body syntonic and ego syntonic effect.

Our examples are based on two curve generation techniques. First, we put together the curve by "jumping out" of the origin and build the curve from "stretches". Second, the curve is the combination of the work of several cooperating turtles, what we call "electronic drama pedagogy".

2 Epistemology

Logo's role in education is clearly due to its impact on the development of thinking. To realise Polya's methods for improving problem solving skills, Papert thinks, Logo is the most effective tool in many respects. Based on our experience, we agree. In understanding and creating the algorithms for drawing certain mathematical curves, the body syntonic impact of Logo is rather useful, namely the empathy with the execution of the motion sequence. For example, to draw a square in Logo the algorithm "repeat four times: step some units and turn *90* degrees (or one-fourth of a full turn)" is well-known. Clearly this can be generalised to draw an *n*-polygon by turning *360/n* degrees at each vertex. To draw a circle, "repeat 360 times [step one, turn one]" is the widespread Papert algorithm. This generation is based on the intrinsic feature of the curve; it could be the symbol of turtle geometry. The circle is approximated by a regular polygon. The circle in fact is the limit of the polygon sequence when the number of sides tends to infinity. By this approach the circle can be defined in a non-classical way, and it can also present the concept of limit in a very expressive way in the secondary school.

We can go one step further. Using turtle geometry, most of the geometry curriculum of the secondary school can be taught and learnt more easily. Moreover, further mathematical curves can be demonstrated and taught using turtle geometry that had been taught in higher education only. Concepts such as angle or parameter, that were traditionally introduced in the fifth or sixth class in the elementary school and were not easily understood, can be taught earlier and in a more student-friendly way.

We think that both in secondary and higher education it is very important that mathematics should be presented in a clear, understandable way. Therefore, the curves in Logo should be first presented based on body syntony. How is the curve created? How could we walk its path? The most plausible (but not the only) sequence: step some length and turn some angle, like Papert's circle.

Real turtle geometry uses polar coordinates instead of Cartesian. Analysing the child's learning, polar coordinates are more natural: we look around to see and extend our hands to touch. When we first experience motion the point of reference is our own body. The child creates a circle when he builds a toy train around himself or draws a line in the sand with a stick.

3 Demonstration using "stretches"

In this section we present a series of examples that we find interesting and stimulating. Some of these examples are not elementary and not part of the compulsory material; nevertheless they are useful in developing thinking skills.

After the step-and-turn algorithm, we find it useful to present another one where we draw the curve pixel-wise in a polar coordinate system. We stretch r pixels in the start direction (e.g. north) and draw a point (a segment of length of one turtle step), lift the pen and retreat r+1 pixels, turn one degree and repeat the above until we get back to the start direction having done a full rotation. When the points are close enough we need not connect them as the circle can be clearly seen without connecting the points. If r does not change we have drawn a circle around ourselves. Decreasing the angle – and thus increasing the number of repetitions – the circle will be finer. We think that this method of drawing a circle is easier to empasize with, it is more syntonic than the original Papert algorithm. Another advantage is that many other curves can be generated similarly.

If r is not constant but varies according to some rule, various curves can be generated in a body syntonic way. Rotation can be defined with respect to a given angle or also with respect to the starting northern direction. The direction of the turtle need not be stored by us as this is a parameter of the turtle that can always be referred to. The length of the radius can be expressed in terms of the previous value, or the starting value, or rotation parameters. The first is intrinsic, the second is extrinsic and the third is mixed. Intrinsic Logo commands: forward, right. Extrinsic Logo commands: setx, sety, setheading.

Our circle drawing algorithms:

```
repeat 360 [penup forward :r pendown forward 1 penup back
:r + 1 right 1]
```

or

```
repeat 360[penup forward :r pendown forward 1 penup back
:r + 1 setheading heading + 1]
```

We favour the first version that uses only intrinsic commands. The intrinsic features of the circle are independent from the selection of the coordinate axes or the starting point. To draw further curves we only have to substitute *:r* with a function of rotation. Let us consider some examples.

Ellipse

On the various ways of how to generate the ellipse consult Pavel Boytchev's film [2].

A polar coordinate equation of the ellipse is:

$$r = \frac{a(1 - e^2)}{1 + e\cos(\theta)}$$

Here *a* is ½ of one of the axis and e is eccentricity:

$$e = \frac{c}{a}$$

where *c* is one-half of the focal length. The eccentricity is a fraction. The smaller the eccentricity, the flatter the ellipse. Based on the equation of the ellipse, this is how *r* can be expressed in Logo:

```
make "r :a * (1 - :e * :e) / (1 + :e * cos :fi)
```

This is how we can draw an ellipse combining 360 points:

```
repeat 360 [make "r :a * (1 - :e * :e) / (1 + :e * cos
heading) pu fd :r pd fd 1 pu bk :r + 1 rt 1]
```

The value of *:e* is less than 1, so we have avoided division by 0. When *:e* is greater than 1 it will not be an ellipse any longer. Our formula may still be used if we insert if `(1 + :e * cos heading) = 0 [rt 1]` after repeat 360 [.

Before the command, *:a* and *:e* must be given, e.g. `make "a 100 make "e 0,5`

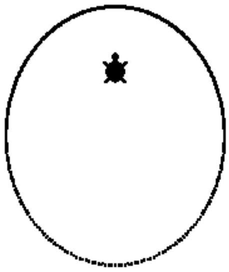


Fig. 1. Ellipse from 360 points.

We can also make a finer drawing:

```
repeat 3600 [make "r :a * (1 - :e * :e) / (1 + :e * cos
heading) pu fd :r pd fd 1 pu bk :r + 1 rt 0.1]
```

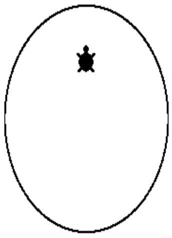


Fig. 2. Ellipse from 3600 points.

To draw an Archimedes spiral we can do the following:

```
to spiral :r :a
repeat 360 [pu fd :r pd fd 1 pu bk :r + 1 rt 1 make "r :r
+ :a]
end
```

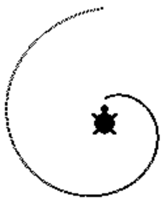


Fig. 3. Spiral 20 0.2

The general algorithm looks like this:

```
to curve :f
;f is a list expressing the radius with the angle
repeat 360 [make "r run :f pu fd :r pd fd 1 pu bk :r + 1
rt 1 ]
end
```

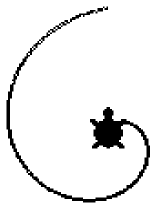


Fig. 4. Curve [0.2 * heading]

With trigonometric functions, the use of heading is practical, e.g. the polar coordinate form of sine can be seen in Fig. 5. We may note that the density of the points in the line is not uniform due to the use of sine:

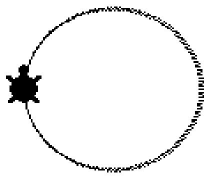


Fig. 5. Curve [100 * sin heading]

Here is the cardioid:

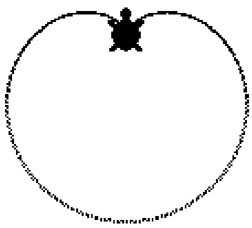


Fig. 6. Curve [50 * (1 ï cos heading)]

Now we are going to draw the nephroid (for a nice demonstration see [10]) whose polar coordinate equation is the following:

$$r(\varphi) = a \cdot \sqrt{1 + \frac{3}{2}\left(\sqrt[3]{(1+\cos\varphi)\sin^2\varphi} + \sqrt[3]{(1-\cos\varphi)\sin^2\varphi}\right)}$$

where *a* is the distance between the two breakpoints of the kidney-shapes. The formula in list processing will be

```
r = a * sqrt (1+3/2 * (power (1+cos φ) (sin φ)^2) 1/3 +
power (1- cos φ) (sin φ)^2 1/3))
```

We will get such a curve if we roll a circle of radius *a/2* along the perimeter of another circle of radius *a*. A fixed point of the rolling smaller circle will draw the nephroid. The two breakpoints will be *(a; 0)* and *(-a; 0)*.

Drawing a nephroid in Logo is:

```
to nefroid :a
repeat 360 [make "r :a * sqrt (1 + 1,5 * (power (1 + cos
heading) * sin heading * sin heading (1 / 3) + power (1 -
cos heading) * sin heading * sin heading (1 / 3))) pu fd
:r pd fd 1 pu bk :r + 1 rt 1]
end
```

Unfortunately, neither Imagine, nor MWLogo yields a correct curve:

Fig. 7. It is not nephroid

How can we modify the code to get a correct curve? Logo has misinterpreted the first parameter of power as it was a complex mathematical operation. Let us do the exponentiation separately (*s* and *c* denote the directions of *sin* and *cos*):

```
to nefroi :a
   repeat 360 [make "s sin heading
               make "c cos heading
               make "r :a * sqrt (1 + 1,5 * (power (1 + :c)
               * :s * :s (1 / 3) + power (1 - :c) * :s * :s
               (1 / 3)))
               pu fd :r pd fd 1 pu bk :r + 1 rt 1]
end
```

The curve has become symmetric but still not a nephroid. The problem is that Logo still misinterprets our mathematical expressions. Therefore, we reformulate them by further articulation to make the algorithm easier to understand:

```
to nefroid_stepwise :a
repeat 3600 [
make "c cos heading
make "s sin heading
make "cplus 1 + :c
make "cminus 1 - :c
make "ssquared :s * :s
make "esquareroot :cplus * :ssquared
make "msquareroot :cminus * :ssquared
make "esqrt power :esquareroot 0.333333
make "msqrt power :msquareroot 0.333333
make "product 1.5 * :esqrt
make "sqroot 1 + :product + :msqrt
make "root sqrt :sqroot
make "r :a * :root
pu fd :r pd fd 1 pu bk :r + 1 rt .1]
end
```

The result is:

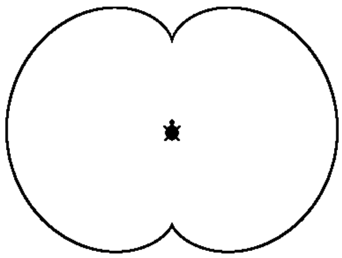


Fig. 8. Nephroid

We should avoid complex mathematical expressions! Logo prefers simple sentences. As Petőfi writes, "I cannot understand that even common people do not know or do not believe that simplicity is a general rule; those who do not have simplicity have nothing".

4 Electronic drama pedagogy

Divide the problem into subproblems, advises Polya in his book *How to Solve It* [3]. In the following, we generate curves by superposition using several turtles. Think of one as the tune and of the other as the accompaniment, they jointly create harmony.

Let us create three turtles. This way more special, more mathematic motions can be described with simpler motions and their superpositions. Adam's (blue), Eve's (red) and Cain's job is to execute the commands *:a, :b,* and both *:a* and *:b*, respectively. When *:a* is continuous movement ahead, *:b* is turning, then Cain will rotate [1]. Such a generation of *sin* was presented at the EuroLogo conference [4]. The superposition of two harmonic motions results in a Lissajous curve [5]; a special case of which is an ellipse [6]. When Eve partly adapts to Adam and lets herself being rotated, but also moves radial; her path is a spiral. Changing Eve's relative motion we will get various spirals [6].

We get a spectacular group of second order curves if our three turtles move while being tied to one another: Adam rotates, Eve rotates around Adam and Cain rotates around Eve [6].

After the above published examples, let us have a look at a new one.

5 Combining the two approaches to generate Maurer roses

A Maurer rose can be described as a closed route in the polar plane. A walker starts a journey from the origin, (0, 0), and walks along a line to the point (*sin(nd), d*). Then, in the second leg of the journey, the walker walks along a line to the next point, (*sin(n·2d), 2d*), and so on. Finally, in the final leg of the journey, the walker walks along a line, from (*sin(n·359d), 359d*) to the ending point, (*sin(n·360d), 360d*). The whole route is the Maurer rose of the rose $r = sin(n\theta)$. A Maurer rose is a closed curve since the starting point, (0, 0) and the ending point, (*sin(n·360d), 360d*), coincide. [7] Alternatively, a Maurer rose is a plot of a "walk" along an *n*- (or *2n*-) leafed rose in steps of a fixed number d degrees, including all cosets. [8]

Clearly turtle geometry is the best way to describe a motion like "closed route" or "walk along" type. This has been done in Scratch:

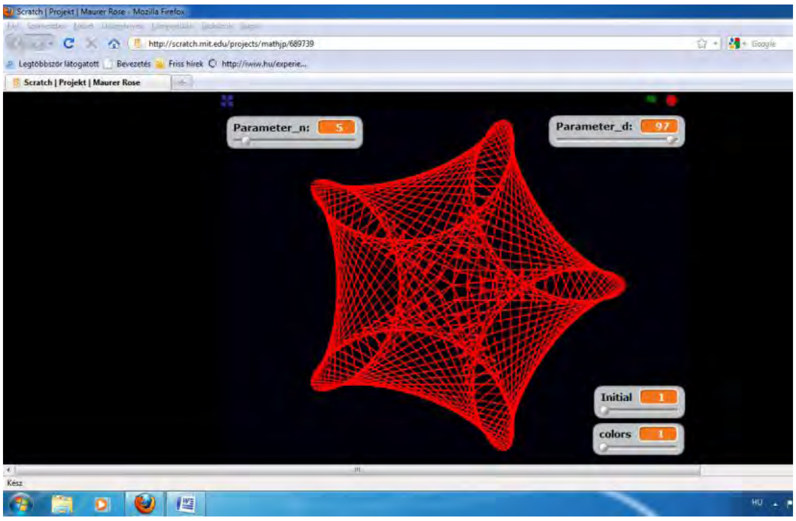


Fig. 9. A Maurer rose in Scratch

```
let us start from the origin,
the first target of our walk is the point [sin n*d, d]
the next segment goes from here to [sin n*2d, 2d]
the third segment takes us to [sin n*3d, 3d]
and so forth…
```

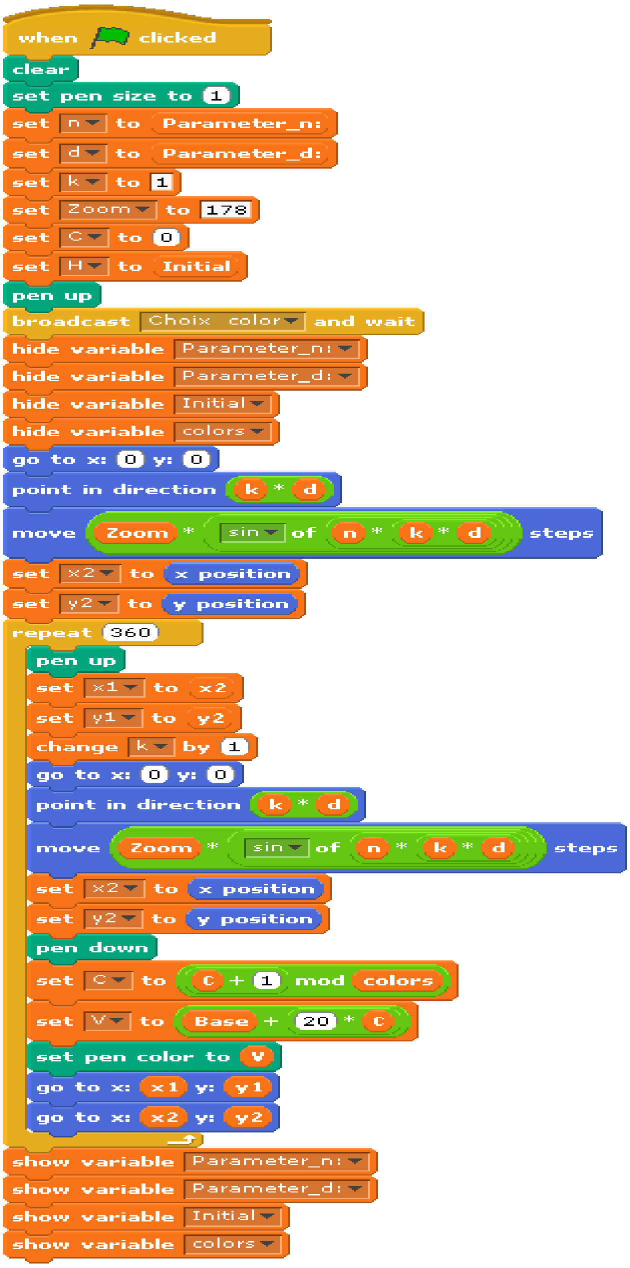


Fig. 10. Scratch code of Maurer rose

It is clear that the general form of the vertices is [*sin n*k, k*], where *k=0*d,1*d, 2*d, 3*d, ... , 360*d* , and *d* is a positive integer.

Our turtle's first position is the origin, the second position is the point P1 whose distance from the origin is *sin :n * :d* and it lies on the ray forming angle *:d* with the base direction (north).

The third position is P2 whose distance from the origin is *sin :n * 2 * :d* and it lies on the ray forming angle *2 *:d* with the base direction (north).

The code in Fig. 10 was downloaded [9]. We can see that the program Scratch definitely demonstrates the program structure using various colours and symbols. We shall write this in another way using MicroWorlds Ex Logo.

Let us have two turtles:

```
to stage
newturtle "Adam setcolor 116 st
newturtle "Eve setcolor 16 st
end
```

Eve designates the vertices one after the other. Adam will, after finding a new rose point, run to Eve. Connecting these points, results in a Maurer rose:

```
to maurer :n :d
make "k 1
Eve, pu
repeat 361 [Eve, seth remainder :d * :k 360 fd 200 * sin
:n * :d * :k
    Adam, towards "Eve fd distance "Eve pd
    Eve, bk 200 * sin :n * :d * :k
    make "k :k + 1  ]
end
```

The spanning curve which connects the vertices is constructed as follows:

```
to spanning :n :d
Adam, setpensize 3 setcolor 56 pu
repeat 360 [
Eve, seth :d  fd 200 * sin :n * :d
Adam, towards "Eve fd distance "Eve pd
Eve, bk 200 * sin :n * :d
make "d :d + 1]
Adam, setpensize 1 setcolor 116
end
```

The spanning curve, drawn in green, is the envelope of the Maurer rose. The Scratch rose above looks like this in our program:

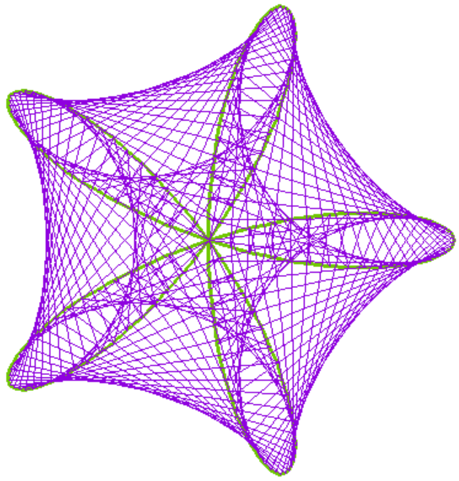


Fig. 11. spanning 5 97 maurer 5 97

If we insert wait 3 before the bracket in line 3, the generation of the polygon can be clearly followed.

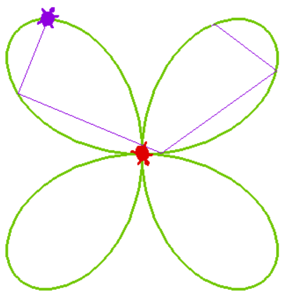


Fig. 12. The first 4 phases of one of the Maurer roses

One can clearly see the connection between the Maurer roses and closed cycloids.

6. Summary

In this paper we have discussed the epistemological issues regarding our approach, as well as presented some additional novel algorithms we have used in teaching informatics in a humanities secondary school.

We think that syntonicity is the vital concept of real turtle geometry. We think that intrinsic Logo commands are better than extrinsic ones when we students to understand the programs. Our examples are based on two model generation techniques. First, we put together the curve by "jumping out" of the origin and build the curve from “stretches”. Second, the curve is the combination of the work of several cooperating turtles, what we call “electronic drama pedagogy”.

We demonstrate how these two approaches can be combined to generate the Maurer roses in a novel way.

References

1. Csink, L., Farkas, K.: Turtle’s Curves. In: R. T. Mittermeyer M. M. Sysło (ed): Proceedings Informatics Education Contributing Across the Curriculum, Torun, Poland, 76-86 (2008)
2. Boytchev, P.: http://www.youtube.com/watch?v=1v5Aqo6PaFw [12 February, 2011]
3. Polya, Gy.: How to Solve It; Induction and Analogy in Mathematics. Princeton University Press, Princeton; and (1969) Patterns of Plausible Inference Princeton, N. J. Princeton. (1954)
4. Farkas, K.: Logo and native language. Intrinsic procedures of some curves. In: Secundino Correia (ed.): Proceedings of the 9th European Logo Conference, Porto pp. 69-79 (2003)
5. Foltynowicz, I.: Cicloids and limacons in the turtle graphics. In: Kalaš, I. (ed.) Proceddings of the 11th European Logo Conference, EuroLogo, Bratislava, Slovakia, August. p. 50 (2007)
6. Csink, L., Farkas, K., Tasnádi, I.: Playful Turtle Geometry in the Paradise. In: James E. Clayson and Ivan Kalas (ed): Constructionist approaches to creative learning, thinking and education: Lesson for the 21th century, Paris, p. 51 (2010)
7. http://en.wikipedia.org/wiki/Maurer_rose 28 [March, 2011]
8. http://mathworld.wolfram.com/MaurerRose.html [28 March, 2011]
9. http://scratch.mit.edu/projects/mathjp/689739 [January, 2011]
10. http://www.youtube.com/watch?v=XnOpOqWWP5o [28 June, 2011]

Information Education in ICT teacher education at the Faculty of Education in Prague

Miroslava Černochová

Charles University in Prague, Faculty of Education, Dept. of Information Technology and Technical Education, M.Rettigové 4, 116 39 Praha 1, Czech Republic
Miroslava.Cernochova@pedf.cuni.cz

Abstract. The author analyses and summarises the experiences of teaching of three compulsory pedagogical courses (ICT in Education, Systems of Open Teaching and Pedagogy for Information Education) to ICT student teachers and focussing on knowledge of instructional methods as a key component of pedagogical content knowledge in ICT teacher professional competency. In these courses, the student teachers are trained to be able to apply appropriate teaching methods for teaching ICT and Informatics in basic or secondary schools. The courses cultivate student teachers theoretical knowledge and practical abilities and facilitate their skills' development to be able to implement ICT into school practice in three main directions: ICT specialisation, ICT application across curriculum and ICT as a technological platform of educational and working environment.

Keywords: information education, ICT teacher education, informatics, curriculum, Moodle, programming

1 Introduction

The Faculty of Education at Charles University in Prague educates student teachers for teaching subjects to be found in the basic and secondary school education curriculum, including subjects concentrated on computer technologies. As a consequence of Bologna process, most of the five-years' Master´s degree study programmes at the Faculty of Education (including those for technical and information education) contained a set of compulsory, pedagogical and teaching methodological subjects. Experiences gained in these pedagogical subjects on how to teach ICT in schools are now exploited via a new set of pedagogical courses, re-conceptualised as two-years' Master´s degree study programmes for ICT or Technical and Information Education. The graduates of these study programmes will be able to work at basic or secondary schools as teachers of subjects focused on computer technology, ICT, Informatics or technical education.

1.1 ICT and Informatics in the curriculum for Basic and Secondary School Education

In the Czech Republic, Information and Communication Technology (in short ICT) is integrated into curriculum for basic school education (aged 6 -15) which is aimed at two crucial topics:

- Searching information and communication
- Information processing and exploration.

In these topics, great attention is paid to development of pupil´s skills of how to search information on the Internet, portals, libraries and databases, how to communicate via the Internet or other ordinary communication technology and how to work with texts and pictures in text and graphic processors.

Primary school pupils (aged 6-11) are trained to work with a computer in order to be able to use fundamental standard computer functions and hardware (HW), to respect rules for save work and behaviour with HW and software (SW) and to protect data against their damage, loss and misuse.

The educational domain, ICT, enables pupils to achieve a basic level of information literacy to be able to orient in the information world, to use information creatively and for learning and practical life [21].

The ICT literacy development continues at secondary schools (gymnasium) in the educational domain, Informatics and ICT, which contributes to systematic structuring of student´s knowledge [22]. It is focused not only on practical applications of ICT for learning and knowledge but also on the theoretical basis of ICT and Informatics. The Informatics and ICT domain is concentrated on three main topics:

- digital technology;
- information sources and searching information, using commonly available services of information networks (Internet, data, information, information systems, e-learning, ethical aspects of communication, copyright, personal data protection);
- and information processing and presentation (multimedia, algorithmic approach to solving problems, document publishing, typographical editing), databases, modelling, simulations, algorithms and programming languages.

The preparations to include Informatics and ICT into a state part of the graduation secondary school-leaving exam for which there has been developed a catalogue of requirements, demonstration exercises and tests ([2], [3], [16]) emphasise the importance of the educational domain, Informatics and ICT, in gymnasium education.

2 Teachers of subjects oriented on Informatics and ICT

The majority of teachers of subjects oriented around Informatics and ICT in basic schools or gymnasium in the Czech Republic are university graduates with a pedagogical specialisation. In spite of the fact that, in Czech schools, women are in the majority, in basic schools the representation of both male and female teachers of Informatics and ICT education is equal [20, p. 331]. Subjects focussed on ICT and Informatics at basic schools are taught by teachers with a teaching qualification in science, mathematics, informatics, social science or primary education. At the gymnasium these subjects are taught mainly by teachers qualified in informatics, mathematics, physics or technical education (computer science). There are fundamental differences between schools in what pupils do in ICT or Informatics lessons, what and how they learn these subjects. Teachers, their expertise and experiences are the most important factor for decisions as to which topics from ICT, Informatics or computer science will be taught (or learned). The teachers qualified in Informatics or teachers who have been programmers for computer companies struggle to implement the basics of programming into ICT education; for this purpose, they use Baltik, Baltazar, Imagine or KAREL as an example of educational programming languages [7]. The majority of teachers of ICT or Informatics at basic schools consider algorithmisation and programming as a marginal educational activity. This was established from a research study in 2007 at more than 900 basic schools in which the research team explored teachers´ preferences for teaching ICT topics. Among 14 items of ICT topics the teachers ranked algorithmisation and programming 13th just ahead of the introduction into working with database system and database design [15, p. 22]. The experiences of ICT student teachers gained during teaching practice in schools confirm that pupils in basic or secondary schools only very seldomly learn programming [7].

3 Information education in ICT teacher education

The Faculty of Education in Prague has ICT teacher education in the bachelor´s and master´s study programmes: “Information and communication education,” and “Technical and information education” which derived from reflections on the experiences of teaching of compulsory pedagogical ICT courses (see Table 1) integrated into a five- years study programme “Technical and Information Education”. In these pedagogical courses there have been developed didactic-methodological competences of ICT student teachers required for ICT and Informatics subjects at basic and secondary schools. The graduated ICT teachers are expected to be able to support teachers at schools to apply ICT in their teaching and actively participate in a transformation process of educational school environment into a modern educational institution.

Table 1. A list of compulsory pedagogical ICT courses implemented into a five-year master´s degree study programme “Technical and information education”

Semester	Course
8th semester	ICT in education
9th semester	Systems for open teaching
10th semester	Pedagogy for information education

3.1 Pedagogical courses focused on information education

Although the educational content defined in the curriculum [21], [22] is common ground for the above-mentioned pedagogical courses in Table 1, these courses are not concentrated just on particular (or typical) "the most appropriate" methodological approaches on how to teach each individual topics in ICT educational domains. These courses serve as a platform to investigate and search for and examine non-traditional teaching methods suitable for the teaching of ICT and informatics subjects and for information education in schools. These courses give the opportunity to develop competency in pedagogical content knowledge [9] as an important component of a teacher´s professional skills. Designing new pedagogical approaches the ICT student teachers must apply knowledge and skills gained in pedagogical, psychological and other professional branches and experiences from teaching practice.

In these pedagogical ICT courses, we endeavour to concentrate on the knowledge of instructional methods as a key component of pedagogical content knowledge[1] [12]. ICT development is extremely dynamic and it will be reflected in the educational content and in the role of ICT and informatics subjects in schools. Therefore, it is important that ICT student teachers are able to transform and adapt a new content "into shapes which are from the pedagogical point of view efficacious and tailored to pupil´s abilities" [23, p. 15].

ICT can be implemented into school education in three fundamental ways:

A) ICT specialisation (ICT literacy, computer science, informatics);
B) Application of ICT in subject areas and infusing ICT across the curriculum;
C) ICT as a technological platform of educational environment and working space of pupils/teachers (e-learning, cyber-learning, e-portfolio, personal learning environment).

Table 2. Specifics of pedagogical ICT courses in ICT teacher education

Course	Key questions	Focus	Assignments
ICT in education	Why integrate and use ICT in school education? How to apply ICT in school education?	A, B	Design of methodological approach on how to learn with selected applets
		B	Video-measurement
		A	Design of teaching aids for use in instruction of ICT topics
		A	Demonstration on how to teach ICT topics without using computers
Systems for open teaching	How to make use of VLE, LMS as a support for daily school teaching?	C	e-learning course designed in Moodle
Pedagogy for	Concept of Information Education	A	Taxonomy of assignments for

[1] Schulman [23] has introduced the concept of pedagogical content knowledge as „subject matter knowledge for teaching“ (selection of topics, useful forms of presentation, analogies, illustrations, examples, explanations and demonstrations) which "also includes an understanding of what makes the learning of specific topics easy or difficult, including knowledge about conceptions and misconceptions that students bring to the subject". [17, p. 23]

information education	specification		ICT and Informatics subjects
	Process data-information-knowledge-wisdom	A	Analysis and evaluation of textbooks for ICT and Informatics subjects
	Non-traditional teaching approaches to information education (constructivism, constructionism, inquiry learning, wiki, collaborative activities)	A	Samples of outcomes designed in educational programming languages

Pedagogy for information education is a key feature of ICT pedagogical subjects in ICT teacher education at the Faculty of Education in Prague. Information education is interpreted as a universal complete concept in which the implementation of computer technologies should dominate regardless of the subject being taught. Information education is education aimed at *information and communication thinking* development needed for learning, knowledge process, creative activities and collaboration in communities of learners. Information thinking can be applied in:

- Information acquisition and processing in knowledge structures' development;
- Process development of a personal information space using knowledge and skills from other branches;
- Solving problems of a creative nature demanding, in some cases, computational thinking;
- Seeking answers for questions demanding algorithmic thinking or database thinking, knowledge in programming or skills of how to use applications for modelling or animation;
- Data processing using appropriate methods and computer applications with the aim of testing a hypothesis;
- Learning with understanding using personal working space which integrates computer technology, tools for sharing and communication to sources, applications and experts and non-digitised study materials.

The ICT student teachers on these three pedagogical courses (see Table 2) work in enduring two-member teams. For collaboration on given assignments they can use any kind of technology including Moodle [18].

ICT in education:

This course is aimed to introduce ICT student teachers to the main functions and fundamental ways of using ICT applications in teaching and learning from different perspectives. The main emphasis is placed on both theoretical questions related to the manner and techniques of using ICT in educational processes (including life-long learning) and practical trials of particular models of how to integrate ICT at schools.

The key question WHY apply ICT in school education? serves as a starting point (terminus a quo) for lessons with ICT student teachers. This question usually takes them by surprise. ICT student teachers don´t expect this question. They can´t see their life, study and work without computers; they represent a generation who has come into a world full of computer technology [5]. In lessons we endeavour to identify the real, undisputed, pedagogical potential of computer technology and to expose key roles of ICT for the learning and teaching process. Student teachers frequently argue that computer technology contributes and supports learners in easier understanding of themes owing to its illustrative nature and that it makes teachers' work easier (e.g. comparison between written records and schematic drawing in geometry on the blackboard and using computer applications). The ICT student teachers also highlight the importance of ICT as a motivator for pupils to learn.

ICT student teachers need to understand the pedagogical potential and values of ICT for learning and teaching: how and what computer technology serves in schools. They should be able to argue for it makes sense ICT are inseparable from school education and learning environment. ICT student teachers cannot use in their rationale any hackneyed phrases. Understanding particular examples, which illustrate problems that could not be solved without computer technology, facilitates their thinking about the positive and real value of ICT in education.

Design of a methodological approach on how to learn with selected applets: In this task ICT student teachers prove not only their professional knowledge related to a problem illustrated by selected applets [19] but also the pedagogical competence to design activities for pupils to be able to work with applets, to ask questions and to understand and learn from illustrated processes. The tutorial material includes not only instruction on how to manipulate with applets and theoretical explanations needed for understanding the interactive model demonstrated by the applet but also a set of tasks, questions, activities and working sheets dedicated to pupils. The thing that appears to be the most difficult for ICT student teachers is asking questions and the ability to formulate interesting activities for pupils. In many cases, the reason is that they do not understand the applet from a professional point of view.

Video-measurement: The main reason why ICT student teachers do this activity is to show them how they can collect and analyse data for measurement using a digital camera and also to give them a chance to try an example of inquiry-based learning. Using a digital camera students record an appropriate object being in motion. They adapt the digital record using the appropriate SW (e.g. AVISTEP), analyse a time course of fundamental kinematic quantities and test their hypothesis [13].

Demonstration of how to teach ICT topics without using computers: What is typical for youth of today the so-called, "digital generation," is immediately after they are given a problem, they use mobiles or the Internet to seek counsel or help [1]. The notion of teaching some topics (ICT or Informatics subjects) without using computers or out of computer labs astounds the majority of our ICT student teachers. It is one reason why we ask them to design lessons and activities for particular topics for ICT and Informatics subjects without using computers - using real models and teaching aids. In such cases, we draw inspiration from YouTube (e.g. [4]) or in manuals and tutorials for teaching of other subjects.

Systems for open teaching:

The aims of this course are to introduce ICT student teachers to the theoretical underpinning of using ICT as a virtual educational environment and to enable students to try in a practical way some teaching models arranged in virtual environment, to design and create their own educational situations using technology of different types for communication and collaboration, educational content distribution, motivation pupils and teaching and learning assessment. Great emphasis is placed on a constructivist approach to learning organised in virtual education environment applications (see Table 3). A theme about learning objects and learning object repositories is also included in this course. ICT student teachers are primarily familiar with Czech portals [14]. Attention is given also to copyright and adherence to ethical rules (copyright, licence, etc.) when using different types of materials.

The theoretical part of this course is dedicated to explanation of the e-learning concept and how ICT can support fundamental processes important for learning (learning content distribution, communication, collaboration, feedback, etc.). A competition (who can find the highest number of e-learning concepts) stimulates and motivates ICT student teachers to read and think about e-learning. The concepts are then discussed in seminars. The competition is organised in Moodle and it also serves as an example of how to motivate and stimulate on-line students. In this course, ICT student teachers are introduced in main characteristics of virtual learning environments (VLE) and learning management systems (LMS) to support and manage education and the functions and services for learners, tutors, course designers, and administrators of Moodle. The students look for inspiration and drawing a lesson in analysis and evaluation of e-learning courses and activities for children offered by various companies and institutions.

In the course, we focus primarily on Moodle through which ICT student teachers design their own on-line courses and to criteria on how to evaluate e-learning courses. At the end of the course, each author team presents its Moodle course. During their university studies, some students work in schools as non-qualified teachers develop their Moodle courses for their teaching with pupils in schools and they present not only a Moodle course but also gain some personal experience with Moodle in schools.

Table 3. Main topics of "Systems for open teaching"

Topic	Theme
Introduction	Theoretical management basis of VLE and LMS. Basic concepts.
e-learning as a model of open teaching in school practice	ICT for learning support. Teacher's activities and the role in instructions with on-line support. Pupil´s activities in ICT support learning. Advantages and disadvantages of on-line support of F2F education in schools.
Technologies for e-learning in school practice	Technologies for educational content design and distribution via networks. Technologies for collaboration in networks. Technologies for synchronous and asynchronous communication, collaborative activities via networks. Example of using video-conference for education (activities, sharing SW)
Computer systems for management of pupil's learning	Examples of commercial and Open Source VLE/LMS for full-time F2F education
Introduction to Moodle	Main characteristics and options of Moodle for different groups of users. Introduction to particular Moodle functions and tools for designers and teachers.
Assignment: Scenario Moodle course design	Scenario design project for e-learning in which it could be applied to LQs and different technologies to support pupil's learning. Criteria for e-learning course evaluation.

Moodle and its educational potentialities: Most of the exercises given to ICT student teachers in the course, “Systems for open teaching,” are included with the aim to give students the opportunity to practise some specific and interesting learning activities in Moodle. A ***Glossary*** is one of them. During the semester, students develop a glossary of concepts related to e-learning and systems for open teaching. Each student publishes explanation of concepts including bibliographic references, links and resources. Students make remarks and comments to all published concepts and discuss them. ***Database*** is another very interesting activity which we train with ICT students, for example, when they look for events in “on-line live transmission” suitable for education (e.g. scientific expedition in the Antarctic, watching a life of animals, ZOO).

Moodle course design: e-learning courses design in Moodle is an opportunity to contribute to student teacher’s pedagogical competencies’ development. Student teachers elaborate a scenario of their Moodle course in which they describe a target group to which the course is dedicated, course aims (what on-line learners have to be taught), a course structure, a way how to work and study in the course, if there will be a tutor or not, if the course will support school activities in formal education or learning activities as a part of informal education.

Some outcomes and issues:

Students can opt for a topic of their Moodle course from disciplines and specialisations of their studies. Since 2004, there have been developed more than 60 on-line courses for various school subjects: ICT (31%), Math (19%), Visual Arts (17%), Social Science (14%), or Languages (10%). They differ not only in quality, but also in structure, number of sections and design [6]. The most common components are study materials (14.4 materials/course), assignments (usually 9 assignments per course), tests and cross-word puzzle in HotPotatoes. Lectures or database activities have very seldom been implemented. The student teachers gained step-by-step experience with on-line activities at the Faculty of Education. They see the role of discussion forums in the educational process therefore the number of discussion forums involved in their course design has diminished [6].

Some students have developed flash animations for better understanding of study materials of on-line courses. Constructivist approaches to pupil learning prevail in courses designed by Mathematics and Art Education student teachers.

Not all on-line courses created by student teachers are done well. During the on-line courses development the student teachers are faced with different types of problems:

- There is no theory on how to design e-learning courses for children. Course designers develop their on-line courses intuitively.
- Student teachers cannot verify their on-line course with pupils. In the course, “Systems for open teaching,” there is no time to test designed Moodle courses in schools with children or to enable student teachers to try the role of on-line teacher; therefore, they have no experience of how to integrate ICT into a teacher daily planning (how to organise work, etc.). Nevertheless, some students exploit their Moodle courses for research activities in their diploma thesis, which means they get feedback to what degree the on-line course fulfil its function.
- ICT student teachers make a lot of mistakes in assignment formulation; most of them aren´t good at asking questions to pupils and articulating interesting problems. We could meet with this incompetence in “ICT in education” course. Although student teachers have completed a compulsory course how to

construct tests they still have a lot of difficulty in formulating high-quality test questions.

- ICT student teachers are not sufficiently competent to design on-line courses with a constructivist approach. These professional abilities prevail mainly among Mathematics and Art Education student teachers. The most likely explanation lies in the teacher education ideas and approach of the Department of Art Education and the Department of Mathematics Education where very sensitively constructivist approaches are applied in all courses.
- Some ICT student teachers lack good professional knowledge. Our experiences indicate that designers who don´t understand a topic are not able to develop a good on-line course.
- Although student teachers are taught to apply copyright and licences in their on-line courses, they do not very often respect these principles.

Pedagogy for information education:

This course is focused on pedagogy-methodological questions for teaching of compulsory or facilitative ICT and Informatics subjects at basic or secondary schools. The main emphasis is on non-traditional teaching and learning methods (wiki, inquiry learning, etc.). Information education and its components serve as a starting point for contemplation on how to transform data into information and information into knowledge. Students analyse and evaluate textbooks used in schools for ICT education. Attention is given also to programming languages – we don´t discuss about which language is the best one, we try to identify all positive arguments as to why it is important to programme in the basic or secondary schools. Student teachers are introduced into Imagine and Scratch programming. The course does not serve as a course of introduction into programming, but as a platform to facilitate understanding of Logo culture in a sense described by professor Ivan Kalaš [10] and comprehension of the importance of algorithmic thinking development as an integral part of information and communication thinking and computer culture in the education of a new generation of pupils.

Table 4. Main topics of "Pedagogy for information education"

Topic	Theme
A concept of Information Education A concept of information	Explanation of two key concepts: information education and information Data-Information-Knowledge-Wisdom Knowledge structure (concept mapping) Database thinking. Information thinking
Information education in curriculum	ICT in Curriculum for basic education ICT and Informatics in Curriculum for secondary school education
Teaching methods suitable for information education	Constructivism. Constructionism. Wiki. Inquiry Learning
Taxonomy of assignments for ICT and Informatics subjects	Assignment analysis and evaluation.
Analysis and evaluation of textbooks for ICT and Informatics subjects	Textbook analysis and evaluation.
Algorithmisation and programming	Educational programming Logo. Imagine. Scratch.

Conclusion

Experiences acquired from these three pedagogical courses in teacher education could be applied to the new strategy of ICT teacher education at the Faculty of Education in Prague which proposes to introduce five pedagogical courses focused on information education into two-year Masters' degree study programmes. In the implementation of this new concept of ICT teacher education we would like to pay deep and comprehensive attention to the analysis of professional knowledge and skills of ICT student teachers and to interconnect these five courses with their teaching practice. We plan to introduce e-portfolio as a tool for monitoring and assessment of their professional progress and for the identification of problems in their undergraduate education. In the pedagogical training of ICT teachers we would like to reflect particularities, specificities and the needs of ICT teacher profession. As for knowledge of instructional methods as a key component of pedagogical content knowledge in ICT teacher professional competence, we are at the beginning. For ICT and Informatics subjects, the analysis of knowledge database needed for teacher professional knowledge does not currently exist.

References

1. Brown, J.S. Learning in the Digital. In the Internet and the University : Forum (2001). pp. 71-72.
2. Informatika. Základní úroveň obtížnosti. Katalog požadavků zkoušek společné části maturitní zkoušky platný od školního roku 2011/2012 [on-line]. CERMAT, 2010. [cit. 25.3.2011]. Available on <http://www.novamaturita.cz/katalogy-pozadavku-1404033138.html>
3. Informatika. Vyšší úroveň obtížnosti. Katalog požadavků zkoušek společné části maturitní zkoušky platný od školního roku 2011/2012 [on-line]. CERMAT, 2010. [cit. 25.3.2011]. Available on <http://www.novamaturita.cz/katalogy-pozadavku-1404033138.html>
4. Computer Science Unplugged. http://video.google.com/videoplay?docid=-5129662873097337591#
5. Conole, G. A holistic approach to designing for learning: a vision for the future [on-line]. [cit. 25.3.2011]. Available on <http://www9.code.ouj.ac.jp/sympo-2010/report/pdf/06_conole_en.pdf>
6. Černochová, M. Návrh a vývoj elearningových aplikací jako příležitost k rozvíjení pedagogických kompetencí budoucích učitelů. In Sborník DIDINFO2008. L.Huraj (ed.) Univerzita Mateja Bela, Fakulta prirodných vied, Banská Bystrica, 2008. p.25.
7. Černochová, M. Teaching practice from the perspective of ICT student teachers at the Faculty of Education, Charles University in Prague. In Key Competencies in the Knowledge Society. IFIP TC 3 International Conference, KCKS 2010. Held Part of WCC 2010 Brisbane, Australia, September 2010, Proceedings. N. Reynolds, M.Turcsányi-Szabó (eds.) IFIP AICT 324. ISSN 1868-4238. Springer, Printed in Germany. pp. 44-55.
8. Dede, C. Theoretical perspectives influencing the use of information technology in teaching and learning. In J. Voogt, G. Knezek (eds.) International Handbook of Information Technology in Primary and Secondary Education. Springer, 2008. pp. 43-62.
9. Janík, T. Didaktické znalosti obsahu a jejich význam pro oborové didaktiky, tvorbu kurikula a učitelské vzdělávání. Paido : Brno, 2009.

10. Kalaš, I. et al. Konštrukcionizmus, od Piageta po školu v digitálnom veku [CD-ROM]. In Sborník DIDINFO 2011. Invited Papers. G. Andrejková (ed.) Univerzita Mateja Bela, Fakulta prirodných vied, Banská Bystrica, 2011. pp. 7-19.
11. Katedra informačních technologií a technické výchovy, PedF UK v Praze. http://it.pedf.cuni.cz
12. Krauss, S. et al. Pedagogical Content Knowledge and Content Knowledge of Secondary Mathematics Teachers. Journal of Educational Psychology, 2008, 100(3), 716-725.
13. Lustigová, Z. Záznam pohybu a jeho zpracování (AVISTEP aneb fyzika s videem). [on-line]. [cit. 25.3.2011]. Portál Telmae. MFF UK : Praha, 2009. Available on http://telmae.cz/Experiments/lectures.nsf/0/348CFE980C7821C78025739F0051D4E7.
14. Metodický portál. RVP. Portál DŮM. http://dum.rvp.cz/index.html
15. Mudrák, D. et al. Dotazníkové šetření aspektů informační výchovy na ZŠ. In POŠKOLE2007. Sborník Národní konference o počítačích ve škole. M. Černochová, I. Fialová, B. Mannová (eds.) Praha, 2007. pp. 15-28.
16. Nová maturita. http://www.novamaturita.cz.
17. OECD Teachers´Professional Development. Europe in international comparision. Teaching and Learning International Survey. EU : Belgium, 2010.
18. Online kurzy. Univerzita Karlova v Praze Pedagogická fakulta. http://moodle.pedf.cuni.cz
19. PhET. Free Interactive Simulations. http://phet.colorado.edu/index.php
20. Rambousek, V. et al. Výzkum informační výchovy na základních školách. Koniáš : Plzeň, 2007.
21. RVP ZŠ Rámcový vzdělávací programme pro základní vzdělávání. VÚP : Praha, 2007.
22. RVP GV Rámcový vzdělávací programme pro gymnaziální vzdělávání. VÚP : Praha, 2007.
23. Shulman, L.S. Knowledge and teaching: Foundations of the new reform. Harvard Educational Review, 1987, 57, pp.1-22. Cited in JANÍK, T. Didaktické znalosti obsahu a jejich význam pro oborové didaktiky, tvorbu kurikula a učitelské vzdělávání. Paido : Brno, 2009.
24. Na středních školách se začal naplno projevovat demografický pokles, na základních školách se pokles žáků částečně zastavil [on-line]. [cit. 24.4.2011]. Tisková zpráva. ÚIV : Praha, 31. srpna 2010. http://www.uiv.cz/soubor/4330

Status and Trends in Educational Robotics Worldwide with Special Consideration of Educational Experiences from Greek Schools

Nikolaos Detsikas & Dimitris Alimisis,

School of Pedagogical and Technological Education (ASPETE), Achaikis Simpoliteias 20, 26441 Patra, Greece

detsikas@gmail.com
alimisis@otenet.gr

Abstract. Teaching programming concepts to novice learners and particularly school students is often a difficult task for numerous reasons. Innovative technologies, and specifically robotics, have allowed for constructivist theories to be applied in that cause with encouraging results. This work describes educational initiatives involving the use of robotics throughout the world with special focus on educational experiences coming from Greece. In addition to that, a case study is presented in which robots built with Lego Mindstorms kits were used in a Greek secondary education school classroom of informatics to support the learning of decision and repetition control programming structures.

Keywords. Robotics, constructionism, LEGO Mindstorms, programming, control structures.

1 Robotics in education at an international level

1.1 Projects and activities in school settings

Modern technology tools can be used in school classrooms to promote the teaching and learning processes. In order for technology to be adopted effectively in the context of the above cause, suitable learning theories have to be employed. Educational robotics, being inspired by the theory of knowledge construction (constructivism, [1], focus on the constructivist use of educational technology (constructionism) as learning tool [2]. According to the constructivist view, the process of learning should include authentic activities regarding the solution of real world problems and encourage a strong personal involvement by the learners as well as social interaction. In addition to that, constructionism suggests that learners build knowledge more effectively when they participate in activities in which they design and implement objects of the real or digital world, either they are sand castles or Lego robots and computer programs (Papert, 1992).

Throughout the world, many educational projects and activities have taken place in the past years, involving robotics either as a learning object or as a learning tool, though the distinction of the two meanings is not always clear. Some of the most important of these initiatives, varying from classroom activities to competitions and exhibitions, are summarized here.

The Lifelong Kindergarten group, located within the MIT Media Lab (http://llk.media.mit.edu/) aims at using new technologies and a kindergarten style of learning to help people of all ages learn through playful activities. The people that participate in the projects learn by designing, creating, experimenting and exploring, in a variety of activities ranging from engineering to drawing and constructing melodies projects. Many of the projects involve robotics like the Interactive C and Learning Engineering by Designing Robotics.

The Science, Engineering, NASA Site of Remote Sensing (SENSORS) project was an effort towards introducing upper elementary and middle school audiences to tele-robotics and remote sensing. The robots were rover vehicles based on LEGO RCX units and were remotely controlled by the students through the WWW.

Significant initiatives have taken place in Italy an example of which is the Robot@Scuola network. Robot@Scula is a network of Primary and Secondary Professional and Vocational Education schools that use robotics in their teaching processes (http://www.scuoladirobotica.it/progettieng.htm).

Another example is The PIONEER (PIedmOnt NEt for Educational Robotics) project which is also an Italian schoolnet for the educational use of robotics in school classes. Its goal is to promote Papert's constructionism in a cooperative environment for setting up a model of minirobot programming experiences that can support the standard curricula for school years K-12 [3].

[4] describes the efforts undertaken by a small community of teachers towards boosting science education in the school district of Verona (Italy) by promoting constructivism with the help of various configurations of robotic devices. These efforts have been going on for the last eight years, slowly gaining momentum and impact. They emphasize that the most striking difficulties have been encountered with the educational environment rather than with students themselves.

The Roberta-Goes-EU project (http://www.iais.fraunhofer.de/3845.html) aspires to encourage young people, and especially girls, through Robotics to take up engineering studies, providing training courses and comprehensive teaching materials to teachers and others who wish to increase students' enthusiasm for technical professions.

The Computers in Education Group of South Australia (CEGSA) suggests the use of Bee-Bots http://cegsa.editme.com/EDET3302-BeeBot#uses) and Lego Robotics WeDo (http://cegsa.editme.com/EDET3302-LegoRoboticsWeDo#uses) in classrooms Bee-bots are programmable floor robots designed for young children (4-7 years). They can be used in the teaching of many curriculum areas including arithmetic, literacy, arts and science. Similar uses and classroom scenarios are suggested for older students (7-11 years) with the use of he Lego Robotics WeDo kits.

As described in [5], three higher education initiatives have taken place in Sri Lanka, Ghana, and the USA. The project focused at integrating robotic technologies in developing communities, studied the future involvement of robotics in education and their ability to contribute to sustainable development. The work presents the challenges faced as well as the benefits of the activities and the factors that contributed to their success.

The use and effectiveness of robotics in the teaching of other scientific topics has also been demonstrated by [6]. Their work describes an educational activity based on robotics and compares it with a similar computer-simulated activity. The subject was related to the study of graphs and kinematic concepts. The activity was carried out with the use of a set of handheld devices and a robot with a wireless connection. The robotic activity proved to be almost twice as effective as compared with the computer-simulated activity, since the results showed that students demonstrated significant better understanding of the graph concepts. Moreover, the robotic activity proved to be highly motivating for the students and fostered collaboration among them.

There are also numerous research efforts that have focused on the integration of Robotics in Early Childhood Education with the design, development and practicing of attractive activities with digital technologies at preschool age [7], [8].

Educational projects targeting technical and vocational school students have also taken place, engaging them in designing, building and programming a robotic device that allowed them to explore phenomena of mechanics like the gear-aided transmission of motion [9] or the gear function and mechanical advantage [10]. [11] describes a project-based learning environment in which various robotic construction tasks based on LEGO Mindstorms have been undertaken by middle-school students and highlight some sample products of their work.

Finally, the use of robotics in the teaching of computer programming concepts, from elementary ones to most advanced, has been explored and attempted numerous times. The famous Logo turtle has been widely used to support the first steps towards programming for many young students throughout the world. Similarly, Karel the robot [12], a robot with built-in capabilities that allow it to move in its world and manipulate simple objects in it, has been instructing students in the practice of programming since 1981. In addition to that, [13] argues about the benefits of using robots (Lego Mindstorms) in the teaching of object-oriented languages and provide sample exercises. The appeal of computer controlled models in teaching programming with the Java language has been studied by [14].

2 TERECoP Project and new European initialives

The TERECoP Project (2006-09) (Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods, www.terecop.eu), takes a different view in the involvement of robotics in education. Spanning over 8 educational institutions and 6 European countries (Greece, Spain, Czech Republic, Italy, France and Romania) and coordinated by ASPETE (School of Pedagogical and Technological Education) in Patras, Greece, it focuses on teachers' training in the effective introduction and use of educational robotics in school classrooms.

The TERECoP Project distinguishes robotics projects and activities in two categories according to the role that robotics play in the learning process:

- Robotics as *learning object*: where robotics is being studied as a subject on its own.
- Robotics as *learning tool*: robotics is proposed as a tool for teaching and learning other school subjects at different school levels. Robotics as learning tool is usually seen as an interdisciplinary, project-based learning activity drawing mostly on Science, Mathematics, Informatics and Technology and offering major new benefits to education in general at all levels.

The TERECoP partnership has developed a training methodology for future and in-service teachers aiming to engage them in robotic activities that they could implement in a creative way with their own students. The proposed course curriculum, being consistent with the proposed use of robotics as a tool for constructivist learning, is meant to train teachers in the same way in which they are expected to educate their school students.

Teacher training courses have been taking place in several European Countries participating in TERECoP since October 2007. The courses were evaluated by the TERECoP partnership. The trainees have been prospective and in-service teachers. The LEGO Mindstorms NXT system (http://www.legomindstorms.com) was selected among others as an appropriate technological framework that attempts to partner technology with the ideas of constructionism.

The pilot course curriculum (Alimisis, Moro, Arlegui, Pina, Frangou, & Papanikolaou, 2007) included briefly the following sections (30 hours):

- “Breaking the ice” introductory activities and agreement on a "didactic contract"
- Robotics as learning object
- Constructivism, Constructionism and project-based learning principles
- Robotics as a learning tool

The TERECoP training activities are also consistent with the idea of “learning by design”. The learning tasks of the course are organized as small or large scale robotics projects enabling trainees to design and develop their own products. All the training materials, guides for trainers and teachers, exemplary robotics projects and other resources have been included in a book co-authored by the TERECoP partnership [15].

The continuously increasing interest in Educational Robotics was confirmed in the Workshop organized at the SIMPAR Conference 2010 held in Darmstadt, Germany (http://www.terecop.eu/SIMPARworkshop.htm). Researchers and practitioners from 12 European Countries (plus participants from Brazil, Japan, Panama and USA) participated in the workshop, which demonstrated both the diversity and similarity of European and international education robotic projects represented by very simple robots suitable for kindergarten to very complex ones for higher education.

New plans for future initiatives were presented in the workshop, among them the e-Robot project which aims to investigate educational robotics principles through a longitudinal on-line research collaboration and to set up a community-based online research collaboration that will inspire, gather data and collate multiple research projects using the Roamer® robotic system [16]. The event concluded that cooperation at European Level would offer great benefits by enhancing the existing expertise in the field. Suggestions to this direction were discussed for future joint activities including new workshops on educational robotics within the major robotics or education conferences, contacting and lobbying to governmental education authorities in order to integrate robotics in school curricula, submission of proposals for funding a network at European level to bring together the different initiatives and competitions, to exchange experience, to coordinate the relevant activities and to run long-term evaluations [17].

3 Robotics in Education in Greece

3.1 Projects and activities in school settings

Several innovative educational activities have been designed and attempted in Greek schools at all levels in the past years, involving robotics.

Teachers who have been trained by the TERECoP partnership were encouraged and supported to introduce robotics in their classrooms in a variety of activities and in different school subjects. For example, [18] following the methodology developed in the TERECoP Project, designed and executed an educational activity for the teaching of computer science, physics and technology topics in informatics school courses. The activity involved the design and programming of a catapult device with the use of Lego Mindstorms Robotics kit.

A pilot program for the introduction of robotics in schools of Athens was realized during April-May 2010 by a consortium consisted of the non-profit organization World Robotics Olympiad Hellas, the Athens Municipality and the Educational Technology Lab of ASPETE in Patras, Greece following the TERECoP methodology. 68 pupils aged 11-14 years from 6 schools (3 primary and 3 lower secondary ones) located in 6 different areas of the Athens city centre participated in the program. The educational objectives of the program included the familiarization of pupils with robotics technology, the acquirement of technological skills in mechanical construction and in programming, the cultivation of valuable mental skills such as creativity and critical thinking and the development of teamwork and collaborative spirit of work [19].

The reported results were in all the above cases encouraging and constituted the integration of robotics in the standard curriculum of schools almost a necessity.

Other initiatives have also been undertaken in Greece. [20] have studied the use of LEGO Mindstorms Robotics kit as an instructional tool in technical and vocational secondary education. In the context of their study, a robot car was developed and programmed, in order to be used in teaching Mechanical Engineering topics in classes of the field. The study also describes the construction of the robot car and suggests a course plan exploiting this technology.

[21] have carried out a research concerning the effectiveness of the use of robotics in elementary school classes in teaching environmental issues. During the study, the students have been involved in designing and building robots that could assist in recycling, garbage collection and fighting water pollution. The project has shown that, through the use of a teaching tool, as interesting and appealing to young students as the robots are, the concern and interest of young people in environmental issues can be increased.

It is worth noting the pilot projects realized by the Educational Technology Lab of our School in Patras, which offer distance teacher training in educational robotics based on the use of Synchronous Audiographic Conferencing. During those distance training sessions, the learners were provided the opportunity to use distantly programming software available only in the trainer's computer in order to program a remote robotic vehicle located at the trainer's desk, download their program to the robot and watch through a video-camera the reaction of the robot while executing their commands sent distantly [22].

3.2 Educational Robotics in public events

WRO Hellas is a non-profit company whose aim is to promote the use of science, modern technology and in particular robotics and automation in education and social life. WRO Hellas is constantly active in educating all interested citizens, with a strong emphasis on youngsters and students, in matters of technology and robotics and their integration into social life. More information can be foun at www.wrohellas.gr (in Greek).

WRO Hellas coordinates its efforts with the WRO (World Robotics Olympiad). WRO is an annual international robotics competition, whose participants are young students interested in robotics.

WRO Hellas has been organizing the "Panhellenic Educational Robotics Competition" for the past three years. The "Panhellenic Educational Robotics Competition" is an annual robotics competition, whose participants are students and schools of all education levels, as well as all interested individuals. Participants build robots and compete with each other on various scenarios.

WRO Hellas has also been organizing educational activities at schools of Athens and other parts of Greece. During the class sessions, students, grouped into teams, had been designing and implementing robots with the use of the Lego Mindstorms platforms.

Finally, 'Polymechanon', located in Athens, Greece, is a recreation and exhibition center that hosts multiple high technology games related to robotics informatics, communications and virtual reality. 'Polymechanon' accepts visitors of all ages, but is especially suited to school visits, during which students are involved in fun, entertaining and educating activities with the high technology games that are exhibited there [23].

4 Case study: Using robotics to support the learning of control structures in programming languages

4.1 Context

The traditional use of a blackboard and chalk in teaching various science topics at school is often not enough to demonstrate the main ideas, key concepts examples and applications. Particularly, in teaching Computer Science and informatics subjects even the use of the standard computer programming environments and simulators is often incapable to introduce the students to the new topics, since they appear too abstract and out of reach to them at the beginning.

The teaching of the notions of programming to secondary school students, who have little or even no prior programming experience, is a difficult task that involves the use of specially designed computer programming environments adapted to the students' level. Students face a lot of difficulties in understanding the programming concepts, as is shown by numerous studies [24]. In particular, as [24] describe, "conditional statements", in other words computer commands that control the flow of execution of a program, cause a lot of difficulties to students.

There are many reasons for these difficulties the most frequent of which are pre-programming knowledge that leads to misconceptions and programming languages that are usually too advanced for novice learners. It is also of great importance the fact that the problems that students are called to solve in their initial programming steps, such as mathematical problems, are often inadequate for teaching purposes. On the contrary, meaningful computational models able to provide immediate feedback to their programmers [25] are more appropriate for educational purposes.

It quickly becomes evident that apart from the programming environments that are often used in the computers of secondary school computer laboratories, less abstract teaching tools and activities should also be incorporated to introduce students to the ideas of programming.

The teaching of Computer Science and informatics in secondary Greek schools, covers the topics of the standard programming control structures which are loops and decision statements i.e. IF-THEN-ELSE, DO-WHILE etc. Since these programming concepts are the fundamental part of meaningful programs, their understanding is crucial to students.

Although the above structures can be taught with the use of various programming environments in computers, their use can be made a lot more intuitive if robots are incorporated into classrooms. Robots, such as moving vehicles, can be used to vividly demonstrate repetitive behavior and also changes in behavior according the external conditions observable by students.

A method of programming the robots is also essential in the above process of teaching these programming concepts. It is absolutely necessary that this method or environment is simple enough for students without significant prior programming knowledge to understand and use. Otherwise, instead of aiding, the above process would add an extra difficulty in the course of understanding control structures, which would be the difficulty of learning the robot-programming environment.

Lego Mindstorms, which were introduced in 1999, is a great tool that fulfills the requirements of the activity described above. Lego Mindstorms extend the idea of standard Lego kits with the use of sensors and motors and a programmable unit called “the brick”. The brick accepts input from the sensors and sends commands to motors according to the logic it has been programmed with. The variety of sensors available (sound, light, ultrasonic, collision etc.) combined with the ability to use multiple motors, allows for very complex and interesting designs from moving vehicles to robots with moving limbs.

Lego Mindstorms programming environment allows for students to program very simple to very complex robot behavior without writing any code. Instead, commands, statements and structures are represented by graphical blocks, which can be combined into programs with the well-known drag-n-drop method. Their parameters, whenever needed, can be selected in drop-down menus. Therefore, it becomes evident, that the skills needed by students in order to use the Lego Mindstorms environment are already present in them due to their experience of computers in the school computer laboratory and possibly from everyday use.

The robotic activity can greatly motivate students to participate in it since it gives them the opportunity to get involved in a palyful task which is meaningful to them and has a strong playful character. Allowing the students to think of their own robotic behaviour, according to the principles of constructionism, enables them to create their own little stories and become part of them.

4.2 Activity

This section describes a teaching activity that was carried out to secondary school students, in the context of the Computer Science class. The activity took place in the school computer laboratory.

The goal of the activity was to aid the teaching of decision and repetition statements in programming, with the use of robots. Robots, appropriately programmed, can perform simple actions, which can demonstrate the use of decisions and repetitive behavior in programs.

The robots used in this activity were simple cars with four wheels, one motor and one ultrasonic sensor. The motor was attached to the front wheels. The car lacked the ability to turn, which was not necessary for the simple projects demonstrated in the class. The ultrasonic sensor could measure the distance to nearby objects. The motor and sensor were both connected to the robot-processing unit (“the brick”), the first to an output port and the second to an input port.

Since the activity focused on the programming concepts, constructing the robots was beyond its scope. In addition to that, the available time was very limited. Therefore the robots that were finally used in the classrooms were all constructed in advance.

Nevertheless, beside the standard Lego parts, the building blocks of the robots, i.e. the motor, sensor and brick, as well as the connections had to be explained to students. Students were also explained the flow of information from the sensor to the processing unit and then to the motor. In addition to the above, the students were also demonstrated the basic operating instructions of the processing unit, in other words how one can start it and stop it.

Students were also shown the programming environment that accompanied the Lego Mindstorms robots, which was also installed in their computers. Through examples, the basic building blocks of a program (move, wait, conditional wait, loop, switch etc.) were explained to them along with the steps necessary to build a program and download it to the processing unit installed on a robot.

The examples presented to the students, were those of a robot moving forward until it detects an object at a certain distance and of a robot moving forward and backwards for a certain number of times. The examples involved the whole process of constructing the program, building it, downloading it to the robots, executing and verifying that the robot behavior was actually the one programmed.

This concluded the first part of the activity, which was the introduction to the notion of robot programming.

Students were asked to freely think of a behavior, involving decision making and/or repetition and then describe it using paper and pencil before programming it to their robots in the second part of the activity. They were grouped into five teams of three to four persons and each team was given a robot to execute its programs on. At the end of activity, the groups were asked to present the behavior they thought of and demonstrate it with their robots in front of the whole class.

4.3 Results and conclusions

Students described the behaviors they thought of in their working sheets, as requested. Although their actions resembled the ones presented by the instructors they tried to deviate from them while still using repetition and conditional execution. As results shown, conditional behavior was the most difficult and not always present. Two indicative examples from the actions the students presented below, while the rest of the projects were similar to these:

Case 1

1. The robot waits for 1 second
2. The robot moves forward forever at speed 100%
3. If the robot detects an obstacle 20cm away it stops
4. The robot moves backward at speed 75% for 2 seconds
5. The robots repeats steps 2-4 twice

Case 2

1. The robot waits for 1 second
2. Moves forward for 4 seconds
3. Moves forward for 2 seconds
4. Moves backward for 1 seconds
5. Repeats steps 3-4 twice

With some help from the instructors most groups managed to program the actions to the robots with considerably little difficulty after some trial and error attempts. Although all teams completed the activity, one of them did not use conditional statements at all and another did not manage to program it effectively. Although the SWITCH or CONDITIONAL WAIT blocks of the programming environment were explained, students in many occasions had trouble to implement their thoughts on their own and instructors had to describe the use of these blocks again according to individual needs. The repetition proved to be less difficult but in some cases students tended to repeat the action blocks by cascading them in the programs instead of putting them in LOOP blocks.

At the end of the activity, the instructors collected the work sheets of the groups. In addition to that, the programs the groups built were also collected. Students also anonymously filled in simple questionnaires about the course of the activity, the difficulties they faced and what helped them understand the discussed topics.

The collected feedback as well as the instructors observations in the classroom, have verified the initial assumption that a robotics activity would be appealing to the students and could be used successfully in bringing abstract concepts closer to the students.

While working in the activity, students had the opportunity to gain experience and take advantage of their own errors. In this manner, they were able to make their own plan towards solving the problem and analyze every possible case and outcome. While cooperating in their groups, they exhibited group thinking and team spirit. Through the LEGO Mindstorms activities, students developed a better view of their own learning abilities. The robot activity enabled them to see the results of their actions and get immediate feedback, which increased their self confidence not only in the cases of success but also in the error cases, since the learners could see their mistakes and correct them. The above are better described by their reactions towards the instructors after a final successful attempt, which were of the form “we did it!”, “we were good, weren’t we?”, “It works! Do you like it?”, “What else should we program?”. Often, they were willing to show their work to their friends and explain them how they had done it and how they had programmed the robot.

Our role as teachers was also different from the traditional teacher role in the classroom. We acted mostly as experienced advisors, encouraging the students towards the solutions but not doing the work for them. We also had the opportunity to discover the difficulties students faced when they worked out the new concepts, to understand how students preferred to work and how they thought and felt and finally to gain insights on how our future educational activities should be planned and designed.

In general, programmable robots can be used as a preparation step into the actual programming for young students but can also be used as simulators in Computer Science courses in all education levels. They can also be instrumental in teaching subjects involving physics, motion and mechanics.

This project constitutes a small-scale study and for that reason general conclusions cannot easily be drawn from the findings. The areas that emerged in the analysis raised new questions for exploration. The activity, as described here, should take place in more schools and classrooms. In addition to that, allowing for more complex robot designs to be used in the activities, with more sensors and motors, will enable learners to think of and program more elaborate actions, that will cover the aspects of programming more extensively. Finally, the activities should be allowed to take up more time than a few classroom sessions. This way, students will have the opportunity to test more designs and programs and try harder to accomplish them. A potential competition among the classes that participate in the activities might greatly motivate students towards this direction.

References

1. Piaget, J.: To understand is to invent. New York (1974)
2. Papert, S.: The Children's Machine. New York (1992)
3. De Michele, S., Demo, B., & Siega, S.: A Piedmont SchoolNet for a K-12 Mini-Robots Programming Project: Experiences in Primary Schools. In: SIMPAR 2008, Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, (pp. 90-99). Venice(Italy) (2008)
4. Fiorini, P., Galvan, S., Giuliari, L., & Pighi, L.: It Takes a Village... to do Science Education. In: SIMPAR 2008, Intl. Conf. on SIMULATION,MODELING And PROGRAMMING For AUTONOMOUS ROBOTS, (pp. 43-53). Venice(Italy) (2008)
5. Dias, M. B., Mills-Tettey, G., & Nanayakkara, T.: Robotics, Education, and Sustainable Development. In: 2005 IEEE International Conference on Robotics and Automation, (pp. 4248 – 4253) (2005)
6. Mitnik, R., Recabarren, M., Nussbaum, M., & Soto, A.: Collaborative robotic instruction: A graph teaching experience. Computers & Education (2009)
7. Bers, M., Ponte, I., Juelich, C., Viera, A., & Schenker, J.: Teachers as Designers: Integrating Robotics in Early Childhood Education. In: Information Technology in Childhood Education Annual , pp. 123-145 (2002)
8. Pekarova, J.: Using a Programmable Toy at Preschool Age: Why and How? In: SIMPAR 2008, Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, (pp. 100-111). Venice(Italy) (2008)
9. Alimisis, D., Karantrantou, A., & Tachos, N.: Technical school students design and develop robotic gear-based constructions for the transmission of motion. In: Gregorczyk G., Walat A., Kranas W., Borowiecki M.. (eds.). Eurologo 2005, Digital Tools for Lifelong Learning, Proceedings, (pp. 76-86). Warsaw (2005)
10. Chambers, J., & Carbonaro, M.: Designing, Developing, and Implementing a Course on LEGO Robotics for Technology Teacher Education. Journal of Technology and Teacher Education , 209-241 (2003)
11. Carbonaro, M., Rex, M., & Chambers, J.: Using LEGO Robotics in a Project-Based Learning Environment. The Interactive Multimedia Electronic Journal of Computer-Enhanced Learning (2004)
12. Pattis, R. E., Roberts, J., & Stehlik, M.: Karel - The Robot, A Gentle Introduction to the Art of Programming. New York: Wiley (1995)
13. Lawhead, B. P., Bland, G. C., Barnes, J. D., Duncan, E. M., Goldweber, M., Hollingsworth, G. R., et al.: A Road Map for Teaching Introductory Programming Using LEGO Mindstorms Robots. In: ACM SIGCSE Bulletin, (pp. 191-201). Reno, Nevada, USA (2003)

14. Barnes, D. J.: Teaching introductory Java through LEGO MINDSTORMS models. In: Proceedings of the 33rd SIGCSE technical symposium on Computer science education, ACM (2002)
15. Alimisis, D.: Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods. ATHENS: ASPETE (2009)
16. Catlin, D., & Blamires, M.: The e-Robot Project: A Longitudinal On-Line Research Collaboration to Investigate ERA Principles. In: Proceedings of SIMPAR 2010 Workshops, (pp. 411-420). Darmstadt (Germany) (2010)
17. Bredenfeld, A., Hofmann, A., & Steinbauer, G.: Robotics in Education Initiatives in Europe - Status, Shortcomings and Open Questions. In: Proceedings of SIMPAR 2010 Workshops, (pp. 568-574). Darmstadt (Germany) (2010)
18. Vounatsos, G., & Mega, A.: The TERECoP methodology and LEGO Mindstorms in secondary education: a case study. In: Proceedings of the 6th Panhellenic Conference on ICT in Education. Syros (Greece) (2011)
19. Alimisis, D.: Introducing robotics in schools: post-TERECoP experiences from a pilot educational program. In: Proceedings of SIMPAR 2010 Workshops, (pp. 575-585). Darmstadt (Germany) (2010)
20. Moundridou, M., & Kalinoglou, A.: Using LEGO Mindstorms as an Instructional Aid in Technical and Vocational Secondary Education: Experiences from an Empirical Case Study. In: Dillenbourg, P. & Specht, M. (Eds.): Times of Convergence: Technologies across learning contexts, Lecture Notes in Computer Science, Vol 5192, Springer-Verlag, Berlin Heidelberg, 312-321 (2008)
21. Anagnostakis, S., & Makrakis, V.: Educational Robotics as a tool for technological literacy and environmental sustainability: an action research with primary school pupils. In: Proceedings of the 7th Panhellenic Coference on ICT in Education. (pp. 127-136). Univ. of Peloponnese (2010)
22. Alimisis, D., & Plessas, A.: Teacher Distance Training in Educationl Robotics through Synchronous Audiographic Conferencing: towards a Learner-centered Approach. In: Proceedings of ED-MEDIA 2011-World Conference on Educational Multimedia, Hypermedia & Telecommunications. Lisbon (Portugal) (2011)
23. Kynigos, C.: Black-and-white-box perspectives to distributed control and constructionism in learning with robotics. In: SIMPAR 2008, Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, (pp. 1-9). Venice(Italy) (2008)
24. Doukakis, D., Tsaganou, G., & Grigoriadou, M.: Using animated interactive analogies in teaching basic programming concepts and structures. In: Proceedings of the ACM Conference on the State of: Informatics Education Europe II, (pp. 257-265). Thessaloniki (Greece). (2007)
25. Guzdial, M.: Programming Environments for Novices (2003)

Thinking Informatically

Karl Josef Fuchs[1] and Thomas Schiller[2],

[1] Paris Lodron University Salzburg, Hellbrunnerstr. 34,
5020 Salzburg
karl.fuchs@sbg.ac.at
[2] BG / BRG Ramsauerstraße,
4020 Linz
th-schiller@gmx.net

Abstract. The paper accommodates the move-in of basic informatical concepts into schools. The discussion is focused on Modelling representatively. After required theoretical inputs the article describes Modelling-aspects in an instructional concept for character recognition. Referring to the instructional conception the paper tries to catch a glimpse of evolutions in practice by a slight questioning on Modelling. The views of the presentation are strongly governed by the Austrian developments in secondary level education.

Keywords: Functional and Object Oriented Modelling, Secondary school, Character Recognition, Students' Perceptions of Modelling

0 Prolog

As teaching Computer Science has only been with the Austrian schools for a short time of 25 years ([3], [11]) teaching has geared towards central informatical concepts by and by in recent years initially. Some various characterizations and catalogues of ideas ([2], [5], [13], [16]) have come to the fore. *Modelling* has evolved into a general accepted strategy among Computer Science Educators.

1 The Prototypical Character of Modelling

We bring the high profile of *Modelling* in educational publications up as a first argument for our decision to choose this idea as a prototype for the development of Computer Science Education. Representatively we want to quote the books of Peter Hubwieser where the idea is a pivotal structuring element ([9], [10]) or the publication of Andreas Schwill and Sigrid Schubert (2004, [14]). These authors instance *Modelling* as an Fundamental Idea explicitly.

Modelling as technique in terms of Computer Science has a strong focus on *Implementing* [6] wherewith our view centers on *Programming* and *Algorithmising* which are traditional fundamentals of Computer Science. We bring up this fact as a second argument for our choice. We base the evidence of our statements on the following definitions. The examples are taken from publications written in German. We hope that the message which we want to put over to the reader don't suffer from our translations.

We start with Hans Haas and Detlef Wildenberg (comp. [8]) who carry on our strategies mentioned before in their definition published 1982: '*Programming* aims at the *implementation* of *algorithmus* (and data structures) on Computers.' Later on Gerald Futschek and Peter Hubwieser built a bridge to the idea of *Modelling* by their characterizations: 'When *Programming* the computer scientist draws up a *model* to solve a given application problem firstly. The initial *model* will be modified stepwise to other *models* which will always become more detailed and more formal. This iterative process will be run through until a final *model* coded in a program language will come out' [7] and '*Programming* means *Implementation* of abstract *models* with the aim to visualize and to prove the character of the *models* through simulation' [9].

Finally the target-orientation of teaching is our third argument for choosing *Modelling*. In particular we refer to one of the central items in Lorin Anderson's and David Krathwohl's well accepted Taxonomy (2000, [1]). It addresses the ability to apply strategies and concepts in new situations.

Now we change our focus and look at the practice.

2 Modelling Aspects in an reality related Instructional Concept - Character Recognition

This chapter illustrates prototypically possible content which could be treated in classroom. It requires modelling aspects, especially functional[1] and object oriented modelling[2], essentially.

2.1 Functional modelling

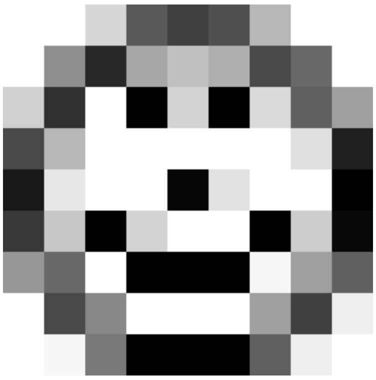
Before you deal with character recognition you will have to work on principles of digital image processing. This can be done without many special requirements in class, only simple knowledge of spreadsheets is required. The functional model behind spreadsheets implies the importance of the role that *functional modelling* plays.

There are many types of digital raster images such as photos, colour and greyscale images, screenshots, fax documents, radar images or ultrasound images in practice. Raster images are rectangular usually and made of regularly arranged picture elements so-called pixels. In general raster images are rectangular and differ mainly by the values stored in the pixels. Contrary implementations to raster images are

[1] Photos (e. g. greyscale images) normally are represented as (greyscale) functions. (comp. chapter 2.1 – e. g. [4], p. 5 and p. 10)

[2] Modelling of lines (and characters) as objects is useful to be able to detect those objects in raster images. (comp. chapter 2.2 – e. g. [4], p. 155ff)

Vector graphics ([4], p. 5). Tracing back to those one can interpret a recorded raster image as a matrix with numbers. ([4], p. 10)



255	255	215	88	64	80	183	255	255
255	143	40	167	191	175	72	104	255
207	48	255	0	211	0	218	96	159
72	183	255	255	255	255	255	223	32
24	231	255	255	6	225	255	255	0
56	199	0	211	255	255	0	204	8
151	104	255	0	0	0	247	159	96
255	72	135	255	255	255	159	64	239
255	247	120	0	0	0	96	239	255

Fig. 1. Part of an image (magnified) and related greyscale function (comp. figure 2.5 in ([4], p. 11))

The digital image is more formally. A greyscale function $f(i, j) = g$ with positive integers as arguments i and j e. g. generates non-negative image values from 0 to 255. In this way images can be represented, stored, processed, compressed or transmitted by computers. It doesn't matter which way an image has emerged actually. We simply understand it as numeric data. ([4], p. 10) Character recognition deals with finding special patterns in such a matrix with numbers to be able to convert it into a vector graphic or a string.

Many effects such as the sharpening or smoothing of an image can be realized with the help of filters. ([4], p. 89). Students can reconstruct elementary principles of digital image processing through functional modelling which means by manipulating pixel values in a spreadsheet to simulate linear filtering. They will be enabled to detect possible edges (changes of the intensity (of the pixel values) (comp. [4], p. 117)) in pictures for example.

The steepness of the greyscale function represents the strength of an edge. It is nothing but the amount of the gradient and the direction of the edge perpendicular to the direction of the gradient. ([15], p. 174f) Since edges separate objects (e. g. characters) from the background the knowledge of edges may be useful for example in character recognition and therewith in the detection of complex objects (e. g. faces). Students can reconstruct the edge detection by simulating the differentiation of the greyscale function by approximating[3] the derivatives by differences in Excel by the use of linear filtering (comp. [15], p. 175). As mentioned above an edge is a place of great change in intensity according to Burger et al ([4], p. 117). Thus this basic idea behind the edge detection using functional modelling can be implemented in classroom much earlier because the demands will become more suitable even for the lower grades.

[3] Approximation is considered as another well accepted fundamental idea in computer science and mathematics. [13]

To get a visualization of results (new pixel values) you can convert the values back to an image or just zoom out of the matrix:

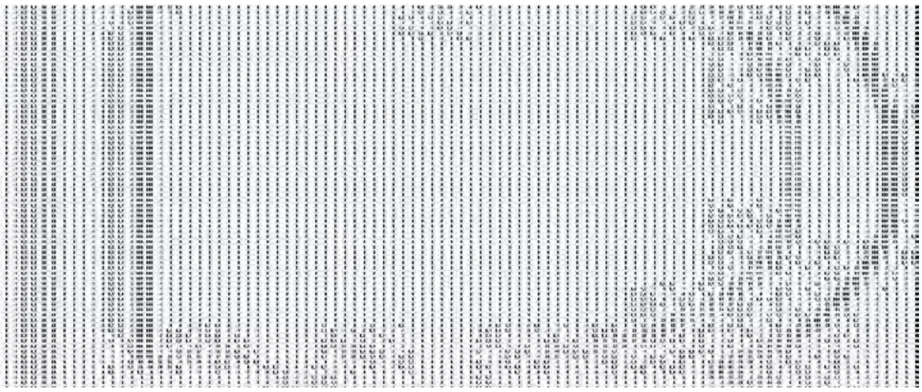


Fig. 2. Matrix of pixel values (part of the letter “B” after simulating a (functional) edge detection algorithmus)

As mentioned above, digital raster images are represented as functional models and therefore functional modelling helps students to reconstruct elementary principles of digital image processing and to be able to detect possible positions of edges.

2.2 Object oriented modelling

Now we will change our point of view and focus on patterns (e. g. straight lines) in our images which means detecting characters (e. g. consisting of straight lines). *Object oriented modelling* will come to the fore. This helps us to detect objects (e. g. straight lines or characters) in an image (a matrix of separate pixel values, e. g. after running an edge detection algorithmus).

The so-called Hough transform (e. g. described in Burger et al ([4], p. 155ff)) is a concept to detect parametric geometric forms (e. g. lines and circles) described by few parameters. For example lines can be described by the radius[4] and an angle[5]. You must find these parameters to detect a line. ([4], p. 155ff) Radius and angle are attributes of lines. Hence it will be useful to discuss an object oriented model of lines.

If you decide to use a professional method for detecting lines (e. g. the just mentioned Hough transform) in class it will be profitable to consult an already arranged code for discussion from a methodological point of view. Thus the students will be able to concentrate on central modelling and accuracy aspects. Even the supplementation of dynamic geometry or computer algebra software is useful to get a better visualisation of the modelled situation.

Certainly students can implement (primitive) line detection algorithms on their own, e. g. a brute force algorithm searching for all possible lines in the image.

[4] distance to the origin ([4], p. 160)
[5] deviation from the horizontal axe (comp. [4], p. 160)

Anyway the implementation should be an object oriented one taking care of the attributes (and methods[6]) of a line, because the algorithm should detect objects (e. g. straight lines and later on characters) in a raster image (a matrix of separate pixel values, e. g. after running an edge detection algorithmus). For example the students may decide to implement lines by choosing a two points-strategy in continuation of the functional modelling results (raster images).

Possible implementations for points and lines (without getter- and setter-methods) could be the following:

```
public class Vec2D {
          // coordinates of the pixel in the image
          private int x;
          private int y;

          public Vec2D(int x, int y) {
            this.x = x;
            this.y = y;
          } // Vec2D

          public Vec2D minus(Vec2D v) {
            return new Vec2D(this.getX() - v.getX(),
            this.getY() - v.getY());
          } // minus
}         // Vec2D

public class Line {
          private int a;
          private int b;
          private int c;
          private Vec2D p1;
          private Vec2D p2;

          public Line(Vec2D p1, Vec2D p2) {
            // line through points p1 and p2
            this.p1 = p1;
            this.p2 = p2;

            // direction vector
            Vec2D rVec = p1.minus(p2);

            // orthogonal vector
            Vec2D nVec = new Vec2D(-rVec.getY(),
            rVec.getX());

            this.a = nVec.getX();
            this.b = nVec.getY();
            this.c = nVec.getX() * p1.getX() +
            nVec.getY() * p1.getY();
          } // Line
```

[6] A line is an object that „does“ nothing. “Acting” methods are necessary therefore.

```
            public boolean equals(Object o) {
             Line g = (Line) o;
             if ((this.k() == g.k()) && (this.d() ==
             g.d())) {
              return true;
             } else {
              return false;
             }
            }

            // k ... slope of the line
            public double k() {
             return (p2.getY() - p1.getY())*1.d /
             (p2.getX() - p1.getX());     // or k = -a/b
            } //k

            // d ... distance on the y-axsis (-> section
            point line/y-axis)
            public double d() {
             return c*1.d/b;
            } //d
}           // Line
```

The last class consists of the two defining points as attributes and different methods for calculating the slope and other attributes (based on the two points). These other attributes may be needed for the implementation ideas of the students for implementing a (brute force or (later on) systematic) detection algorithm, e. g. to be able to compare the slope of given lines.

3 Students' Perceptions of Modelling

Additionally we were interested in the effects of the instructional concept especially in those in the direction of Programming in terms of Modelling. Hence we added a slight questioning finally.

3.1 General conditions and Questionaire

The evaluation was made in a special form in the BG / BRG Ramsauerstraße, a grammar school in Linz, Austria, where computer science is taught from grade 9 up to grade 12 (age 15 to 18) obligatory (see following table).

Table 1. Amount of computer science lessons a week in a special form of a grammar school in Linz.

Grade	Amount of computer science lessons a week

9	3
10	2
11	2
12	1

Furthermore these special forms are separated in classes where each student has his / her own notebook at all hours (attribute N) and computer-classes where lessons are held in computer rooms whenever access to computers is needed (attribute S). We want to mention that we do not bring this separation into account in our evaluation as we decide to concentrate on effects in these forms in general.

Theoretically and practically we decided to apply methods of action research in the way of Peter Posch and Herbert Altrichter [12].

The questionnaire is structured in a specific and a general part. The single items of the specific part deal with attributes concerning modelling (Remembering, analysing, illustrating, implementing, applying and synthesizing – q1a-q1e), those of the general address the relevance of modelling for general education (Maturity in terms of everyday life, operating- and decision-, communication-competences – q2a-q2c). Each item is bipolar (from -5 as lowest rate up to 5 as highest rate).

3.2 Selected results

Additionally two side comments:

The free software PSPP[7] was used for evaluation and LibreOffice Calc[8] for histogram paintings.

The *validity* of our items was tested via the correlation between Remembering (q1a) and 'Knowing the vocabulary' -item (q1ei) as well as illustrating (q1c) and synthesizing (q1eiV) which should be strong as expected. The results (based on a population of 87 students) are as follows: $r_{q1a,q1ei} = 0.38$ and $r_{q1c,q1eiv} = 0.85$. Both values are positive. Especially the second one indicates high correlation.

Additionally we present the analyses of the attributes analysing (q1b) and applying (q1eii) graphically as they indicate core properties of modelling after our conviction.

[7] PSPP – a software for statistical analysis (http://www.gnu.org/software/pspp/)
[8] www.libreoffice.org

q1b

Value	Frequency	Percent
-4	3	3,45
-3	2	2,30
-2	4	4,60
-1	3	3,45
0	6	6,90
1	6	6,90
2	15	17,24
3	19	21,84
4	21	24,14
5	8	9,20
Total	87	100,0

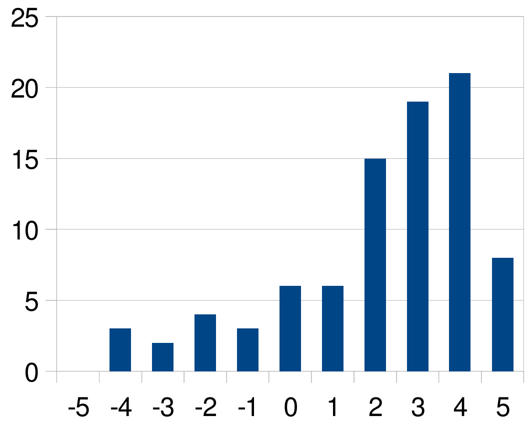


Fig. 3. Frequencies of question q1b “Mark the rate of the following attribute playing a role in Modelling: Analysing (which means stepwise structuring of a given problem)”.

q1eii

Value	Frequency	Percent
-4	2	2,30
-3	7	8,05
-2	7	8,05
-1	5	5,75
0	14	16,09
1	10	11,49
2	12	13,79
3	17	19,54
4	7	8,05
5	6	6,90
Total	87	100,0

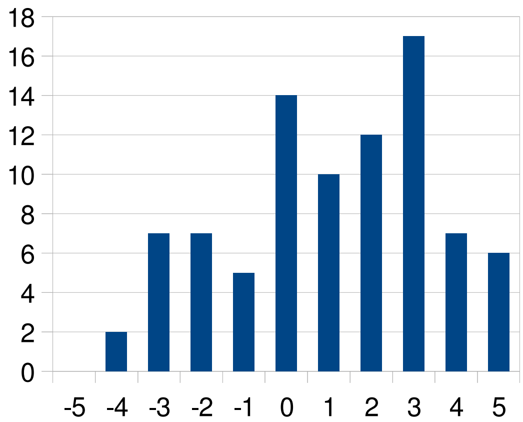


Fig. 4. Frequencies of question q1eii “Mark the rate of the following attribute playing a role in Modelling: Implementing II (which accents the 'programming'-competence) → Applying efficient program structures (conditional branch, iteration, recursion)”.

Even essential aspects of general education can be identified in the evaluation results. Exemplarily we state the graphical representation for Maturity in terms of everyday life (q2a).

q2a

Value	Frequency	Percent
-5	7	8,05
-3	5	5,75
-2	7	8,05
-1	2	2,30
0	13	14,94
1	12	13,79
2	13	14,94
3	15	17,24
4	6	6,90
5	6	6,90
.	1	1,15
Total	87	100,0

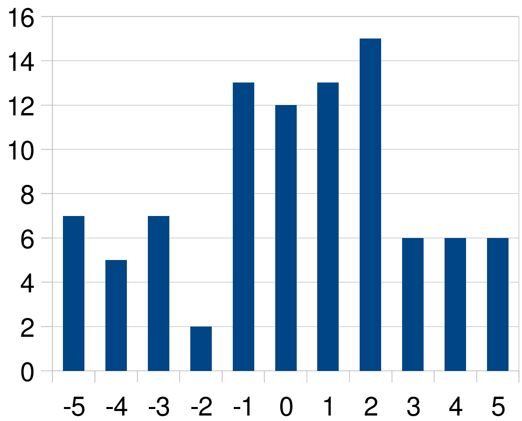


Fig. 5. Frequencies of question q2a "Rate the relevance of Modelling in terms of ... Maturity (which means the ability of coping with everyday life problems (ranging from social to technical questions))".

We interpret the results as very convincingly. One can accept that the instructional concept centered on modelling left desired effects with the students.

4 Perspectives and future duties

Computer science in school is a very new subject in many countries worldwide. Hence big application must be invested by teachers and computer science educators in formulating basic principles independently of time as these ideas will help to structure modern purposeful teaching. The bundle of well-discussed and well-approved techniques will trendsetting characterise the students' attitude 'Thinking Informatically'.

References

1. Anderson, L & Krathwohl, D et al (2000). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Allyn & Bacon:Boston.
2. Armoni, M (2010). On Teaching Topics in Computer Science Theory – Part II: Making it Possible by Using the Prism of Fundamental Ideas. In: ACM Inroads, Volume 1, Issue 4, p. 18 -19.
3. Brandhofer, G et al (eds. 2010). 25 Jahre Schulinformatik in Österreich: Zukunft mit Herkunft (*25 years of Computer Science in schools in Austria: Sourced future*), books@ocg.at: Vienna.

4. Burger, W et al (2005/2006). Digitale Bildverarbeitung, Eine Einführung mit Java und ImageJ. 2. Überarbeitete Auflage. Springer: Berlin, Heidelberg. ISBN 978-3-540-30940-6 (Print) bzw. 978-3-540-30941-3 (Online) (SpringerLink), ISBN-13 978-3-540-30940-6; (This book is also available in English: Burger, W et al (2008). Digital Image Processing, An Algorithmic Introduction using Java. Springer. ISBN 978-1-84628-379-6)
5. Fuchs, KJ (1994). Didaktik der Informatik: Die Logik fundamentaler Ideen (*Computer Science Education: The Logic of Fundamental Ideas*). In: Medien und Schulpraxis, 4+5/1994, p. 42 – 45.
6. Fuchs, KJ (2005). How Strict May, Should, Must the Borders be Drawn? In: Innovative Concepts for Teaching Informatics (Micheuz, P. et al eds.). Carl Ueberreuter: Wien, p. 7 – 14
7. Futschek, G (1990). Informatik als Wissenschaft (*Computer Science as a Scientific Discipline*). In: Didaktik der Informatik (*Computer Science Education*) (Reiter, A & Rieder, A. eds.). Jugend & Volk: Vienna.
8. Haas, H & Wildenberg, D (1982). Informatik für Lehrer – Band 2: Komplexere Probleme und Didaktik der Schulinformatik (*Computer Science for Teachers –Volume 2: More Complex Problems and Computer Science Education*). Oldenburg: Munich, Vienna.
9. Hubwieser, P (2007). Didaktik der Informatik: Grundlagen, Konzepte, Beispiele (*Computer Science Education: Basic principles, concepts, examples*). Springer: Berlin.
10. Hubwieser, P et al (2004, 2007, 2009, 2010). Informatik 1, 2, 3, 4 (*Computer Science 1, 2, 3, 4*). Klett: Stuttgart.
11. Micheuz, P (2010). 25 Jahre Schulinformatik (*25 years of Computer Science in schools*). CD – Austria Sonderheft des bm:ukk.
12. Posch, P. & Altrichter, H. (2007). Lehrerinnen und Lehrer erforschen ihren Unterricht, 4. Auflage (*Teachers analyse their lessons, 4th edition*). Verlag Julius Klinkhardt: Bad Heilbrunn
13. Schwill, A (1993). Fundamentale Ideen der Informatik (*Fundamental ideas in computer science*). In: Zentralblatt für Didaktik der Mathematik (*International Reviews on Mathematical Education*), Volume 25, Number 1, p. 20 – 31.
14. Schwill, A & Schubert, S (2004). Didaktik der Informatik (*Computer Science Education*). Spektrum Akademischer Verlag / Elsevier GMBH: München.
15. Tönnies, K D (2005). Grundlagen der Bildverarbeitung (*basics of image processing*). Pearson Studium. ISBN 3-8273-7155-4
16. Zendler, A. & Spannagel, Chr. (2008). Empirical foundation of central concepts for computer science education. ACM J. Educ. Resour. Comput. 8, 2, Article 6 (June 2008), 15 pages.

Let’s talk about Internet [1]

František Gyárfáš
Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská Dolina, Bratislava, Slovakia, gyarfas@ii.fmph.uniba.sk

Abstract. Internet is a new world. It is young, growing extremely fast, changing rapidly. For our children it is a natural part of their environment. There are many serious questions concerning Internet: ethical, philosophical, social, legal, etc. We do not have a set of answers, but it is important to discuss these questions with young people. It could help them to understand this new digital world and develop proper ethical and social behavior, and trust social relations and also to be careful with them.

Keywords: Internet, digitalisation, social networks, virtual love, plagiarism, file sharing.

1 Behind the mirror

Another world grows close to us. In fact, it is very close: the distance is one click. We have built this world. Its name is Internet.

Internet as we know it exists for only about a quarter of century. It started with few pages serving scientists for communication and information exchange. Nowadays it serves billions of people for almost everything. It consists of hundred billions of pages, addresses, pictures, music, videos. It is growing unbelievably fast: exponentially. It is not only growing, it is also changing continuously. We did not know five years ago how it would look like today and now we have no clue how it will look like three years from now.

There are no courses of Internet literacy. We are learning it every day, adapt to new software, new content, new customs, new netiquette, and new dangers. All of us are testers of its usability. We judge its quality: criteria are acceptability, popularity and usefulness. More and more of our information, duties, working tasks, social activities, cultural events, and love affairs are moving into this world. We spend on Internet large parts of our days and nights. We are starting to be net citizens.

Most of adults still remember times before Internet. We grew up with books, cinemas, phonebooks, maps, working places. Nowadays many of these things are

[1] Acknowledgement: this article has been written at the Department of Applied Informatics of Faculty of Mathematics, Physics and Informatics at Comenius University in Bratislava, and VEGA MŠ SR, Project n. 1/0688/10 supported it.

disappearing and their contents move to Internet. We have to adapt to it and we do. We are changing our schedule, customs, behavior, and principles.

Situation of our children is different. They are growing directly into Internet. For them Internet is not a new world, but an integral part of their living environment. They communicate, play, learn, invent, shop, travel, love and hate in Internet. They download music, watch movies, write school essays, chat with friends, publish photographs, write blogs, create videos, play multiplayer games, socialize on Facebook, ask Google, and learn from Wikipedia.

Alice went through the mirror and came into a world of miracles [1]. We go through such mirror every day. But our children are inhabitants of both worlds.

A lot of things in this strange world are different from our customs, moral, even law: torrents, hackers, open source, Google, Wikipedia, Wikileaks, plagiarism or netiquette. But we can still recognize strong similarities. We should take lessons from our ethical principles, knowledge management, copyright laws, freedom of speech, and protection of property.

Even if we do not know the answers, we should talk to our pupils and students about these questions. Discuss phenomena of Internet, show examples of similar situations, and warn about apparent or hidden dangers. Ask them about their opinion and try to formulate solutions acceptable for them. Discussion about Internet should be a part of education.

2 What to discuss

2.1 Limits of growth

Internet is a giant network of computers, servers, routers and cables. They cover almost the whole planet. A map of these technical devices is another layer to a map of our continents, countries, and cities. But this is not the real map of Internet. Such map is organized differently; it does not resemble our globe but an outer space. The visualization can look like this:

Fig. 1. Internet map. Image: Internet Mapping Project, Bell Labs/Lumeta Corporation http://www.physorg.com/news151162452.html

Shining dots on this map are web pages, portals, social networks, email boxes and other Internet addresses. It does not matter if a web page is physically stored in Europe or in Australia. The map does not change if we move a portal from one server to another, from one country to another continent. This map depends on links.

It contains hundred billions of sites, web pages, Wikipedia entries, photographs, music or movies. Number of them is growing exponentially. Intuition tells us that this increase must crash into limits of growth. Up till now it did not.

What helps until now is Moore's law. Computer specialist Gordon Moore discovered already in 1965 that number of transistors per square inch on integrated circuits had doubled every year. Ten years later he increased this period to 18 months. But nevertheless his law is valid for a half of a century. Not only a capacity of integrated circuits grows with this speed, but also memory, speed of calculation, etc. Similar growth can be found also in Internet. [2]

Questions

Are there any limits of Internet growth? How would these limits look like: would it be an end of our knowledge or a capacity overflow?

2.2 All knowledge of the world

It seems that we are transferring all our knowledge into the Internet. It is obvious with new texts. If we produce a new text for publishing, either we type it directly into a computer or it will be typed there in the process of editing, printing or reproduction. A similar happens with pictures, movies, or music. There are still some limits though: three-dimensional objects like statues or architecture, food, wine, or furniture. But at least their recipes, plans and documentations are stored on Internet.

On the other hand, there are billions of old documents stored in archives of this world, millions of books, paintings, symphonies, maps, movies, or photographs. They are stored on paper, canvas, vinyl and celluloid, in formats far from digital.

There are strong arguments to digitalize complete archives, libraries, and galleries. The main reason is access time and cost. It is much more expensive to get a piece of data stored somewhere on paper or celluloid. It is complicated to search for it, find out, where it is located, visit that institution, ask for permission and get it.

The comparative advantage of digital archive is so high that effectiveness puts pressure on digitalization of whole archives, no matter how many millions of artifacts they contain. We see this movement already now: Google Books, BBC, national programs of digitalization, and many other projects.

Mere digitalization is not enough. It is necessary to transform scans of texts into digital letters, describe content of pictures, add names to faces and locations, add written dialogues to movies. The ultimate goal is not only a digital reproduction, but completion of semantic analysis as well to be able to provide effective search.

Questions

Should all knowledge of human history be digitalized? Who will pay for that? Should be an access to digital knowledge charged or should be for free? Should we protect our knowledge against misuse or against future Artificial Intelligence?

2.3 Power of search engines

There are billions of objects in this digital world. Most of them are linked together. We can jump from any place to the other. But a problem is how to find something. Fortunately there are specialized programs that help us to search through Internet – search engines (SE). The most popular SE are Google, Yahoo!, Bing, Baidu (China).

SE are based on three main activities: **web crawling** – programs travel over the Internet to access and bring all pieces of digital knowledge to SE, **indexing** – analyzing found pages and incorporate them into huge indexes, **searching –** finding in indexes relevant answers to user questions. It is not only important to find all relevant answers but also to sort them reasonably. With several thousands or even millions of answers, the best of them must be sorted within first few pages of results. Very successful in proper sorting is a specialized algorithm **PageRank** used in Google. It sorts pages according to number of links that are pointing to them. More popular pages (with higher rank) linked to a page give a page higher rank as well. This means that not all pages in Internet are equal.

Unfortunately, a large part of the Internet is hidden from SE. It is called **invisible** (**deep) web**, where private, unlinked or dynamic data are stored. Some studies expect that this part of the web is still larger than a visible part of the web, covered by SE [3].

SE are extremely powerful. What they do not recommend is forgotten. Nowadays they are available not just on computers, but they start to be an integral part of mobile phones, cars, TVs, music players. SE improve their functionality and transform themselves into intelligent assistants. They are able to use additional information as user location, profile, history of previous searches, but also recommendations of people with similar profiles and taste.

SE are trying to incorporate into search a lot of additional information collected from various sources. It would be enough to direct your mobile phone at a persons face and SE will find you his or her identity. It will recommend wine in a wine store, movie you could like and how you should dress. It will remind you your schedule or tell a content of articles before you start to read them. They will be able to answer various *what if* questions.

SE are not only our windows into Internet knowledge base. They also collect a huge amount of information about us, about our interests, habits and daily lives. They know when and where we are planning our holidays. They follow our dates, weddings, and divorces. They know our health. All these information they got from us or they infer them from our queries. It can be used for our comfort like precise recommendation or advertisement, but as well as for spying us out or making us victims of targeted spam mails or other attacks [4].

Questions

Could we live without SE? Can we trust them? Who is an owner of information collected in SE? Could we protect ourselves and still use SE? Should exist a legal protection of our privacy? Are SE the intelligence of Internet?

2.4 Intelligence of crowds

Nowadays, millions of people cooperate and collaborate on various creative projects. These people live somewhere at this planet and usually do not know each other. Collaborations and peer production of such mass was not possible ever before. Internet is a first tool that allows organizing it.

There are several conditions collective projects usually fulfill: **large reservoir of experts**, **low investment costs** (computers connected to Internet), **common environment** (wiki), split of work into **small segments**, **iterative development**, **low cost of integration, benefits for all [5].**

There are many types of collective intelligence products: **encyclopedias** (Wikipedia, IMDb, etc.), **open source software** (Linux, Firefox, MySQL, Apache, Moodle, etc.), **self-presentation portals** (blogs, YouTube, Flickr, etc.), **social networks** (Facebook, Twitter, etc.), **collective forums** and many others.

Collaborators invest their time, expertise, and creativity without a salary or personal profit. They very often participate anonymously. Their motivation is neither financial reward, nor fame. Appearance of such projects surprised many. Anonymous collective long-term effort without reward is contra intuitive. Motivation is a mystery, but results are real [6].

Important conditions of success of collective projects are: **openness** (everyone is allowed to join), **share** (results are for all), **parallelism** (development can continue in many parallel threads), **share of knowledge** (collaborators communicate with each other), **evolutionary approach**, **global access** (not limited to territory) [7].

Important question is the quality of collective products. It is not guaranteed by formal methods, just by active participation and collective feedback. One of the most famous cases of quality controversies happened in 2006. Magazine Nature published a study that compared quality of Wikipedia with Encyclopedia Britannica. Britannica won but the difference was not big enough. The study proved that the quality of Wikipedia is acceptable [8].

There are several explanations of collective intelligence projects quality. Some of them were published in forms of law:

Linus law of quality [9]

Given enough eyeballs, all bugs are shallow.

Grahams law of quality (evolution law) [10]

People just produce whatever they want: the good stuff spreads, and the bad gets ignored.

Questions

What are driving sources of collective intelligence? Is open source really public? Could a collective intelligence be more intelligent than a sum of all participants?

2.5 Digital copying, torrents, and plagiarism

A production of copies was always an important part of intellectual properties distribution. Books, paintings, music, movies were reproduced in thousands or even millions of identical exemplars made of paper, vinyl, celluloid or other materials. Each copy had its price. If someone wanted to have it, it was necessary to pay the price, or at least to borrow it. There were thousands of legal owners of the same work. Individual reproduction was complicated and very expensive.

Each of us was an owner of some books. Private libraries were production tool for scientists, or artists. But even large private libraries contained only thousands of exemplars. The distribution of books was specialized and unequal.

Digitalization changed this completely. There are no technical obstacles for anyone to duplicate e-book, CD, DVD. There is no recognizable difference between a digital original and a duplicate.

Very controversial distribution of intellectual properties is sharing them by torrents. There were many attempts to protect digital artifacts against such copying or to forbid it legally. There were several legal cases like Napster, or Pirate Bay but they did not help to stop copying and sharing.

Today there are millions of book, movies, and music records accessible by few clicks. It takes just few minutes from having interest in a book, article or music to having an access to it. The ultimate goal is to provide immediate access to all books in all languages of all ages. This does not mean that everything would be for free. But it would be available. We do not know how this accessibility will influence relationship of our children to education and knowledge.

The real change is not only accessibility. It is also a relationship to originality. On Internet, there are already now thousands of written materials for almost any topic. It is very simple to produce an essay, or other work just by using them. They could serve as sources of information but also as building blocks of new solutions. Copy-paste is a fast and effective method of production. But it is not allowed in current schools because of plagiarism. It is not easy for young people to understand why something that is a daily routine in their lives (looking for information on internet and use it) is forbidden and morally doubtful. Punishment is not an explanation.

Questions

Is knowledge a property? What is a difference between finding and knowing information? What is wrong with copy-paste? Should be all information for free?

2.6 Social networks, privacy, netiquette

First thing in the morning many people do is to check statuses of their internet social connections. There are messages that arrived during the night, new photographs from parties, emails from friends, living on the opposite part of the world. People are being permanently socially connected.

There are many tools used for this: emails, ICQ, Skype, Facebook, Flickr. These tools give us impression that we are living in small worlds. That there are friends close to us we could always talk to and rely on. Somehow it is true. Nowadays friends do not loose their intimate contacts because of traveling or migration. People never spent so much time with their friends. They share and comment daily lives without necessity to be together. We add new friends and do not leave the old ones. Many people prefer communication on Internet to being together. They underestimate the chemistry of real social life, and move into a world of pure information without feeling physical closeness.

There are many new and unknown consequences of this social change. First is an illusion of permanent being with friends. Many people already recognize that friends in Facebook are not real ones. Lonely people stay alone, no matter how many Facebook friends they have.

Second problem arises from the fact that on Internet nothing really disappears. Whatever you did, wrote, asked Google or commented on social network stays there. People change their lives, their opinions, their friendship, their work, but evidence of their previous opinions, crazy parties, love stories, smoking experiments stay there for future judgments. The past was never so close.

Another danger of social communication on Internet is its non-physicality. Words are only a small part of messages we talk to each other. In normal communication we help ourselves with gestures and body language. On Internet we do not have such tools. There is no sign of humor or self-irony. We have to learn how to communicate and not destroy politeness and trust. Netiquette is an important part of our good manners on Internet. [11]

Questions

How real is a social life on Internet? What value has a number of Facebook friends? Is a presence on social network a necessity? What is a purpose of smilies? Would you publish negative information about your friend? Would you mind if someone publishes private information about you?

2.7 Virtual love

Love is a passion that happens among living people. It affects their bodies and soul. It is a mixture of body attraction, spirit and chance. There are rules, but they are broken all the time. Love is connected to sex. But there are also exceptions: platonic love without sex and sex without love.

We believe that love needs physical world, dates, kisses, and touches. Internet changed this reality. People start to visit chat rooms, where they meet virtual personalities; they chat, flirt and fall in love with them. They do not know how the others look like, how old they are, what is their sex, education, status. Internet opens a strange, unreal world where a skinny teenage blonde can be an old, married fat man, or a skinny teenage blonde. In words the difference is unrecognizable.

There are many reasons why people are going there. It is an escape from loneliness, possibility to find partners, whom they can talk to. Some people are afraid of relationships, some are too shy, and some find themselves unattractive. It is much easier to find people with same interests on Internet than in suburbs, where most of us live. Internet offers anonymity, plenty of identities, invisibility, and security, neutralization of own status. People can escape borders, conventions, and social control.

On the other hand, Internet also brings new dangers: lies about own identity, false interpretation of other persons, relations to dangerous people. Internet is full of pornography and crime. You do not see to whom you are talking your intimate secrets.

It is not easy to talk about these questions with young people but at least it is important to tell them, that they exist.

Questions

Can You imagine to fall in virtual love? Is anonymity a protection for intimacy? Would You mind to have an Internet relationship? Do You trust Your Internet *friends? Are You telling true on Internet? Are You expecting that people tell true to You there?*

2.8 Hackers, crime and politics

Internet is young and new. It is a place of freedom, courage, adventures of different kinds. Most of its discoverers are young people, full of romantic enthusiasm to change old rules and replace them with better ones. Open source initiatives, file sharing torrents are just examples of their new ethic.

Another example are hackers. Computer adventurers, who can brake into any computer, circumvent all locks and make good. There is a lot of romantic idealism in an image of a typical hacker. Especially young people do not see danger from their activities. Media and movies often support this positive image. The main argument is that hackers fight the evil, they brake private places but do not make any harm. One of the most controversial cases nowadays is WikiLeaks; a portal specialized in publishing secret documents of governments, banks, corporations and industries.

Unfortunately hacker ethic is just an ideal. There is no guarantee that braking into private computers does not cause any harm. There are many examples of the opposite. Internet is full of criminal activities, political repression, spying, spam and terrorism. There is no reason to trust someone who feels in your computer as at home.

Internet is more and more a political arena. Election campaigns use viral effects of social network; they manage revolutions; some countries like Estonia use Internet for general elections, people use absolute freedom of speech there.

Questions

Should we punish hackers as criminals or accept their right to brake into private computers if they do not make harm? Is absolute freedom of speech on Internet a positive or negative attribute? What should we do with infringement of international law by states, governments, secret services, international terrorist networks? Are elections on Internet safe?

Conclusion

We use Internet as a daily routine. We move there our collective knowledge and use it for all purposes. We meet there our friends, publish our ideas, present our lives, and tell our secrets. Our children are growing also there.

There are no final answers about Internet; it is changing too rapidly and we do not know enough. If there were any, they would be ridiculous in just few years, sometimes even shorter.

Absence of answers does not mean that discussions do not have sense as well. We have to live there as well as we live in our physical world. We need rules, manners, netiquette, safety and security. We should behave there according to our collective expectations. All of the mentioned starts with discussions, agreements, and solutions. Finding answers to new questions.

Let’s talk with our pupils about Internet.

References

1. Carrol, Lewis: Alice's Adventures in Wonderland. Penguin Classics, London, (2010)
2. Zittrain, Jonathan: The Future of the Internet. Penguin Books, New York (2008)
3. Battelle, John: The Search. Portfolio, Penguin Books, New York (2006)
4. Conti, Greg: Googling Security, How Much Does Google Know About You? Addison-Wesley, New York (2008)
5. Tapscott, Don, Williams, Anthony D.: WIKINOMICS, How Mass Collaboration Changed Everything. Portfolio, Penguin Books, New York (2006)
6. Surowiecki, James: The Wisdom of Crowds, Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economics, Societies, and Nations. New York: Doubleday (2004)
7. Sunstein, Cass R.: Infotopia, How Many Minds Produce Knowledge, Oxford University Press (2006)
8. Giles, Jim: Internet encyclopaedias go head to head, Nature, 438, 900-901
9. Raymond, Eric S.: The Cathedral & the Bazaar. O'Reilly. ISBN 1-56592-724-9 (1999)
10. Graham, Paul, 2005: What Business Can Learn from Open Source. Essay derived from a talk given at OSCON (O’Reilly Open Source Convention) 2005 (1–5 August, Portland, Ore.), http://www.paulgraham.com/opensource.html.
11. Weinberger, David: Everything is Miscellaneous, The Power of the New Digital Disorder. A Holt Paperbacks, New York (2008)

Minimally Invasive Education for Computer Literacy

Claudia Horner

University of Salzburg
Hellbrunnerstraße 34, 5020 Salzburg
claudiahelena.horner@stud.sbg.ac.at

Abstract. *Minimally Invasive Education* is an Indian approach that intends to give children of poor social background the opportunity to achieve Computer Literacy. The central part of the paper concentrates on the discussion of the term *Computer Literacy* and the skills that children are expected to acquire in the process of working with computers. Furthermore, it focuses on the question of how *Minimally Invasive Education* can be integrated in European computer science courses.

Keywords: Computer Literacy, ECDL, Minimally Invasive Education

1 Indroduction

In 1999, an experiment was conducted in the course of which a computer was built into a wall and people observed how children in the adjacent slums made use of the appliance. In the newspapers this experiment was titled *Hole in the Wall*. Mitra and Rama who observed this experiment referred to it as *minimally invasive*. This term was borrowed from medicine. Mitra said in an interview [8]: *"I'm saying that, in situations where we cannot intervene very frequently, you can multiply the effectiveness of 10 teachers by 100 – or 1,000 – fold if you give children access to the Internet. […] This is a system of education where you assume that children know how to put two and two together on their own. So you stand aside and intervene only if you see them going in a direction that might lead into a blind alley. That's just so that you don't waste time."*

In the meanwhile the *Hole in the Wall* experiment has also come to be known as *Minimally Invasive Education* and is being tested in several urban and rural areas in India. Usually observations show similar results: a learning-effect occurs through arbitrary experiments and exploration. The children invent imaginary words to communicate the results to others. *Minimally Invasive Education* learning stations are in use in several other countries as for example Cambodia, Botswana or Mozambique. It is very important to mention that none of these projects aims to replace a teacher in class.

2 *Computer Literacy* or What should kids be able to do with computers?

Nowadays it is expected that everyone can handle a computer at work, at school, in the industry and the economy. This ability to work with a computer is known as *Computer Literacy*. Schubert and Schwill [16, p. 15] define the term as follows[1]: *"[Computer Literacy] expresses the need for the purposeful use of computers to become a central cultural technique as is reading, writing and calculating. It covers the knowledge of computers and their variety of uses as well as the ability to utilize them in different tasks in business and professional life, at home and for recreational activities. Basically, this ability is to be expected from all students; without it, that is computer-illiterates [sic!], a controlled participation in everyday life becomes impossible."*

It is especially Schubert and Schwill [16] who assume that a computer-literated person should have knowledge of the history of computer science, of functional units and the meaningful use of language. Mitra and Rama, on the other hand, define computer literacy as follows [12]: *"Turn on a PC, use MS Paint to create a designed picture, move objects using folders, shortcuts, cut-and-paste, drag-and-drop, copy and delete methods, move from one web page to another and back and send and receive e-mail through a PC that is pre-configured to do so."*

These two statements clearly show how multifaceted the term *Computer Literacy* can be. On the one hand it is the simple use of the devices that is addressed, and on the other hand the focus is on the consolidated knowledge of the processes running in the background. The issue of cross-cultural influence is mentioned in both publications.

A lot of teachers may consider the ECDL, the European Computer Driving License as a measurement for Computer Literacy in computer science education. In the following I will briefly outline the ECDL principles for readers who are not aware of the concept: The ECDL consists of individual modules, mostly based on the Microsoft Office package. Each module is taught like a cooking recipe. In this approach computers are generally used as trainers. Each change of program versions will necessitate a follow up training as the adopted skills will hardly be transferred. Hubwieser [6, p. 48] addresses this issue too[2]: *"Unfortunately the term computer science is often misused in school for any type of work with the computer. This ranges from a computer aided video-tutorial to finger exercises on the keyboard."*

Therefore computer science education is settled in a field of competing interests between the use of the computer as a medium, its use as a trainer and the mastery of basic concepts (see Fig. 1, literal translation [6, p. 49]). Hubwieser does note that a meaningful use of computers as a medium requires basic skills in the use of the computer together with the knowledge of basic concepts. But then again, simple user

[1] The original publication is written in German unfortunately. Hence I decided to translate the text passage and all following publications in German into English. I hope that the intentions of the authors will not suffer from this translation.

[2] Original publication in German.

training emerges as a mere exchange of *recipes*. From the point of view of a student however, the mere teaching of concepts leads to a computer science education far apart from practical usage.

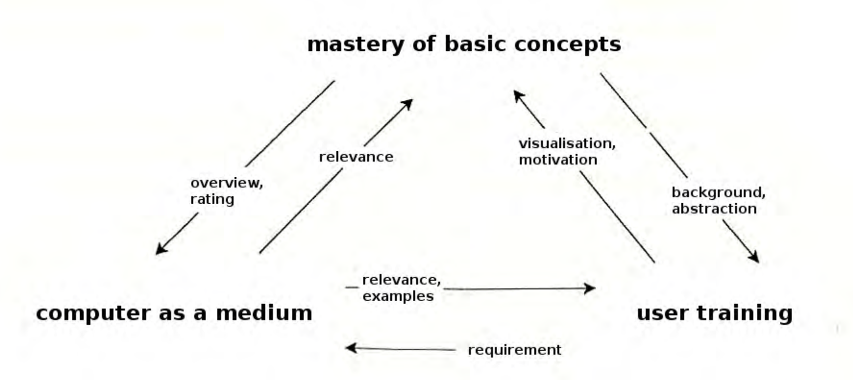


Fig. 1. Integration of teaching aids, user training and teaching basic concepts

1984 the *Gesellschaft für Informatik* (German Society for Computer Science) introduced a concept of information technology in basic computer education in an attempt to reconcile these different points of view [6]:

- Knowledge of basic structures,
- practice of simple applications of information technology,
- presentation of the opportunities and risks of information technology,
- processing and classification of student experiences in the environment of information technology.

One main goal of *Hole-in-the-Wall* is that computer literacy is no longer a privilege of people who have access to a computer at home and/ or at the office. Everyone regardless of his/ her origin should have the chance to acquire computer literacy [11].

3 Ways to *Hole-in-the-Wall*

Mitra comments on the first achievements of *Hole-in-the-Wall* [13]: "*Within six months, the children of the neighborhood had learned all the mouse operations, could open and close programs, surf the Internet and download programs, surf the Internet and download games, music and videos. When asked, they said they had taught themselves. They were describing the computer in their own terms, often coining words to describe what they saw on the screen. The hourglass symbol was "darmu", the mouse cursor "sui" or "teer"."*

The kids learned how to deal with the computer independently regardless of the language they used and outside the school. Some experiments preceded the *Hole-in-the-Wall* project to explore and understand the effect of collaborative learning:

3.1 LEDA experiment

The term LEDA [10] is derived from *Learning through Exploration, Discovery and Adventure* and was initiated as a one-week summer school for kids between 4 and 16 years from urban areas. These experiments were conducted in the period from 1991-1996. Part of this program included that the children tried to achieve basic computer skills while playing. Over a period of 4 years it was observed that a learning effect occurred without great efforts but only when the children had enough free space and motivating content. What did these children achieve after a week? They were able to use a computer in a network, to exchange data, create computer graphics, to ponder philosophically about the computer as a living being and much more.

3.2 Udang experiment

The Udang experiment took place in 1994 in a village in West Bengal in India. In the course of it a school was equipped with computers. Teachers as well as students gained their computer knowledge and their skills in private studies. During this process they only needed a few explanations. In the end the experiment showed that the teachers as well as the students were able to build up a local database without help.

3.3 Kalkaji experiment

The previous experiments had prepared the grounds for *Hole-in-the-Wall*, and thus in 1999 in the south of New Delhi a computer was installed on an outside wall of the office building of NIIT Limited. The building is adjacent to a slum that is also home to many children of all age groups. Few children attend school – mainly public schools – which do not have the best reputations. Only some children are able to understand or speak English fluently. The children's perception of the computer has been observed over three months. During that time no instructions were given. After this period all children reached the level of computer literacy defined by Mitra and Rama [12] through trial and error. Furthermore it has been observed that the children were even able to understand the English alphabet although they had no secured knowledge of English. After some weeks they were able to draw simple pictures, create, move and cut a folder. The Kalkaji experiment was continued in other slums and rural areas with similar results in the following years.

4 Design and target group of a *Hole-in-the-Wall*

The original *Hole-in-the-Wall* [13] consisted of a monitor which was built into a wall and was visible through a glass plate. A touchpad was attached to the wall. Behind the wall there was a computer connected to the network of NIIT Limited. Hence internet access was possible. After some time they found out that a touchpad did not work in the outside reasonably as dust and dirt stopped the computers from working. With minor modifications, like a reversed attached fan or/ and a cover for the keyboard, it was possible to work with the computer outside. Furthermore, special software was developed. So it was possible to monitor the devices over the Internet, adjust desktops and prevent crashes.

The children were informed about these observations that mainly focused on under-15-year-olds. Monitors were placed lower to ensure that adults did not make use of it. Additionally a sunshade was installed. Later on keyboard and mouse were hidden behind a protective cap that only offered enough space for a child's hand. Furthermore a bench was set up offering only limited space between the bench and the computer.

5 The *Minimally Invasive Education* approach (MIE)

The major goal of *Hole-in-the-Wall* is that a group of children without direct intervention by others, such as teachers, is able to learn independently. Hence Mitra has coined the term *Minimally Invasive Education* (MIE) [4]: *"[...] a pedagogic method that uses the learning environment to generate an adequate level of motivation to induce learning in groups of children, with minimal, or no, intervention by a teacher."*

As mentioned previously, [14] schools do foster computer literacy, usually through ECDL courses or computer science as a separate or optional subject. In these courses however learning cannot happen spontaneously but only at certain time on a given day. Papert [14] also laments that classes consist of homogeneous groups with similar knowledge. Hence the possibility to learn from each other is reduced considerably.

When we speak of *learning* we think of it in psychological terms:

- *Behaviorism:* It assumes that the learner is essentially passive and learning is caused by external stimuli. In other words, the teacher decides about the content and about the learning methods.
- *Cognitivism:* In contrast to behaviorism the learner is actively integrated in the learning process and learning is seen as an internal mental process. Therefore the teacher decides in cooperation with the student what methods will be adequate and what contents should be learned.
- *Constructivism:* The learner is seen as an information constructor. He/ she brings past experience that is linked to new information. The students will determine their learning process and outcome on their own.

The last approach represents an essential idea that is taken into account by MIE. Children acquire their knowledge actively instead of incorporating given knowledge.

This process happens in a playful and experimental manner. Another important aspect of MIE is learning in groups, so called collaborative learning [5][3]: "*Cooperative learning is mainly individually and highly structured. Most participants simply add their results at the end. In collaborative learning however there is a permanent, mostly self directed team work of the group available.*"

Mitra and Rama [12] found out through the observation of the experiment in Kalkaji and of subsequent experiments that independent learning had taken place. For example new functions of the computer were discovered with the graphical user interface by trial and error. Each step is noticed by other children who try to reproduce the result themselves. During this process more discoveries are made, other children notice these steps and this circuit starts again. These events are transferred among other children with specially invented terms and vocabulary. This exchange of knowledge takes place in exchange of friendship.

This learning process should not be interfered, but supported. Mitra and Rama [13] are of the opinion that adults should not interrupt the learning process. Furthermore the performance of the equipment should be secured permanently. This is done by the network-based monitoring. Finally learning should take place in heterogeneous groups [14].

5.2 Results of MIE

How can the success of MIE be measured? For this purpose a unique test, the *Graphical User Interface Icon Association Inventory* test was developed. An icon is a small image and whenever you click on it a computer based command or action is activated. This test includes 77 icons from the Windows environment. The children were asked to describe which icon is meant and what happens when you click on it in their own words. Several NIIT Limited employees tried the test in advance. 70 percent of the icons were described properly no matter whether the symbols had already been used or not. The children in the slums were tested after 9 months of computer usage and 40 percent of the icons were identified correctly. For Mitra [13] this result was proof enough that the children have gained their knowledge and skills independently.

6 The Digital Doorway – MIE in South Africa

In the meanwhile the MIE approach has been tested outside of India as well as for example in South Africa. In South Africa the project started in 2002 under the name *Digital Doorway*. The ignorance in using technical equipment is a real problem, as Gush [3], for instance, mentioned: "*Even using an ATM presents tremendous difficulties for many South Africans, either because of ignorance or because of difficulty in understanding the language in which instructions are provided.*"

[3] Original publication in German.

The high unemployment rate that prevails in this area is part of the vicious circle. Finding employment without the necessary computer skills becomes rather difficult. The big difference between the MIE kiosks in India and the Digital Doorway in South Africa is the operating system. After initial tests on Windows, they chose Debian Linux, an open source system. Gush [3] mentioned some advantages of this change, including the absence of license costs occur, development of open source skills and the multilingualism of the system. The country hopes that the computer literacy will increase in the population through this project. Thus the possibility increases that South Africa will become a leading nation in software development.

7 MIE in computer science education

What options do you have to take the success of MIE into your every day classroom? A regular course of action when we teach is to start with theory and then continue with practice and the training of new skills (see [10]). Theory and practice are not related very often. So the students do not see any relevance in learning the theory. *Learning through trial and error* opens the concept for implementing MIE in schools. This method was introduced by Thorndike an important contributor of behaviorism. The scientist observed how human and animals respond to an unknown situation, namely by random behavior. Different behavior patterns are tested to reach a certain learning success. If a behavior is exploited successfully, it will be intensified and reapplied in similar situations. Thorndike [1] emphasizes that a reliable learning success can only be reached by learning by repetition which means through iterated practice.

Another concept close to MIE is known as *Learning at the model* introduced by Bandura. Humans do not only learn through the consequences of behavior, but through observation [14]. Thus knowledge or observed behavior can be passed over to others. If it appears useful, the learner will try to imitate the behavior. If they succeed, the behavior will be adopted. Bandura's theory is often seen as bridge between behaviorism and cognitivism.

In computer science, especially when you learn programming, the concept of trial and error is not very welcome. It can be compared with pointless trying. This technique means that you start to write your code instead of developing a model in the beginning. Maybe this concept is not the best for learning how to program, but it can be worth considering when you learn how to use the so called standard software. Instead of an ECDL course, you can give your students a specific aim: The result of our next lessons will be X, how can we reach X? We can show them how to use the pre-installed or online help applications for a software package. And we can ask ourselves how *we* can ask the questions to get the right answers.

Although one can see the advantages of MIE, there exist several disputes about the pros and cons as for example in Kirschner, Sweller and Clark [9]. They use the term *minimally guided instruction* and are of the opinion that [9, p. 76]: "*[…] minimal guidance during instruction is significantly less effective and efficient than guidance* *specifically designed to support the cognitive processing necessary for learning."* Their main area of concern is that minimally guided instruction ignores the characteristics of working or long-term memory and no research supports the technique so far.

Their concern is reproducible. In the beginning students are not fully aware of the fact that they are teaching themselves and that they do not get instructions. What they will get is help, when they get lost. Maybe it is not the best way for every student, as some need clear instructions to reach their aim. But all the others learn more than just clicking, they try to understand the underlying concept. You show them how to help themselves and because they find their way on their own it will be something they will not forget.

Brunstein, Betts and Anderson [2] emphasize this point and identified three factors that lead to successful learning with minimal guidance: constrained search space, practice and combinatorial structure. Further they stated [2, p. 801]: "*…that minimally guided discovery learning can be successful if the cognitive demands are limited. One of the benefits of discovery learning is that the processes of generating a solution can lead to a characterization of the domain that will help students generalize when they face a new problem situation."* So if one limits the search space and gives enough time for finding the right answers MIE is an alternative.

Another possibility to integrate MIE in everyday school life are open-access-workstations in schools. In many schools such computers already exist on the corridor. Not only older students use these computers, even the younger ones are really interested. It is noteworthy that usually several pupils group around a device, play around together, look for something in the internet and present each other what they have just found out. Personal observations have revealed that students had no major problems either if the operating system was not Windows, but a Linux system with a graphical interface.

8 Summary

The MIE project has shown and is still showing that children are capable of teaching themselves even complex facts. It mainly requires sufficient time and the necessary materials to implement the concept in *regular* classrooms. MIE offers one more option to get kids to integrate new information with existing information. A concept, one needs to keep in mind.

As conclusion, a further idea of Mitra and Rama [12], how MIE can be implemented in school: "*The educational application of the above theories lie in creating curricula that matches and also challenges children's understanding, fostering further growth and development of the mind."*

Special thanks to Karl J. Fuchs (University of Salzburg, Austria) for the time he spent discussing some of the issues mentioned in this paper.

References

1. Brühlmeier, A.: *Aspekte der Lernpsychologie*. http://www.bruehlmeier.info/lernen_kopie.htm (April 15, 2011) (1994).
2. Brunstein, A.; Shawn, B.; Anderson, J. (2009): *Practice enables successful learning under minimal guidance*. In: Journal of Educational Psychology, 101(4), 790-802.
3. Gush, K. et. al.: *The Digital Doorway – minimally invasive education in Africa*. http://www.digitaldoorway.org.za/publications/ICT in Education conference paper- The Digital Doorway_March_2004.pdf (April 15, 2011) (2004).
4. Hole-in-the-Wall Education Ltd. [Ed.]: *Minimally Invasive Education*. http://www.hole-in-the-wall.com/MIE.html (April 15, 2011) (2007).
5. Hinze, U.: *Computergestütztes kooperatives Lernen: Einführung in Technik, Pädagogik und Organisation des CSCL*. Münster: Waxmann (2004).
6. Hubwieser, P.: *Didaktik der Informatik - Grundlagen, Konzepte, Beispiele*. Berlin: Springer, 2001.
7. Inamdar, P.: *Computer skills development by children using "Hole-in-the-Wall" facilities in rural India*. In: Australasian Journal Technology, 20(3), 337-350. http://www.ascilite.org.au/ajet/ajet20/inamdar.html (April 15, 2011) (2004).
8. Judge, P.: *A lesson in Computer Literacy from India's Poorest Kids' New Delhi physicist Sugata Mitra has a radical proposal for bringing his country's next generation into the Info Age*. http://www.businessweek.com/bwdaily/dnflash/mar2000/nf00302b.htm (April 15, 2011) (2000).
9. Kirschner, P.; Sweller, J.; Clark, R.: (2006): *Why minimal guidance during instruction does not work*. In: Educational Psychologist, 41 (2), 75-86.
10. Mitra, S.: *Children and the Internet: New Paradigms for Development in the 21st Century*. http://www.hole-in-the-wall.com/docs/Paper03.pdf (April 15, 2011) (2000).
11. Mitra, S.: *Minimally Invasive Education for mass computer literacy*. http://www.hole-in-the-wall.com/docs/Paper01.pdf (April 15, 2011) (2000).
12. Mitra, S.; Rana, V.: *Children and the Internet: Experiments with minimally invasive education in India.* In: The British Journal of Educational Technology, 32(2), 221-232. http://www.hole-in-the-wall.com/docs/Paper02.pdf (April 15, 2011) (2001).
13. Mitra, S.: *The Hole-in-the-Wall*. http://www.niit.com/niit/ContentAdmin/images/sugata/DQ--Hole in the Wall.pdf (November 07, 2007) (2004).
14. Papert, S.: *Computer in the Classroom: Agents of Change*. http://www.papert.org/articles/ComputersInClassroom.html (April 15, 2011) (1996).
15. Plassmann, A.; Schmitt, G.: *Lernen am Modell nach Bandura*. http://www.lern-psychologie.de/kognitiv/bandura.htm (April 15, 2011) (2007).
16. Schubert, S.; Schwill, A.: *Didaktik der Informatik*. Spektrum, Heidelberg (2004).

Exclusive Courses for Inclusive Education

Ľudmila Jašková

Department of Informatics Education, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia, jaskova@fmph.uniba.sk

Abstract. The paper deals with the professional training of teachers of informatics to prepare them to use digital technologies for teaching students with special educational needs in mainstream schools. We have developed new courses to improve teacher's competences. We describe the contents and forms of these courses. At the end of the paper we present results of our research evaluating the efficiency of these courses.

Keywords: special educational needs, assistive technologies, teacher's competences.

1 Introduction

Digital technologies (DT) are widely used in many professions and also in everyday life. Independent life, further education and professional success of people with impairment are possible only in case following conditions are fulfilled:

– They can use modern digital technologies in their study and everyday life.
– They can study at mainstream schools integrated in the society.

Many people with special educational needs (SEN) study integrated at mainstream schools but their teachers are not prepared good enough to teach them and to use digital technologies for this purpose. Our department plays an important role in development of study plans for future teachers of informatics. We have started to investigate how to change these plans so that our graduates were better prepared for the inclusive education of people with SEN. Therefore we have performed a research by means of which we tried to answer following questions.

– Which teacher competences are important for the inclusive education?
– How to change the study plans so that informatics teachers were more competent?
– Will these changes bring required results?

You will successively find our answers in further parts of this paper.

2 Teachers and their competences in inclusive education

Technology is a versatile tool for handling information and for communication now. Information and communication technologies are therefore essential tools for inclusion [3].

For learners with SEN, technology can facilitate the process of knowledge absorption if appropriately utilized within their individual education plans.

DT not only make education easier for students with SEN but they are necessary in integrated schools. Therefore all teachers should know [5]:

– using DT as a tool for **transformation of study materials** into an accessible form for students with SEN,

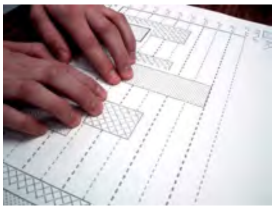


Fig. 1. Bar-chart in a tactile form

– how can a student with SEN use DT for taking notes, for his/her project work, for testing of knowledge, and for communication with teachers and schoolmates,
– how can mainstream teachers or non-disabled students use DT to discover the way of life of their disabled students or schoolmates (where it is possible to find information about disabilities on the web, where are visual or hearing simulations, where to find information about Braille and sign language, etc).

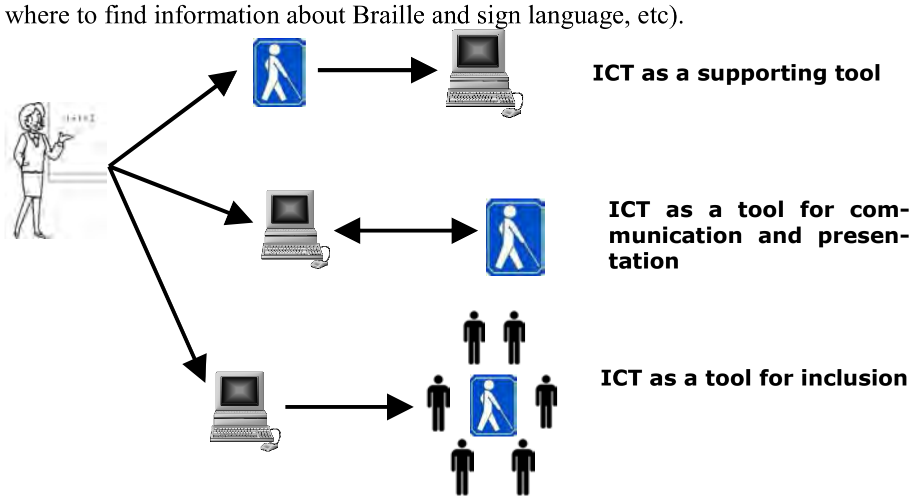


Fig. 2. Teacher competences in inclusive education

3 New courses for informatics teachers

To improve competencies of mainstream teachers we have made several changes in educational study at our faculty and we also have prepared online course for mainstream teachers of informatics [5].

3.1 Modifications in educational study

We have modified the study plan for educational study of informatics in the following way.

We added new topics to existing subjects, e.g. we added the topic about ways of using computers by disabled students into the basic course aimed to gain basic computer skills, etc.

Besides the modification of existing subjects we offer new eligible courses as follows.

1. **Integrated education of people with SEN** – basic course about people with SEN and potential problems with their integration in mainstream schools. This course is designed for all students of our faculty.
2. **ICT in education of handicapped students** (ICTH course) [4] – basic course about digital technologies for students with SEN (assistive technologies, e-accessibility, adaptations of study materials, universal software design, etc.). This course is primarily designed for students of educational study of informatics.
3. **Accessible web design** [1] – advanced course about disabled users of the web, critical objects on the web, accessible web design and about accessibility tests. This course is primarily designed for students of educational study of informatics.
4. **Accessible software design** – advanced course about universal software design. This course is designed primarily for students of applied informatics.

Further we will describe courses focused on students of educational study of informatics – **ICT in education of handicapped students** and **Accessible web design**.

3.2 Course on ICT in education of handicapped students

Course on ICT in education of handicapped students consists of these topics [4].

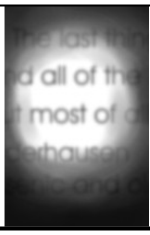
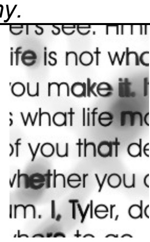
1. Main advantages of using DT in education of students with SEN.
2. Visually impaired people and assistive technologies for visually impaired people.
3. Methodology of teaching computer skills for the visually impaired students.
4. Hearing impaired people and assistive technologies for hearing impaired people.
5. Methodology of teaching computer skills for hearing impaired people.
6. People with mobility problems and assistive technologies for them.
7. People with cognitive problems and their way of using computers.
8. Accessible web design.
9. Accessible web testing.
10. Adaptation of study materials for students with SEN.
11. Principles of universal software design.
12. Digital technologies in activities facilitating integration of students with SEN into mainstream schools.

The aim of this course is to make an introduction to the world of students with SEN and to answer the following questions.

- What does it mean to be impaired?
- How do visually impaired students see?
- How do hearing impaired students hear?
- How do students with dyslexia perceive a text?

– How do students with low visual comprehension perceive pictures?
– How do some objects look like for people with attention problems?
– How to improve orientation in study materials for people with memory problems?

Table 1. Simulations of various vision difficulties.

A simulation of vision with glaucoma.	A simulation of vision with macular degeneration	*A simulation of* vision with *diabetic retinopathy*.	A simulation of vision with advanced cataract.
			

– How do students with SEN use computers? (Attendants of the course come into a contact with people with SEN. They observe how the others use computers and they have an opportunity to try to use computers in the same way as people with SEN do.)

Table 2. Assistive technologies for people with SEN.

Unsighted person is using the screen reader	A person with motor disability is using a mouth stick	A person with low vision is using a TV magnifier
		

– How to prepare study materials for people with SEN?
– How to access pictures and scientific information if someone is person with SEN?
– What are critical objects on the web?
– How to use ICT in activities which facilitate integration of disabled students into mainstream schools?

3.3 Course on Accessible web design

Course on Accessible web design consists of two parts.

The first one is focused to handicapped users of the web. The aim is to answer following questions [11].

– How do they use computers?

– How do they use the web?

The second part of the course is dedicated to critical objects on the web [1], [2], [6], [8], [9], [12], [13]. It answers following questions.

– How to make them accessible?
– How to test their real accessibility?

This course contains several special features – **e-learning**, **learning by doing**, **examples of bad and good practice**, **discussions about projects**. We shall briefly describe these features now.

E-learning

E-learning courses have becomed to be very popular because connection to the internet is not a problem for most of potential attendants. This form of learning saves time and allows studying whenever and wherever anyone wants.

Duration of the course is thirteen weeks. Each week is dedicated to the different topic.

From the organizational point of view tasks of each week have following structure.

– At first the study materials and related exercises are published in Moodle environment.
– Discussion on the topic is held in discussion forum.
– Lecturer evaluates solved exercises and sends his comments to attendants.
– Sample solutions of exercises are published in Moodle.

There are some special exercises in the end of the course.

– Attendants work on a project in the last part of the course.
– Attendants pass a test in the end of the course.

Learning by Doing

The course aims to teach attendants the accessibility standards for web pages. Attendants discover these standards on their own. They use computers in the same way as disabled users do.

Examples of bad and good practice

Attendants analyze many web pages – properly chosen examples of bad and good practice. Besides that attendants have to find pages with specified features (e.g. with wrong navigation or wrong text structure, etc.) or they find wrong features on specified web pages.

Fig. 3. Web page of an unnamed secondary school – Items of active sub menu are mixed with the main content of the web page. We use this web page as an example of the bad practice.

Discussion about projects

Attendants develop accessible web documents in the last third of the course and they perform tests of their accessibility. Web documents are thereafter classified by the other attendants and the whole group discuss about documents in discussion forum.

4 Evaluation research

We have performed qualitative evaluation research to answer following questions.
- Which competences of attendants were developed or improved during the courses?
- Are courses good enough to acquire necessary knowledge and skills?
- Are study materials satisfactory?

To answer these questions we have analyzed these types of data:
- solutions of exercises and evaluation of the final project and the final test,
- multiple choices questionnaire performed at the beginning of the course,
- interview done at the end of the course,
- contributions to discussion forums.

4.1 Evaluation of the course on ICT in education of handicapped students

We have performed 6 runs of the ICTH course (since the year 2006 till 2011). The average number of attendants was ten.

During our research we have found following information about our attendants.
- They have learned how impaired people perceive the world and how they use computers but they should have contacts with individual people with SEN more frequently to gain more experiences.
- They have acquired familiarity with assistive software for visually impaired people.
- They have learned accessibility principles for web pages but they were able neither to do proper accessibility tests nor to create an accessible web page.
- They have acquired familiarity with different formats of study materials suitable for people with SEN but they still were not able to create a suitable study material for them.
- The most difficult for them was to understand the way how people with learning difficulties (problems with attention, memory and text perception) access information. In case of people who are unhearing since their birth it was similar. Attendants couldn't imagine that these people have problems with understanding texts and also to communicate with hearing people. The easiest for them was to understand the situation of motor and visually impaired people.

Fig. 4. The web page of an unnamed secondary school contains too much information for people with attention problems. Our attendants could not see this problem.

4.2 Evaluation of the course on Accessible Web Design

We have performed two runs of the course for practical teachers from mainstream schools. We were surprised by the high interest of teachers in the course. Unfortunately, we could manage just 20 to 30 attendants in one run of the course.

In the first run there were 23 teachers (22 teachers of informatics and one teacher of art). The second run attended 29 teachers (21 teachers of informatics and 8 teachers of other subjects) and 7 students of informatics.

During our research we have found following information about our attendants:

– They have learned problems of handicapped web users.
– They have gained basic information about testing and development of accessible web pages.
– They have gained the ability to look at web pages in a critical way.
– They wanted to redesign their formerly created web pages into accessible form.
– They have found it important to teach their students and colleagues to make accessible documents.

But there were also several limiting factors with negative consequences.

– Many attendants had not even the passive knowledge of English so we couldn't use web pages written in English (e.g. online simulators, online accessibility testing tools).
– Attendants couldn't use web browsers and search engines effectively enough.
– Most of teachers of informatics had no skills in web development and they knew nothing about HTML, CSS and Java Scripts.

Fig. 5. Web page of an unnamed primary school with wrong contrast – texture on the background makes the text less readable. This web page has not changed yet, despite of the fact that the teacher from this school attended our course.

5 Conclusions

Our experience with above mentioned courses showed that attendants found these courses useful and interesting.

They have learned how to percept the world in the way of people with SEN.

They have learned that there is no need to protect disabled students but better make equal conditions for them as for their intact schoolmates.

They have learned that it is important to prepare their students for inclusion also in case they have no student with SEN in the classroom because each of us has occasional contacts with somebody with some kind of difficulty. Therefore we will try to perform our courses in a closer cooperation with people with SEN.

We plan to expand our activities to non-disabled pupils from primary and secondary schools. We have included some texts and exercises concerning people with SEN to their text books of informatics. We believe that this will help them to be more tolerant and sensitive to other people needs.

References

1. Accessible Web Development (in Slovak: Prístupnosť elektronických dokumentov), http://edi.fmph.uniba.sk/~jaskova/ped/.
2. Bernadini, A. et al. Web for All: A User-Centred Design Approach for Making More Usable and Accessible Web Sites. In The Good, The Bad and The Irrelevant conference. Helsinki, Finland, 3-5. September 2003. (2003), http://goodbad.uiah.fi/info/000_general_info.
3. Blaimers, M. Enabling Technology for Inclusion. London: Paul Chapman Publishing Ltd, A SAGE Publications Company. 194 p. ISBN 1-85396-394-1, (1999) .
4. ICT in education of handicapped students (in Slovak: IKT vo vzdelávaní zdravotne postihnutých), http://edi.fmph.uniba.sk/~jaskova/ IKTH.
5. Jašková, Ľ. Informatics teachers and their competences in inclusive education. In Miesenberger, K. et al. (Eds.). Computers Helping People with Special Needs. 10th ICCHP, Linz, Austria, July 11-13, Proceedings, Berlin; Heidelberg; NewYork: Springer. 2006. pp. 552-559.ISBN, (2006) .
6. Course on the accessible web design (in Czech: Kurz tvorby přístupného webu), http://www.h1.cz/kurz-pristupnosti.
7. Miesenberger, K. et al. (Eds.). Computers Helping People with Special Needs. 10th ICCHP, Linz, Austria, July 11-13, 2006, Proceedings, Berlin; Heidelberg; NewYork: Springer. 2006. p. ISBN 3-540-36020-4. (2006) .
8. Section 508, http://www.section508.gov.
9. Špinar, D. Developing the accessible web pages (in Czech: Tvoříme přístupné webové stránky). Praha: Zoner Press. 360 p. ISBN: 80-86815-11-0, (2004) .
10. The European Union Disability Strategy, http://europa.eu.int/comm /employment_social/soc-prot/disable/strategy_en.htm.
11. Web Accessibility in Mind, http://www.webaim.org.
12. Web Content Accessibility Guidelines 1.0, http://www.w3.org/TR/WCAG10.
13. Web Content Accessibility Guidelines 2.0, http://www.w3.org /TR/WCAG20.

Supporting students' development of computer science skills

Wanda Jochemczyk and Katarzyna Olędzka

Computer Assisted Education and Information Technology Centre, Raszyńska 8/10, 02-026 Warsaw, Poland
{wanda, katarzyna}@oeiizk.waw.pl

Abstract. The aim of this article is to present some ideas connected with introducing students into programming world and to share our experience. In the first part we present the idea of introducing students to learning algorithmics by solving puzzles and playing games or other kind of thought-provoking tasks. In the second one, we present the idea of first steps in programming describing microworlds for young learners. By steering a turtle children learn mathematics and develop their programing skills. Next, we show some examples of problems for more matured learners. In our teaching practice we prepared multilevel or open tasks to encourage students to gain new knowledge. We hope that sharing these examples will stimulate reflection.

Keywords: teaching, algorithmics, computer science

1 Introduction

As teacher trainers in Computer Assisted Education and Information Technology Centre in Warsaw we organize courses and workshops for teachers. Our institution provides support teachers' professional development. We also work with schoolchildren both directly by organizing e-learning courses or contests for them, and indirectly by helping teachers in the didactic work.

The aim of this article is to present some ideas connected with introducing students into programming world and to share our experience. We present our article in four parts. In the first one we describe the concept of preparing to learning algorithmics by solving interactive thought-provoking tasks. In the second one, we present the idea of first steps in programming while in the third and fourth we present problems for more matured learners. We hope that sharing experience and examples will stimulate reflection.

2 Non-creative algorithms and creative thinking

Discussing matters related to learning and teaching algorithmics we find challenging. We use algorithms which are defined as *systematic procedure that produces – in a finite number of steps – the answer to a question or the solution of a problem* [5]. Thus, they should be correct, unambiguous, completed and if possible efficient. If one wants to be a programmer and wants to write programs, should be constructive and creative to discover and implement solutions to various problems. Presenting an effective method of solving given problems expressed as a finite list of well-defined instructions is quite an interesting task. Learners should focus on exploring ideas, generating possibilities, looking for many correct answers and be able to choose the best one. An introduction to such activities can be solving puzzles and playing games as introduction to computational thinking and learning programming. This practice helps students to move their thinking to abstract level, introduce problem-solving strategies, develop creativity and attract more interest [4]. Examples of such applications which were prepared in our project for early school teaching are presented below.

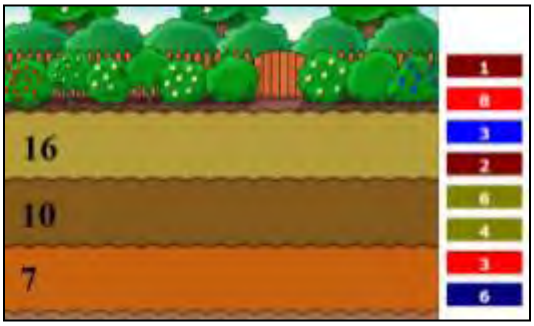


Fig. 1. With mathematics in allotment

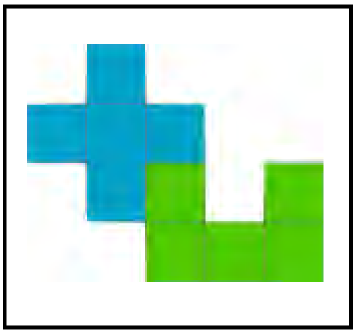


Fig. 2. In the same colour

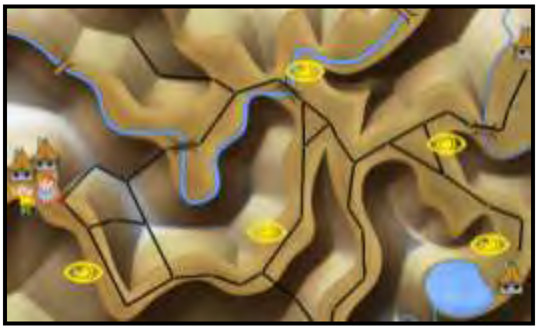


Fig. 3. On the track

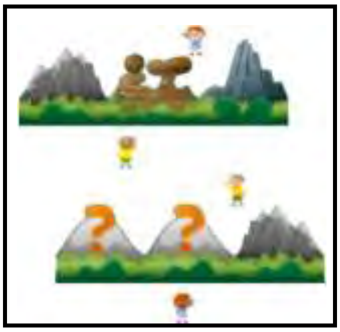


Fig. 4. On the other side of mountains

The above-mentioned projects similarly to computer games are a specific form – they teach skills such as data manipulation, strategic planning and decisions making.

Players win by successfully navigating and meeting challenging tasks. In strategic games children's emotions strongly affect their motivation. When the children play first time, they usually fail, and then they have to do it over and over again, until they master the skill. In the end, they gain the knowledge. Thus, they win. [6]

3 First steps in programming

Another way of learning with computers is trying to talk with them. Both humans and computers have their own languages. As children from early years learn how to read, write and count, nowadays they also naturally acquire some ICT skills. By using a computer, they learn how to write and draw or find useful information on the Internet. They can also control how the turtle moves on the screen. In pre-prepared microworlds, they start with simple drawings. A turtle can go forward, backward, turn right, left and change the colour and width of the pen. When a child clicks on a button the turtle executes the corresponding command. The next step is to drag and drop Logo instructions to make a list of commands. Pupils learn how to create their own procedures. According to Papert: *Once programming is seen in the proper perspective, there is nothing very surprising about the fact that this should happen. Programming a computer means nothing more than communicating to it in a language that it and the human user can both "understand." And learning languages is one of the things children do best. Every normal child learns to talk. Why then should a child not learn to "talk" to a computer?*[1] Moreover, such projects stimulate logical thinking and are good introductory exercises for learning algorithmics.

To exemplify children's activities some drawings of specific shapes can be shown. Even with simple commands, a teacher can prepare a challenging set of tasks. Not only mathematical knowledge does count, but also programming skills are required. Even small children try to automate their work, by applying copy-and-paste strategy or by using the repeat instructions.

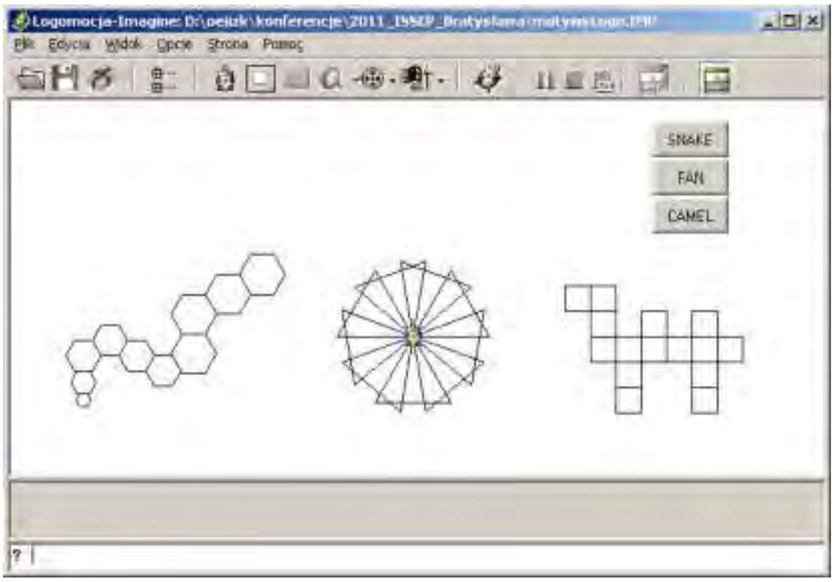


Fig. 5. Children's activities

In practice the most problematic area of promoting programming at school is to encourage teachers to teach such lessons. Most of them are not familiar with programming computers and what is more important they have a barrier to perform such lessons in their schools. It raises a question what can be changed in teacher education to make programming more popular? Observing the school practice there are many children who want to deepen and broaden their programming skills but they suffer from lack of support from adults. We get emails requesting help.

4 Quite difficult problems

The older the children are the more sophisticated tasks are required. Young teenagers build their own knowledge by solving appropriate for their age problems. They learn how to use efficiently the most important procedures of turtle graphic or find recurrent elements to apply iteration and recurrence. To deal with more complicated problems they have to divide a problem into sub-problems to implement subsequent procedures. Mathematical knowledge helps them to calculate proportions, scale a drawing and establish measure of angles.

Fig. 6. Examples of tasks [7]

Solving such tasks requires accuracy and patience, one has to test solutions step by step and the final result. I has to be done with changing values of parameters with special consideration of boundary conditions. From the teacher's perspective, it is very interesting to analyse various solutions got from students. They are so innovative in their way of thinking. However, the main problem teachers usually face at this stage is how to motivate students to a long term effort.

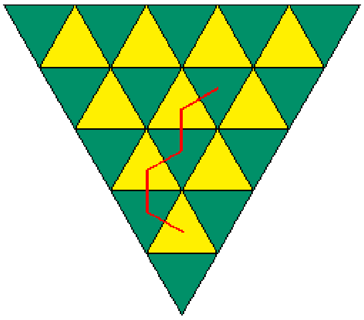
5 Multilevel and open tasks

In some cases there is a need from multilevel tasks which can be solved by beginners and more experienced students. Some examples were shown in "Take up the Challenge – reflection on POLLOGIA Competition". [3] We plan such tasks to meet expectation of different level preparation and experience from the students' perspective.

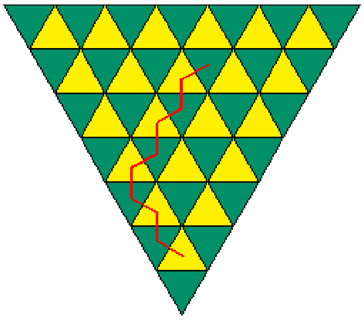
Example of task – the route [8]

A turtle moves on a triangle board starting from the lowest yellow triangle. It can move to the left and right, but it cannot leave the board. When it is not able to move, the movement is omitted. Write a procedure ROUTE :n :description, after calling it on the screen it will we drawn two-coloured board with marked route of the turtle. Parameter :n defines the number of green triangles in the highest row, and the :description presents the route of the turtle.

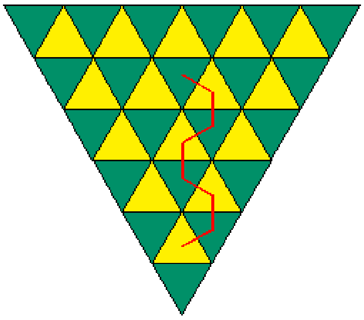
The descriptions consist of small letters ***l*** *(the movement to the left) and* ***r*** *(the movement to the right). The length of the big triangle is 500. If you know how to do it, do not draw these parts of the route, in which the turtle goes on its previous position. If drawing the turtle route is too difficult for you, draw only the board.*



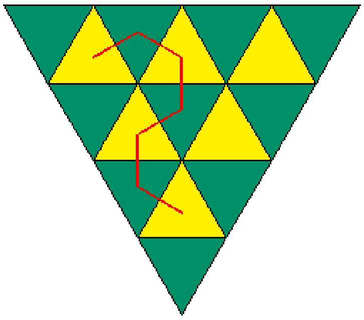
ROUTE 5 "lrrlr



ROUTE 7 "lrlrrlrlr



ROUTE 6 "rrrrrllrrll



ROUTE 4 "lrrlrrrrrll

Fig. 7. Task *The route*

In the example above one can draw only triangles. The next step is to draw the way the turtle moves. More experienced students can restrict movements only to the board. The most difficult part of this task is to find cycles and eliminate them.

A good way to interest students and develop their skills is to give them some open problems with some guidelines. In our e-learning courses we prepared such tasks to encourage students to gain new knowledge also by surfing the Internet. The example of such task is “Break the Cesar’s message”. The problem is connected with ciphering information. It is a type of substitution cipher in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, if there is a key equal 3 and we want to cipher letter **a** it would be replaced by **d**, **d** by **g**, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. Ciphering is quite an easy task. The more difficult one, is to break the code without knowing the key.

Fig. 8. A screenshot from the application *Break the Cesar's message*

The prompt which we give students is to find typical frequency of letters and to compare it with frequency of deciphered message for different keys. The value of a key which fit the best is most probably the correct one. Another example of open task is to draw mountains with option of covering or to explore L-systems.

One can ask what will students learn by making such projects? They will learn some technical things, for example to programme computers. Moreover, they will acquire knowledge traditionally incorporated and even not included in the school curriculum. For example in order to decipher message, they have to learn about frequency of letters in their language and where to find this information, in the project connected with drawing mountains some geometry knowledge is needed. They will develop some psychological, social and moral aspects of thinking in other projects. *Most important of all in my view is that children will develop their sense of self and of control. For instance, they will begin to learn what it's like to control their own intellectual activity*. [2] It is a good instrument to create open exercises, stimulate thinking and make one's own creative activity.

From our own perspective there is a difficulty in finding or in creating such tasks. There is a lack of collections of such problems which are either approachable for students on one hand and developing for the other. Some source of such tasks are programming contents both Polish and international. We hope in the nearest future there will be more and more student friendly places with such problems so that they can find the most interesting for them.

6 Conclusion

We live in a society which is information centred and computerized. Many people, especially young, use for their everyday life computers. Some of them know or learn how to programme them. Apart from reading, writing and arithmetic, such skills like thinking, problem solving, synthesising, communicating really count. Our response to 21st century demands is teaching algorithmics from early stages of children development in a constructvistic way to help them become mature and knowledgeable digital citizens and not to get lost in this modern world.

References

1. Papert, S.: The Mindstorms: children, computers, and powerful ideas. Basic Books, New York (1980)
2. Papert, S., The Connected Family: Bridging the Digital Generation Gap, Longstreet Press (1996)
3. Borowiecka A., Borowiecki M., Jochemczyk W., Oledzka K., Samulska A., Take up the Challenge – reflection on POLLOGIA Competition. In: Proceedings of Constructionism 2010., Paris (2010)
4. Levitin A.: Puzzles and Analysis of Algorithms. In: 36th SIGCSE technical symposium on Computer science education, ACM New York (2005)
5. Encyclopedia Britannica, http://www.britannica.com
6. Learning by Playing, http://www.lauerlearning.com/learning
7. Logo competition for primary school portal, http://minilogia.oeiizk.waw.pl
8. Logo competition for gymnasium portal, http://www.pdp.edu.pl

Automated Online Identification of Learner Problem Solving Strategies – A Validation Study

Ulrich Kiesmüller[1] and Torsten Brinda[1],

[1] University of Erlangen-Nuremberg, Didactics of Informatics, Martensstr. 3, 91058 Erlangen, Germany
{Ulrich.Kiesmueller, Torsten.Brinda}@cs.fau.de

Abstract. In the field of learning programming, environments providing visual programming are employed in computer science education. Learners have to complete tasks using these environments. To guide learners to an independent problem solution from a constructivist perspective [2] teachers have to find out, what the learners' concepts are. For this purpose it is interesting to identify learners' problem solving strategies. As described in previous work [1, 2] a system based on pattern recognition methods was developed to identify automatically the learner's problem solving strategy. A validation study of this method and the software implemented is described in this article. In addition to the results of the software, data from thinking aloud studies was used. The data collected was evaluated using empirical research methods based on Cohen's κ [5]. In this work we compare the automatic identification of learners' strategies with the human expert ratings and find the high agreement of κ = 0.747. Conclusions concerning the validation of the technique and some remarkable observation in the data are drawn.

Keywords: Computer Science Education, Secondary Education, Problem Solving Strategies, Qualitative Content Analysis.

1 Introduction

During their first steps in programming, learners solve small programming tasks often using learning and programming environments developed for certain age groups, such as Alice [6], Scratch [17] or Kara, the programmable ladybug [20].

1.1 Observed Problem Solving Strategies

In this context, different strategies of solving the given problems can be observed. An obvious aspect to differentiate the various strategies is the way of structuring the problem. There are learners, who structure the problem before solving, whereas others solve the problem in not prestructured single steps. The first group contains those learners who are using a *top down* strategy in terms of[14]. First of all they construct

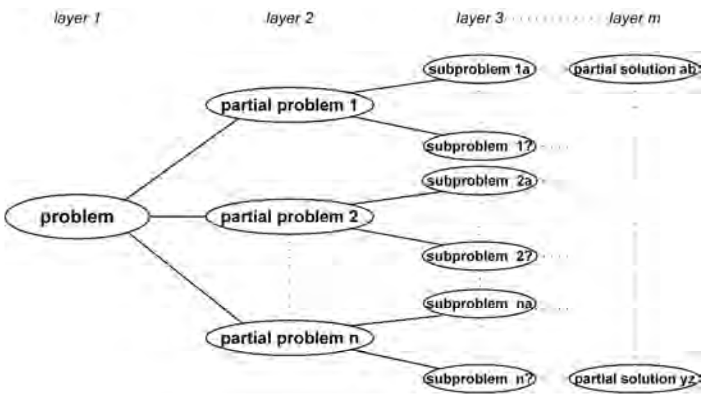


Fig. 1. Structure of a problem from the view of learners using a structured problem solving strategy.

(or evoke, if present in solvers' long time memory [19]) the problem space. They structure the problem as a whole. Therefore they divide the problem into several partial problems, where necessary, these again in smaller subproblems and so on, until they reach the partial solutions. In this way they approximate to the solution in layers (Fig. 1 and left side of Fig. 2). If they complete the set task employing Kara, the programmable ladybug, they will create all states required at first. Subsequently they justify Kara's sensors, with that all branches of the program are created. At last they fill in and if necessary they edit the commands to solve the small subproblems finally. In practice the observation of a learner, who uses a top down strategy will not show exactly this sequence of learner-system-interactions (LSI) but a very similar one. At this point one must take into account that this strategy is hardly to be maintained even by programming experts like described in [10].

The second well structured working group contains learners, who solve the separate subproblems one by one like shown on the right side of Fig. 2. At every step they go from the problem as a whole all the way to the solution layer. This strategy is

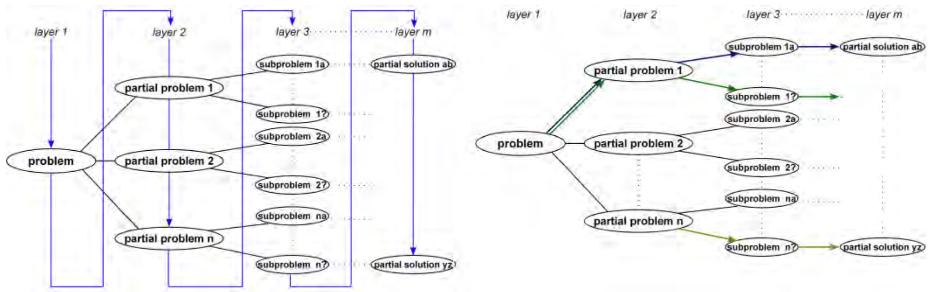


Fig. 2. Learners' approach to a problem solving using the top down strategy (on the left hand) or the bottom up strategy (on the right hand).

called below ***bottom up*** strategy in terms of [14]. By using the Kara environment, this strategy manifests itself in creating one state and set the sensors for only one branch before adding the commands in exactly this branch. Subsequently these steps are repeated. In this manner, the learners complete the tasks set by solving the subproblems one by one. The sequence of creating/editing states, editing branches, adding commands is repeated until every single subproblem is solved.

Even in the group of learners, who are solving the problems in not pre-structured single steps, there are two different types of strategies.

First of all we find learners, who focus on one single programming step and try to optimize it (loops in Fig. 3). Therefore they verify each step of their incremental solution. With a general concept in mind how their program should work so far, they start the program execution. As soon as they recognize whether the current tackled subproblem is solved correctly or not, they actively stop the program execution. They correct their mistakes if necessary and execute the program again. In case of a correct partial solution these learners look for the best next step to proceed (dotted arrows in Fig. 3). In this way, they bring themselves closer to the solution layer. But they not necessarily follow one path like learners using the bottom up strategy. In fact they change from one branch to another only focusing on the step, which is the best to proceed from their point of view. This strategy is called below ***hill climbing*** strategy.

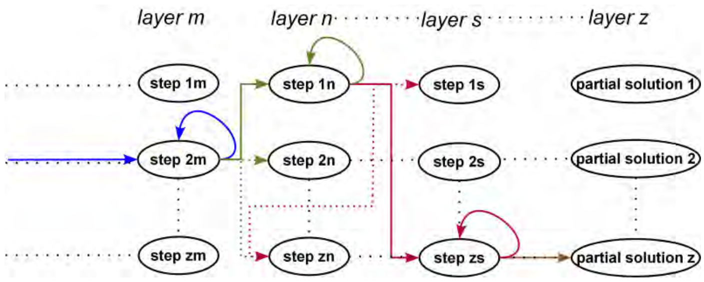


Fig. 3. Incremental approach to a problem solving using the hill climbing strategy.

Their sequences of LSI using Kara show frequent program executions followed by single actions like adding single commands or editing a single branch. There are only rare system error messages because the learners do not need them to proceed in programming. Messages only appear if the learners are not fast enough to stop the program when recognizing an error.

The other strategy in this group is the well known ***trial and error*** strategy. In this case learners focus on a single step again, but they have no general concept in mind when they execute their program. Thus they frequently produce system error messages. They are not actively controlling their programming progress but only reacting to the system messages. They jump around the different branches in a rather random order. Therefore there is no strict approach to the completion of the task. It may be that the learners work at the same programming step again and again without any progress (Fig. 4).

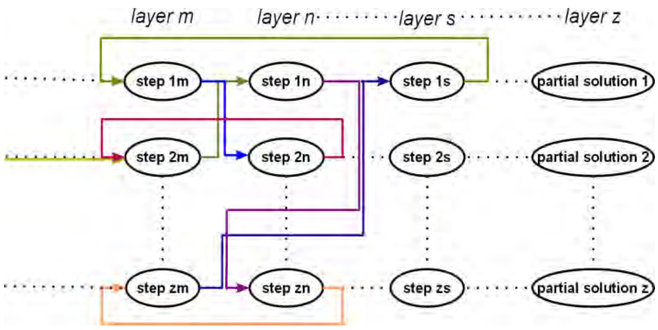


Fig. 4. No strict approach to a problem solving using the trial and error strategy.

1.2 Related Work

In previous studies [13] the chronology of LSI, which were categorized before was reported in log files. All of the learners observed, showed one of the four strategies described in Section 1.1 at a time. But only a few of the observed sequences of LSI match exactly with the ideal learner system interaction pattern mentioned above. From time to time there are missing single interactions, there are appearing additional interactions or some single interactions are replaced with others. An example for the bottom up strategy is shown in Tab. 1 – on the left hand an ideal sequence of LSI, on the right hand a real observed sequence. The sequences are very similar but differ in some LSI. Finding the most probably strategy which causes a certain observable sequence of LSI is a problem similar to problems well known in the field of pattern recognition. In [13] a software was designed and implemented in 2009 which identifies learners' problem solving strategies automatically with the help of pattern recognition methods (hidden markov models).

Table 1. LSI-sequence for the bottom up strategy (ideal versus real observed).

ideal	observed
TRANSITION ADD	TRANSITION ADD
TRANSITION CHANGE	
SENSOR SET	SENSOR SET
SENSOR SET	SENSOR SET
	TRANSITION CHANGE
	PLAY
	STOP
COMMAND ADD	COMMAND ADD
COMMAND ADD	COMMAND ADD
COMMAND ADD	COMMAND ADD

This article is structured as follows. After a short introduction, we give a brief insight to the problem solving strategies observed during the studies described here. Subsequently previous work is shown. In Section 2, we describe the design of the study including the research questions, the research field and the methodology. In Section 3, the results using the proposed method are analyzed. In Section 4 we discuss the results. Possible conclusions are drawn in Section 5. We end with an outlook on future work presented in Section 6.

2 Study Design

A validation of the results of the strategy identification software requires a different way of obtaining information about the learners' problem solving strategies than observing the problem solving process step by step as it is done by the research software. A first idea was to get information about the learners' problem solving strategy with the help of questionnaires, which the learners have to fill in. The questionnaires are based on the fundamental concepts of Ajzen's and Fishbein's Theory of Reasoned Action and Theory of Planned Behavior [1]. In this way information about the learners' problem solving strategies can be used without considering certain steps during the problem solving process. Because of the necessary extent of those questionnaires, there would be a large impact to the learner's problem solving process. To avoid this, another observation method was employed at last. We decided to use the "thinking aloud" method. Learners using the Kara environment to solve the set tasks are observed by a researcher, who records all statements, which the learners make during the problem solving process, employing shorthand notices. At two times during a practical course for girls at the University of Erlangen-Nuremberg in the years 2009 and 2010 a group of seven learners (14 to 15 years old) at a time were observed at working with the Kara environment and completing the set tasks. One session takes 21.5 min in average (minimum: 15 min, maximum 25 min). The learners solved tasks of the collection included in Kara. These tasks are categorized concerning their difficulty in groups "easy", "medium" and "hard". Additionally they got two complex problems – the first, enabling the ladybug to find the exit of a random labyrinth; the second, continuous inverting a pattern of fields with or without leaves. All learners can choose at any time which task they complete. Subsequently some records do not start exactly with the beginning of working on a certain task and respectively do not automatically end, when the current task is completed.

2.1 Research Questions

"Teachers should meet learners where they are" is an important aspect from a constructivist view [2], lightly modified from where we tend to "... meet learners where they walk". From this vantage point it would be helpful to find out, to what extent it is possible to identify automatically learners' problem solving strategies. Teachers can adjust their teaching style to the learners' strategies. The system messages of learning and programming environments can be adapted to the learners' problem solving strategies as well. In this article is searched for an answer to the questions:

- To what extent is it possible to identify learners' problem solving strategies using the methods mentioned above?
- To what extent do the results of the identification software accord with analogue data collected with the "thinking aloud" method?

2.2 Research Field

Problem Solving Strategies. In the field of problem solving there are known many various strategies and differing denotations, resulting from many different points of view. The four strategies used for the studies described in Section 1.1 result from a look at the overlap of publications like [4], [14], [19] and [23] concerning problem solving strategies mentioned there. The labeling for the four strategies used here, was created having regard to the varying denotations found in the publications, but as similar as possible to them.

Collecting Data. All learner-system-interactions relevant to learners' problem solving process are recorded in log-files by a tracking module included in the identification software. Like shown in [13] the recorded learner-system-interactions are categorized for further data analysis considering results and methods described in [3] and [11]. Additionally in the studies described here the learners' statements while "thinking aloud" are recorded by a human researcher like mentioned in Section 2.

Pattern Recognition. Like mentioned in Section 1.2 identifying the most probably problem solving strategy belonging to an observed sequence of learner-system-interactions is a similar problem to those, which occur in the field of automatic speech recognition. In this field a method using hidden markov models [22] improved itself in practice. The identification software used in the studies was developed in [13]. It employs this method to identify single patterns in the sequences of LSI. It is used again to identify the learner's problem solving strategy considering sequences of sequences of learner-system-interactions [13].

Empirical Methods. The shorthand notices of learners' thinking aloud during the problem solving process are transcribed employing methods of qualitative content analysis as described in [18]. Hereby a single character was used to label each of the occurring strategies (Tab. 2). For working out the similarity of two encoded sequences of strategy patterns metrics in the field of symbol sequences are required. The most obvious alternative would be using the hamming distance [9] for calculating the "distance" of two strings. But in that case the gaps occurring sometimes while recording (especially employing the "thinking aloud" method) could only be regarded for the computation of the similarity by a complex data preparation. Therefore the

Levenshtein Distance [21] was used for the comparison of the resulting strings. This algorithm was slightly modified with the help of weighting factors as explained in [15] to concern the higher similarity of the structured (A, B) and the not prestructured (C, D) strategies among themselves. Using weighting factors is a successful method in linguistics since a long time. For the analysis described here every matching pair of characters appearing in the strings to compare is scored with 1.0, pairs of "similar" characters (A and B, C and D) are scored with 0.5, other character combinations with 0.0.

Table 2. Coding of problem solving strategies.

Strategy	Character
top down	A
bottom up	B
hill climbing	C
trial and error	D

2.3 Methodology

The shorthand notices mentioned above were transcribed in a table. Each column represents in average a time slot of 28s. For every one of the four problem solving strategies found in previous studies [12] detailed transcription rules were deduced from publications in the field of problem solving and listed in tabular form for the raters. Additionally hints such as those concerning the learners' choice of words were made available to them. The tables conclude a description of every strategy, some example records and the rules, mentioned above. They were given to three raters. One is a graduated computer scientist, the second a researcher in computer science education and the last one a high school teacher (post-graduate psychologist). First of all they transcribed only one of the 14 records. In a subsequent discussion misunderstandings were avoided and points of disagreement were resolved based on the transcription rules. After this, the remaining records were transcribed by each rater.

Table 3. Correlation of Cohen's [5]/ Fleiss' κ [7] and agreement (based on [16]).

κ_C / κ_F	agreement
$\kappa \leq 0.40$	poor agreement
$0.40 < \kappa \leq 0.60$	moderate agreement
$0.60 < \kappa \leq 0.75$	substantial agreement
$0.75 < \kappa \leq 0.90$	very good agreement
$0.90 < \kappa \leq 1.00$	(almost) perfect agreement

Analysis. The interrater reliability was calculated based on Cohen's κ (in case of two raters) [5] in form of

$$\frac{observed\ agreement - random\ agreement}{1 - random\ agreement} . \tag{1}$$

For computing the kappa label related to the measure of agreement as mentioned in Section 2.2, the agreement of two random strings with n characters must be calculated by

$$\frac{\sum_{j=0}^{n}\sum_{i=0}^{n-j}\binom{n}{i}\binom{n-i}{j}\cdot 2^{n-i-j}\cdot(i+0.5\cdot j)}{n\cdot\sum_{j=0}^{n}\sum_{i=0}^{n-j}\binom{n}{i}\binom{n-i}{j}\cdot 2^{n-i-j}} . \tag{2}$$

In this way you get a value of 0.375 for the random agreement. In case of more than two raters Fleiss' κ [7] related to the measure of agreement as mentioned above was used. Kappa values were attributed to the measure of agreement based on the ideas described in [16]. Tab. 3 shows this attribution. For every transcribed record the measure of agreement was calculated. In Tab. 4 is shown an example of the resulting strings. Calculating Fleiss' κ for this record results a value of F = 0.846. In the same way the agreement of raters' results and the string, which results of the identification software mentioned in Section 1.2, was calculated.

Table 4. Strings resulting from transcription of one "thinking aloud" record.

rater	transcribed string
1	BBBBBBCCCCDDDDDDDCCCCCCCDDD DCCCCCCCCCCCCCCCDCDDCCCCC
2	BBBBBBBCCCCCDDDDDDCCCCCCDDD DCCCCCCCCCCCCCCCDCCCCCCCC
3	BBBBBBBBCCCCDDDDDDCCCCCDDDD DCCCCCCCCCCCCDDDDDDDDCCCC

3 Results

3.1 Encoding and Interrater Reliability

An important point arose at the discussion between the raters before encoding. Not every single statement indicating a special strategy may lead to the identification of this strategy. For strategy records, such as the top down strategy, LSI sequences result in quite long LSI sequences. A single statement not matching to the strategy is not automatically a sign for changing a strategy. In fact it must be interpreted as not strict maintenance of a certain strategy in the sense of [10]. While comparing the strings resulting from the transcription of a "thinking aloud" record (Tab. 4) is noticeable, that all strings start with a number of B followed by alternating groups of C and D. Only the lengths of the groups are slightly differing from one rater to the others. The result of the interrater reliability (Fleiss' κ) calculated employing algorithms and methods mentioned above over all records was 0.816. This implies a very good agreement of the raters resulting encoded strings.

Table 5. Result from identification software.

BBBBBBCCCCCDDDDDDDCCCCCDDDD
DDDDDCCCCCCCCCCDDDDDDDDD

3.2 Comparison and Results

The string resulting from the outcome of the identification software described in [13] for the sample mentioned above is shown in Tab. 5. Already at a first glance there is a high similarity to the strings encoded by the raters. There is also a group of B at the beginning of the string, followed by alternating groups of C and D.

For every rater Cohen's κ was calculated as measure of agreement (Tab. 6). The average of 0.747 implies an agreement, which is substantial, nearly very good.

Table 6. Cohen's κ for comparison of rater's coding and software results.

rater	Cohen's κ
1	0.746
2	0.714
3	0.780
average	0.747

4 Discussion

With a closer review of the encoded records of the raters and the software it is evident, that there are more difficulties and lower agreement in parts of the records with frequent strategy changes. By the help of the learners' statements the human rater is enabled to predict the strategy the learner will show next. As opposed to this the software has to get some information in terms of LSI to compute the strategy, which the learner has been using. So the strategy change in raters' strings often appears one or two steps before they occur in the string resulting from the identification software. Consequently a perfect agreement of raters and software results cannot be expected. After encoding the raters reported that the encoding rules are overlapping in some parts. This lead to disagreement at these points of the records. Both aspects may indicate the existence of further strategies not taken into account so far. This will be examined in further studies as mentioned in Section 6. For the results of the studies, described in this article, this issue means lower agreement results, too. To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

5 Conclusions

The answer to the second question asked in Section 2.1 is the high agreement (κ = 0.747) between raters encoding records of the “thinking aloud” method and the results of the identification software. The methods employed by the software in its current version, are already valid with reasonable certainty. Consequently the software provides a possibility to identify automatically learners’ problem solving strategies during solving problems using programming environments.

6 Outlook

For a further validation whether the four strategies and their respective patterns of LSI mentioned above are selective and sufficient there will be a field study conducted with a larger sample of learners. Employing statistical methods like multiple regression analysis and explained variance, will answer the questions, whether the four patterns are selective and whether there exist additional patterns. In this case respective hidden markov models must be developed and added to the identification software. The pattern recognition algorithms will be the same. Only the models have to be trained one time, like described in [13]. In this way an adaption for a more differentiated strategy identification is easily possible.

The results of the software developed can be used in different ways. One is for studying the learners’ preferred problem solving strategies at the beginning of teaching algorithm. Based on this, curricula in computer science can be improved, regarding to the requirements for certain age groups. Additionally a more detailed knowledge about the learners’ (preferred) problem solving strategies, will provide teachers’ selective interventions in treating novice programmers. Another benefit is to generate system feedback adapted to the learners’ problem solving strategy by using the results of the software developed. In this way the entire teaching and learning process can be improved. Guiding learners to an independent problem solving will be supported to a greater extent than before. Consequently there is more time for teachers to help learners at other crucial points on their way to a solution. At this term it is not to design “learning without teachers” (like programmed learning). In fact, teachers should be provided by enhanced programming environments and therefore more time will remain for intensive advising on questions for example about structuring of problem solving using algorithm control structures (sequence, branch, iteration). Learners should *neither* be *forced* by the adapted feedback *to change* their strategy to a special other one (because it might be better for solving the set task) *nor* be *forced to maintain* their strategy over the entire problem solving process. They have the free choice, which problem solving strategy they will use and are provided with each strategy. Another reason for employing the methods of identifying learners problem solving strategies is the knowledge explained in [8], which reminds that the teaching style should be addicted to the learning style for reaching the most effective learning progress. In this connection an enhancement of the software components, which were designed and developed until now, would be helpful. During learners solving the set tasks, teachers get a statistical evaluation regarding to preferred problem solving strategies out of the enhanced identification software. In this way teachers would be enabled to adapt their teaching style to the learning style in further lessons.

References

1. Ajzen, I., Fishbein, M.: Belief, attitude, intention, and behavior: An introduction to theory and research. Addison-Wesley, Reading, MA (1975)
2. Ben-Ari, M.: Constructivism in computer science education. SIGCSE Bulletin, 30(1):257--261 (1998)
3. Chi, M. T. H.: Quantifying Qualitative Analyses of Verbal Data: A Practical Guide. The Journal of the Learning Sciences, 6(3):271--315 (1997)
4. Chi, M. T. H., Glaser, R.: Problem solving ability. In: Sternberg, R. J. (ed.), Human abilities: An information-processing approach, pp. 227--257, Freeman (1985)
5. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement, 20(1):37–46 (1960)
6. Conway, M. J. Alice: Interactive 3D Scripting for Novices. PhD thesis, University of Virginia, Charlottesville, VA (1998)
7. Fleiss, J. L.: Measuring nominal scale agreement among many raters. Psychological Bulletin, 76(5):378--382 (1971)
8. Felder, R. M., Spurlin, J. E.: Applications, Reliability and Validity of the Index of Learning Styles. Engineering Education 21(1): 103--112 (2005)
9. Hamming, R. W.: Error detecting and error correcting codes. Bell System Technical Journal, 29(2):147--160 (1950)
10. Hoc, J.-M.: Psychology of programming. Computers and people series. Academic, London (1990)
11. Hundhausen, C. D., Brown, J. L., Farley, S. and Skarpas, D.: A Methodology for Analyzing the Temporal Evolution of Novice Programs Based on Semantic Components. In ICER, editor, ICER ’06: Proceedings of the second international workshop on Computing education research, pp. 59--71. New York, NY, USA : ACM (2006)
12. Kiesmüller, U.: Diagnosing Learners’ Problem Solving Strategies Using Learning Environments with Algorithmic Problems in Secondary Education. ACM Transactions on Computing Education, Vol. 9, No. 3, Article 17, 1--26 (2009)
13. Kiesmüller, U., Sossalla, S., Brinda, T., Riedhammer, K.: Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods. In: Proc.

ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), pp. 274--278. New York, NY, USA : ACM (2010)
14. Koenemann, J., Robertson, S. P.: Expert problem solving strategies for program comprehension. In Robertson, S. P., Olson, G. M. and Olson, J. S. (eds.), CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems: CHI Conference proceedings: Reaching through technology; 27 apr - 2 may 1991, pp. 125--130. New York, NY, USA : ACM (1991)
15. Kondrak, G.: Phonetic alignment and similarity. Computers and the Humanities, 37:273--291 (2003)
16. Landis, J. R., Koch, G. G.: The measurement of observer agreement for categorical data. Biometrics, 33(1):159--174 (1977)
17. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M.: Scratch: A Sneak Preview. In: Proc. IEEE Int'l Conference on Creating, Connecting and Collaborating through Computing (C5), pp 104—109. IEEE Press, New York (2004)
18. Mayring, P.: Qualitative content analysis. In: Flick, U., v. Kardorff, E. and Steinke, I. (eds.): A companion to qualitative research, pp. 266--269. SAGE, London (2008)
19. Newell, A. and Simon, H. A.: Human problem solving. Prentice-Hall, Englewood Cliffs, N.J. (1972)
20. Reichert, R.: Theory of computation as a vehicle for teaching fundamental concepts of computer science. PhD thesis, ETH Zürich (2003)
21. Sankoff, D. and Kruskal, J.: Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. The David Hume series. CSLI, Stanford, Calif., reissued edition (2001)
22. Schukat-Talamazzini, E. G.: Automatische Spracherkennung: Grundlagen, statistische Modelle und effiziente Algorithmen. Vieweg & Sohn, Braunschweig (1995)
23. Sullivan, F. R. and Lin X.: The ideal science student and problem solving. In: ICLS, editor, ICLS '06: Proceedings of the 7th international conference on Learning sciences, pp. 737--743. International Society of the Learning Sciences (2006)

Logic programming in informatics secondary education

Barbara Linck and Sigrid Schubert

University of Siegen, Didactics of Informatics and E-Learning
Hoelderlinstr. 3, D-57076 Siegen, Germany
{firstname.name}@uni-siegen.de

Abstract. This article presents a structure model of competence (SMC) as a result of research on logic programming. The SMC was theoretically derived from didactic concepts and from empirically proofed learning and teaching experiences. The aim of our research is to improve informatics secondary education. The SMC shows how valuable the logic programming paradigm is as second and alternative programming paradigm in the curricula. Four dimensions of competencies were discussed to cluster all relevant cognitive and non-cognitive competence details in a tree structure. Each leaf of the SMC tree will be a later test item. So the SMC is the scientific basis of our further research of measurement of the competence development in secondary schools, which are related to the logic programming paradigm.

Keywords: Competence Model Research, Logic Programming, Informatics in Secondary Education, Active Learning, Didactics of Informatics

1 Motivation

The Association for Computing Machinery (ACM) curriculum „*should seek to identify the fundamental skills and knowledge that all computing students must possess*." [1, p. 13] The authors of the curriculum were convinced that students need to learn more than one programming concept: "It is also important for students to recognize that the choice of programming paradigm can significantly influence the way one thinks about problems and expresses solutions of these problems. To this end, we believe that all students must learn to program in more than one paradigm" [1, p. 19]. Object-oriented, functional and scripting languages are mentioned. However, the ACM curriculum does not mention logic programming.

This was the motivation for our research project called 'Competence modelling and measurement of logic programming in informatics secondary education' with the aim to improve the informatics education with the focus on 12th grade students at secondary schools. Both authors of this article have intensive teaching experiences in this field. Logic programming was the second and alternative programming paradigm of our students in secondary education. The research methodology starts with the following two phases, which are presented in this article in chapters three and four:

1. National and international curricula e.g. [1], [2], [3], learning and teaching concepts, examples of best practice, and learning obstacles in the field of logic programming are analysed and discussed.

2. The identified competencies are clustered in a structure model of competencies (SMC) of logic programming. This leads to a tree structure with the root “competencies in logic programming as second and alternative programming paradigm”. This tree structure is divided into branches of competence dimensions with competence sub dimensions until a leaf is reached. The SMC is theoretically derived and has to be empirically proven.

High quality of the SMC allows successful measurement of the learning outcomes.

2 Applications of logic programming

Several criteria are important for the learning process of logic programming. Five criteria are that students are motivated to learn, that prior knowledge is considered, that there is a classification of tasks and their possible answers, that there is a measuring of competencies, and that students gain a comprehensive view of the subject. Therefore, the article starts with examples about applications of logic programming outside of the school environment, which may motivate students in the learning process. Such examples further show the importance of logic programming to be included as an alternative programming concept in school curricula.

‘Cleverbot’ is a chat system. It answers questions and furthermore, it can ask questions (see Appendix). It was developed by Rollo Carpenter who received several prices for his researches. Cleverbot is special, because it extends its knowledge base, while chatting with people (see figure 1, [4]).

Fig. 1. Chat system ‘Cleverbot’

RoboCup is a competition, in which AI strategies are used to play soccer with a robot. It can be used to motivate students to learn logic programming as well. In addition, RoboCupJunior is a competition developed for schools. To ensure that boys and girls are equally motivated to participate, the cup is divided into the three leagues Dance, Rescue and Soccer. A European Cup and a World Cup are organized to advance the exchange of creative ideas. Logic strategies can be also used in other games (see [5]).

Furthermore, AI strategies are also present in the current media. ‘Watson’, a computer programme developed by IBM, won against two players at the famous TV show Jeopardy in 2011. It formulated questions to the given answers (see [6]). Its ancestor beat Garri Kaparow in a chess game in 1997.

Another motivation for students to learn logic programming are logic games. There are several puzzles available which are fascinating and which can be solved, e.g. with Prolog. In this respect Tate comments: “*Prolog*. Yes, I know it’s old, but it is also extremely powerful. Solving a Sudoku in Prolog was an eye-opening experience for me. I’ve worked hard to solve some difficult problems in Java or C that would have been effortless in Prolog” [7, pp. 3-4]. Logic puzzles can support different competencies. Students train their logical thinking by solving these puzzles without programming, they can test and analyse their own solution with logic programming, and logic programming can be used to find a solution. With respect to the latter, a comparison between the implementation using different programming paradigm is advisable.

Another application of logic programming is a robot scientist called ‘Adam’. It analysed genes and enzyms in collaborations with logic programming. It was able to formulate and test 20 hypotheses autonomously. As a result, twelve novel hypotheses could be confirmed (see [8]). Such robot scientists can be used in school to exemplify the examination of phenomena in everyday life with logic programming. Furthermore, interdisciplinary teaching, such as a combination of biology and informatics, is possible. It could also be discussed, how informatics and logic programming can help to improve our living standards. E.g. these automatic robots may be applied to discover new development in medical investigations.

As a conclusion, there are several examples, which can illustrate the useful application of logic programming beyond the school context and can motivate students to learn logic programming.

3 Learning and teaching experiences

The research process to cluster the competencies started with the analysis of teaching experiences in higher education because there are many very detailed reports available. Due to the fact that first semester students of higher education are only one year ahead of learners in grade 12 of upper secondary education, it can be assumed that the learning obstacles are similar. All didactic recommendations of higher education however, need to be modified for the target group of secondary students.

Higher education

Logic programming and in particular, learning obstacles to logic programming have been discussed in several studies. Taylor and du Boulay [9] analysed very detailed why learning logic programming was difficult for novices:

– “So at first sight Prolog looks as though it should be fairly easy to learn. However, the language’s apparent simplicity is beguiling, beginners occasionally being lulled into a false sense of security” [9, p. 157].
– Another main problem is ‘Interpreting problem descriptions’, which demands competencies in ‘Entities and relations’ (I) and also in ‘Generality of solution’ (II):
  I. “But this very flexibility may be a trap, because the minimal amount of structure is imposed on the knowledge bases” [9, p. 161].
  II. “A widespread problem that beginners face when interpreting problem descriptions is deciding how general a solution should be” [9, p. 162].
– “The crux of the argument is this: although most adults are capable of formal reasoning, the suggestion that such reasoning either conforms to, or is captured in, the precise rules of predicate logic is unwarranted” [9, p. 164].

The above study highlighted the main obstacles of logic programming to be overcome by beginners. The structure of the knowledge base and the representation of a problem description in facts and rules are often very difficult for them. Even though our target group are learners with prior knowledge of programming, these difficulties may also be experienced by them.

Another obstacle was highlighted by Tate in an interview [7]: “I recall one of my first experiments with Prolog, writing something along the lines of x = x + 1. Prolog responded no. Languages don’t just say no. They might give the wrong answer or fail to compile, but I had never had a language talk back to me. So, I called Prolog support and said that the language had said “no” when I tried to change the value of a variable. They asked me, why would you want to change the value of a variable? I mean, what kind of language won’t let you change the value of a variable? Once you grok Prolog, you understand that variables either have particular values or are unbound, but it was unsettling at the time” [7, p. 92]. This example was very authentic and gave us a first glance of the main problem ‘logical variables’. In particular, the prior knowledge about variables has an influence on the learning of logic programming.

In another study Stamatis and Kefalas [10] created a ‘Logic programming didactics (LPD)’ and evaluated their recommendations empirically. The target group consisted of learners with prior knowledge of imperative programming. LPD is based on a so-called student model. A student model is an assumption of seven classes of specific, cognitive misconceptions, which every learner has to overcome in the learning process of logic programming, e.g. “all parameters of functions/methods should have a value when called” [10, p. 137]. LPD continues exercise classes with solutions, which support to learn from misconceptions mostly through a comparison of imperative solutions and Prolog solutions. This LPD is very useful in teacher education because it improves the awareness of a teacher to recognise typical misconceptions of the learners in upper secondary education and how to guide the learner by his or her transformation from such a well-known misconception into a successful conception.

Secondary education

Several studies on logic programming are also available that focus on students at school level. For example Di Bitonto, Roselli and Rossano [11] developed software as a learning tool for novices of informatics and of logic programming, aged 9-13 years, and evaluated the usability of the software. 23 pupils learned with the software for two weeks. Afterwards they were tested in building facts (1), reading facts (2), building rules (3), explaining the virtual Prolog machine (4) and problem solving with Prolog (5). Surprisingly task (4) was solved by 20 pupils and task (5) by 17 pupils, even though only 14 pupils could solve task (1) (12 pupils could solve (2) and 11 pupils task (3)). This implies that some pupils could write a Prolog program, but were not able to write the components of such a program (facts and rules). However, it was emphasized that learning tools are highly important for the learning process. In our opinion a learning tool can support in particular the understanding of the virtual machine and the tracing. Therefore, the use of such tools does have an influence on competencies of logic programming.

In another study Scherz and Haberman [12] prepared Prolog rules of abstract data types (ADTs), e.g. lists, sets, trees, graphs, which the learners used as black boxes. Therefore, they recommend: “[...] instructional model is centred on using ADTs [...]” [12, p. 332]. They also recommend logic programming as second and alternative paradigm, but they only focussed on novices of informatics and of logic programming from the age of 14 years. The learners were in this study divided in two groups. The first group used the prepared Prolog rules of ADTs in there solutions. The second group was able to implement the Prolog rules of ADTs by themselves. Both groups could successfully solve their project work at the end of the yearlong introductory course. Two didactic questions are left open in the study:

- Firstly, it is uncertain when learners should switch from one group to the other to develop their competencies in abstraction and formalization.
- Secondly, it is assumed that learners face obstacles in understanding list as a recursive data type in Prolog if they used the prepared Prolog rules of the ADT list.

Even though no answers were given to these questions, the prepared Prolog rules of an ADT may in our view be used as a first learning step. They enable the students to apply and to comprehend e.g. the data structure list, before the students have to develop their own application of an ADT.

In many of the sixteen states of Germany logic programming is the second and alternative paradigm in informatics curriculum of upper secondary education. We focus on Roehner [13], because this book is considered to be a masterpiece of didactics of informatics. He published in Hesse 1995, 2002 and 2007 three versions of exercise classes and solutions of logic programming together with teaching recommendations.

In the beginning of the course logic programming seems in general to be very easy. But by designing rules with the recursive predicates of the data structure list, this is changing. The learners face intense obstacles, because their experience with the first and familiar kind of informatics modelling and programming is no longer applicable. It is the task of the teacher to support the overcoming of frustration by the learning group, which is often a result of the missing statements of loops and conditional branching and the new kind of binding of logical variables. “Goals can be generalized by the use of Prolog variables. They do not behave like the variables in other languages, and are better called logical variables. The logical variables replace one or more of the arguments in the goal” [14]. Logic programming is to use without graphic user interface. Therefore, the preparation of the course has to consider motivational aspects even more. One possibility is the pizza-task [13, p. 14], a simple task of implementing facts and rules as well as performing simple queries to use the data of a pizza place. This pizza-theme can be also expanded to explain and-relations, or-relations, calculations, entities, trace modus and box model.

The functionality of the virtual machine should be visualized to facilitate the learning process. One opportunity to do so is the box model and the trace modus. Another important point is to consider the prior knowledge of the learner. In the most cases the imperative programming paradigm is learned before the logic programming. Therefore, Roehner implies to use the imperative prior knowledge to introduce the list programming. The differences, e.g. lists can contain different data structures and need no pointers, should be elaborated. It is also important to exemplify that the structure [element 1, element 2, …, element n] cannot model the dynamic of the data structure list. Therefore, the virtual machine works intern with another description. A list can be also seen as a head element and the rest of the list. The description is [head | tail]. Like a term, a list is intern a tree structure. This should also be illustrated to the learners. Afterwards they will work much easier with lists than before in the imperative programming because of the lack of pointers [13, pp. 22-24].

Learning potentials

Logic programming gives the learner a new approach to well-known and less well-known fundamental informatics ideas (see [7], [9], [10], [11], [12], and [13]). The learning potentials could be sorted into three groups:

(1) Knowledge extension through a special kind of high-level language:

– design of knowledge bases with sensibly knowledge representation:
  - application of closed-world-assumption,
  - design of facts and rules with recursion,
  - design of data structures list and tree with recursion,
  - recognition of circularities,

– application of declarative and procedural semantics within one language,

– application of cut operator as methodology to make programs efficient,

(2) Knowledge extension through a special kind of human computer interaction:

– application of a query language,

– application of trace modus to debug faulty programs,

(3) Knowledge extension through a special kind of virtual machine, the Prolog machine:

– application of automated reasoning through unification and depth-first search with backtracking mechanism,

– interpretation and evaluation of internal states of the virtual machine.

Furthermore grammar and formal languages, vending machines, Turing machine and automatic language processing could be part of the more advanced teaching units (see [13]).

Recommendations

The learning and teaching experiences show how logic programming expands the competencies of programming. The imperative programming and the logic programming should be compared. Learners need to explicitly understand the following:

1. There are different classes of problems, which will be solved with a different kind of thinking (a different kind of programming paradigm).
2. The data structures are not the same. With this reflection the understanding of data structures will increase.
3. Imperative programming includes iteration and recursion, but logic programming applies recursion. Some students may wonder why they learn recursions if they can express the same in iterations. When they start to learn Prolog, they will be motivated to learn them.
4. Those variables play key roles in different concepts. The associated discussion about it will support the understanding of the underling mechanism.

The learners should be able to express the differences. This means that they have to review and to evaluate both paradigms in a new context.

4 Modelling of competencies of logic programming

Our aim is to measure the learning results. Therefore, we focus on modelling of competencies as basis of test items. Two kinds of competence models are required: structure models of competence (SMC) and models for levels of competence (MLC). The SMC contains the competencies, which are required to solve problems of specific domains. E.g. students are supposed to create a knowledge base referring to a given problem. Furthermore, MLC contains a differentiation into levels. E.g. at the beginning of the course students are supposed to add new facts and rules, which are similar to given facts and rules. A different level would be that students are supposed to formulate own facts and rules in a following lesson. The taxonomy of Anderson and Krathwohl will be used to separate competencies into different levels of the cognitive process [15, pp. 67-68]. This article will focus on the SMC as a first research result.

As a conclusion of learning and teaching experiences (see [7], [9], [10], [11], [12], and [13]) and an additional analysis of German school curricula (see [16]) all detected competencies were clustered. But the decision about the main components of the SMC is very difficult. More than one option is justifiable. We used the research results of [17], which gave us a lot of inspiration and concrete advice for the structuring of our competence model, e.g. *C2 Basic competencies* and *C3 Informatics views*. "Each dimension comprises different competency components, which characterize in more detail the requirements within the two mentioned informatics domains" [17, p. 514]. The domain is now logic programming. According to Weinert´s notion of competency [18], competencies encompass both cognitive and non-cognitive skills and abilities. This leads us to the component *C4 Non-cognitive competencies*. The first component *C1 Prior knowledge* is a speciality of the target group, which should be learners with explicit prior knowledge in two fields, predicate logic and a first programming paradigm. As result of this collection the SMC has four competence dimensions (*C1-C4*) with sub dimensions (see figure 2).

C1 Prior knowledge

Prior knowledge needs to be considered in every lesson. E.g. predicate logic can be taught in mathematics. This prior knowledge is not obligatory in all school curricula. Hence, it can vary. It is necessary to determine whether this knowledge should be repeated or included into the teaching unit of logic programming.

As mentioned before, this structure model assumes that the target group learned another programming paradigm beforehand, because all reviewed school curricula included logic programming as a second or third concept. Imperative or object-oriented programming is included in the curricula first. Researches and experiences in school and at university level showed that known programming paradigms, especially the imperative concept, have a remarkable influence on the learning of logic programming (see [10], [13], and [19]). Hence, the first paradigm has to be considered as prior knowledge. One should bear in mind, that if such prior knowledge has not yet been gained, the competence *C2.1 Computer programming in general* would need to be discussed in a very different way.

- C1 Prior knowledge for logic programming
  - C1.1 Predicate logic
  - C1.2 First programming paradigm
    - C1.2.1 Imperative programming
    - C1.2.2 Object-oriented programming
- C2 Basic competencies
  - C2.1 Programming competencies related to any paradigm
  - C2.2 Logic programming
    - C2.2.1 Application of logic programs
    - C2.2.2 Comprehension of logic programs
    - C2.2.3 Development of logic programs (including debugging)
- C3 Informatics views
  - C3.1 Knowledge base
    - C3.1.1 Structure
    - C3.1.2 Closed-World-Assumption
    - C3.1.3 Dynamic modification
  - C3.2 Queries
  - C3.3 Tracing
    - C3.3.1 Unification and backtracking
    - C3.3.2 Trace modus
    - C3.3.3 Box model
    - C3.3.4 Parse tree
- C4 Non-cognitive competencies
  - C4.1 Attitudes
    - C4.1.2 Expectations for Informatics Literacy & Professional Practice
  - C4.2 Social-Communicative Skills
    - C4.2.1 Cooperation & Teamwork
    - C4.2.2 Empathy: Change of Perspectives & Roles (User, Developer)
  - C4.3 Motivational and Volitional Skills
    - C4.3.1 Openness to new Ideas & new Requirements

Fig. 2. Structure model of competencies of logic programming

C2 Basic competencies

There are two main basic competencies regarding logic programming. Students are supposed to learn *C2.1 Programming competencies related to any paradigm* and *C2.2 Logic programming*. One example of *C2.1 Programming competencies related to any paradigm* is the ability of interpreting problem descriptions. "This includes getting a precise enough understanding of what the problem is in order to determine what might count as a solution" [9, p. 158]. Another example of *C2.1* is the mapping from understanding of the problem to an understanding of the simplified informatics system. "[...] at no point are we talking about the actual physical machine [...], but idealized mental models of part of the computer's functioning" [9, p. 158].

C2.1 requires prior knowledge about at least one other programming paradigm. Before solving a given task with a programming language, students should reflect about modelling of the task. The method of modelling differs from paradigm to paradigm. But in every programming concept a reflection about the task and a specification (see

[12]) is necessary before the implementation can follow. The similarities of programming paradigms should be discussed. Furthermore, *C2.1* includes a comparison between different programming concepts. E.g. control structures or variables should be discussed, because their structures vary in different paradigms. Both parts of programming competence do not support only the understanding of one paradigm but also the understanding of programming itself.

In *C2.2 Logic programming*, the concept should not only be compared exclusively to other paradigms. Furthermore, logic programming basics can be taught separately. The logic programming concept should be applied, comprehended and developed by students. E.g. students apply logic programming while performing queries to a given knowledge base. Students comprehend world-close-assumption while analysing a given answer. Moreover, students develop logic programs while formulating own facts and rules referring to a given task.

C2.2 Logic programming has a strong connection with *C3 Informatics views*. The different informatics views have to be applied to gain the basic competencies. Therefore, these dimensions are related to each other. The SMC structures the competencies. Although, *C2.2* and *C3* are connected, they describe different components of competency and thus, are both clustered in separate dimensions.

C3 Informatics views

In logic programming there are three different kinds of views, namely *C3.1 Knowledge base, C3.2 Queries and C3.3 Tracing*. With respect to *C3.1 Knowledge base* students are supposed to understand the internal structure of logic programming. They apply, comprehend and develop facts and rules in a knowledge base. The *C3.1.1 Structure* of the knowledge base is a main theme. Furthermore, students are supposed to understand the *C3.1.2 Closed-World-Assumption* in order to analyse the facts and rules. *C3.1.3 Dynamic modification* of the knowledge base needs to be discussed, too.

With respect to *C3.2* students are supposed to perform queries. Hence, they analyse the external structure of logic programming. On the one hand, students formulate queries to solve a given problem. On the other hand, they should be able to predict the answer. *C3.1.2 Closed-World-Assumption* has a strong connection with this competence.

C3.3 Tracing combines the understanding of the internal and the external view. Students are supposed to understand *C3.3.1 Unification and backtracking*, especially intern and extern backtracking. *C3.3.2 Trace modus*, *C3.3.3 Box model* and *C3.3.4 Parse tree* can be used to illustrate and to explore the structures and to support the competence *C2.2.3 Development of logic programs (including debugging)*.

C4 Non-cognitive competencies

The non-cognitive competencies are applied from [16]. A separation of cognitive and non-cognitive competencies in empirical researches allows reflecting on their correlations [20, pp. 4-5]. Therefore, this research project starts with the measurement of the cognitive competencies.

Summary

The presented first version of a SMC of logic programming is the basis for the development of test items to measure the learning outcomes of students in informatics secondary education. Our main research aim is to evaluate and to improve this education. The improvement of this first version of a SMC is an intermediate research aim. Further researches will focus on a first version of a model for levels of competence (MLC) regarding logic programming.

5 Conclusions

Important criteria for a good learning process in informatics education, which were mentioned in chapter two, are fulfilled with logic programming. E.g. applications of logic programs can be used to motivate and to support a comprehensive view on informatics. Because of the competence orientation the other criteria, classification of tasks and measurement of competencies, should be based on a SMC. This article presented a first proposal of a SMC to classify logic programming competencies based on school curricula. This SMC needs evaluation through further empirical studies. Therefore, test items based on *C1-C3* will be developed to measure the learning outcomes of students at schools.

The work on the SMC showed us that there are different opportunities to structure the competence dimensions and sub dimensions. Starting point for a further discussion can be the strong connection between *C3.1.2 Closed-World-Assumption* and *C3.2 Queries*. Although closed-world-assumption occurs by formulating the knowledge base, it might be more correlated to queries. A major point for discussion might be the connection between *C2.2 Logic programming* and *C3 Informatics views*. The different views are necessary to gain the basis competencies of logic programming. Despite their relationship, both competencies should be tested separately. This might be a difficult task. The application of the test items will show us if a separation is possible and helpful or not. With further research results we will be able to review the first version of a SMC of logic programming.

Another argument could be that unification and backtracking can be illustrated with trace modus, box model and parse tree. Hence, unification and backtracking might be seen as a superior level. On the one hand, it could replace *C3.3 Tracing*. On the other hand, *C3.3.2* could be changed to *C3.3.1.1* and so on. In this respect, further research on this is still required.

Furthermore, the use of media is essential for the success of the learning and teaching process in logic programming. Learners need different learning aids for simulation, animation and visualisation of structures of logic programs and of the special kind of virtual machine, the Prolog machine. Therefore, there is a need for learning tools to be reviewed on their didactics potential. This will show if there is a lack of appropriate tools.

Our intention is the continuation of measurements, to explore the learning of logic programming in detail, and to use this knowledge to improve the test items together with the SMC and the MLC. The competence orientation in informatics education is an encouraging development. With respect to competence orientation the SMC is one important step for our research process.

References

1. ACM and IEEE (eds.): Computer Science Curriculum 2008, (2008), www.acm.org/education/curricula/ComputerScience2008.pdf
2. Tucker, A. (ed.): A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee. 2nd Edition. New York, ACM, (2006), www.csta.acm.org
3. Tucker, A.: K-12 Computer Science: Aspirations, Realities, and Challenges. In: Hromkovic, J., Královic, R., Vahrenhold, J. (eds.): Teaching fundamental concepts of informatics. Proceedings of the 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives (ISSEP), Springer, Berlin, pp. 22-34, (2010)
4. Cleverbot: www.cleverbot.com
5. RoboCup: www.robocup.org
6. National geographic Daily News: Watson Wins Jeopardy! Published February 17, 2011, news.nationalgeographic.com/news/2011/02/pictures/110217-watson-win-jeopardy-ibm-computer-humans-science-tech-artificial-intelligence-ai
7. Tate, B.A.: Seven languages in seven weeks: A pragmatic guide to learning programming languages. The Pragmatic Bookshelf, Raleigh, (2010)
8. King, R.D. et al.: The Automation of Science. In: Science, vol. 324 no. 5923 pp. 85-89, (2009), www.sciencemag.org/content/324/5923/85.full
9. Taylor, J., du Boulay, B.: Studying novice programmers: Why they may find learning Prolog hard. In: Rutkowska, J., Crook, C. (eds.): Computers, Cognition and Development: Issues for Psychology and Education, John Wiley, Chichester, pp.153-173, (1987)
10. Stamatis, D., Kefalas, P.: Logic programming didactics. In: Proceedings of the Informatics Education Europe II Conference, pp. 136-144, (2007), www.seerc.org/ieeii2007/PDFs/p136-144.pdf
11. Di Bitonto, P., Roselli, T., Rossano, V.: Formative Evaluation of a Didactic Software for Acquiring Problem Solving Abilities Using Prolog. In: Paolini, P, Garzotto, F. (eds): Proceedings of the 8th International Conference on Interaction Design and Children, ACM, New York, pp. 154-157, (2009), portal.acm.org/citation.cfm?id=1551815
12. Scherz, Z., Haberman, B.: Logic programming based curriculum for high school students: The use of abstract data types. In: Laxer, C. et al. (eds.): Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education, Tennessee, pp. 331-335, (1995), portal.acm.org/citation.cfm?id=199848
13. Roehner, G.: Informatik mit Prolog. 3rd edn. Amt f. Lehrerbildung, Frankfurt, (2007)
14. Amzi! inc.: Adventure in Prolog. www.amzi.com/AdventureInProlog
15. Anderson, L., Krathwohl, D.: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom‘s Taxonomy of Educational Objectives. Longman, New York, (2001)
16. Logic programming curricula: www.die.informatik.uni-siegen.de/logicprogramming
17. Magenheim, J., Nelles, W., Rhode, T., Schaper, N., Schubert, S., Stechert, P.: Competencies for Informatics Systems and Modeling. In: Proceedings of the 1st Global Engineering Education Conference - Educon 2010, IEEE Computer Society, pp. 513-521, (2010), Digital Object Identifier: 10.1109/EDUCON.2010.5492535
18. Weinert, F.E.: Concept of Competence: A Conceptual Clarification. In: Rychen, D., Salganik, L. (eds.): Defining and Selecting Key Competencies. Seattle, (2001)
19. Kumar, A.N.: Prolog for imperative programmers. In: JCSC 17, 6, pp. 167-181, (2002)
20. Klieme, E., Leutner, D., Kenk, M. (eds.): Kompetenzmodellierung. (2010), www.pedocs.de/volltexte/2010/3324/pdf/Beiheft56_komplett_D_A.pdf

Appendix: Conversation with Cleverbot

Rollo Carpenter, who received several prices for his researches, developed Cleverbot (see [4]). A special version of Cleverbot won the Machine Intelligence Competition 2010 with average rating of 42.1% human (see www.bcs-sgai.org/micomp2/index.html, http://www.cleverbot.com/machine). Cleverbot (Cb) extends its knowledge bases, when people interact with it, e.g.:

- Hello!
- Cb: How are you?
- Fine.
- Cb: Good.
- How are you?
- Cb: Fine
- How is the weather?
- Cb: I do not know.
- Have a look out of the window.
- Cb: I can't, I'm in mom's basement.
- Oh, I'm sorry for you.
- Cb: Why are you sorry for me? I am glad you have sympathy, but there is no reason for you to be sorry.
- You like to live in your mom's basement?
- Cb: I have never left the country.
- Where would you like to travel to?
- Cb: I would like to travel to South America.
- Bye.
- Cb: Where are you going?
- Back to work.
- Cb: So you're leaving?
- Yes
- Cb: Okay, Cya.

Social competencies appraisals in CSE and in a computer science contest

Gabor MeiÇner

Friedrich Schiller University Jena, Ernst-Abbe-Platz 2
D-07743 Jena, Germany
gabor.meissner@uni-jena.de

Abstract. Jury members of a national computer science contest and teachers use different ways to appraise social competencies. On the one hand social aspects are more important for teachersô appraisals than for jury membersô appraisals. On the other hand if a contest participant e.g. shows a bad behavior the result could be not winning a price or an award without regarding other aspects of the overall performance. Furthermore teachers show other criteria to appraise social competencies.

Keywords: Social competencies, contest, teachers, appraisals

1 Why talk about social competencies?

The complexity of modern life confronts learners and individuals with new challenging demands. The Definition and Selection of Key Competencies (DeSeCo) Project defined three broad categories of such competencies: use tools interactively, interact in heterogeneous groups and act autonomously [7]. This article focuses on the second category, the interacting in heterogeneous groups from the perspective of teachers of computer science education in contrast to the jury members of a computer science contest [8].

Computer science is closely linked to social interaction in two different ways. To develop informatics systems the developer has to comprehend different roles to design these systems. Such systems should be used at different levels of experience, can be modified by other developers and have to meet the conditions of the contractor. The other perspective on social interaction in computer science is the need to develop informatics systems in heterogeneous groups [4].

The objective of this article is to provide hypothesis to point out the differences between appraisals of social competencies in computer science education and in a computer science contest. In this study, a semi-structured interview with five jury members and six teachers was used to get a general idea of social competencies appraisals in computer science.

In the second part of this article the different contexts will be pointed out. Later the method will be presented and in the fourth part the results of the interview will be discussed.

2 Context of the appraisals in CSE and in the contest

The curriculum of the interviewed teachers in the German federal state of Thuringia provides a semiannual project phase in 12th grade in which students have to develop computer programs with self-chosen objectives. Regulations demand to work in a team (ñ[The students] have to organize and coordinate their work in project groupsò[1] [11]).

The teachers have to measure different aspects of the studentôs achievements, e.g. the topic selection, the implementation, the evaluation and the usability [11]. Mostly social interaction of the group members will be measured. Each student gets an individual feedback on his or her achievement.

On the other hand jury members measure the achievements of the participants of the task-based Federal Contest in Computer Science (German: Bundeswettbewerb Informatik, shortened: BWInf). The contest lasts one year and is open for youths up to the age of 21. To get a price, an award and/ or an invitation to the International Olympiad in Informatics (IOI) the participants have to solve problems in three rounds. Price winners are entitled to the German National Academic Foundation. Because the foundation requests social competencies on a high level it is an important reason for observing social interactions between the group members.

The contest mainly addresses excellent secondary school students, but in the first round a broad effect is also intended [10]. Group work is allowed in this first (and third) round. The second round demands much more skills and knowledge in variable fields of computer science. Both rounds have to be done as homework. The most problems can be solved by (delivering the source code of) self-written computer programs including documentation; but also other types of tasks are possible [6].

The best 30 participants of the second round will be invited to a third symposium-like final round which last four days. On the second and third day the participants take part in a 30 minute specialist discussion and a six-hour task-based group work, mostly with three other participants. As proposed by Cormack et.al. [2] informatics-orientated collaboration is an essential part of the third round. The group work is observed by two associated jury members. The goal of this phase is to solve some (hard) problems without additional aids, especially without computers. Afterwards the group has to present their results and put them up for discussion. This article focuses on the juryôs appraisal of the group work.

---

[1] All translations (German ï English) were made by the author.

To compare the appraisals of the jury members and teachers it is necessary to point out some differences caused by different frameworks and guidelines. Teachers have to make their decisions by themselves; the appraisals of the jury are a result of a social negotiation process of about 15 jury members. The juryĝs discussion bases on a numerical score for a participant. A jury member who observed a group gives a score to every group member and a jury member who held an interview gives a score to the interviewed participant. The appraisals are not determinate by a strict guideline [9]. Nevertheless this process will not be investigated, because descriptions of jury sessions could not be done by regarding privacy standards. Also the group work phase during the contest is much shorter and could be observed all the time. In school context students could also do their work at home. In the classroom the teacher cannot observe all groups at the same time. From the perspective of the teacher there is a lack of information which can be compensated by knowing the student over a longer period or using self-evaluation methods. Therefore the expressed views of the teachers and the jury members will be treated equally in this article.

The DeSeCo Project group defined three social competencies which are important for both kinds of group work [6]:

- *the ability to relate well to others,*
- *the ability to cooperate and*
- *the ability to manage and resolve conflicts*.

For such projects, as the mentioned school project, long-term objectives have to be set in a negotiation process. It is important to present, understand and discuss ideas and to make decisions (second competency). Furthermore it is important to make sure that inter-personal conflicts do not have a negative impact on the objectives; the students have to deal with such conflicts (first and third competencies). The group work during the contest also requires such competencies. In contrast to the school project long-term organization is not needed. But because of the cooperation with unknown (and not self-chosen) group members abilities to relate well to others could be more important, e.g. empathy or effective management of emotions. In both scenarios competencies to bridge the gap between different levels of subject-related knowledge and skills are needed to ensure the project objectives [4].

3 Method

The interviews were divided into two parts: On the one hand, the objective was to find out how teachers and jury members measure subject related achievements. On the other hand, the interviewed should explain how to appraise social competencies. Only the answers of the second part will be evaluated in this article.

In November 2010 two pre-test interviews took place. The results of the pre-test interview were only used to edit the questionnaire. Between December 2010 and February 2011 eleven semi-structured interviews were held. The interviews were conducted by phone and were recorded. For the evaluation of the transcribed interviews the qualitative content analysis by Mayring [5] was used. Two experts rated the qualitative data of the transcribed interviews[2]. Both experts currently work at the University of Jena and research in CSE. In order to increase the inter-rater reliability a list of categories and anchors were added. The author developed categorized statements and anchors to five different topics.

3.1 Sampling

To ensure that the most aspects of the research question can be identified a statistical sampling was chosen. The sample

- includes the groups of jury members and teachers,
- only includes jury members or teachers who are experienced in appraising students or contest participants and
- has a similar distribution of gender, age, qualification and experience as the population.

The sample considers of six teachers and five jury members of the third contest round. The participated teachers live and work in the German federal state of Thuringia. They are aged between 41 and 55 years (. . 3 female and 3 male). The jury members are aged between 36 and 52 years (. . . Four jury members work as Professors (all male) and one female jury member work as a research associate. All jurors research in Theoretical Computer Science. Because of the focus on social competencies the main research topic could be a confounding variable, because of the fact that social interacting in different areas of computer science could have a different importance. As well as in the sample the most jury member research in Theoretical Computer Science. So only a small impact of this variable could be expected.

The interviewed teachers and jury members have a lot of experience in appraising students or contest participants. This might be a confounding variable, but it was necessary to ensure that the interviewed teachers and jury members can be assumed as experts. In this article, experts are defined by their appraisals experience and their subject-related knowledge (qualification) [3]. Both conditions are met.

3.2 Analysis Method

As mentioned the qualitative content analysis by Mayring was chosen for analyzing the statements (see table 1). Categories were found inductively because of missing researches and theories in this field and because of the implication of the analysis method. Quantitative results are not important in this study. On the other hand the objective was to find characteristics of the different groups.

---

[2] For all categories, the inter-rater reliability was higher than 0.7 (Cohen-Kappa ə .

Table 1. Simplified steps and operations of the analysis method

Step	Operation	Comments
1	Selection of material	Parts of interviews of 6 teachers and 5 jury members.
2	Summarize	The answers of the interviewed were reduced step by step to a similar annotation. Developing of questions.
3	Explication	Using material to clarify open questions (e.g. guidelines).
5	Inductive development of categories	Developing of categories, anchors and coding rules (see chapter 3.3).
6	Ensuring of the inter-rater reliability	Two raters evaluated the categories according to the material (Redefinition of the categories, anchors and coding rules).
7	Interpretation	

The following questions were posed to the material:

1. How do social skills or the ability to cooperate impact your appraisal?
2. How important are social skills for your appraisal?
3. Why are social skills important for the contest/ the computer science education?
4. What is more important: Having a good idea or the implementation of the solution?
5. Which criteria do you apply when appraising social skills?

In several steps categories, anchors, descriptions and instructions for coding were developed. In this study it will be differentiated between social and non-social aspects of the appraisals. Social aspect will be investigated; but non-social aspects must be treated as a black box.

3.3 Category systems

The first question focuses on the structure of the appraisal model. With the help of this question the interaction between these two aspects should be illustrated. Thereby two different types of structure are possible: If a test person says that social aspects are part of the overall performance and have a specific weight, the impact will be called ñadditionalò. In contrast to that if a student or contestant misbehaves so that other aspects of the performance get unimportant and the overall performance get poor, the impact will be called ñmultiplicativeò (see table 2)[3]. Propositions of a specific weight of social aspects (in contrast to the weight of non-social aspects) could not be done with the help of question 1.

[3] In this article the term ñmultiplicativeò bases on the productivity model by Walberg [12]. The term ñadditiveò was often used to describe other model types [1].

Table 2. How do social skills or the ability to cooperate impact your appraisal?

Code	Category	Anchor
A	Additive	ñI do appraise the product and, with one or two points[4], the cooperation of the group members.ò
B	Multiplicative	ñGroup work depends on something like social competencies. Many participants who performed well in the interview disqualified themselves because they misbehave during group work.ò

If a test person gives several answers and it is possible to assign them in both categories, than mostly the overall answer to question 1 must be ñmultiplicativeò, because the impact of a multiplicative structure has a higher weight. The answers to the second question express the weight or the importance of social competencies in the overall performance. Three categories similar to Likert scales were found (important, moderately important and of little importance). Only one category per test person can be assigned. If answers of a test person can be assigned to different categories a specific category has to be chosen by the help of the context (see table 3).

Table 3. How important are social skills for your appraisal?

Code	Category	Anchor
A	Important	ñI think it is very important. I think it is indispensable.ò
B	Moderately important	ñThatôs good, but it has little to do with computer science. I would appraise this, but not overvalued in this conte t.ò
C	Of little importance	ñIn real life, in the real society it could be important to have some social skills. Nevertheless in the contest I would appraise it only very low.ò

With the help of the answers to the third question the reasons of the importance could be pointed out. The statements were assigned to four categories. The first category, guidelines, sums all answers which mention guidelines. In school context this could be the curriculum or internal school programs. The contest has to ensure that the expectations of the German National Academic Foundation are fulfilled. The second category is the general importance of social competencies and skills for everyday life or the carrier. Social competencies could also be important for CS (third category). In the fourth category all other answers were summarized, e.g. contest publicity (see table 4).

Table 4. Why are social competencies important for the contest/ computer science education?

Code	Category	Anchor
A	Guidelines	ñThe national winners will receive a scholarship of the

[4] In this conte t ñpointò is a synonym for grade. The worst grade is 0 points and the best grade is 15 points.

		German National Academic Foundation (german: Studienstiftung des deutschen Volkes) without any other demands. The foundation e pects social skills on a very high level.ò
B	Important for everyday live or carrier	ñIn the society we are living in we have to cooperate always with others at some point.ò
C	Important for CS	ñI think the idea that group work should be established in all subjects is a little overrated. The way we do it in CSE is just the right way.ò
D	Others	ñIf a national winner or an award winner [é] misbehaves, it would be bad publicity for the contest [é].ò

In order to get an idea of the importance of a specific social aspect (cooperation and the willingness to implement an idea of another student or participant) relatively to a specific non-social aspect (having a good idea to solve a problem) question 4 was posed. Both alternatives are necessary to solve a problem in both scenarios. The question was formulated as a forced choice question, but because of openness of the interview neutral answers were possible. Neutral answers could be ñboth aspects are importantò (category C or ñthe important depends on the problem context, the idea or the complexity of the implementationò (category B, see table 5). The answers to that question will be summarized in one category. If answers of a test person can be assigned into different categories, the main idea of the answer has to point out with the help of the context.

Table 5. What is more important: Having a good idea or the implementation of the solution?

Code	**Category**	**Anchor**
A	Idea is more important	ñ[é] I would prefer the idea. In my opinion the objective is to identify those who are able to solve hard problems.ò
B	Depends on the complexity of the idea or the implementatio n	ñThere is not a generalized answer to that question. If someone has a trivial idea and it is a lot of work to realize it, then I would not appraise the idea highly. On the other hand if the idea is brilliant and it is not so much work to do, this is mostly relevant for scientists, not for participants, I would rate the idea highly.ò
C	Neither idea nor implementation is more important	ñBoth could be important. I would not go so far and say something is more important. It depends on the cooperation.ò
D	Implementatio n is more important	ñIn school conte t it is more important to implement a reasonable solution because the result matters [é]ò

The last question was about criteria for appraising social skills (see table 6). Four categories were found (social behavior, functioning of cooperation, equitable distribution of group work and others). In contrast to the questions 1, 2 and 4 all answers will be covered (also answers that were given more than once).

Table 6. Which criteria do you apply when appraising social skills?

Code	Category	Anchor
A	Social behavior	ñI think in such a contest you have to ensure to promote well-behaving participants instead of nerds.ò
B	Functioning of cooperation	ñI appraise the functioning of the group work.ò
C	Equitable distribution of group work	ñIs the workload well-distributed?ò
D	Other criteria	ñMy appraisals? I appraise the product, especially the program code and the documentation [é]. What else? The students have to present their product in front of the class [é] and the other students have to ask questions. The quality of these questions is part of the appraisals.ò

4 Results

A clear difference between the investigated groups turned out by analyzing the answers of the first question. Four times the interviewed teachers gave an answer that can be categorized to an additive appraisal structure (category A). On the other hand four jury members tended to a multiplicative appraisal structure (category B). Two teachers and one jury member gave no (clear) answer to that question.

In order to see the importance or the weight of the social aspects in the overall performance question 2 were posed. The teachers tend to appraise social skills higher than the jury members. The teachers said that social aspects are important (n=3) or moderately important (n=2). The importance of social aspects in the overall performance was assumed by the jury members as moderately important (n=2) or of little importance (n=3).

With the help of the third question the reasons for the importance of social aspects can be pointed out. In this question there were no clear differences between the groups. Teachers said that guidelines (n=2) require the appraisals of social aspects, that they are important for everyday life or career (n=2) and that they are an essential aspect of a good performance in CS (n=2). Similar to that the interviewed jury members answered only once that guidelines are the reason for the importance of social aspects. Answers that can be categorized into category B or category C were given twice by the jury members and also other answers were given twice (for an example see Table 4).

The fourth question focused on a statement if an idea or the implementation is more important for the overall performance. The jury members tend to a higher importance of the idea (n=4) or to an appraisal that depends on the complexity of both aspects (n=1). The statements of the teachers were more diverse. A higher importance of the idea (n=2) was also mentioned as like as a higher importance of the implementation (n=2) and similar importance of both factors (n=2).

Table 7. Answers to question 5.

Test person	**Answers**	**Test person**	**Answers**
J1	AABCC	T1	BCDDD
J2	AABCC	T2	D
J3	AAC	T3	CCDDDD
J4	ADD	T4	CDDDD
J5	*No criteria*	T5	BC
		T6	ABCDDDD

Another difference can be shown with the help of the answers to question 5 about criteria of the ñsocial performanceò (see table 7). On the one hand jury members often gave the answers that can be summarized as ñsocial behaviorò (category A), functioning of group work and equitable distribution of the group work. On the other hand teachers gave only once an answer that can be categorized as ñsocial behaviorò. Other criteria were mentioned more often. Sometimes these criteria were not social aspects (see table 5). Similar to the group of the jury members they often mentioned answers of the categories B and C.

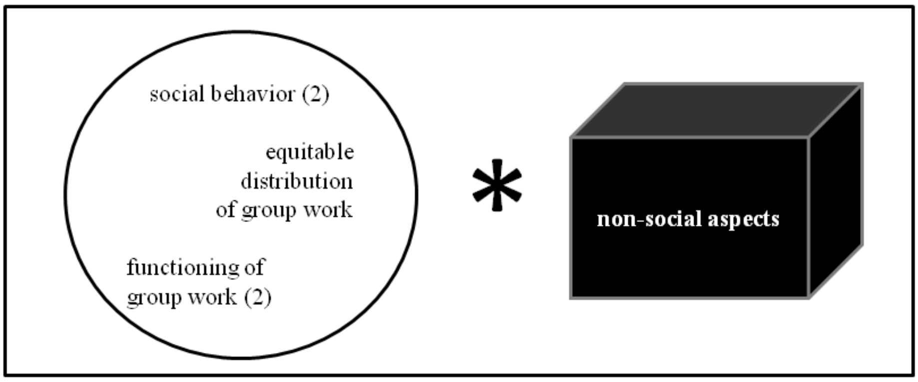


Fig. 1. Prime example for jury membersô appraisals

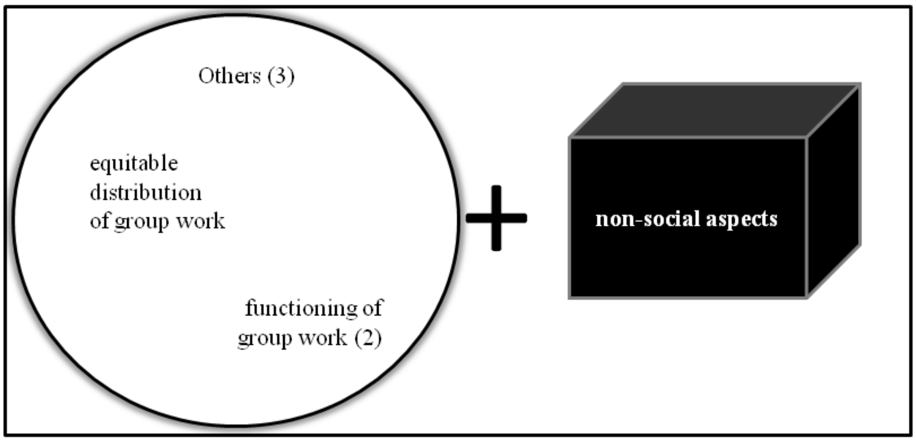


Fig. 2. Prime example for teachersô appraisals

Typical answers of jury members and teachers are illustrated in figure 1 and 2. The figures show specific test persons of both groups, but can be treated as prime examples.

5 Discussion and Challenges

As mentioned the interviewed teachers use an additional structure to break into specific criteria and summarize the parts to an overall performance. On the other hand the interviewed jury members use a holistic way. The social and non-social parts of the overall performance have to be connected multiplicatively. A multiplicative structure implies a higher importance to low performance factors because these factors make a good overall performance impossible. The answers to question 2 seem to be contradictive. Teachers attribute social aspects a higher importance than jury members do (see table 8).

Two reasons for that difference can be pointed out with the help of material. First, contest guidelines demand that price winners have social skills on a high level. Jury members have to exclude participants with an almost bad behavior. Teachersô guidelines demand a comprehensible and fair grading. From the teachersô point of view it would not be fair to give a bad mark because of only one factor. A teacher described the school as a ñprotected roomò. Maybe the contest does not offer such a protection by the appraisals of the jury members. The second reason for the differences between the answers to question 1 and 2 could be the subjective perception of the own role and objectives. From this perspective the interviewed jury members said that they want to find out which participants could be the best computer scientists. In contrast to that teacher may focus mainly a general education. Social competencies seem to be an important part of that general education. This impression could be underlined by analyzing the answers to question 4 about the role of ideas and implementations. In the mentioned context the implementation of an idea has to be done in group work, so social competencies are necessary to ensure a good product. Jury members tend to attribute a higher importance to the ideas, so it can be assumed that they percept their task in finding the best computer scientists (and not the best team worker). On the other hand teachers cannot observe the group work all the time and because of that they cannot ensure to assign an idea to a specific person. In this context the product and a comprehensible way to the product could have another importance.

Table 8. Generated hypothesis

Question	Hypothesis
1	Teachers use an additive and jury members use a multiplicative structure to summarize social and non-social aspects.
2	For teachers social aspects are more important for the overall performance than for jury members.
3	-
4	Jury members focus the aspect of having a good idea. Teachers do not prefer the aspect of having a good idea nor the implementation.
5	Jury members mention social behavior more often than teachers.

Jury members often focus social behavior as an aspect for their appraisals. Meanwhile, only one teacher mentions that aspect. Maybe a reason for that could be a more specific catalog of criteria which teachers developed in their pedagogic education, with their experience and by reflecting guidelines and other impacts (e.g. the demand of comprehensible marking). The answers to question 3 do not show such differences between the investigated groups.

In order to test the generated hypothesis a questionnaire or other quantitative methods could be used. Some hypothesis (hypothesis 1, 2 and 4; see table 8) could be tested directly by developing similar questions. For testing hypothesis 5 it could be useful to formulate questions according to the importance of the different aspects.

References

1. Beck, E. et al: Adaptive Lernkompetenz: Analyse und Struktur, Verª nderung und Wirkung handlungsstreuenden Lehrerwissens, Waxmann, M¿nster 2008.
2. Cormack, G. et.al.: Structure, Scoring and Purpose of Computer Competition. In Pohl, W. (Ed.): Perspectives on Computer Science Competitions for (High School) Students (2006), http://www.bwinf.de/competition-workshop/Submissions/11_Cormack+.pdf.
3. Glª sel, J., Laudel, G.: Experteninterviews und qualitative Inhaltsanalyse, 3rd Edition, VS Verlag, Wiesbaden 2009.

4. Kollee, C. et al: Computer Science Education and Key Competencies. 9th IFIP World Conference on Computers in Education - WCCE 2009, http://www.die.informatik.uni-siegen.de/e-publikationen/Publikationen/2009/WCCE2009_pap147.pdf
5. Mayring, P.: Qualitative Content Analysis [28 paragraphs]. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research, 1(2), Art. 20, http://nbnresolving.de/urn:nbn:de:0114-fqs0002204 (2000).
6. MeiÇner, G.: Aufgaben der ersten Runde des Bundeswettbewerbs Informatik ï Kontinuitªt und Wandel. Didaktik der Informatik - Mºglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik, 81-92 (2010).
7. OECD: The Definition and Selection of Key Competencies. Executive Summary (2005), http://www.oecd.org/dataoecd/47/61/35070367.pdf.
8. Pohl, W.: Computer Science Contests for Secondary School Students: Approaches to Classification. Informatics in Education (2006).
9. Pohl, W.: Computer Science Contests in Germany. Olympiads in Informatics, 141-148 (2007).
10. Schwill, A., Romeike, R.: The Development of a Regional CS Competition. Proceedings of the 4th conference on Informatics in Secondary Schools ISSEP, Z¿rich, 106-116 (2010).
11. Th¿ringer Kultusministerium: Lehrplan f¿r das Gymnasium Informatik (1999), http://www.thillm.de/thillm/pdf/lehrplan/gy/gy_lp_if.pdf
12. Walberg, H. et al: Syntheses of educational productivity research Original Research Article International Journal of Educational Research, Volume 11, Issue 2, 1987, Pages 147-252.

Exploring Computer Science Teachersô Subjective Theories on Designing their Lessons

Ana-Maria Mesaroк and Ira Diethelm

Carl von Ossietzky University
Computer Science Education
26111 Oldenburg
Germany
ana.maria.mesaros@uni-oldenburg.de, ira.diethelm@uni-oldenburg.de

Abstract. In this paper we will show first steps in the research area of subjective theories of computer science teachers at secondary schools and lay the theoretical basis for that. After clarifying our approach that uses semi-structured interviews we will present selected results from the pretest. This initial analysis confirms that exploring computer science teachers’ subjective theories on designing their lessons is an important and promising research field. The pretest showed that computer science teachers really have very diverse perceptions on how to design lessons on the topic networks and the Internet. After knowing teachers’ subjective theories we can include them in designing effective teacher training programs and thus improve teacher education.

Keywords: subjective theories, teacher training, educational reconstruction

1 Introduction

From a constructivist point of view and based on the idea of the Educational Reconstruction [8] effective lessons can only be designed if students’ perspectives are recognized as being that important as the scientific clarification of the subject matter. Therefore, the structuring of lessons needs to include students’ perspectives.

For the design of teacher education for computer science teachers at secondary school this basic assumption can be adapted. Therefore the subjective theories of computer science teachers have to be taken into account when it comes to the design of teacher educational programs. In order to activate teacher to learn and to change certain behaviors or teaching patterns one needs to comprehend and include the existing teachers’ perspectives [12].

Consequently, we need to investigate teachers’ subjective theories [6] on designing computer science lessons. Our approach is to investigate them with semi-structured interviews and to analyze them afterwards with the qualitative content analysis [9].

Here, we show the motivation for this research approach on computer science teachers’ subjective theories on designing lessons. Additionally we set the theoretical framework for research in this field. We first describe the research questions which we focus on in this research area. Further we show the theoretical framework in which these questions can be answered. Therefore, we present the model of Educational Reconstruction and its adaption for computer science teacher education. Afterwards the concept of subjective theories is explained.

After clarifying the theoretical framework the research methodology is presented, clarifying which design we chose to answer the research questions. Than we describe the structure of this study, the data collection instrument and selected preliminary results. As a forecast we present first conclusions and the expected results from further investigation.

2 Situation of Computer Science Teachers

Compared to teachers of other subjects computer science teachers are in an inconvenient situation. The biggest problem is that they are a minority within the teaching community and that many schools are provided with only one or even no computer science teacher. This fact makes collaboration amongst computer science teachers very difficult [3]. Often they do not have colleagues to share their ideas and materials and to get feedback on them. This sometimes inhibits the development for field specific pedagogical confidence amongst the computer science teachers.

Another problem is caused by the fact that computer science is a very young discipline that has not been established in secondary schools yet. As a consequence there is a diversity of materials and no mandatory standards [3] nor a mandatory curriculum. The materials are based on different concepts for teaching computer science and often not tested. The missing standards lead teachers to the challenge of choosing their own topics for computer science education. That underlines the importance of the subjective theories of computer science teachers because they have a significant impact on the topics that are taught during computer science lessons in school.

A further hint on the wide range of computer science teachers’ subjective theories is the fact that there are several ways to become a computer science teacher. Some of them studied computer science education; other ones have studied computer science without any pedagogical content and others were teacher for other subjects and participated in additional computer science courses [3]. Fast changing technological innovations make computer science even more difficult to teach. Computer science teachers often have to deal with changing conditions and new tools [3], more than teachers of other subjects.

Some of the facts mentioned above do not seem to be an observation specifically in Germany but may show tendencies that can be applied to more places. Changing conditions and different educational concepts are widely spread difficulties of computer science education.

3 The Research Questions

In such special and challenging situation teacher trainings for in-service teachers become even more important. For the aim of designing sustainable in-service teacher training we need to know the teachers, their thinking and the preconditions about computer science education, in order to meet their needs with an adjusted training program.

This research tries to give a first overview of the patterns that guide teachers in their planning processes. In this way we get an impression of the subjective theories of computer science teachers. The planning of courses for computer science teacher training will benefit from the results of this study. If the existing subjective theories of computer science teachers are known, teacher education can be designed based on them. Researches show that incorporating teachers’ perspectives into teacher training lessons makes them more efficient [12].

Therefore, the main question of this research is: *What are teachersô subjective theories on designing computer science lessons?*

Therefore, the following two questions have to be answered:

- Which subjective theories guide teachers’ thoughts while designing lessons on the topic networks and the Internet?
- What importance do students’ perspectives have in computer science teachers’ planning for these lessons?

The theoretical framework for this research is given by the model of Educational Reconstruction, adapted for teacher education. The model shows a possible way to design teacher training lessons based on the perspectives of computer science teachers.

4 The Theoretical Framework

These research questions are embedded in the framework of Educational Reconstruction for teacher education which shows the significance of teachers’ subjective theories on the design of their lessons. This model is used in several educational research areas and can be adapted for computer science education as well.

To clarify the deduction of the research questions we show the theoretical environment in which they are embedded. Therefore, we first present the model of Educational Reconstruction and describe how it can be adapted for teacher education in computer science. Afterwards we introduce the concept of subjective theories which serves as the basis of this research.

4.1 The Educational Reconstruction

The model of Educational Reconstruction [8], as shown in figure 1, can be used as a framework for educational research, for curriculum development and to improve teacher education. It was developed for natural sciences, but it can also be adapted for other areas as well. The main idea of the model is the integration of the students' perspectives on a chosen topic and the clarification and analysis of the subject matter. Out of this integration guidelines for the design of learning environments can be generated.

Therefore the design of learning environments includes not only the scientific view of a subject matter but also the students' perspectives on this topic. Furthermore, students' perspectives are as important as the clarification of the science content for planning lessons on a subject matter.

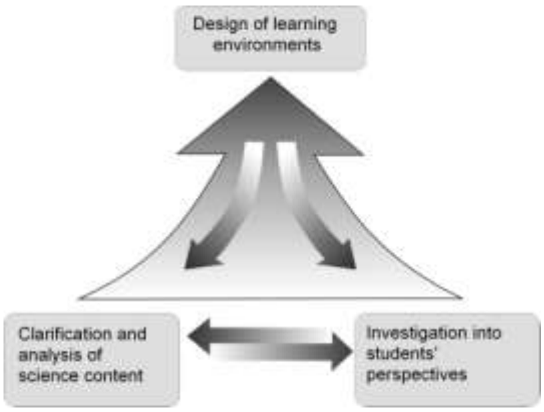


Fig. 1 The Educational Reconstruction see[8]

Following, the idea of Educational Reconstruction means that the science content is expanded by students' perspectives so that it can be arranged for teaching. The science content is being decomposed and enlarged by students' perspectives. Therefore the content structure for teaching becomes much more complex than the science content structure itself ([8], p. 174). This process of rearranging and expanding the science content for designing learning environments is the main principle of Educational Reconstruction.

4.2 The Educational Reconstruction for Teacher Education

The core ideas of the Educational Reconstruction can be adapted for teacher education, to improve not only the lessons at school but also teacher training. It might also be used for designing guidelines for the education of pre-service and in-service teachers [8], see fig. 2.

This adaptation has varying consequences. First, teachers need to recognize the importance of students' perspectives for learning and they need to be familiar with typical perspectives on the topic, in order to include them in the computer science lesson planning process. Another point of view is designing courses for teacher education where the teacher takes the role of the learner. Teacher education courses should not only show teachers the importance of including students' perspective into their lessons. They need to meet teachers’ needs themselves and ñpractice what they preachò as well.

The educational structuring of computer science lessons is influenced by the subjective theories of teachers. As a consequence a description and analysis of teachers’ perspectives status quo is needed. Therefore an approach is necessary for investigating computer science teachers’ subjective theories on designing their lessons on a certain subject matter. As a result of this investigation we will show typical conceptions of computer science teachers.

Fig. 2 The Educational Reconstruction for Teacher Education see [8]

These subjective theories of computer science teachers on their teaching are as important as the domain specific educational concepts. That means that designing teacher education can not only be grounded on educational concepts of computer science. They must include the personal aspects of teachers, their pedagogical knowledge, their subjective theories and their conceptions of educational structuring as well [8].

Through connecting teachers’ perspectives on educational structuring of their computer science lessons with concepts of computer science education, in-service teacher trainings can achieve a fulfillment of the actual needs. To sum up we keep hold on the fact that teachers’ perspectives on educational structuring are influenced by their subjective theories on designing computer science lessons. Therefore we need to define the concept of subjective theories.

4.3 Subjective Theories

Kelly illustrated 1955 what subjective theories are and what function they have, without explicitly naming them: ñMan looks at his world through transparent patterns [é] which he creates and then attempts to fit over the realities of which the world is composedò ([7], Page 7). He named these patterns ñpersonal constructsò and explained them as ñways of construing the worldò ([7], Page 7). Teachers also have this personal constructs on the design of learning environments. Many scientists name them differently, like ñnaiveò or ñsubjectiveò theories. Groeben [11] named them ñsubjective theoriesò and designed a research approach for educational research on these personal theories of teachers.

Based on Kelly [7] and Groeben et al. [11] we define subjective theories as individual cognitive structures of self- and worldviews that have function of explanation and prediction.

Because of the missing standards in computer science, computer science teachers can, and have to, decide on how a specific subject matter is taught at school, where they set priorities and in which intensity they handle a topic. Following Dann [1], these decisions are not only affected by their content knowledge but also by their personal theories that determine their thoughts about computer science lessons.

5 The Design of the Investigation

The intention of this study is to explore computer science teachersç subjective theories on designing computer science lessons. Groeben et al. [11] define the human being as ña reflective and (potentially) rational subject, capable of language and communicationò, see also [6]. The fact that subjects are reflective and capable to communicate this reflection implies that subjective theories can be investigated with a dialogue-hermeneutic method [6] such as an interview. The presented inquiry relies on the fact that subjective theories can be explored through interviews. The method we use is a semi-structured guided interview. After transcribing these interviews they will be evaluated with the method of qualitative content analysis, following Mayring [9].

The aim of this research is to create guidelines for teacher training. As a consequence it is not important to illustrate how often a belief is represented. But it is important to identify a wide range of computer science teachersç subjective theories that occur during our investigation. These theories are considered to be valuable for the inclusion within the design of programs for teacher education. Because the focus of the research is on each subject it is not important to have a representative number of interviews but to have the opportunity to understand each subject. That makes a qualitative research approach necessary.

We are aware of the gap between planning a lesson and a carried out lesson and we do not claim to solve this problem. Therefore we decided to set the focus on the planning action, so that the subjective theories that lead teachersç decisions preliminary to the lessons are investigated. Because of the predictive function of subjective theories investigating teachersç thoughts on designing lessons provides enough information about teachersç perspective on educational structuring of computer science topics.

To enhance the quality of the answers we decided to do the investigation on a specific topic. This also helps to compare the results of different teachersç interviews. For several reasons we chose the topic networks and the Internet. Some of these reasons are listed below:

- Studies show that the Internet has a high relevance to the everyday life of children [10].
- The topic of communication is named by Denning as one of seven great principals of computing [2].
- The Computer Science Education Standards published by the German Association of Computer Science (Gesellschaft f¿r Informatik ï GI) [5] includes the topic of networks and the Internet.
- Students' perspectives on this topic have been investigated lately [4].

ñNetworks and the Internetò is not just because of the computer science content like the idea of communication an important topic in computer science education. Beyond that, the role of the Internet in the everyday life of children increases daily. Therefore, it should not be missing in computer science lessons as it often does. Furthermore, this topic can be connected to other subjects such as politics, history or economics as well.

Of course there are other important topics in computer science as well, but we think that this is one of the most important. Therefore, the data collection is based on the design of a lesson on networks and the Internet. In the following we present the structure of the interview.

6 Data Collection and First Findings

For the data collection we chose a semi-structured guided interview. This means, that additionally to the planned questions other questions can be asked as well. This may happen if utterances have not been understood or if there is a need for more detailed explanations. Sometimes questions already have been answered before they were actually asked by the interviewer. In this case, the interviewer has to include the given answers into the question and ask if the teacher has something to add. Thereby the teacher gets the opportunity to elaborate on the first answer and give more information which can be revealed.

In the following we try to give a general overview on the different key aspects of the interview. Therefore we list some questions and show selected, interesting answers.

6.1 Planning of the Current School Year

As an introductory question we asked the teacher how he or she planned the current school year. This question gives an overview on the topics the teacher has been planning for this year. Furthermore, it directs the focus on the (professional) expertise of the teachers, of their planning in other topics and helps to generalize the results of this study. Additionally, this question provides the opportunity to talk about their special situation in their school and about the problems they have to solve. This is very important because computer science teachers often do not have computer science colleagues at their school to exchange experiences. Thereby this question opens a new conversation level at the beginning of the interview in which the teacher feels understood. They start to speak quite openly and mostly start to state that they feel a lack of computer science knowledge.

6.2 Designing Lessons on Networks and the Internet

Then, questions follow about the concrete planning of lessons on networks and the Internet. Teachers are asked about the order of contents, teaching methods and learning objectives. These are well known main aspects in designing lessons.

Since none of the teachers that we interviewed in our pretest taught or planned these lessons yet, a short pause followed the question on how they would design a computer science lesson about networks and the Internet. This was followed by an astonished: ñNow?ò or ñReally?ò Afterwards they started thinking about how they would plan this lesson for the first time. The following descriptions show the variety in designing computer science lessons:

One of the teachers would try to avoid technical input in these lessons and set the focus on the impact of the Internet on the society. He would start by generating an error like a homepage that can not be found. In this way he intends to increase the interest of the students on this topic. As an aim for his lessons he would like his students to know why this error occurs and how it can be fixed. Additionally, he explained that first time teaching a topic is just an exercise or experiment in which he finds out what to improve next time.

Another teacher would split the teaching unit into two focus areas. The first one would focus on the usage of the Internet and the second one on the structure and the functionality of it. He could not elaborate on the structure of the second focus. He explained his lack of knowledge on this subject matter and uttered he would wait for his colleague to plan the second part and afterwards he would adopt this planning.

The third teacher had a very clear picture of how he would structure the lessons. He thought about an example to start with and about the order of the themes. He was not sure which example to take at the beginning but mostly he liked the idea of students trying to seek informations about them in the Internet. Although he does not clearly know how he can build a bridge to the technical contents he knows that topics like network topologies, protocols and the OSI model would follow.

Another teacher had a different point of view on this unit. From his point of view students should learn how to design and build up a computer network and use their knowledge to improve the school infrastructure. He would just give a short theoretical input at the beginning of the lessons and afterwards let the students work quite independently because his students are very good at researching information and self-guided learning.

6.3 The Studentsô Perspectives

Afterwards, questions about the students' perspectives follow. Teachers are asked to think of the topic and evaluate whether it was interesting for their students and of possible difficulties.

Our pretest has shown that teachers expect their students to have high interest in this topic. The difficulties they can imagine are strongly connected to their own challenges on this topic. As soon as it becomes more detailed and the scientific level increases teacher see difficulties of learning. There were just a few answers to these questions because the teachers we asked could not really imagine where difficulties might arouse. That is why we are going to add some additional questions to the main survey and present a studentsø perception to the teachers and ask them how they would react on that. This may hopefully lead us to more results.

6.4 Teachersô Attitudes Comparing Computer Science to other Subjects

The following questions focus on the other subjects taught by teachers. In Germany teachers normally teach at least two subjects. After getting the information of their second subject we ask them to reflect on any differences to the planning for other subjects.

Analyzing these answers we detected an interesting phenomenon: One teacher always searches for fun in his computer science classes, but he does not search for fun in his math classes. Therefore, he spends a lot of time planning something impressive and entertaining for his computer science lessons. For the math classes he has a scheme how lessons have to be constructed. There is a theory part, in which he gives an input, then he gives an example how to use the new theory and afterwards the pupils have to do some exercises. When planning math lessons he implements a lesson based on the scheme and he does not search for something impressive. This phenomenon may be an explication for another encountered circumstance: the unhappiness with existing material for computer science lessons. An explanation for this might be the fact that computer science teachers have huge expectations on computer science materials.

6.5 Teachersô Biographical Data

To close the interview questions about the teachersø biography are asked. These questions are intentionally at the end of the interview, because there is a phenomenon specific to computer science teachers that can ruin the interview from the beginning: Computer science teachers often say ñI havenøt studied computer science, so Iøm not a proper computer science teacher.ò They perceive themselves as an interim solution due to the lack of properly educated computer science teachers.

Our investigation so far shows that there are several ways and diverse motivations on becoming a computer science teacher. Two of the interviewed teachers had taken additional courses to become a computer science teacher. The other two had the qualification to teach other subjects (math, physics and chemistry) and teach computer science without any additional training.

These findings confirm that in Germany computer science teachers have highly diverse ways of becoming a computer science teacher. This might apply to other countries as well. The fact that teachers have diverse ways of becoming computer science teachers led us to the conclusion that they most probably have different subjective theories about designing computer science lessons.

7 Conclusions and Future Work

According to the fact that we are just at the beginning of this study, we are only able to present brief results of our pretest. They already illustrate that computer science teachers’ subjective theories is an interesting research area that needs to be explored. These few interviews deliver results that motivate us to investigate further teachers’ perspectives on designing computer science lessons. As first results we can sum up:

Computer science teachers…

- have different ways in designing lessons on a given topic. They set different focuses and handle the topic differently.
- do not have a clear idea of the students’ perspectives.
- have diverse ways of becoming a computer science teacher.
- have various points of views on computer science compared to their other subject.
- often feel a lack of knowledge on computer science.
- think they are not a proper computer science teacher because of their (missing) qualifications.

These findings have to be used carefully because the main investigation follows this summer. As a change to the main survey, we are going to focus more on the knowledge of teachers about students’ perspectives on the topic of networks and the Internet. This might give us more detailed answers about when and where teachers think that students have difficulties in understanding this topic; the first interviews did not really contain information about that. That is why we are going to confront the teachers with a specific perspective of students such as that students think the Internet consists of only one big computer and ask how he would react if this view occurs in his lessons [4].

As a result of the main survey we expect to identify and be able to describe different approaches for teaching the topic networks and the Internet. We also expect that further results will permit a generalization of the subjective theories for other topics of computer science.

The fact that these interviews already showed very diverse perceptions shows a tendency on how different computer science teacher think about designing their lessons. These few interviews already show that we have to include teachers’ perspectives on planning computer science lessons in the process of designing teacher education.

We showed here an approach that leads us to our aim, which is generating guidelines for teacher training. Computer science teachers’ subjective theories should serve as a solid foundation for planning lessons for teacher training. If we understand how teachers think about their classes we have the chance to guide them to the aim of having better computer science lessons in schools. Furthermore, the results of this investigation might show the needs for teacher training of computer science teachers. The results of this study are going to foster the improvement of in-service teacher training.

8 References

[1] H.-D. Dann. Subjective theories and their social foundation in education. In M. von Cranach and W. Doise, G. Mugny. Social representations and the social bases of knowledge, p. 161-168., Lewiston, Hogrefe & Huber, 1992.

[2] P.J. Denning. Great principles of computing. Communications of the ACM, 46(11):15-20, 2003.

[3] I. Diethelm, C. Hildebrandt, L. Krekeler. Implementation of computer science in context - a research perspective regarding teacher-training. In Koli Calling 2009, p.97-100, Koli, 2009.

[4] I. Diethelm, S. Zumbrª gel. Wie funktioniert eigentlich das Internet? ï Empirische Untersuchung von Sch¿lervorstellungen zur Funktionsweise des Internets (How does the internet work? ï Empirical invesitigation on studentsĂperceptions of the functionality of the internet), In: 6. Workshop der Didaktik der Informatik, p. 33-44, Kº llen Verlag, Bonn, 2010, in German.

[5] Gesellschaft f¿r Informatik (GI) e. V. Grundsª tze und Standards f¿r die Informatik in der Schule (principles and standards for computer science at school). LOG IN 150/151(28) : LOG IN Verlag, Berlin, 2008, in German.

[6] N. Groeben, B. Scheele. Dialogue-hermeneutic method and the "research program subjective theories". Forum: Qualitative Social Research, 1(2):9, 2001.

[7] G. A. Kelly. The psychology of personal constructs. Routledge, London. Kelly, G. A. (1991): The psychology of personal constructs. London: Routledge.

[8] M. Komorek, U. Kattmann. The model of educational reconstruction. In S. Mikelskis-Seifert, U. Ringelband, M. Brueckmann (Eds.), Four Decades of Research in Science Education - from Curriculum Development to Quality Improvement, chapter 7, p. 171-188.Waxmann,2008.

[9] P. Mayring. Qualitative content analysis. Forum: Qualitative Social Research, 1(2):10, 2000.

[10] Medienpª dagogischer Forschungsverbund S¿dwest JIM-Studie 2010: Jugend, Information, (Multi-)Media (youths, information and multi-media). Forschungsberichte, Baden-Baden, 2010, in German.

[11] J. Schlee, B. Scheele, N. Groeben, D. Wahl. Forschungsprogramm Subjektive Theorien. Eine Einf¿hrung in die Psychologie des reflexiven Subjekts (research program on subjective theories). A. Francke Verlag, T¿bingen, 1988, in German.

[12] D. Wahl. Lernumgebungen erfolgreich gestalten. Vom trª gen Wissen zum kompetenten Handeln (The effective design of learning environments). Julius Klinhardt Verlag: Bad Heilbrunn, 2005, in German.

Discovering the Creativity in Informatics

Katarína Mikolajová[1], Martina Kabátová[1],

[1] Department of Informatics Education, Faculty of Mathematics, Physics and Informatics, Mlynská dolina,
842 48 Bratislava, Slovakia
{katarina.mikolajova, martina.kabatova}@fmph.uniba.sk

Abstract. In this paper we scrutinize various aspects of creativity in education and namely in school informatics. We have reviewed many literature sources looking for definition of creativity in informatics and the result was not satisfactory. We decided to build a definition and theory about various aspects of creativity in informatics using the qualitative research methods and grounded theory approach. The emerging theory is compared with literature. In the paper we present early version of our findings.

Keywords: informatics, education, creativity, digital technology

1 Introduction

We are living in a creative society, where new ideas and original solutions are becoming more and more valuable. In today´s fast changing world we need to prepare our pupils for uncertain future which is neither possible to imagine nor to define key competencies and skills for living in the world in twenty or even ten years.

We need to prepare pupils for their lives and future professional careers. What will they definitely need is the ability to quickly adapt to new circumstances, to communicate, to cooperate in team and *"think and act creatively"* [1].

In the last three years (since the national school reform in 2008) discourse about creativity in education has become more important in Slovakia. The National Curriculum in Slovakia highlights the importance of fostering creativity in schools and in informatics as well.

Although The National Curriculum emphasizes fostering creativity in informatics in various contexts, the explicit definition of creativity as well as ways how to support it are missing. We need to find a key characteristic of creativity to define this concept and determine relevant factors, which can influence creative process in schools. Since digital technologies (when properly used) can facilitate creative process we are trying to focus on them and characterize digital tools that support creativity.

2 Creativity in Education

Although creativity is often and widely discussed on popular and academic levels, there is still no consensus on explicit definition of creativity. The term creativity is used frequently and with different meanings – there are various perspectives for studying creativity and many different definitions of creativity in the psychology. There are two essential attributes in almost every definition of creativity – **novelty** and **value** [2].

Today we know that creativity is not exclusive to the geniuses or extraordinarily gifted people. Everyone is creative at some level and we should develop pupil´s creative skills during the educational process as one of the key competencies for living in 21st century. Creativity is a complex capacity, not only general ability [3] which requires three variables to appear [4]:

- domain-relevant skills,
- creativity-relevant skills,
- and task motivation.

The definition of creativity applicable to the educational process is one in the NACCCE report, which defines creativity as: *"Imaginative activity fashioned so as to produce outcomes that are both original and of value"* [5]. But what does it mean to produce something original and valuable in the school setting? Banaji et al. [6] refer to two key implicit distinctions in this definition – *"it distinguishes between imagination and purposeful imagination and defines originality as new to the child, not necessarily the world"*.

For better understanding of creativity we need to distinguish between two types or levels of creativity. Boden [7] describes historical creativity (h-creativity or eminent creativity, also labeled Big-C creativity) and psychological creativity (p-creativity, also labeled little-c creativity). Whilst Big-C creativity appears exceptionally and brings historically novel and original ideas which are fundamental in the field and nobody has had them before, little-c creativity is typical for everyday creative experiences that help to express ideas, solutions, emotions or opinions in original way. This type of creativity is also called everyday creativity and brings novelty which is relevant and meaningful for a certain group of people [3], for example classmates.

Beghetto & Kaufman [4] broaden conception of creativity and describe mini-c creativity, which is typical and essential for learning process, when pupils construct their personal knowledge and understanding. Although the problem solution may be well-known or introduced in the textbook, it can be new and original in relation to pupil´s previous work and knowledge, if he or she explores this solution by himself/herself. The main distinction between little-c and mini-c creativity is in judgment - whilst both Big-C and little-c creativity *"rely on interpersonal and historical judgments of novelty"*, mini-c creativity produces something novel and meaningful to the individual which is intrapersonally evaluated [4].

2.1 Creative teaching

The role of the teacher is crucial to promote creativity and facilitate creative activities within the classroom. They establish conditions and atmosphere which support the creative process, conduct creative learning activities, actively engage pupils and through their thinking and acting represent a role model of creative behaviour.

Feldhusen & Treffinger [8] and QCA [9] provided recommendations for establishing atmosphere which stimulates pupil´s creativity and give advice to the teacher how to promote creativity within the classroom:

- teacher should appreciate and foster unusual ideas, solutions and responses of students. It is important to perceive failures or mistakes positively as a part of the creative process,
- teacher should provide the pupils with various resources, prompts and support, enable them to reflect and concentrate,
- teacher should be open to pupils' ideas or interests and adapt them into lesson plan if possible. Unexpected events can motivate and stimulate pupils to work creatively, it´s allowed to put aside lesson plan and 'go with the moment',
- time is very important in the creative endeavor, pupils need sufficient time to think about creative ideas and explore them, that might not occur immediately or spontaneously,
- teacher should encourage pupils to solve divergent problems which support lateral thinking, encourage pupils to ask questions, make connections, work imaginatively, ask open-ended questions such as "What if...?" to explore various perspectives,
- teacher should establish a confidence building approach where pupils can take risk and freely present, communicate and share their ideas within the classroom. Atmosphere in which pupils feel safe is key factor for stimulating creativity,
- teacher should have fun and laugh with pupils, respect them, value what pupils do and say, reward originality and imagination,
- teacher should participate in creative activities, act and think creatively as a role model. Demonstrating that he or she is still learning can help to establish an opened atmosphere,
- teacher should create conditions for teamwork, where pupils can cooperate with classmates or even with pupils from a different age group.

2.2 Digital technologies that support creativity

Digital technologies (DT) have the potential in supporting creativity and creative process. Loveless [10] describes four key features that digital technologies provide for creative process:

- **provisionality** – they provide easy way to make changes or make step back during the performance, to try alternatives, to make drafts or to trace of the thinking process and development of ideas,

- **visualization and interactivity** – DT can provide dynamic, reasonable and immediate feedback on decisions or actions currently made, they can simulate or visualize processes or realities and enable manipulation of variables or changing conditions, so users can better understand and make connections and relations,
- **capacity and range** in which DT provide access to information from all over the world in any time, it's also wide range of tools that digital technologies enable us to use in thinking and problem solving process,
- **speed and automatic functions** which enable users to store, transform, transmit and especially analyze, synthesize, interpret, share and communicate information and ideas more effectively or more intelligibly.

Greene [11] and Schneiderman [12] analyze properties and characteristics of digital tools that support creativity. They define some requirements that DT should fulfill to facilitate creativity:

- DT should support *"pain-free explorations and experimentations",*
- enable users to move back (return to previous steps) or forward, work continuously and step-by-step,
- there should be no big penalties for errors or mistakes and success should be rewarded with meaningful response,
- there should be *"immediate and useful feedback for one´s action"*,
- user should have sense of control over actions and processes,
- DT should support meaningful visualization of data and processes for better understanding and exploring relationships and connections,
- enable explore various alternatives and solutions through 'what-if' scenarios and simulations,
- provide access to large databases of resources and digital libraries to find inspiration and gather knowledge,
- let users to disseminate their outcomes, products or artifacts to gain reputation and broaden databases of accessible resources.

Programmable media [13] (which enable users to create animations, interactive games, stories, simulations etc.) should let users customize their projects in an easy way – change backgrounds, objects, choose from available libraries or make own figures, scenes or pictures up to personal style and preferences.

2.3 Valuing creativity

"To Understand Is To Invent."
J. Piaget

If we want to think about the evaluation of creativity, first we need to answer the raising questions concerning value and originality in creative efforts and results.

Beghetto & Kaufman [4] emphasize creativity as a dynamic process. They propose that a creative process is more important than a creative product. Boden [7] assumes that everyone is creative at some level. Rather than asking if the idea is creative, she proposes to ask how creative the new idea or artifact is and in which way.

The NACCCE [5] proposes three different categories of originality: individual, relative and historical. **Historical originality** is involved in Big-C creativity and such outcomes are uniquely creative in the field. **Relative originality** is a part of little-c creativity and produces outputs, which are original in relation to the classmates or to a particular peer group. Mini-c creativity involves **individual originality** which produces outcomes that are *“original in relation to his or her own previous work”* [15].

In the school setting it is the little-c and mini-c creativity, which are most likely to occur. The mini-c creativity is typical in a situation when pupils construct their knowledge by themselves – in such situation it is very likely that new findings will be original and meaningful for them. This type of creativity is the most important in the learning process and a teacher should be aware of such efforts and reward them. Moreover, the teacher should suggest normative standards, authorities and positive role models in the field.

Runco [14] claims that the creative expression is often *"personal and not easily compared with normative standards"*. The results of the creative effort are novel or original for classmates *"but not in comparison with some larger norms"*. Hence Runco recommend to focus attention on a creative potential *"rather than unambiguous creative performance"* [14].

Nevertheless, it is important to encourage pupils in self-evaluation and critical reflection on their own and others´ work. Craft et al. [15] present some thoughts on how to judge creativity and to overcome the subjectivity of this judgment:

- *“determine clear criteria for excellence”* and requirements for the tasks,
- be flexible in the judging (be opened to various solutions even the surprising ones),
- get variety of evaluative methods (teacher's evaluation, peer review, self-evaluation),
- failure in creative process should not be judged negatively.

3 What Do Teachers Think About Creativity in Informatics?

In this chapter we analyze the data gathered via several interviews with teachers. Our goal is to identify core themes in this context and propose a definition of creativity in informatics. For this purpose we chose a qualitative approach heavily relying on grounded theory methodology (though at this point of research not all the criteria are met and some strategies of the grounded theory have not been used yet).

Sampling - we combined several types of purposive sampling [16] and we chose teachers from primary and secondary schools that we knew are active, innovative and think critically about teaching and learning processes. For the future sampling an exact method on how to choose those teachers is needed (e.g. a questionnaire that would estimate their fit for our research).

Data collecting and preparation - with four participants we digitally recorded the interviews and then transcribed them into digital text form. The interviews were unstructured. We began with general questions about teaching and school life and then we moderated the interview to creativity related topics. Other three participants responded to seven open-ended questions via e-mail (e.g. Are there some lessons you consider to be the creative ones? If so, how are they organized, what is the teacher's role? Are they different form less creative lessons? Describe the activities the pupils do.). Once we collected the data we printed and organized them by type (interview, questionnaire responses).

Coding process - in first reading we explored the data looking for emerging patterns and themes (this partially happened already by interviewing and transcribing). Then we applied open coding and broke text into segments, labeling each of them with code. By axial coding we refined the categories and looked for more accurate relationships among them. The result is a detailed map which depicts the categories and their relations (simplified version is in the Fig. 1.). We describe our findings in the following chapters.

This research is not finished. We are aware that data has not been saturated yet (we need to interview more teachers as well as enhance the methods on choosing them). More detailed selective coding is needed to reduce the categories and their properties. Only then we would be able to state the resulting theory on creativity in informatics. There are also several emerging questions that are interesting to examine closer. We plan to interview more teachers and reinterview the participants we already had and ask them more specific questions.

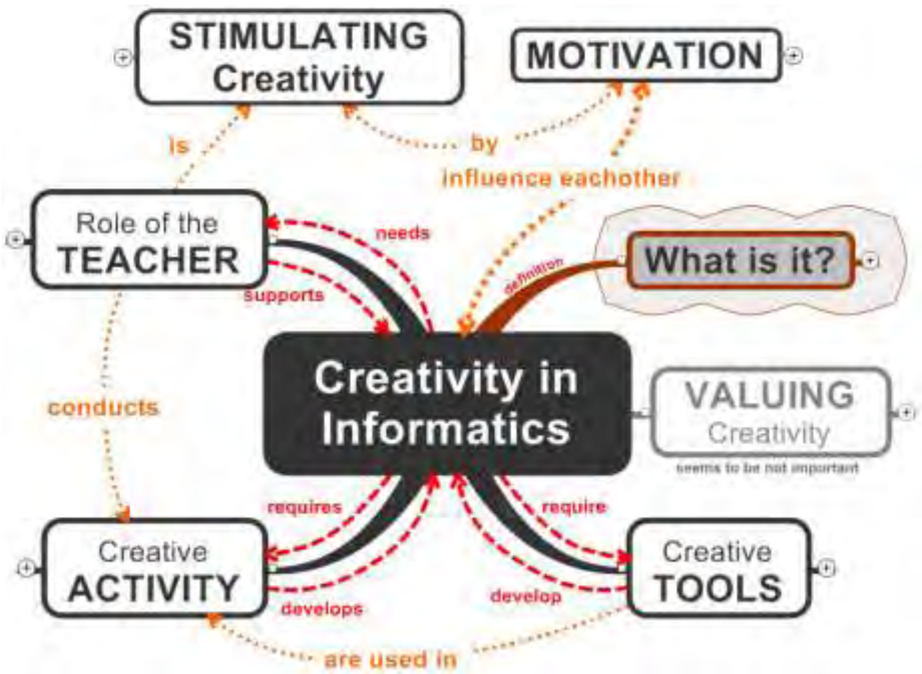


Fig. 1. Map of main themes (categories) emerging around the creativity in informatics.

The **main themes (categories)** that represent creativity in informatics are:

- definition of creativity,
- characteristics of creative activities,
- role of the teacher in this process and it's connection to motivation and means of supporting creativity,
- features of creative tools.

3.1 What is Creativity in Education and Informatics

We have identified **five domains** in which creativity is present in the context of school informatics (and generally in education). Creativity means

- **to have ideas** - ability to come up with something new, different, original, innovative and imaginative,
- **to make things happen** - to do more than the instructions asked for, to work independently,
- **to know how to** use tools and knowledge acquired at school, to know how to connect things together,
- **to discover** solutions, be able to find more than one solution,
- **to react** in different situations, to process various inputs.

As the most important the teachers marked the **ability to creatively use the tools which were a subject of the informatics education** (usually they meant software applications such as graphic editors but some also mentioned this in the context of programming and by tool they meant some programming structure such as a loop).

All teachers implicitly spoke about 'little C' or 'mini C' creativity. They were aware of the fact that their pupils usually did not come up with any revolutionary inventions. They readily appreciated if children discovered something that is new only to themselves or in the classroom environment.

Teachers also identified various aspects of creativity. They mentioned **visual**, **algorithmic** and **literary** creativity. This categorization is not final since only some teachers admitted that solving algorithmic problem might be creative. We suspect that many teachers do not find programming to be the core topic in school informatics and they focus much more on the user skills with various applications (editors).

Other classification of creativity regarded the **creativity of a process** and **creativity of a product**. Again this topic is seen differently - some teachers observe and appreciate both types of creativity, some of them endorse only the product, others only the process. We plan to investigate more what the reason for this disagreement is.

3.2 Creative Tools and Digital Technology

So far we have identified several features that are common for creative educational tools (programming languages and environments, educational software, microworlds, software applications, etc.):

- **simplicity** - tool must be easy to use and to get familiar with, complicated interface restricts the creativity,
- **wide walls** - tool must have many possibilities and wide range of use,
- **visual elements** - this feature relates to motivation, teachers believe that children like working with visual tools, it also links to the visual (or artistic) type of creativity,
- **interactivity** - again interactivity is a feature popular with children and stimulates their motivation and thus also creativity,
- **orientation on the product** - tool has to produce something (so the creativity of product might appear), the product should be interesting, entertaining or useful.

3.3 Creative Activities

The teachers pointed out that any activity in the classroom **has to meet the educational objectives**. Creative activities often jeopardize their fulfillment. A teacher has to ensure educational objectives are met and this is often very difficult to manage. They stated that **creative activities are demanding** and a teacher has to be very flexible. However, during the activity the role of the teacher is minimized and the focus is on the **learner who works on his own**. We assume that during such creative activity a teacher 'teaches less' but the nature of this teaching is much more demanding than during regular lessons.

The teachers often described **improvised activities** they prepare *"only in their head"* due to lack of time or will to prepare them beforehand. Some teachers do this on purpose and improvising is their modus operandi. Regardless of the reason these activities are reported to be very successful. Teachers perceive them as creative (we suspect this relates more to the teacher's creativity than to the learner's creativity).

The most important feature of a creative activity is its **open-ended nature**. Teachers characterized it as

- without given output,
- proposing wide theme,
- the learner has freedom to choose (theme, solving strategy, methods, ...),
- has various solutions.

Examples of such activities:

- the child has to make up its own task (*"make up your own"*) and solve it (e.g. to make up their own picture a turtle should draw, or to make up a new function for the project they programmed at previous lesson),
- the child adds a new feature to existing solution (e.g. to the program),
- multimedia projects (creating movies, clips, music videos, books, radio broadcast - this activities often include writing a scenario, working with DT, using editing software and teachers regard them as fairly difficult to finish).

Convergent activities are perceived as restricting the creativity - it can emerge only in the process not in the product (since it is given in the instructions). Some teachers do not recognize the process as creative and thus convergent activities are not creative either.

3.4 Role of the Teacher

As hinted in previous section the role of the teacher is to **manage the class**. Ensuring the educational objectives are met is important. Other positions involve

- **giving ideas**, stimuli and prompts,
- **interacting with the learner** - the teacher is flexible, is able to continue the lesson in various directions, finds benefits in every situation that appears, he also helps the learner if they go astray from the lesson objectives.

Stimulating Creativity

Most of the teachers admitted that they **do not support creativity on purpose**; it is more of a by-product of other objectives. However, they were able to identify some strategies for supporting creativity:

- give ideas and other extrinsic stimuli,
- conduct open-ended activities,
- give enough space for learner's ideas,
- do not restrict, only guide,
- recognize, appreciate and present learner's creativity.

None of the teachers had a strategy for grading creativity. They either did not know how to evaluate it or did not even feel the need to evaluate it since creativity is not their primary goal of education. Although they stated that recognition and appreciation is needed to maintain the creative environment.

Creativity and Motivation

Motivation is understood as a crucial factor in the process of learning. Motivation and creativity go hand in hand, they influence each other and they both enable a child to learn new things. Most teachers identified the personality of the teacher himself as a core factor for extrinsic motivation. They mentioned '**atmosphere**' of the class created by the teacher, group of learners, environment and 'school spirit'. This atmosphere has to be favouring creativity to stimulate it.

The teachers recognize also an intrinsic motivation of children, but some think they are not able to develop it and take it as a given property of the child. Others think they can work also with intrinsic motivation by giving appropriate prompts.

Valuing Creativity

The teachers do not grade creativity - they do not consider creativity to be their educational objective and therefore grading it is not their goal. They grade only the correctness of the solution. Some of them recognize only the visual (artistic) creativity and since they are not educated in this field they do not feel entitled to judge the originality.

They were not able to word any methods how to value creativity, though they stated they could recognize if a child is creative or not. They also emphasized that creativity should be rewarded - verbally or by presenting before the class. They feel that promoting creativity is important but they have no systematic methods to find out if creativity is occurring in the classroom or not.

4 Conclusion

The data collected from the teachers show that their understanding of creativity is similar to the theory we have learnt from literature. Even if literature heavily recommends focusing on the creative process some teachers do not recognize this process as part of creative performance and they value only the product. The teachers defined creativity in informatics mostly as an ability to effectively use the (digital) tools they mastered in the class, to discover solutions and to work on their own.

Teachers defined creative activities as open-ended, where pupils can modify the instruction, choose theme or make up their own task. They also stated that work with multimedia is creative.

The role of the teacher in the creative process is to facilitate creativity, provide ideas and focus the activity on the pupils. These activities are perceived as more demanding for the teachers because they need to react flexibly in unexpected situations. Important in this process is motivation that supports creativity and vice versa. Appreciating creativity and creating a creative atmosphere in the classroom is equally important. Teachers do not have systematic methods to judge creativity and do not feel entitled to do so.

Our early analysis opened several interesting question we haven't considered before (e.g. what does it mean that some teachers do not recognize creative in the process, why is creativity often limited only to the visual aspect, how should creativity be valued in classroom). We plan to investigate them in greater detail and look for possible explanations.

References

1. Resnick, M.: Sowing the Seeds for a More Creative Society. The Media Laboratory, Massachusetts Institute of Technology (2007)
   On-line: <http://web.media.mit.edu/~mres/papers/Learning-Leading-final.pdf>
2. Kusá, D. et al.: Zjavná a skrytá tvorivosť (Hidden and obvious creativity). Ústav experimentálnej psychológie SAV, Slovak Academic Press, Bratislava (2006)
   On-line: <http://www.psychologia.sav.sk/tvorivost.pdf>
3. Lassig, C. J.: Promoting creativity in education: from policy to practice: an Australian perspective. In Proc. of the 7th ACM Conference on Creativity and Cognition: Everyday Creativity, pp. 27--30. University of California, Berkeley, California (2009)
   On-line: <http://eprints.qut.edu.au/28958/1/c28958.pdf>
4. Beghetto, R. A., Kaufman, J. C.: Toward a Broader Conception of Creativity: A Case for “mini-c” Creativity. Psychology of Aesthetics, Creativity, and the Arts, Vol. 1, No. 2, p. 73--79. American Psychological Association, Washington (2007)
5. Department for Education and Employment (DfEE) All Our Futures: Creativity, Culture and Education. DfEE, London (1999)
   On-line: <http://www.creativitycultureeducation.org/data/files/naccce-all-our-futures-249.pdf>
6. Banaji, S., Burn, A., Buckingham, D.: The Rhetorics of Creativity: A review of the literature. Arts Council England, London (2006)
   On-line: <http://www.creative-partnerships.com/data/files/rhetorics-of-creativity-12.pdf>

7. Boden, M.: The Creative Mind: Myths and Mechanisms (2nd ed.). Routledge, London (2004)
8. Fasko, D.: Education and creativity. Creativity Research Journal, 13(3--4), p. 317--327. Routledge, Philadelphia (2000)
9. Qualifications and Curriculum Authority (QCA): How can teachers promote creativity? (1999). On-line: <http://webarchive.nationalarchives.gov.uk/20100823130703/http://curriculum.qcda.gov.uk/key-stages-1-and-2/learning-across-the-curriculum/creativity/howcanteacherspromotecreativity/index.aspx>
10. Loveless, A. M.: Literature review in creativity, new technologies and learning. Futurelab series, Report 4, School of education, University of Brighton (2002) On-line: <http://archive.futurelab.org.uk/resources/documents/lit_reviews/Creativity_Review.pdf>
11. Greene, S. L.: Characteristics of applications that support creativity. Communications of the ACM, Vol. 45, No. 10 (October 2002), pp. 100--104. CACM, New York (2002)
12. Schneiderman, B.: Creativity support tools. Communications of the ACM, Vol. 45, Issue 10 (October 2002), pp. 116--120. CACM, New York (2002)
13. Monroy-Hernández, A., Resnick, M.: Empowering Kids to Create and Share Programmable Media. In: interactions – Pencils before pixels: a primer in hand-generated sketching, Magazine, Vol. 15 Issue 2 (2008)
14. Runco, M. A.: Education for Creative Potential. Scandinavian Journal of Educational Research,Vol. 47, No. 3. Routledge (2003) On-line: <http://www.coe.uga.edu/torrance/files/2010/08/Scandinavian_Journal.pdf>
15. Craft, A., Jeffrey, B., Leibling, M.: Creativity in Education. Continuum International Publishing Group Ltd., London (2001)
16. Creswell, J. W.: Educational research: Planning, conducting, and evaluating quantitative and qualitative research. Pearson Education, Upper Saddle River, New Jersey (2008)

Op Art or the Art of Object-Oriented Programming

Nikolina Nikolova[1], Eliza Stefanova[1], Evgenia Sendova[2]

[1] Sofia University „St. Kl. Ohridski", Faculty of Mathematics and Informatics,
5, James Bourchier Blvd.,
Sofia 1164, Bulgaria
{nnikolova, eliza}@fmi.uni-sofia.bg

[2] Institute of Mathematics and Informatics, Bulgarian Academy of Science,
Acad. G. Bontchev, Bl.8
Sofia 1113, Bulgaria
jenny@math.bas.bg

Abstract: The paper presents an application of the project-based approach, supported by information and communication technologies (ICT), in the context of informatics education. The experiment under consideration presents classes, performed in parallel in two mathematics high schools in Sofia with 11 graders, working on the topic of *Op Art*. This experiment is another implementation of the findings of the *Innovative Teacher* European project (*I*Teach*), viz. that the integration of ICT into the project-based learning approach can enhance the students' *soft* skills in accordance with the knowledge-based society requirements. The ICT tools are used within the described experiment in three directions: (i) as an integrated development environment (IDE); (ii) as tools supporting education, and (iii) as tools for building a virtual environment for communication and education. The goal of the experiment is to propose a model, which could be applied in the future at secondary school informatics. Furthermore, it demonstrates the authors' strong belief that informatics should not be reduced to *learning how to program*, but should rather be based on *learning by programming*.

Keywords: project-based approach, ICT-enhanced skills, virtual learning environment, integration of informatics and art

1 Introduction

The *specialized classes in informatics* are real challenge for teachers. There are about 30 schools having such classes in Bulgaria, three of them – in Sofia. These classes deal with advanced object-oriented programming. The curriculum is similar to the curriculum for the first two years at the university. The students are prepared to participate in national and international competitions in informatics and IT. This means the learning content is quite heavy and sometimes – quite demotivating in terms of content.

The main efforts of the teacher in these classes usually are focused on acquiring some basic knowledge and skills in programming and data structures, as well as in algorithmic thinking.

In practice, the training is often reduced to teaching the very language constructions, usually by uniform examples, which are not meaningful for the students. Our personal experience and what we have been sharing with teachers in informatics strengthens our conviction that applying this widely used approach leads to a lack of motivation and to superficial skills. In addition, those students who continue their higher education in the area of computer science often comment later that although they have studied hard in school, they either do not remember or just could not apply anything of the material studied. In harmony with the constructionists' spirit we have been applying the credo *learning by programming is more important than learning to program* in our practice in the frames of the Research Group on Education (RGE) [1] and within educational materials influenced by the RGE principles of *integrating school subjects* and *learning by doing*.

The project-based approach has also been successfully applied in a limited number of schools that have adopted the Logo philosophy [2]. The approach however was not spread on a large scale, due to the lack of time and appropriate environment for team work during its application. The contemporary information and communication technologies however *enlarge the classroom in time and space* thus providing new opportunities for the project-based learning.

To introduce on a larger scale the effects of integrating knowledge from various fields in the context of the project-based learning there was a need of examples working in a real setting. We had several recent *theorems of existence* – a *Data structures and programming* [3] at the Faculty of Mathematics and Informatics (Sofia University), and teacher training courses within the Innovative Teacher Project *I*Teach*.

The experience at university level presented in [4], however, is based on the fact, that the most of the required knowledge is being presented to the students in the traditional lecture style, while the project-based approach is only supplementary, as an enhancement. The challenge for the secondary school teacher is bigger – to build all the expected knowledge and skills in the context of the work on a project, within relatively limited time.

Below we give an example how these complex aims have been achieved in a school setting by applying the methodology of the Innovative teacher (*I*Teach*) [5], developed in the frame of the Leonardo da Vinci project [6]. The specific characteristics of the *I*Teach* methodology are:

- The learning process is driven by students' interests.
- The students are faced with a challenge, which motivates them to participate actively in the process of learning.
- The students work in teams on a project, whose goals they formulate themselves.
- *The road* to the goal is a metaphor behind a specific educational scenario (Figure. 1) with *milestones* of intermediate objectives. The teacher guides the students to the ultimate project goal by interweaving his/her own pedagogical

goals concerning the learning content with the building of ICT-enhanced soft skills.

Figure 1. Metaphors of an *I*Teach* scenario

Applying the *I*Teach* methodology turns out to be especially appropriate in informatics education in secondary school by creating a platform for new forms of interactions, a new class management and supporting activities contributing to the development of the 21st century competencies. Here is a specific example.

2 The *I*Teach* methodology in action - the *Op Art* project

2.1 Context and methodology of research

The experiment started in parallel in two informatics classes of 11th grade students from two mathematical high schools, at the beginning of the 2010/2011 school year. We started the classes with the clear idea for a project-based learning, which would not only cover the curriculum-related requirements (as defined by the state standards), but would also account for building the ICT-enhanced skills: *working in a team, working on a project, information, and presentation*.

The teacher diary contains data collection, on the basis of which the phases of the experiment are described below, together with some observations and conclusions. Unstructured interviews have been used for getting feedback from the students.

The main goals of the *specialized classes in informatics* in 11th grade in the Bulgarian high schools are specified as follows: *mastering skills and knowledge about the syntax and the semantics of a programming language*, *algorithms and basic data structures*. The time determined for this is 108 academic hours (40 minutes each). The students are expected to move from the procedure-oriented to the object-oriented style of programming and to acquire the concepts of classes and objects, data encapsulation, composition and inheritance.

Most of the informatics teachers in Bulgaria see the use of integrated development environment for programming and runtime framework as the first and only possible application of ICT in their practice. Right here, we see the first difference in our experiment in which ICTs are applied in two additional directions: *as a* *supplementary tool of education* and for building a *virtual environment of education and communication*.

2.2. Training methods and a learning environment

During the first lesson we presented the *methods of learning and teaching* we envisaged for the informatics classes to the students of both schools*,* viz. providing various learning resources prepared by the teacher (the first author)– presentations on different topics, notes on them, video materials; defining and collecting assignments for class and homework; inquiry-based learning on given topics; projects development in teams. Furthermore, we told them that all these activities would be performed *by active use of virtual environments for education and communication*.

The immediate reaction of the pupils was: *Are you trying to foist on us some American system*? This did not come as a surprise for us because till that moment they were trained only in a traditional lecture-drill style.

In order to organize effectively our work, as we promised to the pupils, it was necessary to build a teaching/learning virtual space (Figure. 2).

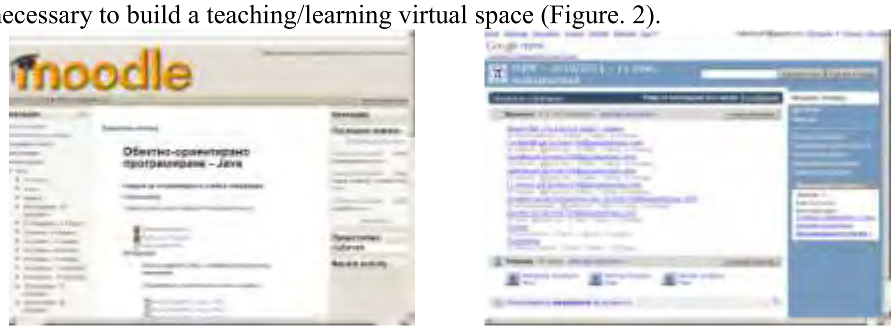


Figure 2. The virtual environments used in two parallel experiments

That problem was naturally solved in the first school, because the *Moodle* course management system had been installed and the students already were familiar with it. At the second school it was necessary to *build* virtual environment for education by means of Web 2.0 technologies: *Google groups* for communication, *YouTube* for sharing video materials, *Google shared documents* for collaborative work, *common internet space* for storing resources, *e-mail, Google forms* for collecting feedback, etc.

2.3. The transition

We were aware that the students would need time to adjust to the project-based ICT enhanced learning and teaching. That is why we started with two projects whose goal was to *cushion* the effect of the transition. Each of the projects introduced new requirements and responsibilities. While in the first project we expected the students to do mainly research on a small scale, finalizing it by comparative analysis, in the second one we asked them to continue the research and to develop their own software application, modelling a mathematical object – geometrical figures, rational numbers, etc. Both projects aimed at teaching the students to find information relevant to their goals, to select and use appropriate tools for communication, to acquire skills for a team work and to present their final product in front of the class. The students were further encouraged during the work on the second project to use new techniques (not studied till that moment), data structures or classes and methods to achieve the best possible result. To foster their critical thinking, we required that they use peer- and self-evaluation by means of assessment forms, developed and shared through Google forms (Figure. 3)

Figure 3. Assessment forms and their summary in Google

As a result of the work on the first two projects, we observed increased motivation and more responsible attitude on behalf of the students. In addition, all of them used more actively the resources and the communication tools provided in the virtual environments.

2.4. *It's a kind of magic*

The work on the first two projects ensured the necessary preparation and the base for the next project. It required not only students to be familiar with technological environments, but also to carry out a very extensive study.

During its first phase the students, in teams of 2-3 members, had the task to make an Internet-based study and to present the result on a topic among the following:

1. Vasarely (who, when, what);
2. Escher (who, when, what);
3. Op Art (art, artists, applied arts);
4. Optical illusion (types, techniques to be achieved, examples).

The formulation of the task surprised the pupils and comments followed:

- *ʃ ooh, madam, enough with this obsession to teach us how to present!*
- *Is this some kind of a physics lesson?!*

There were a lot of emotional comments during the study itself:

- *ʃ ooh, I know him! He is a Dutch! Very cool! Let me show you the waterfall! (enthusiastic exclamation about Escher by otherwise the weakest student in the class)*

One of the girls (good in informatics competitions) was very impressed by what she found on the Internet and by the fact that a lecturer in informatics is interested in arts as well: *And I, myself, began to be interested in it too...*

At the end of the presentations there was a total confusion about why they needed these things. The students still did not know that the study and the presentation of the results aimed at immersing them into the topic, at preparing them for the implementation of the project. It was only at that moment when the theme of the project was announced: *It's a kind of magic*!

In one of the schools spontaneously three students sang the popular Queens' song with the same name, while other students were quite reserved:

– *Now what is left is only to ask us to draw!*

They were not so far from the truth. In the third project students were asked to create Op Art images, programming them in Java language. More complex requirements were put in front of them. Again, it was necessary to work in teams of 2-3 pupils, expected to perform their own research and to have their own contribution, but this time in a more professional way. The requirements included:

- A research of existing models and projects in Java, realizing Op Art images
- Development of a prototype of software application with interaction
- Preliminary study and listing of additional opportunities for programming language which the individual teams will eventually need to realize their ideas
- Completing the overall project
- Presentation, including not only a demonstration, but an analysis of the team's work in terms of their own contribution, problems encountered and solved in the process of work, new matters studied in terms of programming concepts as well as knowledge from other areas (mathematics, culture and arts etc.), social and technical skills developed.

The teams had at their disposal two weeks for work on the project and one – to prepare for the *public presentation*. During the classes in informatics the teams presented the current status of the development of the project, as well as the problems, for which they needed the teacher's help. Most of the work on the project, development itself, was done by the teams out of the regular classes, actively communicating (mostly via e-mail) with each other. As for the teacher, she not only was tracking the project, but also participated through directing and assisting students with additional material, ideas and possible solutions to problems encountered by the students. In addition, each team tried to surprise the other teams with an attractive product.

The implementation started by choosing a model and development of the application prototype. That task seemed easier to the students at the beginning. Most of them selected a model from the Internet and decided to implement it. When getting focused on the requirements to upgrade the model by adding interactivity, many of the teams found that the chosen model, although easy to implement, is not an appropriate basis, because it did not provide enough attractive options to upgrade and add interactivity.

There were teams who approached the prototype after analysing the project requirements. At a very early stage they had specific questions to the teacher regarding the features of the language and the programming environment. In addition, these teams reviewed the tools proposed by the teacher so as to decide whether they could rationalize and implement them in the context of their own projects. Moreover, they experimented with simple examples before making sure that their idea is realizable.

Other teams showed their creativity by upgrading ready models and developing their own Op Art models based on the fundamental elements of the pattern. A girls' team was very happy to discover the *bottom-up* programming approach and to find how easy it is to write methods realizing small similar parts of the figure and then to use the same functions to assemble completely new, arts products of their own.

This project engaged the students emotionally much more strongly than the previous two projects. Even the students, who traditionally would not rush to program, were actively involved – a timid girl, who had not shown initiative until then, participated now in the programming process and gave ideas at the same level with her teammate – the best programmer in the class; other two students who did not consider themselves "mathematicians" drew and calculated coordinates for translation and rotation of objects (Figure. 4).

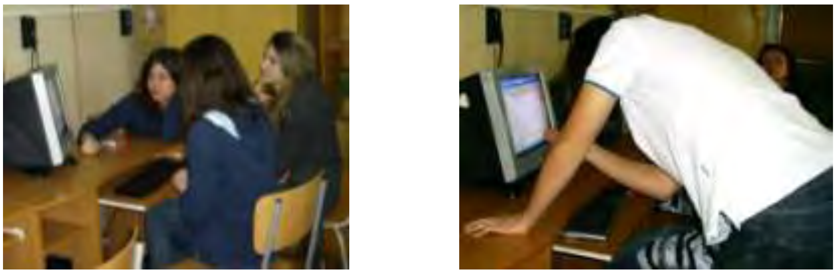


Figure 4. Working in teams on the project

There were also students who failed to organize themselves well. They underestimated the project and began to panic at its end. Some teams were broken and formed new ones. Other students with research spirit, devoted more time to doing research and, while working referred to the papers of such mathematicians as Polya and Poincare.

Overall the last two phases of the project ran very dynamically, even passionately. Many questions and semi-successful experiments flew continuously, both in class and by email. At the end the work flowed more quietly, the first results were already visible, the students felt far more confident in their abilities. When the teacher made an attempt to offer her vision to help a team, its captain exclaimed: *Please, do not put forward your solution! Please, help me to realize my idea!* The team felt mature enough to implement its own idea and expected to be supported by the teacher, *not to be lead by the hand*.

The presentations of the projects were also intriguing. It turned out that only one regular class is insufficient for all teams to present and defend their results. All teams had invested a lot of effort and wanted to share everything new they had found while working on the project – interesting facts and images, features of the language for objects settings, new tools of interaction with users, methods of implementation of animation, math functions and their application to the project. But most of all students wanted to show the final result being *a fruit of their own imagination*!

On the other hand, being in the role of evaluators the students kept asking the presenters about problems they themselves had come across, or effects they were curious about. At the same time, they judged their peers very critically striving to meet all the criteria for evaluation. The evaluation cards were filled in for each project. By them the evaluators justified and argued any grade with concrete examples from the observed presentation.

Some teams asked for time extension in order to improve their project, proposing to take a lower score due to the extended time for work.

Other teams showed unexpectedly attractive results, achieved by independent work at home. Some of them said that they had purposely kept their work in secret so as to surprise the audience, while others shared that their attitude was inherited from previous years: *it was not cool to work in informatics classes*!

In all cases, discussions in both schools continued even after the end of classes. The presentations passed with many emotions, involvement of students, exchange of ideas and projects. Some teams decided to continue the development of their projects, even if they had already received excellent marks and evaluation.

In both schools the project results (Figure. 5) were published at the virtual learning environment thus becoming accessible to all. The principal of one of the schools offered to publish the projects on the school's web site to make the students' real achievements visible to a wider public.

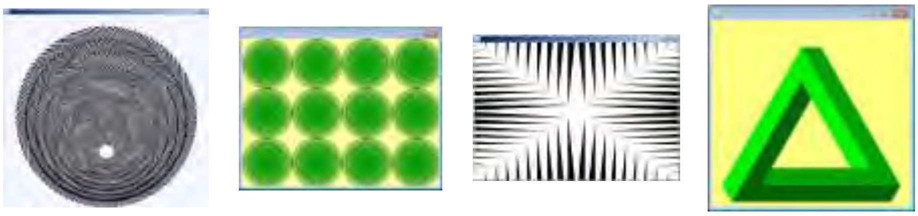


Figure 5. Visible results from the students' work on the project

2.5. Reflection

The project-based learning still puts lots of challenges in front of the teacher.

Before the start of the project she carefully selected the core content to present to the students, so that it would be sufficient as a starting point for the projects. In this case the curriculum requires to introduce the object-oriented programming concepts – *the class, the objects, constructors, access modifiers, method call, data hiding concept, controlled setting and getting private variables' values*. The teacher chose to present these concepts on the base of the JPanel class as a drawing canvas and to demonstrate object's properties and ways to manipulate them through very simple geometrical forms (squares, arcs and ovals). She explored with the students the colours' and strikes' transformations introducing to them the idea of *inheritance*.

During the implementation of the *It's a kind of magic* project, the teacher had to follow in parallel the threads of the mental road-maps of all the teams; to experience their successes and problems; to seek together with them solutions of questions, sometimes new even for her. She had to invest much more time for preparing the next lesson because she wouldn't know in advance what her students would need.

This extra involvement was completely justified by the results and the added value of this kind of education. Some students discovered new (for them) algorithms, mathematical patterns and relationships, programming approaches and techniques. Others were familiarized with the concepts of *containers, basic* and *inherited classes*, *manipulating threads*, etc. Most importantly – all of them were happy to share their findings, to learn from the experience of others, to get a realistic picture about what they had learned while working on the project.

On the other hand, the fact that there were some students who were not so active, or tried to present a development which was not their own, means that there is a need of more strict control on the achievement of an intermediate result. Furthermore, the time for presentation of the final result should be extended so as to provide the opportunity for more questions about specific points in the source code as well as to allow students to edit the program in place.

The students' reflections were very encouraging for us since they showed their enthusiasm to continue to work in the same way:

- *It was the most difficult program I've ever written! It works! It's mine!*
- *Madam, I had a dream that I'm writing a software dictionary for my sister. I've never thought I would be able to do this, but now… I believe that with a little help…*
- *What is our next project about? Please, promise it will be at least as interesting as this one!*

It was interesting for us to see books on computer models of three dimensional optical illusions brought by the students in the school a month after the project was over as well as the exchange of pictures and ideas how to program them in Facebook!

Sharing our experience and the process of work during the experiment, we would like to point out, that all elements (methods as well as ICTs, although different in both cases) play important role for the final results. In the same time, we are fully aware that what could happen in each specific class setting to a great extend depends on the will, the enthusiasm, the skills and the courage of the teacher.

Since the design and the class management of such a type of education are rather difficult tasks, the culture of making them a natural school atmosphere should be promoted at the university level – during the pre-service and in-service teacher education alike. The results will justify the efforts.

3 Conclusions

The work on the project Op Art (as a *kind of magic*) was not just an attempt for Art programming, but rather entering the *art* of learning and teaching *Object-oriented programming* by means of the present *Information and communication technologies*. It was an authentic challenge for the students and for their teacher with real, strong and visible results.

The education of the 21st century highlights skills and competences for formulating open problems, for seeking their solution: original, creative, innovative, required by any employer, could be built, only if the education is not limited to the second, in the best case, to the third level of the taxonomy of Bloom [7], but puts real challenges in front of students and provides them with the relevant tools. For a design of such education, it is necessary for the teachers to set goals going beyond the narrow subject area. If we take the *gauntlet thrown down* by the art project we could better visualise our educational message (Figure 6).

Figure 6. The *natural* Op Art transport of the education to the future

In other words, the education could advance to the 21st century by riding the vehicle of the interdisciplinary project-based learning with ICT-enhanced skills.

References

1. Sendova, E. (2011) Assisting the art of discovery at school age – a Bulgarian experience, in *Talent Development Around the World,* http://worldtalent.fundeta.org/libros/ebooks.php
2. Dicheva, D., Nikolov R., Sendova, E. (1997) School informatics in Logo style: a textbook facing the new challenges of the Bulgarian informatics curriculum, in M. Turcsanyi-Szabo (Ed.) *Learning and Exploring with Logo* , Proceedings of the Sixth European Logo Conference Eurologo'97, Budapest, Hungary, 20-23 August, 1997, pp. 234-239
3. Todorova M., Hristov H., Stefanova E., Nikolova N. (2010) How to build up contemporary software professionals - Project-Based Learning in Data Structure and Programming, In proceeding of S3T conference, Varna, Bulgaria, 11-12 September 2010, pp.47-54
4. Todorova M., Hristov H., Stefanova E., Nikolova N., Kovatcheva E. (2010), Innovative Experience in Undergraduate Education of Software Professionals: Project-based learning in Data Structure and Programming, In proceeding of International Conference of Education, Research and Innovation (ICERI'2010), Madrid, Spain, 15-17 November 2010, pp 005141 – 005150, ISBN: 978-84-614-2439-9
5. Stefanova E., Sendova E., v. Diepen N., Forcheri P., Dodero G., Miranowicz M., Brut M., et al Innovative Teacher - Methodological Handbook on ICT-enhanced skills, Faleza-Office 2000, Sofia, 2007
6. Innovative teacher (I*Teach) project web site, http://i-teach.fmi.uni-sofia.bg
7. Bloom, B., Englehart, M. Furst, E., Hill, W., & Krathwohl, D. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. New York, Toronto: Longmans, Green

The Digital Book in ICT – the New Tool into Learning and Teaching Process in the Primary School

Rumyana Papancheva[1] and Krasimira Dimitrova[1],

[1] University "Prof. Dr Asen Zlatarov", Prof. Yakimov Bul. 1, Burgas 8000, Bulgaria,
rumi@parallel.bas.bg, itlearning@hotmail.com

Abstract. The paper describes basic goal, principles and approaches in creating student's digital textbook for learning ICT in the Primary School. Some examples of developed E-book "ITI" for 3rd and 4th grade in Bulgarian school are presented.

Keywords: ICT, teaching, interdisciplinary, project-based learning.

1 Introduction

The students in Bulgaria from grades one to four study ICT as facultative classes. During last years, more and more schools started to offer such classes. The main obstacle that stops the acceleration of this process is the lack of qualified staff. Primary school teachers have to teach ICT together with all other subjects. They don't feel confident in their knowledge and skills concerning computers.

Another serious problem in teaching and learning ICT at Primary school is the lack of facilities needed at school. Usually there are one to three computer labs with 10 to 15 computers inside. All labs are almost permanently occupied by 5th to 12th grade students that study ICT as compulsory subject. Primary school teachers have limited access to these labs and usually only for ICT classes. To integrate technologies during regular lessons in Math or Science, for example, the teacher relies on his/her personal laptop and some projector at the classroom and on some mobile laptops if they are available at the school. In this way the independent students' work at home, where most of the children have computers, is of great importance.

Good solution for these two problems is the digital textbook in ICT. A teacher with average level of digital skills could use the book to show and explain certain algorithms. A student could work individually at home or at school with minimal adult's assistance. The electronic textbook is a new learning tool, offering wide variety of options of interaction with students and to overcome many restrictions of the regular textbook on paper.

2 Main Principles and Approaches in Developing Digital Book in ICT

The electronic textbook is new challenge to the pedagogical community. It offers advanced tools that give to it an indisputable advantage over well known teaching resources like textbooks, workbooks, posters, worksheets and so on. At the same time, its development should follow the contemporary pedagogical theories.

2.1 Didactic Principles of Creating Digital Book in ICT

Some of the key principles that should underpin the development of an electronic textbook are listed here.

Individuality. The nature of the material studied and its peculiarities lead that individualization and differentiation of the learning process in ICT have to be realized. The implementation of this principle is determined by the specific of the learning tasks where the tempo is determined by the level of each student. Furthermore, by means of electronic communication the teacher has the opportunity to work at different level with each of his/her students. The digital book should be designed for personal use with personally formulated instructions and individual sets of exercises for every student.

Systematic. The systematic principle is determined by C. Bespalko as a fundamental requirement in establishing some educational model [2]. Its essence consists in the necessity that new knowledge and concepts formed from the students have to take their logical place into the system of knowledge already created. The digital form of presentation of the content allow that the material to be systematized in different levels, appropriate for beginners and students at intermediate or advanced digital skills. The establishment of these levels of complexity is based on prior estimation of the experience of groups of students.

Visibility. The visibility is crucial for this age interval (6 – 11 years old students), which is related to the nature of the mental processes. The visibility is defined as "enrichment and expansion of the sensory experience of students, in clarifying their ideas and sensory development of observation"[1]. The computer could be perfect tool for identifying the properties and the dependencies between objects into the space. It is the most powerful visual tool to study objects and events – static and dynamic. The opportunity pupils to use computers in their work, both in ICT and in other subjects increase their motivation and the quality of training. With proper use of the capabilities of the computers, the teacher could visualize highly complex processes, phenomena and objects.

At this age, it is better for the students to watch how to do something and to repeat, than just to listen given explanations. The demonstrations included into one digital book lead to higher level of understanding. The digital book offers rich opportunities for visualizing series of actions in applying certain algorithm – by pictures, by video, be schemes.

Problem-based learning. Most valuable is the knowledge acquired through personal learning experiences. Working on given topic and doing self research should be key points in many of the tasks formulated. Variety of research themes could be provided with an initial set of resources – implemented as galleries and hyperlinks into the digital book.

Consciousness and activity. The principles relate primarily to the motivation for active participation into the learning process. Higher motivation could be achieved by involving children in activities such as: research, monitoring, classifying, problem solving, practical activities, work on thematic projects. In general, students like to use variety of digital devices, in particular – the computer. Higher motivation for work at school is noticed when contemporary technologies are used.

Consistency with the capabilities of the children. It is or great importance to achieve agreement between the curriculum and the students' abilities. According Vygotsky, training should be directed to the nearest area of the development. The content in one digital book should follow the rules formulated by Comenius: from easy to difficult; From known to unknown; From simple to complex; From near to distant.

All principles proposed here have to be used in unity.

2.2 Didactic Approaches in Creating Digital Book in ICT

Leading ideas in the development of an electronic textbook in ICT should correspond to some key approaches, listed here:

Algorithmic approach. There are many formulations of the algorithmic approach into the learning process [3]. The formation of algorithmic culture and development of algorithmic thinking are the main teaching objectives of school education in Informatics. Some methodical studies show the effectiveness of forming of algorithmic thinking skills in early age [4].

The digital textbook offers different ways in introducing algorithms. Appropriate system of hyperlinks helps students in actualization and knowledge summarization.

Activity and Personality. The approach of the active position of the student and the personal-oriented approach are two of the most important. Nowadays there is a new level of interaction between them – personality-oriented approach dominates in order the individuality of the student to be form. This approach determines the direction of the progress of the education system – the child is in the center of learning process, with its needs and desires. From subject of training, he/she becomes an active creator of his/her knowledge (subject-subjective system).

Depending on personal capabilities, each child works with an individual speed. In this way a differentiation of the training process is obtained. Working with personal digital book, the students who have higher interests in informational technologies could go ahead and do more complex work. Students with fewer opportunities would not be embarrassed by their inability. Even working slowly they will acquire the minimum volume of knowledge.

Real individualization of learning process could be achieved in wide aspects – from working with talented students to working with students with some disabilities.

The digital book allows child to be placed in the center of its educational process by enabling him/her to make choices – which software to use, which exercise to complete, tools to implement and so on.

Integrative approach. The integrative approach is used to support learning and to overcome some failings of the subject-based training, which is defined as reproductive. According to educational standards the integrative approach is implemented by internal and interdisciplinary connections.

This approach should be used actively in creating content. In electronic form students could solve problems from different areas of knowledge – from text language exercised, to solving math word problems and so on.

2 E-Book "ITI"

Taking into account all main principals and approaches, described above, the authors with their team, have developed a complete system for teaching and learning ICT into Bulgarian Primary school [5, 6, 7, 8]. Within this system a digital book for 3rd and 4th grade students, named "E-book ITI" was created.

ITI is part of the educational package – paper-textbook, teachers guide for the teacher and educational software distributed on CD. The main goal was to create an electronic book to help the process of teaching ICT and/or integrating ICT into the learning process.

The E-book "ITI" was designed in two parts – E-Practice and E-Guide.

E-practice is a set of electronic exercises. During installation process a personal folder for the student is created with all learning resources in it. To work with the exercises, the student has to follow the instructions included into E-guide module.

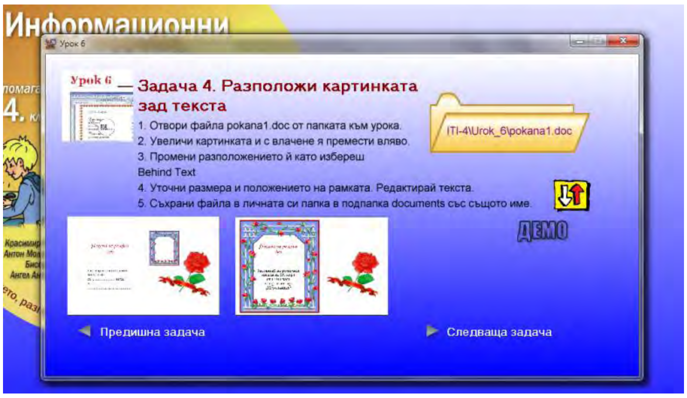


Fig. 1. Page from ITI-4 E-book. The lesson deals with layout of pictures towards the text.

Besides clear and precise instructions for working with the exercises, the E-guide module includes many video-demonstrations, algorithms and samples given. Figure 1 shows a page from the 4th grade book in ICT. There are some basic elements on the interface realized:

- Instructions for the student. The list contains short clear tasks that student has to do in the order given;
- Information about the files, containing the particular digital exercises;

- Demo – created by video capturing of the screen that shows the key actions to do;
- Visual presentation of the start and end point of the task;
- Navigation to next and previous tasks.

In some lessons there are additional elements like hyperlinks, dictionary, i.e.

The main menu of the book is designed by pictures corresponding with the lesson. On Figure 2, we could see the first screen as a pointer to different units.

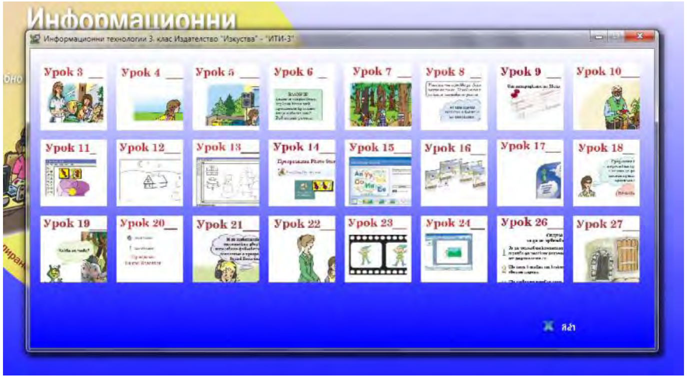


Fig. 2. Main "Table of content" page of ITI-3 E-book. This picture-based menu leads the student to the corresponding theme.

All basic interface elements used into ITI E-book are represented with constant graphical signs. In this way students feel confident within the environment. After some classes they could orient independently between demonstrations, dictionary (English – Bulgarian) sections, information about the files, algorithms, instructions, i.e. On Figure 3 we could see some of examples.

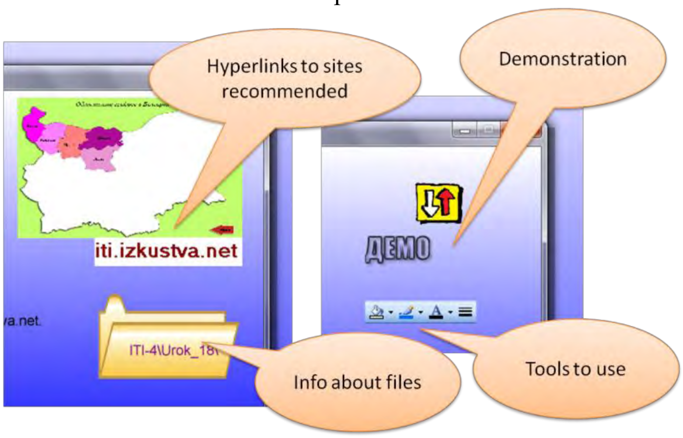


Fig. 3. Basic elements in ITI-3 E-book, providing interactive interface with the students

The content of the course in ICT for the third grade students is structured in some main parts, shortly listed here:

- Computer system. File system – work with files and folders. Save and Open commands. Start menu.
- Working with text. Text selection. Formatting – style and font. Copy and paste commands. Printing. MS Office Word based work.
- Working with graphics. New instruments like color picker, color edition, selection, copy and paste command. Paint.
- Combination of text with graphics.
- Picture processing. Working with Photo Story.
- Importing pictures and sound into presentation. MS Office PowerPoint based work.
- Work with animation. Creating animation by Gif Animator.
- Internet based work.

Let's consider one of the modules – text selection. The digital book includes:

- PowerPoint presentation, illustrating the idea of the new activity – the text selection. One needs to select the text to point out the target for the next operation. Some parallels with selections already familiar to children are made. On Figure 4 we could see route selection within the forest, selection of trees to be cut down, animal selection.

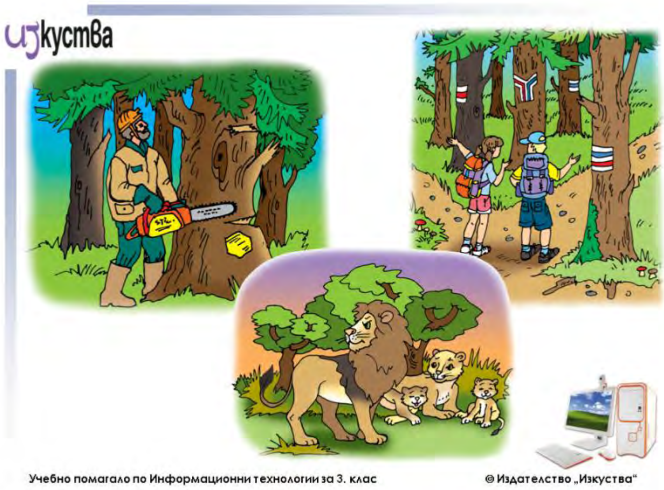


Fig. 4. The concept of Selection.

- E-Guide module with short, clear instructions for students, together with demonstrations and information for algorithms and tasks proposed.
- E-practice module, containing pre-created digital exercises. For example, to acquire skills for row (line) selection students have to color rows into Word document to create different international flags (Figure 5a) or to format well known poem coloring rows in different colors (Figure 5b).

E-practice module includes variety of exercises. One of the main goals is the realization of interdisciplinary connections. The digital exercises, from content point of view, play important role for systemizing and summarizing students' knowledge from different school subjects. Students have to solve math problems working with text formatting, or recall some facts from science and history.

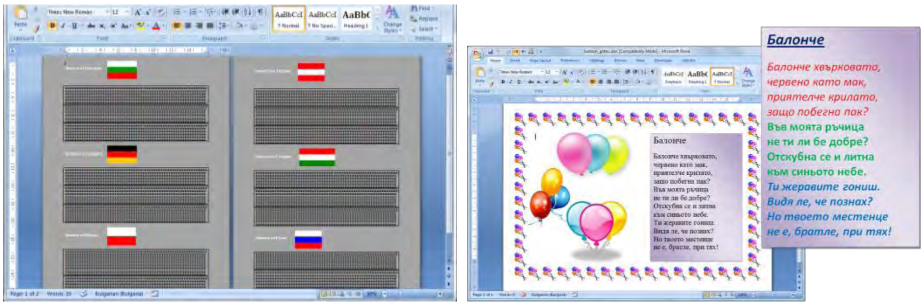


Fig. 5. Exercise examples from E-practice module.

The experimental work and the training already realized into the practice, give us the ground to conclude that the use of electronic textbooks is effective and produces positive results. Its introduction into the teaching and learning of ICT is a natural extension of the tendencies for the digitalization of schools worldwide.

References

1. Andreev, M. The Process of Learning. Didactics, Sofia University Press, Sofia, 1996.
2. Bespalko, V., The Basis of the Theory of the Pedagogical Systems, Narodna Prosveta, Sofia, 1982.
3. Dimitrova, K, R. Papancheva, N. Kaloyanova, The Algorithmic Approach in Learning to work with Internet, Sofia University Press, 2004.
4. KOLCZYK, E., Algorithm – Fundamental Concept in Preparing Informatics Teachers. Springer Berlin / Heidelberg, Volume 5090, pp. 265-271, 2008.
5. Manev, K, R. Papancheva, K. Dimitrova, Informational Technologies for 1. grade, Izkustva, Sofia, 2008.
6. Manev, K, R. Papancheva, K. Dimitrova, Informational Technologies for 2. grade, Izkustva, Sofia, 2008.
7. Manev, K, R. Papancheva, K. Dimitrova, Informational Technologies for 3. grade, Izkustva, Sofia, 2009.
8. Manev, K, R. Papancheva, K. Dimitrova, Informational Technologies for 4. grade, Izkustva, Sofia, 2009.

Art, Literature, and Turtles

Artemis Papert[1], Brian Silverman2
[1] Independent artist, Montreal, Quebec, Canada
artemis@turtleart.org
[2] Playful Invention Company. Montreal, Quebec, Canada

Abstract. In this paper we discuss the need to move beyond computer literacy to computer fluency. We describe TurtleArt, a microworld for creating art through programming. We feel that TurtleArt encourages fluency by providing an easy to learn programming environment coupled with a vocabulary well adapted to deep and varied artistic explorations.

Keywords: TurtleArt, LOGO, turtle geometry, constructionism, programming, art, microworld, fluency.

Knowledge is a polite word for dead but not buried imagination.
e.e. cummings

1 Literacy and Fluency

When computers first started appearing in schools in the 1980's a lot of attention was paid to computer literacy. Less attention was paid to fluency. Being literate in a language means having an understanding of the syntax and semantics of that language. It also means having some comfort with its vocabulary. Fluency is the ability to use the language for the creation and appreciation of substantive works.

We would have hoped that by twenty years later fluency would have replaced literacy as a goal and that we would be seeing more substantive works using technological materials. True, various programmes have provided an opportunity for children to understand and use, for example, spreadsheets and word processing, even programming. However the works tend to be a new media form of the kind of presentations that children have always done in school or simple games and simulations.

In the hope of encouraging a more fluent use of technology we built a system we call TurtleArt. Our goal is to provide an opportunity to engage in deep exploration and produce substantive works. TurtleArt is about art. It is a system that is relatively unsophisticated on the technological front and that is quite narrow in terms of the kind of artefacts that can be produced. TurtleArt is focused on creating static images. It is not a general programming environment or a system for exploring math, language, science, etc.

2 TurtleArt

TurtleArt is a microworld for creating art. It is inspired by Turtle Geometry as descried by Seymour Papert [1]. It is closer in spirit to the early versions of the LOGO programming language than to the more modern constructionist environments.
TurtleArt's language is blocks based rather than text based. Programmes are constructed by snapping together puzzle shaped blocks (Fig. 1).

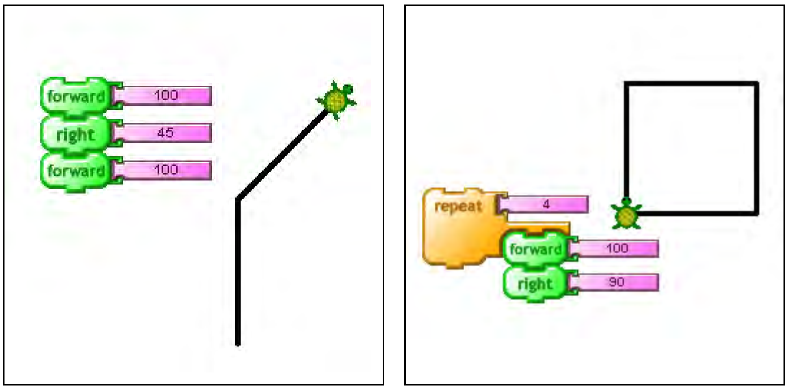


Fig. 1. The mechanics of TurtleArt are very simple.

The vocabulary is very small as compared to other versions of LOGO (Fig. 2). We have left out words, lists, and data manipulation. The colour model, on the other hand, is, unsurprisingly, sophisticated –it loosely resembles the P. O. Runge's farbkugel [2].

In creating our images we try to keep the code very concise. We do this because we believe that the code being elegant is an art form in itself. We also think that it is a way to allow most of the images to be used as introductory examples [3].

In our images we have been exploring the interplay between algorithm, randomness and direct choice (Fig. 3). Producing images programmatically is ideal for this kind of work. Images usually evolve from an initial idea through stages of refinement and often variants are traded between us as part of this evolution [4].

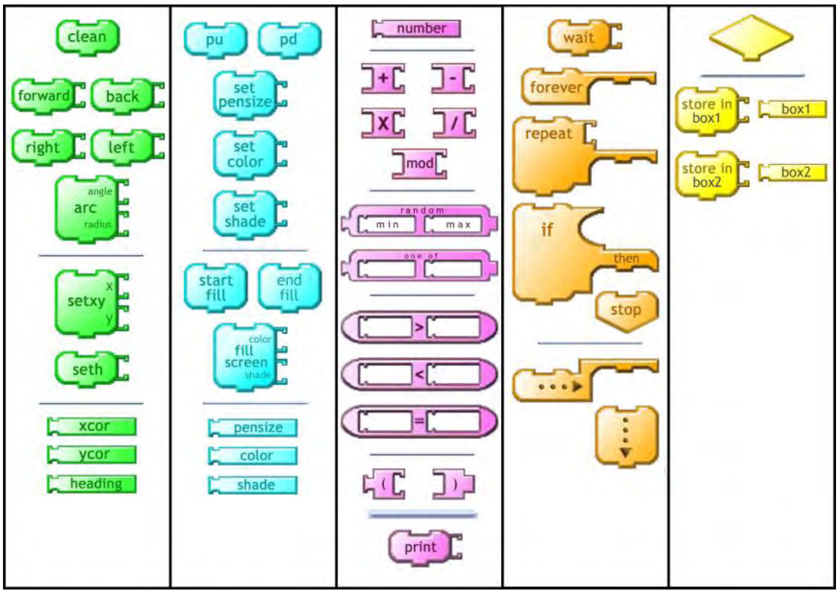


Fig. 2 The entire vocabulary of TurtleArt.

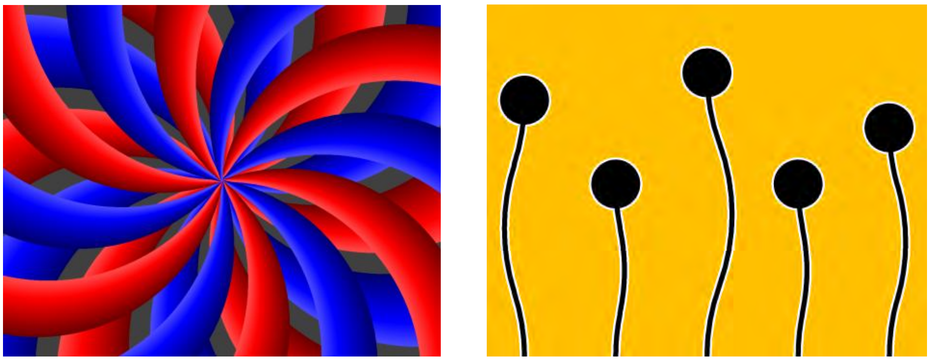


Fig. 3. The art is in the art.

3 A Little Bit of Programming Goes a Long Way

One of the design principles suggested by Resnick and Silverman is that "a little bit of programming goes a long way" [5]. The design of TurtleArt takes this principle seriously. More seriously, we think, than most constructionist environments. There is little reason in TurtleArt to do more than a little bit of programming. We see this as a strength, not a limitation. We recognise as a disadvantage that this makes the "walls" narrow [5]. It may also make the “ceiling” lower. However, while it is true that in TurtleArt you can hit your head against a programming ceiling, we find that, at least for ourselves, the artistic ceiling is very high. Even after a couple of years of intensive work our artistic exploration continues to broaden and to deepen.

In his book Mindstorms [6] Seymour Papert described Turtle Geometry as a microworld and predicted that in the future there would be dozens of other microworlds. We see TurtleArt as a microwold in the sense Papert intended. It is not the only one that has been created since Mindstorms. However much of the constructionist software that exists can be described as “microworlds for making microworlds”, i.e. for learning a programming language. This has led to a situation where, to paraphrase Marvin Minsky, we have “languages without much literature”.

We believe that TurtleArt has a literature. The images are the literature [4]. The language only exists in order to support that literature. We have kept the language “as simple as possible, maybe even simpler” [5]. With this we hope that fluency is a short distance from literacy.

References

1. Papert, S.: Teaching children thinking. Artificial Intelligence Memo No 247 / Logo Memo Artificial Intelligence Lab, MIT, Cambridge (1971)
2. Schopenhauer, A., Runge, P.O.: On Vision and Colors and Color Sphere. Princeton Architectural Press (2010)
3. Getting Started with TurtleArt. http://www.turtleart.org/programming
4. TurtleArt Gallery. http://www.turtleart.org/gallery
5. Resnick, M., Silverman, B.: Some Reflections on Designing Construction Kits for Kids. Proceedings of Interaction Design and Children conference, Boulder, CO., USA (2005)
6. Papert, S.: Mindstorms. Basic Books (1982)

Ten Years of Creative Robotics Contests

Pavel Petrovič

Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia, ppetrovic@acm.org

Abstract. Educational Robotics is a new field that did not win the position it deserves in the educational systems yet. We look at its role from a natural view, provide some didactic arguments and concentrate at robotics contests. We focus on one type of robotics contest that is not very common – creative robotics contest. We have been organizing it in Slovakia for ten years and wish to share our views and opinions with a wider community.
Keywords: educational robotics, contests, creativity, virtual games.

1 Introduction

Learning to be creative, self-motivated, goal-minded, and capable of effective and efficient gathering, organization and application of information should be among the highest-priority goals of the contemporary schools. Regardless of how natural these capabilities might be for a human being, they can easily be left undeveloped when youngsters are growing up in a non-stimulating environment, an environment lacking challenges, interactions, opportunities, and stimuli. A highly hierarchical, structural and very advanced organization of the society we live in can easily become demotivating, boring, or even frustrating for young people. Creating smaller or larger isolated worlds on its own with rules and facts that are comprehendible and contain goals and challenges to be achieved by systematic efforts can be among the best possible scenarios to stimulate creativity, self-motivation, goal-mindedness, and information processing. Let us call these "worlds" virtual games. Participants of virtual games can be individuals or cooperating groups, teams. Finding ways of effective inclusion of virtual games into educational process is a key to its success.

In the context of information processing skills, and other contexts as well, let us underline that it must be recognized how centrally placed virtual games are to the informatics education even when they do not relate to informatics, computers, or algorithms directly. They typically require planning, role assignment, scheduling, task distribution, progress reporting, logging, evaluation and presentation of information and results. Many of these are important elements of thinking about and dealing with information. Having them established in one's mind before attempting to understand and acquire specific concepts and skills corresponds to building a bridge instead of trying to jump over a wide river. The second approach often results in sinking and drowning. And we have witnessed this fate in various attempts. For instance, when trying to integrate computer programming into standard curriculum. In those cases, the mental substrate – the foundations to be built on were simply not there. Every attempt resulted in a collapsed structure, falling down to the river, passing away and leaving no traces. How harmful this has been to our field!

Starting with virtual games that do not directly relate to “computer science” might also be a very suitable way of connecting the children’s interest and buying them into the course framework. Once they feel comfortable, further informatics concepts may slowly be integrated into the scenarios they experience. In this way, we may achieve popularity and passion of informatics courses among the children from the very beginning and further on. In fact, we believe this is the only way that works and it is also what we are trying to do. Provide entertaining activities that ignite the interest and generate self-motivation. We live in a society, where education-by-force does not work any more. That era has passed! We live in a society that should be labeled education-by-motivation. And even though the idea has always been there, it has only recently become possible and it is now flowing into the main stream.

Setting up virtual games demands immense amount of work to be performed when realized by every teacher separately. Sharing and common organization seem to be an implicit requirement. Shared scenarios imply or at least provide for a social aspect among the participating groups or classes. These common activities may involve meetings, shared presentations, and awards. They can take form of independent or interconnected challenges – a virtual game for one class or a team, or *a competition* of multiple individuals, teams or classes. We believe, competitions are an excellent example of virtual games. They are a very potent bridge building activity. Bridges between different parts of knowledge, reasons, arguments, beliefs, facts, constraints, positions, activities, dependencies, sequences, etc. are inevitable for learning. We think that the real contribution of competitions to the educational progress of an individual is more substantial than it is generally believed. We think that they deserve more attention. However, we have to be careful here, and not to organize a competition just for the sake of it. There are various kinds of competitions, and we will elaborate on this idea further below.

In the remaining sections, we will discuss the role of the competitions, the role of robotics in informatics education, and provide a short overview of robotics contests. We describe our creative robotics contests, which are the main focus of the article. In the later sections, we describe how we support the teachers in the educational robotics activities, evaluate the outcomes of our efforts, and finally conclude with a few remarks on the future.

2 Competitions

In this and our previous work [1], we have learned about the key characteristics of the competitions:

Competitions have a fixed deadline that cannot be moved in any circumstances. This provides a learning experience of a hard constraint. In order to succeed, pupils must learn to work well with time, set their priorities, and weigh their decisions based on the finite resource they are dealing with. The deadline also works as a strong motivating factor and helps pupils to learn to perform time-efficient work, and to deal with stress.

Successful competitions are prepared by experienced individuals who are able to asses the difficulty of the task specification and select tasks that are neither too hard nor too simple. Pupils that participate in the competition can usually support their thinking by the assumption that the task is solvable. The task is defined in very clear and graspable language. The task specification defines a small world of its own as we described above. The competition organizers also make sure their task is novel, and the solution cannot be tricked by searching for it on the Internet.

Competitions are organized around standardized platforms. Participants know in advance what they can expect, what tools and skills will be required to take part in the contest. Large user communities form around standardized platforms, online forums, manuals, tutorials, and example projects help the participants to get started, and keep a steady progress without running into a wall and being blocked on an issue that could not be resolved for weeks.

Competitions allow the schools and teams to attract the media and sponsors' attention. They are also a good opportunity to persuade the school administration to provide the best practice space, time, and human resource conditions to the team in order to maximize the chances the participants win an award, helping the school image. It is also a useful opportunity for the teacher to let other students in the school know about the activities and inspire them to join in.

A very important aspect is the one of relating the children's performance against their peers. Young people are permanently and intensively trying to determine and learn about their role in the society. Talented students need to learn about their endowments and build upon them. At the same time, this aspect also poses a high risk of disappointment, and it can result in loss of interest, unpleasant attitudes, loss of performance and other complications. Team leaders must be prepared to deal with the situation in advance, and balance the motivation of the team for the contest with the interest for the subject and activity itself, find different ways to reward the team members for their performance unrelated to the position in the contest.

Competitions are social events. Schools are social institutions, but nothing is more mind-numbing than a daily stereotype in a master-slave configuration of a classroom education. Pupils meet and interact with wider population and its ideas. They exchange experience, learn from each other, and learn to act on their own in a novel situation when asked for a certain type of performance, different from a daily routine.

Competitions emphasize friendly and fair-play atmosphere, they carry the good spirit and create an island of time when everything and everybody is subordinated to a good outcome, result, and feeling of every single individual. Experiencing this atmosphere helps vitally later in the life when a participant faces difficult team or individual challenges.

3 Educational Robotics

The idea of using robots in the educational process is not new, but it may still appear surprising, disturbing or useless at the first sight to many of us. Let us explain why the contrary is true and why it is so natural and inevitable process.

One of the distinguishing features of a human being is the capability to use instruments and tools. Humans were getting involved in a social transfer of knowledge from the earliest time. Young people can handle a much steeper learning curve and thus an establishment of a school as a form of a social gathering of youngsters with the purpose to learn from the more experienced is among the most natural acts of our species. And from the earliest times, tools were used in this process – chalk, stone or a blackboard, abacus, rulers, pencils, books. These all are tools. In every age period, the learning tasks would always be supported by the technology that is available, accessible, and can automate or in any other way support the learning and teaching process: mechanical drawing boards, erasers, calculators, computers with word processors, spreadsheets, mathematical programs, and educational software. In the very same manner, it is just natural to use robots, for instance, to deliver items from place to place in the school; clean the floor, or other surfaces; provide guidance to school visitors; monitor and patrol the environment; help in the kitchen and dining room; perform simple office tasks – copying, stapling, stamping, etc.; operate, control, or measure experiments on chemistry, physics, or biology; throw or collect balls in various sport activities for the purpose of training, or make measurements, or otherwise assist in the physical education activities; explain or demonstrate a particular material to the students by letting them or the teacher operate it or even program it; be part of a student project assignment, allowing the students to extend their range of experimental scenarios.

All of the above are reasonable uses of robots in the school, even though some of them are less frequent yet. There is no separation between the "traditional" teachers who should not be involved with the robots and "weird modern folks" who should be dealing with them. Robots are here and they are entering the school in the same way as the stylus did thousands of years ago in the ancient Egypt or Greece. Robots are greeting every teacher or employee of the school alike, they are for them all.

The last four items in our list describe roles when robots actively participate in the educational process. A common term *Educational Robotics* is sometimes used to identify them. To this end, various systems have been put at the market or constructed in the research laboratories. Among others,

- small wheeled mobile robots, programmable to certain degree, can be used to teach basic elements of control theory, mechanics, electronics, and some programming, these robots typically have a well-defined and limited range of possible applications;
- robot arms and manipulators, can be used to setup various kinds of experiments that involve manipulation, they are typically single-purpose devices;
- dedicated automated or robotic systems that are assembled for a particular purpose, can demonstrate or assist a single type or a limited set of experiments;

- users can chose to build their own robot systems from raw parts and materials, if only they are equipped with the required skills, background, and resources;
- versatile construction sets with programmable control units that can be used for various purposes, in most cases to learn about control and programming, and to provide a platform for exploratory student projects.

All of the listed approaches work well for their purpose, although the most popular certainly is the last item of the list. Among the reasons are: "many in one" solution, accessible price, large community support, availability. Most efforts in the field of Educational Robotics focus on working with the construction sets. The success of their utilization lies in their open-ended nature, supporting the activities based on Constructionists' approaches. Together with the self-assembled robots, robotic sets are the most often used and most suitable platform for the robotics competitions.

4 Overview of Robotics Contests

One could try to successfully argue that events as robotics competitions are in a very corner of any decent educational program or process, unless considering the specialized students in tertiary education program who study robotics, control engineering, or artificial intelligence. Admittedly, there are no particular reasons why robotics would be taught at primary or secondary level. In fact, participants of most robotics competitions are hobbyist individuals, or teams from clubs or free-time centers. Some of the schools, where robots are used to teach programming, may be interested in setting up a school team; however, that still requires a heroic sacrifice of the time and energy of the teacher. This kind of positive deviation indeed is a rarity. Recalling our arguments from sections 1 and 2, we honestly believe, this is a pity.

Organization of robotics contests world-wide depends on individuals who are steeling time from their other duties. This process is not dissimilar to that of the development of local cultures. It is a bottom-up competition for the survival of the fittest. We see both positive and negative consequences:

1) (+) Approaches that are more interesting, having better impact and stronger potential, survive the selection process and become more established;
2) (–) Different players have different starting positions. Distributors of specific construction sets and their network has a much stronger position than a group of academics regardless the actual value of the activities;
3) (–) Efforts, materials and resources get wasted, when teams decide to discontinue challenges they joined earlier. Some interesting ideas can never get implemented.
4) (–) The organization is more exhausting, taking its toll on the organizers, hampering the progress. It usually depends on private sponsorship, which requires enormous efforts in the fund-raising activities, leaving less time to provide a better service and content.

That, however, is a real-life, and the situation in many areas of the society. As [2] nicely put it: "*Everybody in the orchestra tries to convince the others to use their instruments and to play in a local club that is not able to accommodate the* *orchestra*". Somewhat different situation can be observed in some Asian countries, which, undoubtedly, took over the technological lead in various fields already.

To provide an overview of the robotics contests, we must start with the initiatives of the US FIRST association, which probably has the largest impact. US FIRST cooperate with many associations around the World to prepare global educational challenges for children at different age levels.

Namely, *Junior FLL*, for ages 6-9, where children build models with moving, possibly motorized parts containing simple machines. They present a poster on their model. Despite the popularity of this contest, we are not convinced that children are mature enough for it. Playing is vital at this age, not so much the contests, and the stress to produce something. Having a sensitive didactic approach can be very helpful.

FLL (*FIRST LEGO League*) for ages 9-14 in the US, and 10-16 in Europe [12] is the most popular one, students having 3 months in their clubs to prepare 1) a robot to complete a course consisting of multiple independent challenges of varying difficulty, and 2) a little research project on a specified topic. The children's solutions are often so good that college students who are given the same task are unable to equate their performance. We find this competition to be very useful and effective. Watching the showpieces prepared by the children makes the organizers very proud and usually suggests that a very hard work has been done on the side of the team coach, who is typically a teacher. And here we do not mean the work on the model or research project, which, of course is forbidden, but the didactic work done on the team, on the children. The FLL experience leaves a strong learning trace in the children. Our only concern relating FLL is the style of the challenge specification for the research project. The children are expected to perform a study and develop a solution in one of the areas that are of a critical importance to us all. Recent challenges dealt with issues such as climate changes, biomedical engineering, safe transport, or safe food. The children are asked to find plausible solutions to real problems pertaining in our world. While it is excellent to give them a reason and motivation to learn more about the World around them, it is also very unrealistic to believe they can possibly recognize a real problem and suggest a plausible solution to it. Perhaps, one in thousand can, but all play. The challenge thus either becomes just an obligatory ride, or a competition in teams getting hold of a "smartest" consultant who brings both the questions and the answers. We believe, there are interesting creative scientific projects students may be able to work on that are at a suitable difficulty and knowledge level. The outcome, however, would not be a novel solution. Rather, the outcome would be the learning experience, a *personal* discovery of facts, principles, and phenomena. And yes, they could even be demonstrated or approached in novel ways, if the team chose to do so.

FTC (*FIRST TECH Challenge*) for ages 14-18 is an advanced competition where robots of multiple teams perform together on a field to solve a specific task consisting of multiple smaller challenges. Robots are much larger than in FLL, they are built of metal parts, strong motors, and typically have some kind of manipulator arms attached. During parts of the game the robots must navigate autonomously, while in other parts they are remotely controlled using a wireless network modules and joystick consoles. The schedule of the competition follows that of the FLL. A first pilot tournament in "Eastern" European countries was organized in 2011 in Romania.

FRC (*FIRST Robotics Competition*) is very similar to FTC – it has the same age level, and also contains autonomous and remote-controlled sections, different challenge every year, but involves more interaction with human teams, which are larger. It is organized in cooperation with NASA and requires even more advanced robots. Both FTC and FRC require lots of space for the field setup, and have a high budget frame. The format is one of a TV show, but represents a lot of learning and hard work behind.

WRO (*World Robot Olympiad*) is an Asian answer to FLL, with quickly increasing impact. It has a similar format in the sense a new challenge is announced every year, and it has strict and precisely defined limitations on the allowed material. WRO covers all age levels from the primary through junior high school up to senior high school categories (college students can participate in the open exhibit). The main (positive) difference to FLL is that the WRO competition is focused on one complex task instead of multiple smaller tasks typical for the FLL. On the other hand, WRO is crippled by a bizarre requirement that the participants must assemble, program, and test their robots during 150 minute session at the competition site. In practice, they build, program and test their robots in their clubs during the months between the challenge is announced and the tournament, but they bring their kits unassembled. Teams that better memorized the robot morphology and teams that are able to reconstruct the program faster have more time for testing, sensor calibration and tuning and thus higher chances to succeed. Our impression from 2007 when we were involved in this competition in Scandinavia was: this will change soon. It proved false, unfortunately.

RCJ (*RoboCup Junior*) is an academic educational initiative that annually organizes one major world event (combined with senior RoboCup leagues) [11]. It is similar to FLL in that it also has a network of regional competitions to select the teams for higher rounds. Notice though a few important differences to FLL/WRO:

1. RCJ does not restrict the use of materials, sensors, motors and control units. On one hand, this means the better access to electronics labs, hardware workshops, university students or teachers, industrial professionals the team has, the better are its chances to win or succeed. This puts the teams in an uneven starting position. On the other hand, this opens up the really free range of possibilities so important for creative learning. And it also creates important connections out of the conceptual box of the virtual game. The game becomes real.
2. Even though the challenges of RCJ slightly evolve from year to year, the main mission usually remains the same. That allows the successful teams to participate again, and again and eventually pass their robots on to younger team-mates, making the starting threshold for others ever more difficult.
3. The team size of RCJ is not so strictly predefined as in FLL and it is smaller, in fact individual participants are also welcome and not unusual. This leaves the teacher the option to choose the approach that is most suitable for his or her local situation. On the other hand, team focus of FLL gives the option to do more on the didactic side and to provide more tailored instructions and supporting material for the teacher/coach, leaving his or her job easier.

4. RCJ is organized on a voluntary basis (FLL employs some full-time professional organizers) and thus RCJ can never achieve the level of professionalism of FLL. On the other hand, RCJ events last more than one day and include social events.

RCJ consists of three challenges – robotic soccer with infrared ball, rescue challenge, and robot dance.

Line-following contests of various flavors are the most common type of local robotic contests. Building a simple line-following robot can be efficiently done by a complete beginner in two hours, but developing a fast robot with advanced sensors and algorithms that is to perform on complex track is difficult enough for professional engineers.

Sumo contests are inspired by the human sumo wrestling, the robot's goal is to push the opponent robot out of a circular ring. The solutions usually have simple algorithms, the focus lies on the mechanical construction and electronics. How to create a reliable and strong robot? Many different size/weight versions exist from larger 20x20cm, 3kg robots, down to pico-robots of 12.5mm square footprint.

Micromouse contests are traditional single challenge maze-navigating contests that have been active for decades. Even though only little new can be invented here, the contest is a good benchmark and challenge for every truly dedicated roboticist.

RobotChallenge and *Istrobot* are annual Central-European contests that involve various categories including line-following, sumo, micromouse, but also other creative categories such as humanoid sprint, puck-collect, and humanoid sumo. The most creative category is free-ride, where the participants can bring and present their own robots of any kind. In other parts of the world, many other contests are organized, these include solving various maze tasks and racing tracks, for an example.

The contests mentioned above are suitable for young people from secondary or even elementary schools. Many other contests that are suitable for college students exist, among them Eurobot [9], Freescale Race Challenge, multiple categories of RoboCup including RoboCup@Home contest for indoor service robots, FIRA soccer competition, Robotour [10], and other.

5 Creative Robotics Contest

In the description above, we have omitted one distinguished type of contests that we find particularly interesting and useful. It is different from all the other types, and we

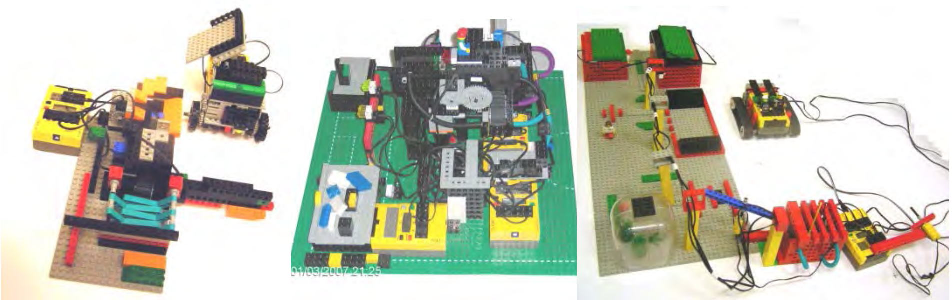


Fig. 1. Example models from the creative contest. See robotika.sk/rcj for more.

are not aware of others who would have been organizing it too.

The origins of the idea can be traced back to the 90s, when LEGO Dacta Control Lab construction sets were distributed to more than 100 schools in Slovakia as part of the government project. During this time, many teachers from elementary schools received training and acquired certificates. Pupils aged 10-14 took part in informatics classes typically in two or three different years, and the construction sets were actively used in the classes. Between 1994 and 1998, pupils from Czech and Slovak republics used to gather once a year in the free time center in the Czech town Hradec Králové. They were given or brought their own construction sets. The pupils were given a workplace with a computer and 4-5 hours of time. They constructed and programmed a creative model. At the end they have demonstrated its functionality and graded their own ideas by attaching cards to the models they liked the most.

The idea has been followed up in a Slovak national competition, with sporadic participation from the Czech Republic. It is organized since 1999 and more than 100 pupils participate annually. From 2001 categories of the RCJ contest (soccer, rescue, dance) were added to the competition, but the creative category remains. In this version of the contest, pupils were always assigned an area. Imagine the following example areas: agriculture, tourism, disabled people, space exploration. Their model was expected to demonstrate some important issue or solve a particular demand of people in the selected area. Pupils prepared a short presentation where they demonstrated the model and software functionality using a story they made up for that purpose. The jury graded the students in various categories, namely functionality, program efficiency, robustness, presentation, user comfort, and design. See figure 1 for examples of the models.

Despite the popularity of the contest, teachers were giving us the following feedback: the task was too loosely specified, pupils often built the model they have prepared in their school or club, and made up the story and small minor adjustments during the contest to fit it the assigned area. Thus it was not the real skills they were showing, but the model they have learned to build in their club or school. In addition, the grading of the jury was complicated and subjective. We responded to this feedback by changing the rules as follows: the pupils were assigned a specific task instead of a general theme. They did not know the task in advance and learned all about it directly at the contest, where we have demonstrated the rules directly on the practice field. Then again, they had 4-5 hours to build, program and test the robots built from the construction sets. At the end, they demonstrated the performance of their robots, and they received an objective score based on points their models earned according to clear and unambiguous task specification.

We have tried this version of the contest in four consecutive years since 2008. The tasks (all prepared by the author) can function as virtual games outside of the competition contest during regular school or club activities. In that case, it is recommended to allot a longer time to this activity, approximately 10 hours divided to 2 or 3 meetings. The following sections describe the tasks we have prepared.

2008 – Ball collecting. The task was inspired by one of the tasks used in the WRO, but we have modified it to form our own version. The goal was to build a robot that can travel along the track marked by a black line that was interrupted at a few places and knock down coke cans that were placed on top of wooden prisms with triangular base. On the way back to the start, the robot could collect six table-tennis balls that were resting in the centers of black circles close to a wall, see figure 2. We have prepared two testing fields, where the students could try and debug their models during the building and programming phase. The score was assigned as follows: 10 points for entering a gate, 5 points for each ball brought to the start/goal, 10 points for each can knocked down, 10 points for successful return of the robot. In case of equal scores, the robot that was faster to knock down the first can won. 15 teams participated, and one of them scored 100 out of the maximum 120 points.

2009 – Hubble Space Telescope mission took place exactly at the same time when the Atlantis space shuttle was out in the space to repair the Hubble. This inspired us to formulate the following task. The space shuttle robot with a random direction heading is placed and started on the Earth, represented by a blue circle. The sun is represented by a strong light coming from a lamp mounted on the floor. Like Atlantis that was launched around midday, the robot was to turn towards the sun, and leave the Earth towards the orbit. After reaching the orbit – a black line around the Earth, the robot was required to start orbiting, i.e. following the line. The Hubble was a white cup, fixed on the floor surface. The goal of the robot was to put three little LEGO models into the cup – install the replacement instruments and unload the astronauts (space walk). Finally, the astronauts boarded the shuttle again, the robot turned back to the Earth and landed on top of it. Scoring: 10 points for starting towards the sun, 10 points for reaching the orbit, 10 points for orbiting, 4 points for each instrument installed, 5 points for each space walking astronaut, 10 points for re-boarding, 10 points for landing, and -2 points for each astronaut lost. This contest appeared to be of a suitable difficulty. 15 teams participated. One team earned the maximum 82 points, the following three teams in the ranking order earned 77, 69, and 67 points. The atmosphere during the building phase could be described as concentrated efficient dedication, and made the organizers were happy.

2010 – Life on Titan was a life-seeking mission to the largest moon of Saturn. The task was to collect a sample (set of table-tennis balls waiting in a tube to be unloaded to the robot), which was left and marked during a visit of previous space probe. Thus the robot should have followed a black line (the trace) which splits in two at some location. The correct direction was determined by a light beacon. Earlier than that, the robot must wait on a traffic light until its technical control will complete, indicated by the change of light color. After the sample was collected, the robot was to navigate to an evacuation location, a place marked by a flashing light. Thus robots needed to be able to distinguish between a steady and a flashing light (an interesting programming task). The field plan is shown in figure 2. Scoring: 10 points for reaching the traffic light, 10 points for reaching the light beacon, 10 points for reaching the sample, 10 points for each sample ball, and 10 points for reaching the evacuation location. 21 teams participated, while the best team acquired 140 points.

2011 – Atmospheric Exploration required the teams to operate a balloon. The field was divided into several atmospheric layers, troposphere, stratosphere, mesosphere and thermosphere. Robot starting on Earth was to load two instruments (blue and red balls), board a balloon and fly up. Troposphere contained biological forms to be studied – birds represented by a black tape segments. Identification required to produce a sound after every bird has been detected. The mesosphere contained an illuminating cloud that was to be studied by one of the instruments. The other instrument was to be released in the direction of the sun. Robot was to jump out of the balloon and land on the Earth again. Scoring: 5 points for loading the instruments, 10 points for boarding the balloon, 5 points for leaving the Earth, 5 extra points for leaving the Earth with balloon, and 5 extra points for each instrument, 5 points for every bird that was correctly identified, 10 points for each instrument released, 10 points for leaving the balloon, 5 points for reaching troposphere, and 5 points for reaching the Earth. 22 teams participated, maximum score was 65 out of possible 95. The task appeared to be difficult and the teams with almost complete solutions were not lucky to earn all their points.

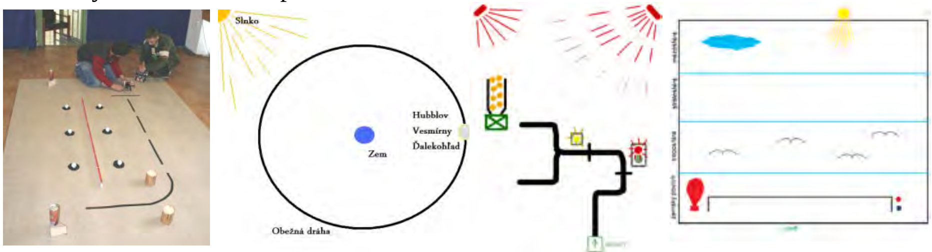


Fig. 2. Playing fields for the creative contest in 2008-2011. Details at robotika.sk/rcj.

Observations. One important observation is that the concept of virtual game works. The tasks kept the children focused, organized, and concentrated for several hours. Even the children that normally had difficulties concentrating accepted the working atmosphere and tried to obtain the best outcome. The scenarios always required creativity, experimentation, open thinking, discussion, and team cooperation. The children showed their true potential and could not be influenced by the advice of the teacher. Every mission was a valuable and memorable hands-on learning experience. We are very happy to learn that our past participants are currently successful and very active students, and continue to participate and even win in further robotics contests.

6 Supporting the Teachers

Despite the fact that LEGO provides good quality learning materials, teachers need inspiration, projects, and other type of help. Unfortunately, the NXT-G software has not been localized to Slovak language until today, which would definitely have had happened if it were an open-source project. We have written a reference manual in Slovak language, and translated the programming manual to NXC language [3], however, this cannot solve the difficulties of a 10-year old child facing English labels of various software controls. The icons are an advantage, but menus, dialogs, and labels are in English.

We have organized several teacher trainings for the recent LEGO NXT construction sets, and made the material with projects with solutions suitable for beginners available online [3]. We have specified a technical and didactic concept for robtivities portal [4], and implemented a prototype [5], which believe will be in operation in multiple languages during the conference at portal.centrobot.eu. Useful studies were compiled in final and master theses of our students and colleagues [6,7,8]. We are preparing further didactical materials to support the teachers.

7 Conclusions and Future Work

We have coined the term virtual games and explained some didactic purposes of competitions. We have discussed the frontline of robotics contests from a critical point of view. We explained what we think is one of the most suitable forms of robotics contests and provided examples that we realized and that can also be re-used as virtual games outside of the competition context. The article is a contribution to the ongoing discussion on the robotics contest. We think that the current initiatives are valuable and should continue, but that they absolutely need help in the institutionalization and coordination so that the organizers may save a lot of efforts and time they currently must spend on recruitment, fund raising and administration. This may allow integration of the efforts in the future – first signs of which can already be seen in a pilot integration of RCJ category into WRO this year.

8 References

1. Petrovic, P., Balogh, R.: Educational Robotics Initiatives in Slovakia, in Teaching with Robotics: didactic approaches and experiences, SIMPAR, Venice (2008)
2. Hofmann A., Steinbauer G., Bredenfeld A.: Robotics in Education Initiatives in Europe - Status, Shortcomings and Open Questions in Teaching Robotics, Teaching with Robotics, SIMPAR, Darmstadt, (2010)
3. Referenčná príručka ku grafickému jazyku NXT-G, Programátorská príručka k jazyku NXC, Stavebnice LEGO MINDSTORMS NXT vo vyučovaní, online: robotika.sk/nxt, (2009)
4. Balogh R., Dabrowski A., Hammerl W., Hofmann A., Petrovic P., Rajnícek J.: in Centrobot Portal for Robotics Educational Course Material, Robotics in Education, Bratislava., (2010)
5. Rajníček J.: Portál výukovej robotiky pre projekt Centrobot, Bachelor thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava (2010)
6. Malík M.: Zavádzanie robotiky do vyučovania informatiky, DVUI záverečná správa, Štátny pedagogický ústav, (2010)
7. Lehocká D.: Didaktické materiály k téme robotické stavebnice a Imagine Logo, DVUI záverečná správa, Štátny pedagogický ústav, 2010.
8. Pataky M.: Robotické laboratórne experimenty pre stredoškolskú fyziku, master thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava (2010)
9. Obdržálek, D.: Eurobot Junior and Starter - A Comparison of Two Approaches for Robotic Contest Organization, In Robotics in Education, Bratislava (2010)
10. Iša, J. Dlouhý, M.: Robotour - robotika.cz outdoor delivery challenge, In Robotics in Education, Bratislava (2010)
11. Sklar, E., Johnson, J.H., Lund, H.H.: The Educational Value of Children's Team Robotics: A Case Study of RoboCup Junior. In: AROB (2002)
12. Petrovic P., Onacilová D., Svetlík J. Skúsenosti s prípravou sútaze v stavbe a programovaní robotov FIRST LEGO League z pohladu organizátora, trénera a rozhodcu, Didinfo (2010)

Information Technology Education Add-on: „Improving Media Literacy“

Rainer Planinc, Elisabeth Wetzinger and Monika Di Angelo

Vienna University of Technology,
Institute of Computer Aided Automation
Favoritenstraße 9-11/183, 1040 Vienna, Austria
{rainer.planinc, elisabeth.wetzinger, monika.diangelo}@tuwien.ac.at

Abstract. New media, especially the internet, have spread vastly and are omnipresent nowadays. Media literacy is seen as a key qualification and considered to be a basic cultural skill. As a consequence, great importance has to be attached to the promotion of media literacy, particularly in regard to school education. The first part of this paper discusses the terms „media“ and „media literacy“. In the subsequent parts we present different (exemplary) approaches that aim at the improvement of media literacy as part of IT classes. The cutting-edge topics, chosen with respect to everyday relevance for teenagers and their interests, are centered around „General Terms and Conditions of Online Social Networks“ and „Digital Images and Videos“.

Keywords: media literacy, digital natives, digital images, videos

1 Introduction

Ulrich Saxer considers media literacy in [13] as a basic cultural skill. He also refers to it as a key skill in today‘s information society. However, the meaning of „media literacy“, particularly in reference to children and teenagers is often reduced to security issues and the careful use of the Internet. Undoubtedly, these are important topics, but certainly „media literacy“ is a term of wider comprehension.

Before going into detail on „media literacy“, it is necessary to have a quick look at the term „media“ in this context: media does not only include new media such as the internet or mobile communication media, but also printed media (e.g. newspapers), television, radio, motion picture and so on.

According to an evaluation by Harald Gapski [6] there exist more than a hundred different attempts aiming at a definition of media literacy. And due to the various types of media, it turns out to be exceedingly difficult to precisely and uniquely define the term „media literacy“. Hence we present a few attempts, which seem to be particularly important in the context of school education.

Dieter Baacke divides the meaning of media literacy into four dimensions [3]:

- Instruction-oriented dimensions: “media critique“ and “media studies“
- Goal-oriented dimensions: “media use” and “media design”

Schiermann et al. describe media literacy in [14] to consist of three complementary components: competence in handling and use of (media, internet and communication) technology, competence in the design of socio-technical (by means of media, internet and communication) technology and competence in sophisticated criticism of these technology.

Media literacy is about dealing with media, looking at it with a critical eye and using it creatively [7]. The latter does not mean the creative use in an artistic manner, but the skills to use media to fulfill one’s own concepts and needs (in an artful way). One aim of the promotion of media literacy is the acquisition of expertise to provide a competent handling of media. This includes an understanding of media as well as media production skills.

A consolidated view at the definitions of the term „media literacy“ indicates the following key points: operational competence and design competence, utilization competence as well as competences in regard to social, ethical, informative and analytic aspects. Due to the fact that nowadays most teenagers widely have access to computers and the Internet [9], its handling and competent use is increasingly seen as the key factor of media literacy.

Owing to the use of social networks like Facebook and the extensive media coverage, the awareness of security issues concerning the Internet has increased. As adults nowadays are somewhat aware of risks within the Internet, also the awareness of students and children needs to be raised. They often lack the understanding for the need to protect their privacy and they often deem these risks harmless. Thus, developing the media literacy of teenagers has to deal with these new risks when using the Internet. But media literacy is much more than just dealing with privacy issues. Media literacy also has to deal with the consequences of using social networks like Facebook. Using Facebook does not necessarily mean that you have poor social relationships in real life. But it definitely changes the every day’s life of people throughout the world, which needs to be addressed when talking about media literacy. Online social networks are mostly used to stay in touch with (real) friends and for communication with them [4]. A way to integrate and use new technologies and especially Facebook in the class is shown in [10].

But media literacy is still more than privacy and social networks. It should not only consider social media, but all types of media. Children and teenager naturally deal with pictures, audio and video every day. But often they do not have a clear understanding of the underlying processes and no consolidated knowledge of the technology they use. Hence, they only use a few simple operations on the media instead of a creative and holistic media composition – they just produce and consume media in an incidentally way. A deeper understanding of the underlying processes (e.g. how can I enhance the picture quality although the file size should be small?) offers much more possibilities. Due to these facts, it is important that media literacy also deals with some fundamental basics of these media. This does not have to be done using ex-cathedra teaching, but can be done in interactive and interesting ways (e.g. working with audio [5]).

In the following section two core areas of media literacy are discussed in detail. First we show the importance of online social networks as Facebook, MySpace and Netlog in the context of its relevance to society and the ***discerning*** use of new media. The focus of the second part is on the improvement of design and utilization competence through hands-on experience, using examples.

2 General Terms and Conditions of Social Networks

Following [8], the majority of young people use social networks extensively. Therefore it is important to encourage a careful and responsible usage of it. This is not achieved by control or penalty. Success is much better achieved by establishing awareness of the possible risks when being online. The first step towards this aim is to instill that awareness in the teachers, which for the most part are not „digital natives“. This is a prerequisite in order to educate the young people appropriately.

The General Terms and Conditions (GTC) of a web service provide an overview of the rights, duties, risks and responsibilities when using it. However, hardly anyone reads the GTC before accepting them. A closer look at the GTC proves to be worthwhile as it uncovers some frightening details. Due to the popularity of Facebook, Netlog and MySpace the GTC of these online social networks were analyzed in [10] and [15]. Some of the detected facts are:

- It is not possible to completely delete any kind of data once published in the internet. This is also stated in the general terms and conditions of Facebook in a way that they have the right to not delete data even if the user asked for it.

- Private information is used to offer personalized advertisements [11]

- Published information is possibly forwarded to third parties.

- From a legal point of view Facebook is allowed to use every information, text, picture, video, etc. for any purpose.

- Facebook, Netlog and MySpace presume that the personal data entered by the user is correct, truthful and exact as well as the user is “sui juris”

- The user is responsible for consequences resulting from the use of the network. These include for example the copyright law, the personal rights, the patent and trademark law.

- The legal basis for jurisdiction is anything but trivial. As the users may access the social network from anywhere in the world, an agreement conferring jurisdiction to a place chosen by the provider is part of the GTC. Facebook uses Californian law, whereas MySpace follows the laws of the State of New York. For Netlog the courts of law of Brussels are responsible and therefore Belgian jurisdiction is applied. The average user has no knowledge and above all, no “feeling” of what is legal with respect to the applied law.

Everyone who wants to use online social networks should be aware of these conditions. Especially young people intensively use such networks, but unfortunately many of them are oblivious of which rights and duties apply. Without doubt online social networks have many advantages, but the disadvantages (especially the ones mentioned above) are often concealed. In order to improve the media literacy of young people, it is exceedingly important to focus on these disadvantages or restrictions and to discuss them. Thereby it is aimed on gaining a discerning look at using Facebook and the internet in general.

3 Digital Images and Videos

Problem-based learning use problems as starting points, which are practically relevant for the students. A specified tangible problem has to be solved with the aid of resources of individual choice [1].

Since videos and digital images are parts of the daily life of teenagers, practice-oriented and every-day relevant problems can be formulated easily. One example could be the purchase of a new digital camera: In order to be able to make a reasonable purchase decision, one has to know the meanings of the different camera parameters and its relevance to the personal usage of the camera. During some research on the important parameters for buying a new camera, students are confronted with terms like resolution, ISO value, noise, HD ready (720p) and so on.

However, the primary goal is not to find the best camera available on stock, but to get an understanding of the different factors, camera-parameters and aspects and their interdependence. Secondarily the students should be able to make a reasonable fictitious purchase decision based on of some given requirements the camera should meet using the knowledge learned before. They should also be able to argue their decision. The theoretical background and a summary of the most important parameters are represented in [11].

It is important to not only theoretically, but also practically gain knowledge and experience with the topic. Therefore we recommend the concept of circuit training, because it is based on open and active teaching methods and advances *self-reliant learning.*

At first, several physical stations are prepared, which contain different working instructions. Some stations may be associated to each other, but they do not necessarely have to share something in common. The instructions at each station are to be worked out in small student teams. Examples of such instructions can be found in [11]. After an appropriate amount of working time a team proceeds to the next station. In general, it is irrelevant in which order the stations are worked through. It is only important, that each team visits every station.

For the promotion of media literacy regarding digital image and video each station is about a different camera type. As almost all modern cameras are able to record videos, both picture and video file formats can simply be analyzed using the following concept. Camera types, which are recommended to be used for stationary learning, are: mobile phone camera, webcam, digital camera, digital single-lens reflex camera (optionally).

Fig. 1: circuit training

Image quality differs due to different camera types and their purpose of use. Possibilities to easily experience these distinctions are provided by using the three/four camera types mentioned above. Figure 2 shows the same motif taken by different camera types under poor lightening conditions: The picture at the upper left shows a reference image representing the desired result took under normal lightening conditions. The following pictures show the same motif, but once taken with a mobile phone camera (upper right), digital camera (lower left) and a digital single-lens reflex camera (lower right). The lighting was reduced to a minimum to better demonstrate the noise performance of the individual devices.

One task of the circuit training exercise could be the reproduction of these images with an evaluation of its differences relating qualitative aspects (e.g. quality, noise performance and white balance) or quantitative aspects (e.g. file size). The reason for these differences is mainly caused by the different sizes of the image sensors: the image sensor of a mobile phone camera is significantly smaller than one of a digital single lens reflex camera. In general it can be said, that the smaller the image sensor, the worse the noise performance and generally the performance in *darker environment*s is.

Fig. 2: Noise performance of different camera types under poor lightening conditions [12]

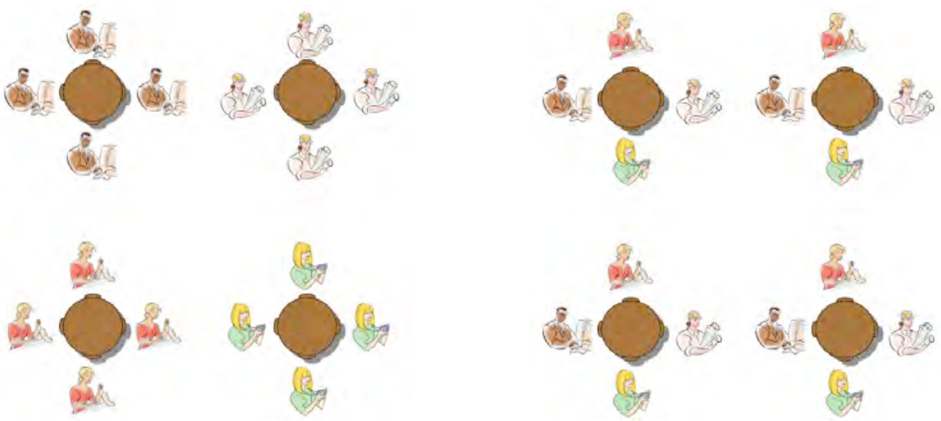
Other tasks could include taking pictures or recording videos using different parameter-settings such as different

- resolutions
- frame rates (video)
- ISO-settings
- file formats/compressions
- file formats

This circuit training exercise aims at the ability of students to discover and understand the different impacts of each parameter on the final result of the picture. Furthermore this knowledge allows for reasonably choosing the appropriate parameter settings for different requirements. For example the students should learn, that especially in darker environments digital single-lens reflex cameras perform much better than digital compact cameras, whereas in light environments the differences may be marginal.

During circuit training the students are asked to write up central questions relating to the topic, which afterwards are worked out and discussed using the jigsaw method [2]. The jigsaw method has two steps: For the first step the class is divided into as many groups as there are subtopics to be worked out – these are the expert groups who work out their subtopic (which may consist of more than one central question depending on the total amount of central questions written up before). So every member of a team becomes an expert on this central question or subtopic (Figure 3 (a)). The work is based on forms prepared by the teacher in order to structure the teamwork and on doing online research. The forms contain the central questions, some basic information on the subtopic and useful hints to online resources.

Once every expert team has gained sufficient knowledge of its subtopic, the teams are re-formed so that each new team consist of at least one member of every expert team (Figure 3 (b)). Now every expert has to teach the other group members the subtopic learned in the first step and should be able to answer questions related to it. The big advantage of this concept is the active attendance of every student, which leads to a sustainable learning success.



(a) (b)

Fig. 3: Jigsaw method (a) first step: becoming experts in a subtopic (b) second step: knowledge exchange

More examples and working instructions related to the improvement of media literacy in contexts of digital image, moving picture and digital rights aspects of online social networks can be found in [11] and [15].

4 Conclusion

In this paper we have presented a discussion of the term „media literacy“, focusing on various different attempts to define it. Regardless which one is used, media literacy must not be reduced to just being able to safely and competently use the Internet. Furthermore, examples representing approaches for IT-classes at school, which focus on different aspects of media literacy with respect to digital images and videos were shown. These examples have been chosen in such a way that the topics are relevant and interesting for teenagers, as anchors can be found in their daily lives. We have shown that media literacy also has to deal with legal aspects since agreements are concluded very easily in the internet – just by accepting the general terms and conditions of a website. Future work will deal with the evaluation of these concepts within school classes.

References

1. Arnold, K., Sandfuchs, U., Wiechmann., J.: Handbuch Unterricht. 2nd edition. Julius Klinkhardt (eds.), Bad Heilbrunn (2009)
2. Aronson, E.: The jigsaw classroom. 1st edition. Sage Publications, Beverly Hills (1978)
3. Baacke, D.: Medienpädagogik. Niemeyer, Tübingen (1997)
4. Boyd, D. M., Ellison, N. B.: Social Network Sites: Definition, History, and Scholarship. In: Journal of Computer-Mediated Communication vol. 13, Nr. 1, pp. 210-230 (2008)
5. Eisenbarth M., Di Angelo M.: "Experimentieren mit digitalem Ton". In: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (eds.), Proceedings of 25 Jahre Schulinformatik in Österreich, pp. 291-295, Melk (2010)
6. Gapski, H.: Medienkompetenz. Eine Bestandsaufnahme und Vorüberlegungen zu einem systemtheoretischen Rahmenkonzept. VS Verlag für Sozialwissenschaften, Wiesbaden (2001)
7. Hamm, I.: Fernseh- und Radiowelt für Kinder und Jugendliche. In: Schriftenreihe der Landesanstalt für Kommunikation Baden-Württemberg, vol. 3, Nr. 1., pp 69-76, Villingen-Schwenningen (1996)
8. Lenhart, A., Purcell, K., Smith, A., Zickuhr, K.: Social Media & Mobile Internet Use Among Teens and Young Adults. Pew Research Center report. http://www.pewinternet.org/Reports/2010/Social-Media-and-Young-Adults.aspx (last accessed on April 29th)
9. Österreichischer Rundfunk: Medienbesitz und Mediennutzung der Jugendlichen in Österreich. Markt- und Medienforschung, http://mediaresearch.orf.at/c_studien/Mediennutzung_Jugendlicher_2008.pdf (last accessed on April 29th)
10. Planinc R., Di Angelo M.: Facebook im Unterricht. In: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (eds.), Proceedings of 25 Jahre Schulinformatik in Österreich, pp. 296-304, Melk (2010)
11. Planinc, R.: Didaktische Aufbereitung bekannter Video-Kompressionsverfahren. Diplomarbeit, Technische Universität Wien, Wien (2010)
12. Planinc R., Wetzinger E., Di Angelo M.: Von der Kamera ins Web, In: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (eds.), Proceedings of 25 Jahre Schulinformatik in Österreich, pp. 285-290, Melk (2010)
13. Saxer, U.: Medien als Gesellschaftsgestalter. In: Medienkompetenz als Herausforderung an Schule und Bildung. Ein deutsch-amerikanischer Dialog. Konferenz der Bertelsmann Stiftung, Gütersloh (1992)
14. Schiersmann, Ch., Busse, J., Krause, D.: Medienkompetenz – Kompetenz für Neue Medien. Studie im Auftrag des Forum Bildung (2002)
15. Wetzinger, E.: Unterrichtskonzept Digitales Bild: Bildkompression, Dateiformate und Anwendungen. Diplomarbeit, Technische Universität Wien, Wien (2010)

iPods in Primary School
A Pilot Project at the Austrian "School in the Park"

Anton Reiter[1], Rosemarie Stöckl-Pexa[2], Peter Sykora[3]

[1] Federal Ministry of Education, Arts and Culture, Präs./IT,
Schreyvogelgasse 2/303, 1010 Vienna, Austria, anton.reiter@bmukk.gv.at
[2] journalist and author (project documentation)
Lichtentaler Gasse 16-18, 1090 Vienna, Austria, r.stoeckl-pexa@chello.at
[3] Öffentliche Volksschule der Stadt Wien ("School in the Park"),
Währinger Straße 43, 1090 Vienna, Austria, direktion@school4u.at

Abstract. For the first time students of an Austrian primary school have been equipped with iPods at the "School in the Park" located in the 9th district of Vienna. The Federal Ministry of Education, Arts and Culture funded iPod touch devices of the 3rd generation for all students of a so called Freinet class.
For a scientific evaluation of the project a single case study is carried out. First results confirm international studies on iPod use in primary school: The students quickly get familiar with the iPod, they are motivated to learn and to try out different applications. The mobile device can also be used outside the classroom which is a key benefit.
For education based on Freinet pedagogy it is particularly important that iPods support individual learning and foster cooperative social forms.

Keywords: iPod, primary school, Freinet pedagogy, mobile computing, 1:1 in education, case study, Austria

1 Introduction

Mobile computing is the future of ICT, also – and particularly – in education! Considering this fact the Austrian Federal Ministry of Education, Arts and Culture (BMUKK) equipped students of a primary school in Vienna with iPod touch devices of the 3rd generation[1]. The students use their iPods in classroom and at excursions, during lessons and also in leisure time.

The iPod project is already the second new media project undertaken at the "School in the Park" with the support of the BMUKK: From 2005/06 to 2008/09, the class of teacher Peter Sykora made use of Web 2.0 resources, such as wikis, podcasts and the video portal YouTube. Now as teacher of the iPod class Mr. Sykora combines

---

[1] Technical specifications about the iPod touch 3rd generation see at http://support.apple.com/kb/SP570

his past Web 2.0 project experiences with the advantages of a mobile tool by using the iPod touch for 1-to-1 education.

2 Background of the Project

Nowadays, **ICT** are considered to be **useful tools in primary schools**[2] – and even in pre-school education[3]. A "Study of the impact of technology in primary schools", funded by the European Commission and coordinated by the European Schoolnet, states positive effects on learners and learning:

" 1. ICT improves children's knowledge, skills and competences
2. ICT increases motivation, confidence and engagement in learning
3. Assessment can be more sophisticated and individualized"

(http://eacea.ec.europa.eu/llp/studies/documents/study_impact_technology_primary_school/brochure291009_en.pdf)

While the use of PCs or Notebooks in primary schools is well researched, the iPod is – even by most teachers – still seen mainly as a device for listening to music, watching videos or playing computer games. Up to now only a few studies in the United Kingdom[4], the USA, Canada, Australia[5] and New Zealand have been carried out to evaluate the educational potential of this cult product of the digital age.

---

[2] ICT in primary schools in Austria:
a) recommendations: Empfehlungen der IKT Grundschulexpertengruppe des bmu:kk (Innsbruck, 31.05.2007) mit Ergänzungen der bmu:kk Abteilungen I/1 (Volksschulen und Minderheitenschulen) und I/9 (Einsatz innovativer Technologien) vom 08.01.1008
b) literature:
Reiter, A., Grimus M., Scheidl, G. (eds.): Neue Medien in der Grundschule. Unterrichtserfahrungen und didaktische Beispiele. Ueberreuter, Wien (2000).
Grimus, M.: ICT and Multimedia in Primary School. In: PC News 70, Nov. 2000, pp. 34-36
Schwetz, H., Zeyringer, M., Reiter A. (eds.): Konstruktives Lernen mit neuen Medien. Beiträge zu einer konstruktivistischen Mediendidaktik. Studienverlag, Innsbruck, Wien, München, Bozen (2001)
Eder J., Reiter A. (eds.): Computereinsatz an österreichischen Grundschulen gestern – heute – morgen. Studienverlag, Innsbruck (2002)
Reiter, A.: 20 Years of Informatics Instruction in Austrian Schools and the Use of ICT in Class. CDA Verlag- und Handelsges. m.b.H., Perg (2005)

[3] UNESCO Institute for Information Technologies in Education: Recognizing the potential of ICT in early childhood education. Analytic survey. Published by the UNESCO Institute for Information Technologies in Education, Moscow (2010)

[4] Flakefleet Primary School
(http://www.lancsngfl.ac.uk/ictservices/ictcentre/index.php?category_id=414)
St. Francis Catholic Primary School
(http://www.lancsngfl.ac.uk/ictservices/ictcentre/index.php?category_id=415)
Forres Primary School
(http://www.thegrid.org.uk/learning/ict/research/casestudies/forres.shtml)

[5] Department of Education and Early Childhood Development, State Government of Victoria: iPod Touch Research Report, Hampton (2008)
(http://delphian.com.au/sites/delphian.com.au/files/files/attachments/ipod-touch-research-report20081215.pdf)

Janet Wilkinson, Headteacher of St. Francis Catholic Primary School, UK, states: "The iPod touch is a useful educational device and there is a great deal of choice in the applications you can use, to some extent there is too much choice. (...) The pupils have found the devices easy to use and work with them well." (http://www.lancsngfl.ac.uk/ictservices/ictcentre/index.php?category_id=415)

For Epsom Primary School, Australia, the quick access to the Internet is a key benefit: "The instant access can give you results straight away which doesn't take over the learning time. Just two taps and you are on the Internet. (...) The iPod Touch gives you instant access and instant learning." (Department of Education and Early Childhood Development, State Government of Victoria: 23)

According to international studies the **advantages of the iPod** can be summarized as follows:

- **Availability of software:** Thousands of apps (applications), often free of charge, centrally available via iTunes, simplify or enhance the teaching and learning processes.
- **Mobile Internet:** The option of unlimited Internet access is ensured via W-LAN, although in practice availability is limited, since hot spots are not found universally. Access via a router or a mobile wireless solution could be helpful.
- **Motivation:** The use of iPods as enhanced media players, offering almost the functionality of a full computer, result in increased motivation.
- **Social skills:** The 1-to-1 ratio, one iPod per child, fosters independence and responsibility, particularly when students are permitted to use the devices also during leisure time. Group work and cooperation are promoted as well.
- **Media competence:** Media use, considered in the positive sense, contributes to media education. Therefore it is necessary to bring the devices that today's children are growing up with into the classroom.

3 Project goals

Through a 2 ½ years lasting **scientific evaluation** should be found out to which extent the **positive results** reported by international studies can be **confirmed** by the specific Austrian iPod project. The following **general goals** were established before starting the project:

- **Identifying organizational and technical conditions** required for iPod use, i. e. reliable options for storing and charging the iPods, maintenance and updating the software, provision of the required peripheral devices and Internet access;
- **Raising ICT competence** which includes training the students in the use of hardware and software, building awareness of responsible handling and of possible risks, in particular concerning the Internet;
- **Exploring the educational potential** of iPods by testing programs and applications (learning software, knowledge databases, podcasts …) with

regard to the benefits for different educational subjects as well as for the use at home;

- **Developing recommendations for parents and teachers** by informing and educating of parents in the use of iPods, particularly during leisure time, as well as offering specific didactic recommendations for teachers.

4 Organizational and Technical Conditions

In the start-up phase of the project the **organizational and technical conditions** for the effective educational use of iPods were identified and fulfilled:

- **Storage:** The iPods are stored in a combination-locked safe in the classroom. This ensures that the devices are always close to hand. The combination is also known by the students, which promotes an awareness of responsibility. To help maintain order, every iPod is stored in its original packaging.
- **Charging:** The devices are charged in the classroom by five chargers for every four iPods. The students in turn act as "minders" to supervise the charging. If an iPod has to be charged at intervening times, it is connected to a running PC via USB.
- **Maintenance:** So far, no external maintenance has been required; the class teacher has exclusively been responsible for maintenance. Major problems arose only in a few cases, such as when a child accidentally deleted all programs and it was necessary to restore the default settings.
- **Updating:** Installation of new versions and additional programs is a time-consuming process. The class teacher downloads the data on his MacBook and then transmits it to the iPods of the students one-by-one. Colleagues as well as parents assist the class teacher.
- **Access to the Internet:** After a test operation until December 2010, the students can use the Internet via W-LAN, inside school and at hot spots in the city of Vienna for instance when being on excursions.

5 Didactics

The iPod project is carried out in a Freinet class; this means that the iPods are used according to the **Freinet pedagogy**. The pedagogical approach of the French educational reformer Célestin Freinet (1896-1966) assumes that children enjoy learning when they are allowed to decide for themselves what to do. The most important concepts of this pedagogic approach are the following:

- "Pedagogy of Work ('Pédagogie du travail') – meaning that pupils learned by making useful products or providing useful services.
- Co-operative Learning ('Travail coopératif') – based on co-operation in the productive process.
- Enquiry-based Learning ('Tâtonnement experimental') – trial and error method involving group work.

- The Natural Method (‘Methode naturelle’) – based on an inductive, global approach.
- Centres of Interest (‘Complexe d'intérêt’) – based on children's learning interests and curiosity. ”

(http://www.freinet.org/icem/history.htm)

Class teacher Peter Sykora’s opinion that ICT are ideal tools for Freinet pedagogy is supported not only by other Austrian teachers like Astrid Sonnleitner, but also by the UNESCO Institute for Information Technologies in Education:

“If Freinet had had available media like Internet, search engine, homepage, computer, printer, e-mail during his lifetime, he would have used these tools adequately.”[6] (Sonnleitner in Schwetz et al.: 234)

Célestin Freinet “integrated the classical and modern pedagogical ideas and ICT existing in the 1930s to formulate a model of school that can be recognized as an educational framework for many modern applications in primary education (...)”. (UNESCO Institute for Information Technologies in Education: 20)

The iPods are mainly used **interdisciplinary**. For example, searching the Internet for the term “bagpipes”, the students downloaded written information (subject “General Education”) and music videos (subject “Music”) from the Internet. Since the students also use apps in English language, they learn English quite automatically.

The iPods are suited for **different teaching and learning forms**. Frontal instruction is used for explaining the operation of the iPod and of new apps. During group work, some students work on a content together, one of them makes notes on the iPod. If they work individually, the students share solution strategies for completing tasks on the iPod. They use the iPod spontaneously as well as planned, according to the work schedule.

The project class has already started to use the iPod as a **communication tool**. The students write e-mails, short blog entries and Tweets; they exchange dates like birthdays via the “Calendar” app.

The **work phases** with the iPods are of varying length; the students are largely free to determine duration and frequency of iPod use. On average, the students work with the iPods for half an hour to an hour per week.

6 Selection of Software (Apps)

There are numerous apps available which are suitable for primary school students. A few apps are preinstalled on the iPod, others can be downloaded either free of charge or purchased – for a lower price than comparable software for PCs.

The followings apps are frequently used in class:

Preinstalled app:

[6] „Hätte Freinet zu seinen Lebzeiten Medien wie Internet, Suchmaschine, Homepage, Computer, Drucker, E-Mail zur Verfügung gehabt, so hätte er diese Mittel adäquat genutzt.” (original German quotation)

- "**Music**" can download and display not only audio files like music and spoken text, but also videos. Educational podcasts are particularly popular, e. g. "Kinderuni" (Children's University) or "The Show with the Mouse".

Downloadable app free of charge:

- "**Maps**" displays maps from the WWW located by street names, geographic terms and points of interest. The program supports voice recognition.

Purchased apps:

- "**Articles**" searches for terms in the Wikipedia online encyclopedia and shows a summary of the entries and an image. Pages inappropriate for children do not appear.
- "**Tree identification**": Trees can be identified based on specific features, such as the shape of the leaves and the trunk, blossoms and fruit color.
- "**Dr. Brain**" trains general education, logic and visual memory.
- "**Times Tables**" is a multiplication trainer, starting with the 2nd row, with increasingly difficult tasks until the 12th row can be solved.
- "**Slice it!**": Geometric figures have to be cut into pieces as close in size as possible using straight lines. The app is used as a supplement to geometry because it trains area comprehension and deduction.
- "**What is what – Dinoquiz**" is a guessing game on dinosaurs.

7 Evaluation of the iPod Project

For a scientific evaluation of the iPod project at the "School in the Park" a single case study is carried out that will be completed by the end of 2011. The aim of the study is to **find out whether iPods are appropriate devices to achieve the educational goals set in the Austrian curriculum for primary schools**.

In the section "General Educational Goals" of the Austrian curriculum for primary schools ICT are mentioned only once: One goal is "the development and transmission of basic knowledge, skills, competences, understanding and attitudes that serve the learning of the basic cultural techniques (including the use of modern communication and information technologies suitable for children)"[7] (http://www.bmukk.gv.at/medienpool/14043/lp_vs_erster_teil.pdf)

How the educational goals can be fulfilled is described in the section "General Didactical Principles"[8]; the hypotheses (see below) are derived from these principles.

As Margarete Grimus states, the way in which ICT are integrated in primary school education falls within the area of school autonomy (see Grimus in Mitzlaff: 216)[9]. At

[7] „Entwicklung und Vermittlung grundlegender Kenntnisse, Fertigkeiten, Fähigkeiten, Einsichten und Einstellungen, die dem Erlernen der elementaren Kulturtechniken (einschließlich eines kindgerechten Umganges mit modernen Kommunikations- und Informationstechnologien) (...) dienen"

[8] Bundesministerium für Unterricht, Kunst und Kultur: Lehrplan der Volksschule, Dritter Teil, Allgemeine didaktische Grundsätze, Stand: BGBl. II Nr. 368/2005, November 2005, pp. 22-27 (http://www.bmukk.gv.at/medienpool/14044/vslpdritterteil3682005frhp.pdf)

the “School in the Park” iPods are used according to the educational approach of Célestin Freinet.

Defining a **research objective**, it must be taken into account that there is reference data on the use of ICT, but not of iPods in Austrian primary schools; therefore one objective is to highlight the strenghts and weaknesses of the iPod in comparison to other ICT devices like PC or Notebook. Financial and personal resources, organizational conditions and technical equipment available in school must be considered as well.

The **research questions** are derived from the research objective:

- Is the **hardware** suitable for primary school students?
- Is sufficient **software** available for educational purposes?
- Which **didactic methods** are appropriate?
- What are the **financial** and **personal resources** required, what **organizational conditions** and **technical equipment** do we need?

7.1 Preparation and Data Collection

Since the project at the “School in the Park” is the first iPod project at an Austrian primary school, the focus is on **exploration**, i. e., the study uses a **qualitative approach**. The hypotheses of the case study are based on the didactical principles of the Austrian curriculum for primary schools and of Freinet pedagogy.

The following table shows **research methods** applied to **research objects**:

Research objects	Research methods			
	Semi-structured interview	Written survey using a questionnaire	Participant observation	Non-participant observation
Class teacher	●			
Team teacher	●			
All students		●	●	
Selected students				●
All parents		●		
Selected parents	●			

7.2 Data Analysis

For the evaluation of the iPod project primary research is complemented by analysis of secondary material.

[9] „Es ist eine der Aufgaben der Grundschule, grundlegende Medien- und IKT-Kompetenz zu vermitteln. Im Unterschied zu den traditionellen Kulturtechniken Lesen, Schreiben und Rechnen sind Lehrplanziele für diese vierte Kulturtechnik in Österreich jedoch kaum ausgeführt. Details der Integration von IKT fallen in den Bereich der autonomen Schulprofile.”

The **semi-structured interviews** are recorded, transcribed, categorized and analyzed using **qualitative content analysis**[10]. This method is also applied to the answers to the open-ended questions of the **written surveys**; the answers to the closed-ended questions are categorized and displayed as a table and as a graph.

Every action of the students during **participant and non-participant observation** is described and categorized; verbal comments are also subject to qualitative content analysis.

Work schedules as well as short texts, written by the class teacher, the team teacher, students or their parents (e. g. class blog entries) serve as **secondary material**. The work schedules are statistically evaluated; qualitative content analysis is applied to the written texts.

7.3 Results

By the end of school year 2010/11 the results of the semi-structured interviews with the class teacher, the team teacher and selected parents, the written survey of all students and the results of the participant observation are available. They **confirm the hypotheses** to a large extent:

- Hypothesis 1: The iPods (**hardware**) as well as the available **software** are **suitable** for handling by primary school students.
  Confirmed: The students understand the graphical user interface intuitively – even more easily than their parents. Most children state it is "moderately difficult" to use the iPod; no child finds it "difficult". For the majority of boys it is "easy", for the majority of girls "moderately difficult". The students prefer the multi touch screen as an input device to the mouse they use on the PC; only for writing longer texts they give preference to the keyboard. Even though the screen is very small there are no safety concerns because the work phases are short. In general the students have good understanding of the software used in class; if they find an application difficult it may be due to the handling (e. g. the compass) or to the required knowledge (e. g. a knowledge quiz). Careful evaluation of the software by class teacher, team teacher and some parents guarantee that there is no harmful content.
- Hypothesis 2: iPods **support social learning** because they foster cooperative social forms (e. g. working in teams or pairs, helping each other).
  Largely confirmed: The iPods are used in individual work as well as in partner or group work. If they may choose the social form the students mostly form groups in which each child works with her or his own iPod, using the same application as the others. The students exchange hints for solving tasks and help one another. The small screen is an obstacle to group work using one iPod together. The students frequently cooperate in class,

[10] Mayring, P.: Qualitative Inhaltsanalyse. Grundlagen und Techniken. Beltz Verlag, Weinheim und Basel (2003)

working with iPods or not; this suggests that not (only) the working tool but the pedagogy is crucial for social learning.

- Hypothesis 3: iPods are **part of the students' world of experience** and communicate contents by appealing to **several senses** (multimedia-based).
  Partly confirmed: iPods are part of the students' world outside school now, but before the iPod project had started most parents only used PCs. The project raised the interest of the parents in iPods, now most of them have either an iPod or an iPhone at home. A few students have learnt to send e-mails or use the Internet from their parents before this was taught in class. Primary school students generally rather play computer games on Gameboys or cell phones, so the iPod is "something special". The project class frequently uses multimedia-based applications, e. g. videos or interactive software, on the iPod as well as on the iPad or the PC.
- Hypothesis 4: The iPod gives a **holistic view** of the subject of a lesson regarding various aspects, e. g. by using the Internet.
  Confirmed: The iPods are mostly used interdisciplinary. The students complement what they have learned by carrying out their own investigations, e. g. by using online encyclopedias.
- Hypothesis 5: According to the curriculum the iPods are **used in different subjects**.
  Confirmed: Up to now the iPods were used in most subjects; there are useful applications even for handcrafts and physical education. For writing longer texts and for the preparation of presentations PCs are more suitable.
- Hypothesis 6: The iPod **motivates** the students to learn and encourages them to **try out different work techniques**, inside as well as outside class.
  Largely confirmed: The iPod motivates the students to learn because even not so attractive contents like multiplication are interesting if practiced with an interactive educational game. In the initial phase of the project the students almost always preferred the iPod to traditional media like worksheets or books, more and more it became a work tool like others. In class the students try out different work techniques on the iPod, e. g. various educational games, e-mail and Internet. At home the children mainly play games like Doodle Jump on the iPod, only a few use it for learning.
- Hypothesis 7: iPods facilitate **personalized instruction**. They are suitable for dividing the class into varying **groups with different tasks**.
  Confirmed: Students use the iPods according to their own paths of learning. Slower children are assisted by the teacher or other students. Since the project class is a Freinet class the students usually form varying groups with different tasks.
- Hypothesis 8: iPods are used in **different learning phases** (exercise, repetition, revision, applying knowledge in a new context). By using the iPod students can **check their knowledge on their own**.
  Confirmed: iPods are used in all learning phases. By using educational games the students frequently check their knowledge on their own.
- Hypothesis 9: iPods are used for creative joint activities like writing a **digital class magazine** or exchanging **e-mails**.

Largely confirmed: The students frequently write Tweets, some exchange e-mails with class-mates and parents. They have started to write short entries for the class blog.

- Hypothesis 10: iPods help students to **explore the world** they live in, e. g. by carrying out their own investigations or during field trips.

  Largely confirmed: The students search the Internet for contents they are interested in. Applications like Tree Identification are used also outside school. Mobile Internet access would be very useful on field trips but is not available up to now.
- Hypothesis 11: The students **decide for themselves** to use the iPod, spontaneously as well as scheduled.

  Confirmed: If the students may choose teaching material they decide in about half of the cases for the iPod (in the beginning they took the iPod in almost any case). They use it more often spontaneously than scheduled.
- Hypothesis 12: Class teacher and students **together set binding rules** for the use of the iPods.

  Largely confirmed: Most rules for the use of the iPods were set by the class teacher, but the students could give input (e. g. "minders" for charging the devices was an idea of the students). The children stick to the rules concerning the iPod.

Hypothesis 1 – 8 base on the didactical principles of the Austrian curriculum for primary schools, hypothesis 9 – 12 on the principles of Freinet pedagogy.

To evaluate the cost-benefit ratio **financial and personal resources, organizational conditions and technical equipment** must also be taken into account. The findings show that without **funding** by the Federal Ministry for Education, Arts and Culture and the **class teacher's knowledge** of Apple devices the project would not have been possible. Except for the mobile access to the Internet the organizational and technical conditions were not a major problem.

7.4 Conclusion

The results show that **iPods are appropriate devices** for primary school education. Without the long boot time typical for PCs, the iPod is immediately ready for operation, therefore it is ideal for **spontaneous use**. iPods are cheaper than PCs, Notebooks and Netbooks, so it is easier to realize **1-to-1 computing**, but still too expensive without funding. Because of their small size they can also be **used outside the classroom**, e. g. during field trips, what cold be improved by mobile Internet access.

References

iPod Manuals

1. Biersdorfer, J. D., Pogue, D.: iPod. The Missing Manual. O'Reilly Media, Inc., Sebastopol, Canada (2009)
2. Bove, T.: iPod & iTunes For Dummies. Wiley Publishing, Inc., Indianapolis, Indiana (2008)
3. Hart-Davis, G.: How to Do Everything iPod, iPhone & iTunes. McGraw-Hill Osborne Media, New York, Chicago, San Francisco (2009)
4. Miser, B.: My iPod touch. Que, Indianapolis, Indiana (2010)

iPods in Education

1. Department of Education and Early Childhood Development, State Government of Victoria, Australia: iPod Touch Research Report. Hampton (2008) (http://delphian.com.au/sites/delphian.com.au/files/files/attachments/ipod-touch-research-report20081215.pdf)
2. Flakefleet Primary School, UK (http://www.lancsngfl.ac.uk/ictservices/ictcentre/index.php?category_id=414)
3. Forres Primary School, UK (http://www.thegrid.org.uk/learning/ict/research/casestudies/forres.shtml)
4. Kervin, L., Reid, D. (Faculty of Education, University of Wollongong, Australia), Vardy, J., Hindle, C. (Wollongong Diocese of Catholic Education): A partnership for iPod pedagogy: Using the technology of millennial learners across educational contexts. In: ASCILITE 2006. The 23rd Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (http://www.ascilite.org.au/conferences/sydney06/proceeding/pdf_papers/p111.pdf)
5. St. Francis Catholic Primary School, UK (http://www.lancsngfl.ac.uk/ictservices/ictcentre/index.php?category_id=415)
6. STEPS: Study of the impact of technology in primary schools (Brochure) (http://eacea.ec.europa.eu/llp/studies/documents/study_impact_technology_primary_school/brochure291009_en.pdf)
7. STEPS: Study of the impact of technology in primary schools (Synthesis Report) (http://eacea.ec.europa.eu/llp/studies/documents/study_impact_technology_primary_school/02_synthesis_report_steps_en.pdf)
8. UNESCO Institute for Information Technologies in Education: Recognizing the potential of ICT in early childhood education. Analytic survey. Published by the UNESCO Institute for Information Technologies in Education, Moscow (2010)
9. Vallance, M.: iPod therefore iWrite. Future University, Hakodate, Japan (2006) (http://associates.iatefl.org/pages/materials/wi4.pdf)
10. Vardy, J., Kervin, L., Reid, D.: iPods and podcasting technologies to support Talking and Listening experiences of Grade 4 Students (http://www.englishliteracyconference.com.au/files/documents/Papers/Refereed%20Papers/Jeff%20Vardy.pdf)

ICT in Austrian Primary Schools

1. Bundesministerium für Unterricht, Kunst und Kultur: Lehrplan der Volksschule, Erster Teil, Allgemeines Bildungsziel, Stand: BGBl. II Nr. 368/2005, November 2005, pp. 1-3 (http://www.bmukk.gv.at/medienpool/14043/lp_vs_erster_teil.pdf)
2. Bundesministerium für Unterricht, Kunst und Kultur: Lehrplan der Volksschule, Dritter Teil, Allgemeine didaktische Grundsätze, Stand: BGBl. II Nr. 368/2005, November 2005, pp. 22-27 (http://www.bmukk.gv.at/medienpool/14044/vslpdritterteil3682005frhp.pdf)
3. Eder J., Reiter A. (eds.): Computereinsatz an österreichischen Grundschulen gestern – heute – morgen. Studienverlag, Innsbruck (2002)
4. Grimus, M.: ICT and Multimedia in Primary School. In: PCNews 70, Nov. 2000, pp. 34-36
5. Reiter A., Grimus, M., Scheidl, G. (eds.): Neue Medien in der Grundschule. Unterrichtserfahrungen und didaktische Beispiele. Ueberreuter, Wien (2000)
6. Grimus, M.: Computer in österreichischen Grundschulen. In: Mitzlaff, H. (ed.): Internationales Handbuch Computer (ICT), Grundschule, Kindergarten und Neue Lernkultur, Band 1, pp. 216-222. Schneider Verlag, Hohengehren (2007)
7. Schwetz, H., Zeyringer, M., Reiter A. (eds.): Konstruktives Lernen mit neuen Medien. Beiträge zu einer konstruktivistischen Mediendidaktik. Studienverlag, Innsbruck, Wien, München, Bozen (2001)
8. Reiter, A.: 20 Years of Informatics Instruction in Austrian Schools and the Use of ICT in Class. CDA Verlag- und Handelsges. m.b.H., Wien (2005)

Freinet Pedagogy

1. History of Freinet Pedagogy (http://www.freinet.org/icem/history.htm)
2. Sonnleitner, A.: Freinet-Pädagogik mit PC und E-mail? In: Schwetz, H., Zeyringer, M., Reiter A. (eds.): Konstruktives Lernen mit neuen Medien: Beiträge zu einer konstruktivistischen Mediendidaktik, pp. 228-236. Studienverlag, Innsbruck, Wien, München, Bozen (2001)

Links to the School in the Park

1. http://www.school4u.at/ipod
2. http://twitter.com/#!/ipodklasse
3. http://www.facebook.com/parkschule

YouTubeVideos

1. Ipod Education at GEMS Royal Dubai School (http://www.youtube.com/watch?v=V9kczm4QWos)
2. iPod in Education - video overview pt2 (http://www.youtube.com/watch?v=2-AROmt1yhM&NR=1)
3. iPods in Education (http://www.youtube.com/watch?v=N6LlA2SztY8&feature=related)
4. iPods used in KCK elementary school (http://www.youtube.com/watch?v=L2GBbLGO9d8)
5. Using iPods in Our Classroom for the First Time (http://www.youtube.com/watch?v=X2UJQ_UrPW4&feature=related)
6. Using the iPod Touch in the Classroom (http://www.youtube.com/watch?v=3bhzjB0TyOE&feature=related)

Creativity in Computer Science Education – Eleven Findings

Ralf Romeike

University of Potsdam
Institute of Computer Science
14482 Potsdam, Germany
romeike@cs.uni-potsdam.de

Abstract. Creativity increasingly receives attention in the context of computer science education in the last years. In this article 11 research based findings concerning creativity in CS education will be outlined. Consequences are drawn from creativity research and research in computer science and computer science education. A short description of an example for a creative introduction to programming illustrates the implementation of the findings and its consequences.

Keywords: Creativity, Computer Science Education, Scratch.

Introduction

Computer science (CS) education provides an important contribution to secondary education. However, its aims and goals are manifold and vary widely in the educational discussion. In the last years there seems to be a shift in the community regarding the perception of computer science and hence the purpose of computer science education. Traditionally, competencies of CS such as problem solving, programming and the understanding of computer systems have been considered as essential. Now also "soft" competencies like creativity come into focus. Modern software and learning environments like Scratch encourage and support creativity in a computer science context. As a new learning objective, empowering youth to use modern technology and CS strategies in a *creative way* is aimed for. More and more CS educators are becoming aware of the potential that technology offers for realizing ideas, and making everyday life better and easier. CS education plays an important role in achieving this goal.

In the following, eleven research based findings concerning creativity in CS education will be outlined. Consequences are drawn from creativity research and research in CS and CS Education. A short description of an example for a creative introduction to programming illustrates the implementation of the findings and its consequences.

11 Findings about Creativity in CS Education

1. In an educational context creativity needs to be considered from an individual's perspective.

The term "creativity" has been used with different meanings and has been discussed controversially in the field of psychology. Correspondingly, research in creativity focuses on a variety of aspects, such as identifying, assessing, and fostering creativity. Moreover, various dimensions such as a creative person, creative process, influence of environmental factors, and creative products are explored in creativity research (see e.g. [1] for further consideration).

In contrast to historical creativity, which describes ideas that are novel and original in the sense that nobody has ever come up with them before, p-creative designates something that is fundamentally to the particular person [2]. In educational context, p-creativity is based on knowledge in both practical and applied form as well as on the willingness to acquire and use this knowledge. We consider learning processes from an individual's perspective as described by Boden [2].

With this in mind, we call a phenomenon "creative" in this paper when it leads to original, adaptive, and useful ideas, solutions, or insights (cp. [3, 4]). Typical characteristics of creativity include high interest, intrinsic motivation, enjoyment, and individual's challenge by the work itself (cp. [5]).

2. Creativity in the classroom enhances intrinsic motivation and thus may result in a better success in learning.

Being creative is fun and motivating. Motivation is the driving force behind any human action and essential for learning processes. Educators know about the importance of motivation. Intelligence, aptitude and social background are factors that can influence learning, but they are out of reach of the teacher. Intrinsically motivated students reach better achievements. This possibility to raise a student's achievement may be used as a powerful tool by the teacher. Interestingly, creativity and motivation determine each other. Studies investigated the motivation of software developers in Open Source projects (e.g. [6, 7]). They identified enjoyment-based intrinsic motivation, namely how creative a person feels, as the strongest and most pervasive driver. Also in computer science lessons the motivating effect of being creative can be put into action. Students report about motivation due to the constructive character of computer science and the possibility of creative self-fulfillment [8]: It is fun to be creative! In being creative one finds self-fulfillment and self-realization. Csikszentmihalyi named such an intensive involvement in a creative activity as *Flow* [9]. For learning such a condition would be optimal if reached by students.

Summarizing, creativity in the classroom promises to enhance motivation, attention, curiosity and concentration, thus resulting in a better success in learning.

3. Research indicates emphasizing creativity improves Computer Science Education.

Several researchers argue that computer science generally is a creative endeavor and hence students need to be motivated to get insights in creative problem solving (e.g. [10]). This will lead to creative design processes which require open ended tasks in real life contexts and exploration [11]. These ideas are supported by a study by Gu and Tong [12] who found that in courses for software development courses design and programming activities were considered as creative by students and preferred to other activities. Similar observations are reported for secondary education [13]. In programming courses, allowing and encouraging students to be creative increased motivation and interest [14-16]. Related ways to foster more creative learning settings include:

- changes in the environment and encouraging creative, hands-on-learning and exploration into the projects in a data structures and algorithms course ([17])
- letting students choose and process their own problems ([18])
- allowing programming as personal creative expression (e.g. [19, 20])
- presenting programming in an entertaining discovering way [21]

Additionally, creativity is called for from several authors in CS education, because:

- Graduates in CS are missing creativity and problem solving skills [22]
- Creativity is underrepresented in the curriculum [23]
- Women drop out because there is no room for individual creativity in CS courses [24]
- Creative abilities are seen as the highest form of literacy, including computer literacy [25]

Summarizing, many of the activities undertaken in computer science are highly creative. In order to address the motivation of our students and at the same time meeting this creative perception of CS, creativity should be highlighted, encouraged and supported in CS lessons whenever possible.

4. Creativity forms a pathway to Computer Science.

By analyzing students' computing experiences Knobelsdorf and Romeike [26] found that characteristics of creativity may form possible pathways into the field of computer science. In an empirical study they examined computing experiences in the form of "computer biographies" of CS majors. These where previously collected to explore how students learn and understand Computer Science. The study had two major outcomes. First, characteristics of creativity were found in computer biographies of many students who chose to major in computer science: *These students particularly perceive CS as fun, creative, and autonomous; which is typically described in the context of programming. Striving for well working software was the main motivator for engaging in programming. In the majority of programming processes, tasks chosen by students themselves are meaningful to them, but often surprisingly irrelevant as a product. In these processes the activity (mostly* *programming) is most important, which is typical for creative processes and known from artists. This group of students is fascinated and interested in the possibilities that a computer offers them. They express a strong desire for gaining further knowledge, exploration, and understanding. We consider the activities of this group as highly creative.*

Second, the students who are attributed a creative approach to computer science describe their lessons in high school as disappointing due to a lack of possibilities to get involved in creative activities: Apparently a bigger emphasis on creativity in CS classes is needed in order to not discourage and lose this part of students. Possibly even more students can be attracted if they discover the creative side of CS.

This aspect is especially challenging when seen in the context of meager interest and high dropout rates of women in CS: Women often drop out because there is no space for individual creativity in CS courses [24] or they just do not consider CS as a subject to study because CS is perceived as uncreative (e.g. [27]). Uncovering the creative aspects of CS in secondary education and higher education may not only keep more women in CS course but also persuade new students to consider further education in the field of CS).

5. Creative learning extends constructionist learning.

Constructionist learning is recognized and documented as a successful approach for computer science education: In his constructionist learning theory Papert describes learning as a constructive process and emphasizes that learning "happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity" [28]. Learners should dive into real situations instead of just looking at them from the outside. The construction of artifacts and a constructionist understanding of learning provides an appropriate basis for meaningful and creative learning of computer science. Creative learning adds to learning in the sense of constructionism the aspect of problem finding with regard to the learner's personal situation and their own capabilities. Resnick [29] proposes a process for creative learning that follows a creative thinking spiral (cp. Fig. 1):

Fig. 1: Resnicks Creative Thinking Spiral.

"In this process, people imagine what they want to do, create a project based on their ideas, play with their creations, share their ideas and creations with others, and reflect on their experiences—all of which leads them to imagine new ideas and new projects. As students go through this process, over and over, they learn to develop their own ideas, try them out, test the boundaries, experiment with alternatives, get input from others, and generate new ideas based on their experiences." ([29], p. 18)

In formal educational settings "imagine" can be interpreted as problem finding and idea generation. Hence, obviously creative learning is based on ideas of the learner, but not the teacher (cp. also [30]).

6. Computer science is considered as creative by computer scientists.

Computer science, as computer scientists see it themselves, is a creative field to work in, where creativity is demanded and encouraged (for example [31-33]). Studies show that generally CS is perceived by non-computer scientists as uncreative, non-social, and theoretical [27, 34-36]. However, computer scientists point out the creative facet of CS:

„Computer science is the first engineering discipline ever in which the complexity of the objects created is limited by the skill of the creator and not limited by the strength of the raw materials" (Brian K. Reid cited in [37]). In contrast to other disciplines the computer scientist can take part in constructing his own world. Unlike in physics or chemistry the reality is not set but can be modified and extended. Especially programming/software development is considered to be creative: "It is there that the original source of creativity lies" [38]. Several pioneers of computer science agree [39, 40]:

Ray Ozzie*: Programmers are very creative, self-directing, self-motivating people.*
Bill Gates*: So much judgment and creativity goes into a programming project.*
Guido van Rossum*: To me, [programming] relates strongly to creativity, which is very important to my line of work.*

Would it not be beneficial if students, who generally refer to creativity in the context of fields like music and art only, supported the statements above?

7. Programming is the "song writing" of computer science.

Probably no other domain of CS is as involved with the creation of creative products as programming. While the creation of music is usually not questioned as being creative, this indeed is different with software design. However, parallels from the field of musical composition to programming can be easily drawn. The composer as well as the programmer needs a stimulus that starts the creative process. Often it is called an inspiration, idea, problem, task, request or any kind of motivation for somebody to write a song/software. The composer/programmer draws from his inventory of knowledge (facts, concepts, etc.) and puts it into the design of his musical/software work. The design process is strongly influenced by constraints, which need to be obeyed. The composer needs to regard his culture group, customs and personal preferences and has to obey certain constraints such as tonality, style, harmonic rules, rhythm and others. The programmer needs to consider constraints like the circumstances of how, where and by whom the software shall be used as well as limitations of the programming language used, computing resources and many more [41]. During the design both use a pool of problem solving patterns, heuristics and experiences and apply them in finding and realizing a musical or software solution. Both processes result in a "creative" product. The creative character in all of the phases of software development is elaborated by Glass [31] and Romeike [42]: When finding and concretizing a goal, when managing and solving problems, when creating the software product, in testing and when presenting the final result. For CS lessons software development is especially interesting when students have the possibility to develop and realize their own (and thus personal relevant) ideas. Only out of this can the intrinsic motivation develop, which is essential for creative performance.

8. Computer Science is Art.

Art is the embodiment of creativity in a traditional sense. As outlined above, creative processes in art and CS are comparable. Hence it is not surprising that some computer scientists describe their field as art. Here you find two different viewpoints. First, the focus is on the creative process itself (mostly programming): the computer scientist perceives the things he does as art, based on his abilities, knowledge and fun in creating fine software (e.g. [43]). Out of this comes the idea of "beautiful code" and appropriate software design. Second, the emphasis is less on the process but more on the computer scientist/programmer who is acting as an artist. Here the focus is on the creative artistic product; mostly a piece of software which expresses expertise, elegance, often efficiency and creativity simultaneously.

Another context for CS and art is where artists use CS and programming in order to produce artwork. The primary interest here is less efficiency, elegance or reliability of software but the development and elaboration of ideas and the aesthetic success of the work. CS contributes to the creation of art by providing ICT that can be used to support artists, ranging from image, audio and video processing software to interactive components for live installations (e.g. with Arduino[1]).

9. Computer Technology supports Creative Learning.

Probably no other school subject is more influenced by its tools than CS. Also, there is probably no other school subject that can rely on such a variety of tools for introducing a topic, simulating facts or as a subject matter. ICT is seen as helpful for the support of creativity; a wide body of creativity research in CS is concerned with the possibilities of supporting creativity in professional areas as well as in learning scenarios [44]. New technologies can support creative processes in collecting, relating and connecting information and ideas, in creating something new and disseminating the outcome. Doing so the roles of ICT can vary from nanny, pen-pal, coach to colleague [45]. A creativity supporting environment is essential for creativity.

[1] Arduino is an electronic platform allowing to create programmable interactive electronic objects, cp. www.arduino.org.

However, an understanding of fundamental CS concepts is necessary in order to use ICT creatively. Research based criteria for choosing a tool with respect to creativity are discussed in [46] and with respect to secondary CS education in [47].

10. Creative computer science lessons can lead to learning success in secondary education.

At a time when standardized tests are getting more and more common the call for more attention to something that is seen as ineffective as creativity may seem a bit odd. We developed and evaluated a prototypical lesson sequence for a creative introduction to programming with Scratch in order to investigate the practicability and outcomes of creativity-oriented CS lessons (cp. [13]). Scratch [48] is a visual programming language that had a tremendous impact on CS education within the last several years and is used increasingly in primary and secondary education. The intuitive user interface makes the implementation of creative CS lessons easy. The design of the lessons follow from the findings stated above: Maintaining students' attention and fostering motivation was supported by showing the use and relevance of the concepts to be learned to the students and by choosing topics meaningful to them, e.g. animating their name or a story from their everyday life or imagination, and the development of games. Often new concepts were brought up by the students themselves after discovering and applying them in their projects before their "official introduction".

Essential for a creative lesson is providing an inspiration to the students, generally by showing an example program or brainstorming about possibilities. This allows the students to spark their creativity: to balance what they may want to achieve and what they can achieve with the concepts learned so far and what the programming language is capable of. Hence the students were challenged by open ended tasks with variable solution complexity and independent working time. The tasks assigned were basically pointing the students into a direction giving a general "frame" about what to do. Thus the students had to solve a problem they needed to clarify for themselves up front ("What do I want to do?"). There was no one right solution that needed to be achieved (openness) and – as time allowed – the solution could be elaborated upon by the students. An example of such a task was: "Design a program, which displays your name and animates the letters to interact to the mouse or keyboard!"

This way the students could get familiar with the concepts just learned, explore the programming environment, find solutions for their ideas, implement and test them. The teacher would go around, encourage the students to try out their possibilities and only interfere if asked or needed. Usually such a working period ended with the end of a lesson. This way those students, who wanted to elaborate their work or to extend or modify their programs, could continue to do so at home.

The teaching unit introducing programming fulfilled the expectations: The students enjoyed the lessons, the learning objectives were met and the students' picture of CS improved. This is in unison with studies where contextualization, personalization, and choice produced dramatic increases: in students' motivation, their depth of engagement in learning, the amount they learned in a fixed time period, and their perceived competence and levels of aspiration [49].

The students' effort was concentrated and intrinsically motivated. Even when a lesson was over many of them did not want to leave the classroom in order to continue in working at their project (Flow). The presentation and dissemination of the students' results led to increased motivation in the next lesson. Even another course at the school was getting to know the results of this course as many students soon started to play online the games created.

After the lessons the students described computer science as a subject that requires creativity but also allows them to be creative. Obviously it was possible to win the interest of the students by encouraging creativity. The learning success was surveyed by analyzing the grades and outcomes of a programming test and in a comparative study with a control group that was receiving lessons in a "traditional" way with problem solving tasks. Results are given in detail in [13]; Fig. 2 shows the outcomes of the survey where the creative group (A) experienced the programming lessons e.g. as more fun, interesting and creative as the control group (B).

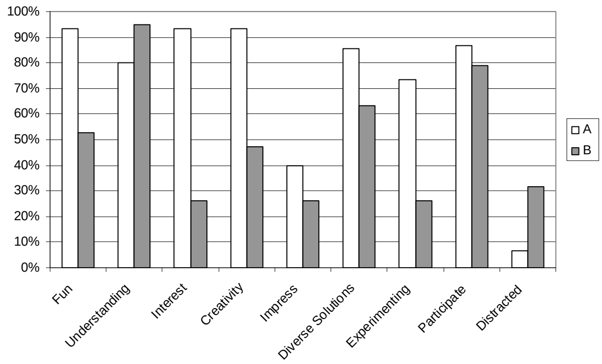


Fig. 2: Student judgments: Agreement in Comparison with Control group.

11. The relevance of creativity in computer science education is increasing.

Up to now mainly complex software products were used on personal computers and computer science secondary education often was misunderstood as training for use of such products. With the App stores of Apple, Google, Amazon, etc. youth have access to smaller and hence more accessible and reproducible software products: Various tools and games are available for smartphones, tablet computers, and social media websites like Facebook and Google. Modern SDKs like App Inventor make the programming of such Apps possible in secondary education. Creativity can be an adequate guide for such learning of programming. Developers of Scratch already emphasize this idea: Programming serves primarily as a tool for personal expression. Hereby students acquire "digital literacy" and skills that are essential for taking part in the so called creative society [29]. The transfer of a creative understanding of computer science to parents and friends may offer the chance to improve the common understanding of computer science as boring and uncreative. This could also lead to an increased interest in the subject in schools and universities.

Computer scientists and teachers of computer science need to verify if the creative perspective of computer science meets their understanding of the field. Based on existing research the possibilities predominantly offered by software development were described. Nevertheless computer science offers many other options for applying creativity in the classroom. With just a bit of creativity at the teachers' side it is not hard to transfer the ideas mentioned in this article to many other learning activities in the computer science classroom.

References

1. Runco, M.A., *Creativity : theories and themes ; research, development, and practice*. 2007, Amsterdam [u.a.]: Elsevier. XI, 492 S.
2. Boden, M.A., *The creative mind: myths & mechanisms*. 1990, London: Basic Books.
3. Runco, M.A. and I. Chand, *Cognition and Creativity.* Educational Psychology Review, 1995. **7**(3): p. 243-267.
4. Kaufman, J.C. and R.J. Sternberg, *Creativity.* Change: The Magazine of Higher Learning, 2007. **39**(4): p. 55-60.
5. Ruscio, J. and T.M. Amabile, *How does creativity happen?*, in *Talent Development*, N. Colangelo and S.G. Assouline, Editors. 1996, Ohio Psychology Press: Dayton, OH.
6. Lakhani, K. and R. Wolf, *Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects*, in *Perspectives on Free and Open Source Software*, J. Feller, et al., Editors. 2005, MIT Press. p. 3-22.
7. Luthiger Stoll, B., *Spass und Software-Entwicklung: Zur Motivation von Open-Source-Programmierern. Dissertation*. 2006, Zürich: Universität Zürich.
8. Romeike, R., *Kreativität im Informatikunterricht.* 2008, Universität Potsdam: Potsdam.
9. Csikszentmihalyi, M., *Flow : The psychology of optimal experience*. 1990, New York u.a.: Harper [and] Row.
10. Scragg, G., D. Baldwin, and H. Koomen, *Computer science needs an insight-based curriculum*, in *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education*. 1994, ACM Press: Phoenix, Arizona, United States. p. 150-154.
11. Hill, A.M., *Problem solving in real-life contexts: An alternative for design in technology education.* International Journal of Technology and Design Education, 1998. **5**(3): p. 1-18.
12. Gu, M. and X. Tong, *Towards Hypotheses on Creativity in Software Development.* Lecture Notes in Computer Science, 2004. **3009**: p. 47-61.
13. Romeike, R. *Applying Creativity in CS High School Education - Criteria, Teaching Example and Evaluation*. in *7th Baltic Sea Conference on Computing Education Research (Koli Calling 2007).* 2008. Koli National Park, Finnland: Conferences in Research and Practice in Information Technology.
14. Lewandowski, G., E. Johnson, and M. Goldweber. *Fostering a Creative Interest in CS*. in *SIGCSE '05*. 2005. St. Louis, MO.
15. Long, J., *Just For Fun: Using Programming Games in Software Programming Training and Education.* Journal of Information Technology Education, 2007. **6**: p. 279-290.

16. Romeike, R. *Applying Creativity in CS High School Education - Criteria, Teaching Example and Evaluation*. in *the 7th Baltic Sea Conference on Computing Education Research, Koli Calling*. 2008. Koli, Finnland.
17. Lewandowski, G., E. Johnson, and M. Goldweber. *Fostering a Creative Interest in Computer Science*. in *SIGCSE '05*. 2005. St. Louis, MO.
18. Meisalo, V., E. Sutinen, and J. Tarhio. *CLAP: teaching data structures in a creative way*. in *Proceedings of the 2nd conference on Integrating technology into computer science education*. 1997. Uppsala, Sweden.
19. Peppler, K.A. and Y.B. Kafai, *Creative Coding: Programming for Personal Expression.* 2005.
20. Resnick, M., *Rethinking Learning in the Digital Age*, in *The Global Information Technology Report: Readiness for the Networked World*, G. Kirkman, Editor. 2002, Oxford University Press: Oxford. p. 32-37.
21. Wilson, B.C., *A study of learning environments associated with computer courses: can we teach them better?* 2004. **20**(2): p. 267 - 273.
22. Mittermeir, R., *Informatik-Unterricht: Bastel-Unterricht, eine intellektuelle Herausforderung oder "Preparation for the information-age".* Medienimpulse, 2000. **9/33**: p. 4-11.
23. Sweeney, R.B., *Creativity in the Information Technology Curriculum Proposal*, in *Proceedings of the 4th conference on Information technology curriculum*. 2003: Lafayette, Indiana, USA. p. 139-141.
24. Guzdial, M. and E. Soloway, *Teaching the Nintendo generation to program.* Commun. ACM, 2002. **45**(4): p. 17-21.
25. Van Dyke, C., *Taking "computer literacy" literally.* Communications of the ACM, 1987. **30**(5): p. 366-374.
26. Knobelsdorf, M. and R. Romeike. *Creativity as a Pathway to Computer Science*. in *13th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2008)*. 2008. Madrid: ACM Press.
27. AAUW, American Association of University Women, *Tech-Savvy: Educating Girls in the New Computer Age*, ed. A.A.o.U. Women. 2000, Washington, DC: American Association of University Women.
28. Papert, S. and I. Harel, *Situating Constructionism*, in *Constructionism*, S. Papert and I. Harel, Editors. 1991, Ablex Publishing: Norwood, N.J.
29. Resnick, M. *Sowing the Seeds for a More Creative Society*. in *Learning & Leading with Technology, International Society for Technology in Education (ISTE)*. 2007.
30. Romeike, R. *Where's my Challenge? The Forgotten Part of Problem Solving in Computer Science Education.* in *3rd ISSEP Intern. Conf. on Informatics in Secondary Schools - Evolution and perspectives, Torun, Polen 2008*. 2008.
31. Glass, R.L., *Software creativity 2.0*. 2006, Atlanta: developer .* Books.
32. Leach, R.J. and C.A. Ayers. *The Psychology of Invention in CS.* in *17th Annual Workshop of the PPIG*. 2005. Univ. of Sussex, Brighton UK.
33. Saunders, D. and P. Thagard, *Creativity in Computer Science*, in *Creativity across domains: Faces of the muse*, J.C. Kaufman and J. Baer, Editors. 2005, Lawrence Erlbaum Associates: Mahwah, NJ.
34. Margolis, J. and A. Fisher, *Unlocking the Clubhouse: Women in Computing*. 2002: MIT Press.
35. Rich, L., H. Perry, and M. Guzdial, *A CS1 course designed to address interests of women*, in *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 2004, ACM Press: Norfolk, Virginia, USA. p. 190-194.
36. Barker, L.J., K. Garvin-Doxas, and E. Roberts, *What can computer science learn from a fine arts approach to teaching?* SIGCSE Bull., 2005. **37**(1): p. 421-425.

37. Frenkel, K.A., *Profiles in computing: Brian K. Reid: a graphics tale of a hacker tracker.* Commun. ACM, 1987. **30**(10): p. 820-823.
38. Noble, J. and R. Biddle. *Notes on Postmodern Programming*. in *ACM conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)*. 2002.
39. Lammers, S., *Programmers at work: interviews*. 1986, Redmond, Wash: Microsoft. 385 S.
40. Littler, J. *Art and Computer Programming*. 2005 [cited 28.04.2008]; Available from: http://www.onlamp.com/pub/a/onlamp/2005/06/30/artofprog.html.
41. Curtis, B., et al. *Empirical Studies of the Design Process*. in *Second Workshop on Empirical Studies of Programmers*. 1987.
42. Romeike, R. *Three Drivers for Creativity in Computer Science Education,*. in *IFIP-Conference on "Informatics, Mathematics and ICT: a golden triangle"*. 2007. Boston, USA.
43. Knuth, D.E., *Computer Programming as an Art.* Communications of the ACM, 1974. **17**(12): p. 667 - 673.
44. Shneiderman, B., et al., *Creativity Support Tools. Workshop Report on Creativity Support Tools*. 2005, Washington, DC: National Science Foundation.
45. Lubart, T., *How can computers be partners in the creative process: classification and commentary on the special issue.* Int. J. Hum.-Comput. Stud., 2005. **63**(4-5): p. 365-369.
46. Shneiderman, B., *Creativity support tools: accelerating discovery and innovation.* Commun. ACM, 2007. **50**(12): p. 20-32.
47. Romeike, R., *Softwaretools für kreatives Lernen im Informatikunterricht.* Tagungsband zur GI-Fachtagung Informatik und Schule INFOS Berlin, 2009.
48. Maloney, B., Kafai, Rusk, Silverman, Resnick, *Scratch: A Sneak Preview.* IEEE Computer Society, 2004: p. 104 - 109.
49. Cordova, D. and M. Lepper, *Intrinsic Motivation and the Process of Learning: Beneficial Effects of Contextualization, Personalization, and Choice.* Journal of Educational Psychology, 1996. **88**(4): p. 715-730.

Implementation of the new Federal State standard of primary education: the first year results on the example of the course “Mathematics and Informatics”

Tatiana Rudchenko

Dorodnicyn Computing Centre of Russian Academy of Sciences, 40 Vavilova, Moscow, Russia
rudchenko1@yandex.ru

Abstract. The article concerns the integrated course “Mathematics and Informatics” by Semenov A.L. and Rudchenko T.A. The authors state the importance of the course as the “launch pad” for ICT-competence formation, and a natural basis for integration of all subjects in the respect of the ICT tools implementation. The authors describe the basic methodological principles and educational aims of the course, various forms of disciple activities in the course, as well as the possibilities of the site which organizes the work of 180 pilot schools using the course. The authors state the advantages of the course and mark positive results of the course implementation in the 1-st grade curriculum based on the references of the teachers who took part in this project.

Key words: metasubject educational results, ICT-competence formation, integrated course “Mathematics and Informatics”, Federal State standard of primary education.

The new Federal State Educational Standard for Primary Education (FSES) declares metasubject and subject results closely connected with “Informatics” subject matter. Such results, in particular, encompass: the use of ICT devices for communicative and cognitive tasks; the use of different means of data search, collection, and processing; the skill of operating within the primary educational information milieu; the mastering of basic metasubject (informational) concepts; the acquiring of algorithmic thinking basics; the acquiring of initial notions of computer literacy etc.

It seems most productive to learn Informatics at primary school together with Mathematics within the framework of an integrated course “Mathematics and Informatics”. As far as Mathematics and Informatics (as disciplines) use common conceptual, logical and algorithmic bases, the integration of these two subjects in primary school enables: to save class time, to highlight, naturally enough, the interdisciplinary ties and links, to show kids the instruction matter from different angles including the demonstration of most general information methods of doing sums.

So far as the majority of subject and metasubject results, planned within the course “Mathematics and Informatics”, feature the concept of ICT-competence, the course itself presents the launch pad for ICT-competence formation. This fact, in its turn, preconditions natural integration of all subjects on the basis of the course in the respect of the ICT tools implementation.

The course called “Mathematics and Informatics” (A. L. Semenov et al.) is being elaborated within the framework of the pilot project of introducing FSES on the platform of 180 Moscow schools. In accordance with FSES the developing course “Mathematics and Informatics” presupposes systemic and pragmatic approach aiming at students’ age consideration and formation of individual instruction paths. It includes independent learning, research and practical activities understandable and attractive for students. The whole variety of disciple activities within the framework of the course “Mathematics and Informatics” may conditionally fall into four groups: work with the textbook, project activity, computer work, subject games.

The work with the textbook is organized so as to give 1st year students freedom in doing it in class almost without any assistance from the very beginning, so that the teacher can concentrate her help on individualized ‘trouble-shooting’. This is attained with the predominant use of graphic explanations, exhaustive, though not large, instruction texts and explicitly introduced game rules. Under the game rules we mean the description of all notions (arrangements), which are exploited in task formulation. To maximally facilitate new material (independent) studying by kids all new concepts are introduced in the form of graphic examples, while the verbal texts serve to comment illustrations. As the textbook contains all information necessary for task solving and formalization, the students regularly have no need to run for any external back up. The work with the textbook in itself is targeted on information competence basics formation (the ability to operate due to the system of rules and restrictions, including those contained within descriptions and instructions; the ability to understand information, displayed in different forms (texts, graphics, tables, charts, schemes); the ability to select proper information and neglect the unnecessary items and so forth). In the course of work with the textbook mastering of basic metasubject concepts and attainment of the majority of subject results come gradually.

The project activity conceives of tasks meaningful for kids and utilizes information methods and tools, including the ICT. The project task has research, practical, creative touch and develops communication, managerial and leadership skills (the ability to distribute responsibilities amongst several members of the group, the ability to negotiate, coming to an agreement with each other, and so forth).

Subject games include manipulating with objects (LEGO bricks, counting aids, bead strings etc.) and also playing parts, including role model ones, with mathematic bias. All this permits to uncover the course matter in accord with age of the primary school students and promotes communicative competence formation and further interest in the subject.

Computer part of the course presupposes several activities: computer lesson task solving, work with simulators, games, communication via the course site.

The work with the computer task complex proposes a consecutive solution of computer lessons based on course topics. At the same time, computer tasks support only those course issues, where the computer use results in clear technological and methodical gain. In the course kids get acquainted with main text and graphic editing program tools, improving their computer literacy. Students can tackle computer lessons independently for all tasks have maximally friendly interface, in particular, all texts have sound versions.

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Working with simulators, used for developing technical skills (mainly computational) and solving various entertaining tasks and those with practical and game content (for instance, “dress the doll” or “ drive a car through the maze”), backs up backward students and stimulates the leaders.

The work of the pilot schools is organized on the site nachalka.seminfo.ru in the Moodle environment. Our “Mathematics and Informatics 1st Form” course is represented on this site by three courses (three parts: 3-11, 12-22, and 23-33 weeks). Every teacher gets his/her own course space on the site where he/she works only with his own class recording all students’ activities. The results of kids’ work with a computer are saved and the teacher not only examines and evaluates kids’ works but also comments on solving the whole lessons and individual tasks. The site, besides holding computer lessons complexes, simulators and games, also contains pages from the textbook or project notebook for each lesson, where all texts have sound, which is convenient for kids with poor reading skills. The site gives various opportunities to kids to converse with the teacher and with each other (chats, forums, short message exchange and the like). The teacher can place his or her own methodic materials in the personal site space, thus enriching kids’ education milieu and individualizing it.

Due to teachers’ estimation (first year intermediate outcome) the majority of objectives outlined by the course “Mathematics and Informatics” turn to be attainable. The manual is available for independent work, and the teacher manages to use different manipulative handlers and run mini-projects and games during the lesson. A very simple description of the first course topics enables kids to feel free working independently and meeting game rules. The further growth of task and topic difficulty goes on almost inconspicuously. The majority of students master the proposed subject matter.

Teachers especially mark changes happened with the appearance of computers and the site in their work with parents. The latter follow their kids’ work on the site and eagerly communicate with teachers in forums. For some technical reasons pilot schools were provided with computers only in the middle of the academic year, henceforth, the computer part of the course was used by the students mainly at home. Parents’ involvement in the work on the site intensified their cooperation with the teacher. There appeared the opportunity to discuss various issues, which had been omitted before for the lack of time during rare parents meetings.

According to intermediate working outcomes evaluation in 2010-2011 academic years the course got unanimous acclaim from the teaching staff and administration of the pilot schools. The teachers hope to continue educating students with the help of this program.

References

1. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Manual-notebook for Institutions of General Education. In 5 Parts. Part 1 (Математика и информатика. 1 класс. Учебник-тетрадь для общеобразоват. учреждений. В 5 частях. Часть 1). MCCME publishers, Institute of New Technologies, Moscow (2010).
2. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Manual-notebook for Institutions of General Education. In 5 Parts. Part 2 (Математика и информатика. 1

класс. Учебник-тетрадь для общеобразоват. учреждений. В 5 частях. Часть 2). MCCME publishers, Institute of New Technologies, Moscow (2010).
3. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Manual-notebook for Institutions of General Education. In 5 Parts. Part 3 (Математика и информатика. 1 класс. Учебник-тетрадь для общеобразоват. учреждений. В 5 частях. Часть 3). MCCME publishers, Institute of New Technologies, Moscow (2010).
4. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Manual-notebook for Institutions of General Education. In 5 Parts. Part 4 (Математика и информатика. 1 класс. Учебник-тетрадь для общеобразоват. учреждений. В 5 частях. Часть 4). MCCME publishers, Institute of New Technologies, Moscow (2010).
5. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Manual-notebook for Institutions of General Education. In 5 Parts. Part 5 (Математика и информатика. 1 класс. Учебник-тетрадь для общеобразоват. учреждений. В 5 частях. Часть 5). MCCME publishers, Institute of New Technologies, Moscow (2010).
6. Semenov A.L., Rudchenko T.A.: Mathematics and Informatics. 1st Form. Project Notebook for Institutions of General Education (Математика и информатика. 1 класс. Тетрадь проектов для общеобразоват. учреждений). MCCME publishers, Institute of New Technologies, Moscow (2010)
7. Semenov A.L., Rudchenko T.A., Posicelskaya M.A., Posicelsky S.E., Soprounov S.F., Soprounova N.A., Hovanskaya I.A.: Mathematics and Informatics. 1st Form. Computer integrated course. In 3 parts. Weeks 3-11, 12-22, 23-33 (Математика и информатика. 1 класс. Интегрированный компьютерный курс в 3 частях. Недели 3-11, 12-22, 23-33), http://nachalka.seminfo.ru/course/category.php?id=240

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

The EasyLogo paradigm

Maria Skiadelli

National Technical University of Athens, *15780 Athens Greece*
`skiadelli@gmail.com`

Abstract. This paper presents a series of educational activities implemented in classroom with children between 9 and 12 years old in a Greek primary school, during a trimester period. It tries to describe in detail the educational scenario that was followed in order to make children acquire basic programming skills using the EasyLogo environment. It mainly focuses on aspects like the teaching path that was followed, the classroom set up, the problems faced by the teacher and the cognitive difficulties of the children. Finally, some samples of the children's work are also presented. The paper is not meant to present however, the results of a research experiment, since none of the educational research methodologies was followed, but to report the classroom experience from the teacher's point of view and to share it with other teachers and practitioners [9].

Keywords: Logo, computer literacy, didactics of informatics, young children

1 Introduction

During school year 2010-2011, a pilot program was implemented in Greece for 800 primary schools, extending the basic curriculum to include new innovative subjects, amongst others one called Information and Communication Technologies (ICT), mostly focusing on informatics education. In order to implement this program the Greek Ministry of Education had to move temporarily specially trained teachers of Informatics from the secondary level to the primary level of education. The teachers were given enough autonomy to implement their own curriculum and methods in a certain given framework of teaching instructions suggesting Logo as the appropriate language for teaching basic programming skills to young children aged between 10 and 12 years old. Since no reference to a specific environment or tool was made, the writer of this paper used the EasyLogo environment [1] in one of the schools of the pilot program after localizing it in Greek. Here it should be mentioned that children that participated in this experiment had no previous knowledge of programming at all and had rather limited experience with computers in general, mostly coming from playing computer games and surfing the Internet.

The educational activities that will be presented took place in a three month period for two teaching hours per week. The work started with children of the 3rd, 4th and 5th grade but only the children of the fifth grade managed to reach the last set of activities. The two others stopped earlier for reasons to be explained later.

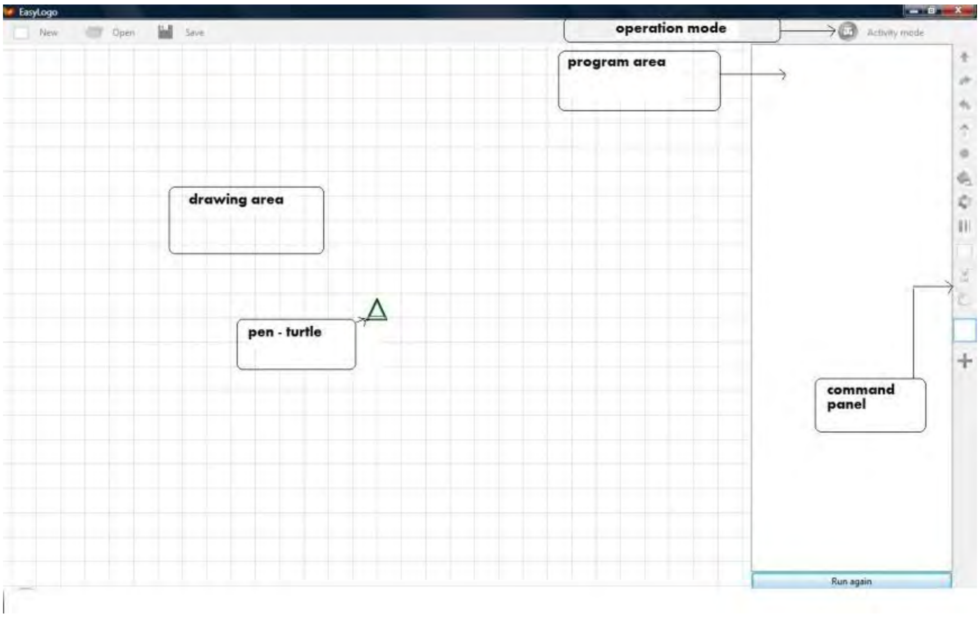


Fig. 1. The EasyLogo environment

2 Classroom set up and previous knowledge

The work with EasyLogo started after the first school trimester which was devoted to familiarize the children with basic computer literacy skills like typing and mouse moving, basic operation system handling, document editing and some very simple internet concepts.

The courses took place in a computer lab of 12 personal computers. The children had to work together in groups of 2 or 3. The choice of partners was up to them, but it had to remain unchangeable throughout a series of activities, unless a major collaboration problem occurred between the members of a group.

In all the activities, children were given some degrees of freedom; they did not have simply to follow and execute commands (the use of worksheets was avoided for this exact reason) but they had rather left to make their own choices and take decisions in a well-defined framework of action set by the teacher. As a result the outcome of their work considerably varied. By letting the children take decisions about their own work and actions is important for it strengthens their autonomy and self-esteem and also helps them understand two basic principles of real life: a problem may have many different solutions and different people view things in different ways.

3 The teaching path

Each series of activities described below is comprised by a number of simpler activities which will be may be also referred later as steps. According to well established learning design methodologies [2] all these activities together comprise a unified Unit of Learning (UoL) [3]. Activities are based on the constructivist theory [4] and constructionism approach [5][6][7] and some of them, especially the most advanced ones, on situative learning principles [8]. Every series of activities, as well as the whole UoL, possess a scalable degree of difficulty, abstraction and complexity and they build upon each other. They form a continuous teaching path which is characterized by uniformity in terms of educational goals and teaching methodologies.

Each of these activities may last for one or more than one teaching hours or classroom sessions of 45 minutes. During each session a certain teaching procedure is followed: reviewing of previously acquired knowledge, setting a new problem that requires the use of new (more advanced) knowledge, experimenting with newly acquired knowledge together with problem solving techniques and finally relaxation and play.

At the end of each series of activities, children take screenshots of their work they save it in a file and they write a small report of what they did and how they did it and of what they think that they have learned. This process helps them to reflect on their work in a holistic way and become more conscious of the knowledge and skills that they have used to accomplish their work.

3.1 Preliminary activities

The first series of activities is based on role playing techniques [9]. Programming is introduced as a robot instructing technique i.e. you want a robot to do something for you so you have to give the right commands telling it exactly how to do it. The robot understands only a limited set of instructions “forward”, “left” and “right”. A child volunteers to play the role of the robot and to be “programmed” by another child, its “master”. A demo (by the teacher) is necessary so that the children can see the impact of the commands to the robot’s movement. At this step it’s very important for the teacher, to insist on the accuracy of the terminology used by the children. When one kid starts giving instructions to his/her classmate, she/he must use the only the right command, for instance “forward” and not “go forward” or just “go” etc. This is necessary for helping children realize through experience the difference of the communication languages between humans and between humans and the machine. The term programming language is not used at this stage, but the children understand that the language that a human uses to control a machine must be strict, formal and unambiguous. After some examples with few pairs of volunteers, it’s the right time to mention for the first time the term “programming”, as an answer to the question: “what is your classmate doing right now to that robot?” New terminology would rather be introduced to young children in an experiential way and not by giving formal definitions.

3.2 Game and role playing (Activity mode)

The second series of activities is carried out using the "Activity mode" of the EasyLogo environment. That is a major step for children of young age since they have to move from the physical-real world to the virtual world of the computer, which should be handled with care. They are not supposed to give verbal commands to their fellow classmate who plays the robot role anymore; instead they have to learn to instruct a virtual entity in their computer by dragging the right command into a certain area called *the program area* (fig. 1)[1]. To help them make this transition smoothly, we take care so that the repertoire of commands of both situations remains unchanged.

By using the *Activity Mode* of the Easy Logo environment students can be engaged into imaginary situations like "the bee that needs to visit the flowers", "the car that runs a race", etc. These are in fact little microworlds [10] constructed by using various background images and an entity that represents the pen-turtle object that can move over this background. While children navigate through these microworlds, they fill like navigating through the different levels of a computer game, which makes them very enthusiastic[2].

The series of activities described in this section focus mostly on the development of algorithmic and design skills as they are basically based on simple graph solving problems [1]. Most of the children find it fairly easy to drive their entity (the "car" or the "bee") as required. However, when one looks more carefully, he/she will notice that most of them do it in a rather ad-hoc way. Look at the following example (fig 2a): the bee moves to whatever direction is possible at each step and at the next step the child just tries to correct its way by moving it towards the right direction. Here the teacher needs to make an important intervention so that children pay attention to the design phase of their entity's movement. In fact they should get used to make their entity move according to a predefined path and not by chance. They should be urged to design this path beforehand by drawing it (electronically or on a piece of paper) or even by negotiating it with their group partners (or both). Designing the path before starting dragging commands into the program area is not only a good design practice; it strongly enhances their algorithmic skills and makes them much more conscious of the programming process.

At this step there is a good opportunity also for the teacher of informatics to talk about the notion of "equivalency" in programming, since there may be more than solutions to a given problem, given by the groups of the classroom.

---

[1] That fact that the commands can be drawn and not written helps them avoid typing and syntactical errors and at the same time is quite similar in a way to giving verbal commands.

[2] Some of these microworlds are found in the original EasyLogo package, or can be constructed by the teacher or activity maker.

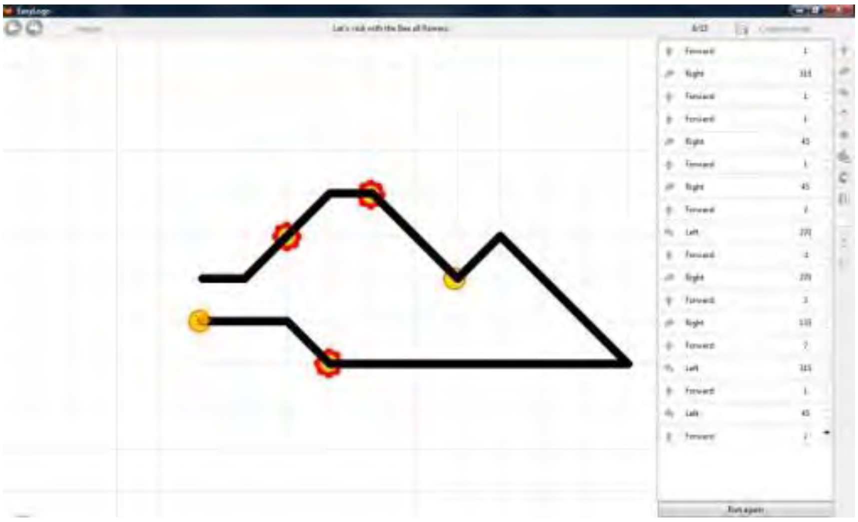


Fig. 2. Ad-hoc movement path

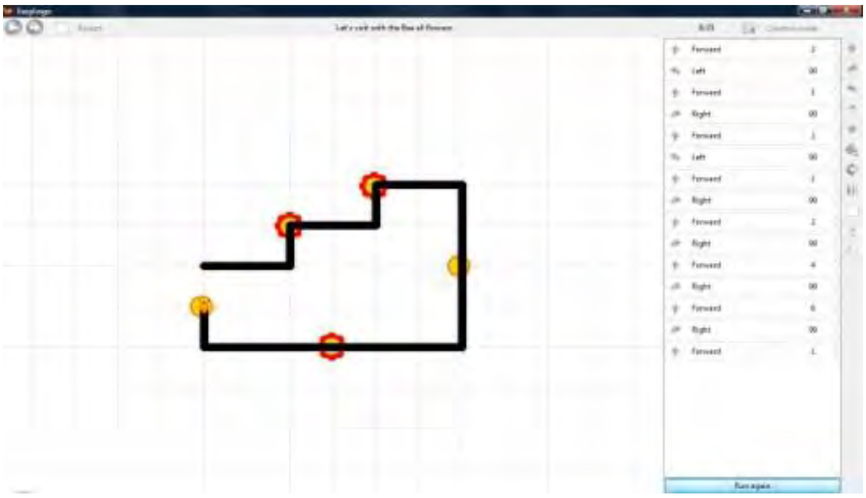


Fig. 3. Pre-designed movement path

Last thing that a teacher should try to help children understand at this step is the idea of good and bad programming practices and how they can be enforced by the application of certain rules. The teacher with the help of the children can elaborate a set of rules, the rules of "good" programming. A rule for instance can be "never use two times the same command successively" or "never use an angle greater than 180° (for left and right)". Children then should be urged to make the necessary corrections to their programs so that they conform to these rules. Moreover, making corrections to their own programs is a highly reflective process that helps them become more and more conscious of their newly acquired programming skills.

3.3 The classical Logo square

The third series of activities is based on the creation of the classical Logo square and is characterized by the transition from the *Activity* to the *Creative* mode of the EasyLogo environment. The children have to move to a more abstract universe where there is no graphic background, just a simple white paper, and their little entity has now been transformed to a simple small triangle. Since the degree of abstraction in the current series is considerably higher than previously, this change should be again handled with care in order not to turn disappointing for children and hinder their cognitive processes.

Taking that into account, it was found that it would be more suitable to implement the first activity of the series again in the *Activity mode*, rather than going directly to the *Creative mode*. So being in the *Activity mode* children are asked to follow the trace of a pre-drawn square path by dragging the right set of commands as usual[3]. Children at this step have to notice the repetitiveness that appears in their program and to find out which is the pattern that is repeated. That is crucial for the next steps of the activity and the teacher should employ various techniques in order for them to achieve it. One useful technique for instance is that he/she starts reading the commands of the program that create the square while the children keep their eyes closed. This helps them to discover the sound pattern that is repeated and memorize it more easily. Memorizing the code is also important for a reason that will become obvious in the next step.

So the next step (better take place in the same teaching session as the previous one) is to move to the *Creative Mode* and start creating the square again by dragging the right set of commands into the program area. They have to do it all alone and they are considerably helped by the memorization of the repeating part that took place in the previous step.

Once they manage to write the program that creates the square, it's time to start making some transformations of this square. One may think of many possible transformations: a square of double (or half) side length, a square that is turned to a rectangle etc. These transformations are really very amusing and the children that should be let free to play and experiment with their program. By playing, they become more conscious of the fact that the drawings on their screen is the result of the movement of the small triangle entity and they change according to the commands

[3] Although the solution of this problem is as well-known repetitive, the *repeat* command is not yet mentioned at this step.

that they use to drive this entity. As soon as they realize this, they are ready to move to the next series of activities.

3.4 Creating interesting shapes

The next series of activities was the last for the children of younger ages (3rd and 4th grade). It seemed that they got cognitively overloaded at this point. This was not only due to the use of the new tools (commands, programming environment, etc.) that they had to learn how to handle, but mostly because of the energy and cognitive power required from them in order to get used to a new philosophy of thinking and working, that needs continuous interaction with the machine . Since this was their very first experience with programming, it is rather justifiable that they reached their cognitive and physical limits rather early. Perhaps next year things will be much easier, since they will know what to expect.

The first step of the activity series is as usually based on the last step of the previous one. Let’s recall that this was the free creation of a square and its various transformations. Now the kids are asked to start changing the colors of the four sides of the square. The new command *Pen colour* is introduced that changes the color of the pen. This task it turned more difficult, especially for the younger children, than first appeared. It seemed rather awkward for them to learn that they had to change the color of the pen just before drawing the line and not after the line was drawn, possibly because of the habits that they had acquired from commonly used painting programs, yet this is a hypothesis that needs to be checked. The right use of the *Pen colour* command was not the only difficulty at this step. The solution to the problem of “how to draw a colorful side square” is obviously to insert some new commands between the lines of the code of the black square. That however seemed hard for them to realize and had to be explained over and over again. Most of them tried to solve the problem by drawing a second square just below the lines of code that draw the initial black square and not by altering this code. Another similar activity that a teacher can use at this point to help them practice more on the insertion of commands between the lines of an existing piece of code is to make a square with sides of different thicknesses. In order to do that, children have to use a new command called *Pen width*. However it seems that the introduction of this new command does not pose a significant problem to them, since its functionality is very much similar to the *Pen colour* command.

The next activity is for playing and relaxation. Kids are asked to combine different thicknesses and colors for the four sides of the square, to create interesting square shapes. They may even change the size of the square or transform it to a rectangle. This activity enforces creativity and imagination and it is ideal to be the final activity for younger children (between 9 and 10 years old), because it gives to them a light and joyful feeling about programming.

The final step of this series is to ask them to fill the square with a color of their choice and then with a random color. The purpose of this activity is twofold: first to learn how to use the new command called *Fill* that fills a closed shape with color and second to get introduced to the idea of randomness, which is of great importance to computing in general. The teacher at this point has to make the children notice what happens when they choose the “random” attribute for the *Fill* command and then press the *Run it again* button. Children should be urged to express their own hypothesis, discuss about it with peers and check its correctness by a trial and error procedure. Once they are able to understand in detail what happens when they press the *Run it again* button, they are not only able to answer to the question “why the color changes each time the program is executed” but more importantly they become more conscious of the execution process of a program and its consequences.

3.5 Drawing a train

After drawing the basic square and playing with it by changing sizes, colors, and thicknesses, etc. children are ready to make a different kind of transformation that will turn the abstract square to an object of the real world: a wagon! That is something quite easy for them: by just adding two dots at the bottom side of the square, the latter all of a sudden turns to a small train wagon. The problem that the kids have to solve this time is how to make a train by using this simple wagon.

This is the right time for the teacher to introduce the *Add new procedure* command with the symbol + on the *Command panel*. By pressing the **+** button, children find out, that they are able to create a new command on the command panel that can keep for later use in a new program. The newly created *Draw wagon* command can be dragged into the program area just like the other commands of the enviroment. The way that the procedure creation is handled in EasyLogo, is so simple, and straightforward that allows most of the children to make their own procedures without nearly any help. Moreover, it helps them understand very intrusively the main purpose of the procedure mechanism in programming which is to create reusable pieces of code. Understanding the procedure mechanism is a major step for children’s programming mastering skills and the teacher should devote 2 or 3 teaching sessions to let them practice with it.

The last step of this activity series is the addition of some more extra wagons to the previously created train. Although children initially find it easy (and amusing) to add more wagons by dragging their new command into the program area, they start getting bored after the fifth or sixth wagon. It is this boredom that the teacher should take advantage of, in order to introduce the *Repeat* command. First he/she should urge them to find which part of their code is repeated. Although this may seem trivial for an adult, for a child it is not. Pattern recognition is a basic mathematical skill that has not yet been entirely achieved by children of younger ages. In fact, locating the part of the code that is repeated is much more difficult than learning how to use the *Repeat* command itself. Once they find the repeating part, it is fairly easy to insert it in a repeat structure and start playing with the “times” argument, which makes them very enthusiastic. In order the teacher to help them practice more with the use of the *Repeat* command, he/she can ask them to rewrite the basic square code of the previous sections or let them create interesting shapes (polygons and others) by just using this new command in an ad-hoc manner.

Fig. 4. The graphical result of the “Drawing the train” series of activities

3.6 Decorative motifs or let’s be creative

The last series of activities is meant to summarize all the previously acquired knowledge and skills and at the same time gives the opportunity to the children to explore new ways of expressing themselves and make their own creations. It is mostly based on situative learning rather than on problem solving techniques; children are not asked to solve specific problems posed by the teacher (which was a common practice in all previous series of activities) but to get themselves involved in a real – like, yet imaginary situation[4].

The teacher starts this series by showing a collection of geometrical decorative motifs that he/she can find on the Internet. Since the experiment took place in a Greek public school, it was found most appropriate to focus this demonstration on the most common ancient Greek motif called **meandros**. This is a geometrical motif that is still widely used in modern times and by other civilizations, as well. What is important at this step is the children to see that **meandros**, like most other geometrical motifs, is made by the repetition of a simple pattern. The teacher shows many examples of different forms of the **meandros** motif and asks the children to recognize which is the part that is repeated each time. Then he/she sets up the learning situation: the children should imagine that they are modern designers that they have to create modern decorative motifs using not a drawing tool but a programming environment like EasyLogo. They are advised to start by creating a simple motif and then produce more complex ones by repeating it in imaginative ways. There are no limitations or

[4] The activity was inspired by the African motifs scenario presented in [4]

restrictions to this process, as long as they use proper programming techniques and their own aesthetical taste.

This approach liberates creative power; kids feel free to experiment and play, try out their newly acquired programming skills and see where they can lead them. They work as usually in groups and the results of their work may considerably vary as shown in fig. 5.

If they are not aesthetically satisfied with the result of their work produced by programming means in EasyLogo, they can use a paint tool to add some more features to their motifs for instance to add more colors in certain areas of their motif, delete certain areas, etc. When finished, members of each group, have to think and discuss about the objects or surfaces that would be mostly suitable to be decorated with their motifs and send a relative post to the blog of their classroom. The post also contains the images of the motifs as well as some details about how they got inspired for these creations. Blog posts are a very nice way to share thoughts and experience with peers. By looking at the work of others, children have the chance to reflect about their own work as well, get more ideas, make comparisons, give their opinion and get highly involved to the activity that took place. The whole process of blogging is very fruitful and helps the children also get used to the idea of publicly share the result of their work.

4 Conclusions and further work

The work presented in this paper is not the result of a research experiment. The reaction of the children, although has not been methodologically measured, can be considered as positive and the results of their work is a good indication for that. For them the whole experiment was a complete novelty, not only because of the new knowledge domain that they were approaching but also because of a whole new set of principles of teaching that were applied in classroom. The whole learning experience was set up in such a way to urge them be confronted with innovation, creativity and a sense of freedom. Their self- esteem was also considerably boosted, by feeling proud of the quality of their work and by letting them take their own decisions. It's important to say at this point that very often during this experiment they kept on asking “what all this has to do with informatics?” And when they were asked (as an answer to this question) “what does informatics means to you?” they mentioned the use of the office tools (word, powerpoint, etc. and the Internet searching) although in fact they started to realize their idea about informatics is rather vague. This is something should be thoroughly considered by the researchers, as it might have both negative and positive results on the didactics of informatics at school.

Although kids considered the experience in general as positive, it would be rather difficult to prolong the activities time duration for more than 10-12 weeks. That possibly will have a negative impact on kids and it would reverse the results that they have achieved so far. In fact the younger the children, the more often one has to change tools and activities. But this is of course a hypothesis that needs to be carefully studied. As a proposal for further work, I would suggest that children should

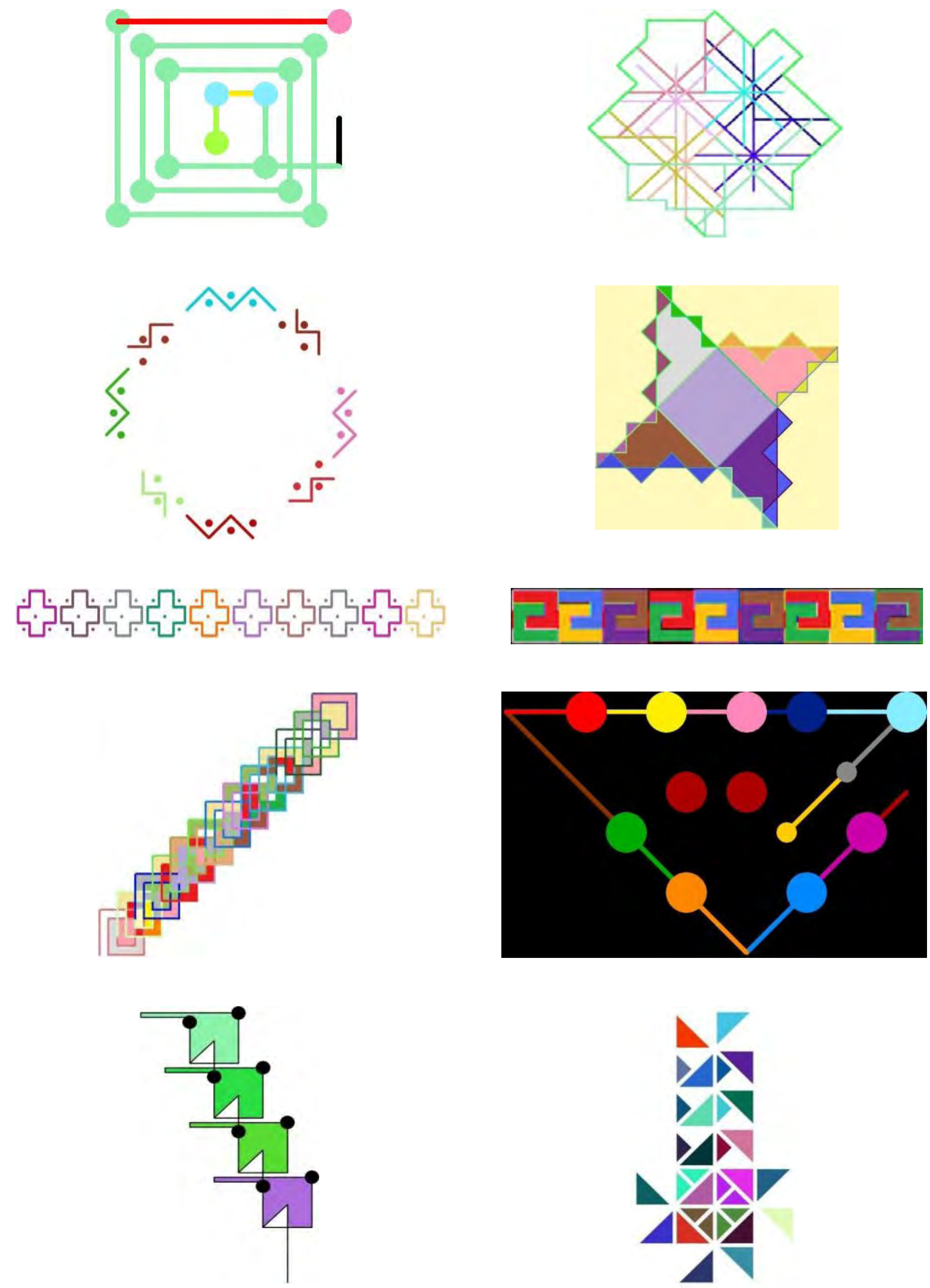


Fig. 5. Decorative motifs of young designers created with EasyLogo

be involved in more meaningful (for them) activities, like for instance the creation of an interactive board game. This kind of activities take advantage of EasyLogo's newly added animation (they were not present when this experiment took place).

These are more advanced activities in terms of algorithmic and programming skills but also in terms of rulemaking and design issues that they can serve as an advanced series of activities with the EasyLogo environment that follow the ones presented in this paper.

Finally in the future, it would be interesting to implement a systematic research that measures the educational result of the activities presented with regard to children's acquired skills, knowledge and attitudes towards programming.

References

1. Salanci, L.: EasyLogo – discovering basic programming concepts in a constructive manner. In: Constructionism 2010, European Logo Conference, Paris (2010).
2. Beetham, H.: An approach to learning activity design. In Beetham, H. and Sharpe, R. (eds.) Rethinking Pedagogy for a Digital Age pp. 26-40, Oxford, RoutledgeFalmer (2007).
3. Agostinho, S., Bennet S., Harper B., Lockyer, K.: Handbook of Research on Learning Design and Learning Objects: Issues, Applications and Technologies, Hersey, PA, IGI Global (eds) (2008).
4. Piaget, J. Science of educations and the Psychology of the child. New York: Orion Press (1970).
5. Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York (1980).
6. Papert, S.: The Children's machine, Rethinking School in the Age of the Computer. Basic Books, New York (1993).
7. Kafai, Y. & Resnick, M.: Constructionism in practice: Designing, thinking, and learning in a digital world. Mahwah, NJ: Lawrence Erlbaum Associates (eds.) (1996).
8. Brown, J.S., Collins, A., Duguid, P.: 'Situated cognition and the culture of learning' in Educational researcher, 1989, 18 (1): 32–42 (1989).
9. Van Ments, M.: The Effective Use of Role-Play: Practical Techniques for Improving *Learning*, 2nd Ed., Kogan Page, London (1999)
10. Adamson, R., Hoyles, C., Noss, R.: Rethinking the Microworld Idea. *Journal of* Educational Computing Research, 27 (2002).

Informatics Classes in Austria's Lower Secondary Schools – a Survey

Peter Smejkal[1], Monika Di Angelo[1]

[1] Vienna University of Technology, Institute for Computer Aided Automation, Treitlstrasse 1/183, A-1040 Vienna, Austria

monika.diangelo@tuwien.ac.at

Abstract. At present, the curriculum for Austria's lower secondary schools does not comprise informatics as a subject. This unfortunate situation led to a wide range of informatics classes both, optional and mandatory, at the lower secondary level, that were established solely through the initiative of the respective schools. These classes are offered in addition to the regular curriculum, which is enabled by the (small) autonomous space for curriculum design that the ministry grants its schools. In this work we present the current status of offers for informatics classes in Austria's lower secondary schools. In particular, we are interested in what kind of topics are offered, how numerous those offers are, and what the legal frame for this outside of curriculum classes is. This study is based on data that was collected from the schools in June-September 2010.

Keywords: Informatics classes, lower secondary school, survey

1 Introduction

The computer is a standard instrument in today's society which cannot be imagined without it. This fact is hardly reflected in the national school curricula, though: informatics lessons simply are not scheduled in the lower secondary school level in Austria. Computer classes only appear in the curricula for upper secondary schools. If a pupil decides to end the compulsory schooling with a polytechnic year, this is also a decision for no computer classes at all. Despite this unfortunate situation, there is a whole range of informatics classes at the lower secondary level, which are offered at school, outside the regular curriculum.

In this paper we seek for answers to the questions: "What is the current status of offers for informatics classes at the lower secondary level in Austria?" In particular, we are interested in a) what kind of topics are offered, b) how numerous those offers are, and c) the legal frame for outside of curriculum classes.

We first present a summary of recent studies with respect to informatics in Austria's schools. Then we briefly summarize the legal framework for classes which are offered in addition to the regular curriculum. This is followed by a presentation of data we collected from the lower secondary schools in Austria during June to September 2010. The gathered information has been evaluated in regard to the legal framework, to the single states, to the two types of lower secondary schools that exist in Austria, and to the subjects of the classes.

2 Informatics in Austria's Schools

Obligatory education standards for Austria in the area of informatics are not very numerous. There are simply none for the general-education secondary schools (AHS). Education standards were developed for the profession forming secondary schools (BHS) in cooperation with the ministry. [1] comments that here "uncharted waters are entered, because there are no approaches neither at the national nor at the international level that can be adapted to the Austrian situation".

After several years of fruitful discussion in the German-speaking countries the "Gesellschaft für Informatik" (GI) finally "enacted" education standards for informatics at school [9] in 2008 which unfortunately are not binding anywhere, yet. According to [4] there seems to be quite a resistance in Austria towards the further development of education standards since informatics is "already" a mandatory subject in year 9 in the AHS. In this respect, Austria is far from putting the GI education standards into action and no trend in this direction on the part of the responsible ministry has been located either.

An investigation in the state Kärnten [10] evaluated, whether the goals of the state standards for the lower secondary level (which were defined in the school year 2003/04) were accomplished and, after two years of informatics classes, how much knowledge was still available in the following year (year 7). The study confirmed the acceptance of the standards on the part of the teachers. However, the teachers were unsure, whether the standards could be achieved in a one hour per week class. In addition, it was confirmed that the informatics knowledge is lost, if the classes are not continued in the following years.

In the state Vorarlberg, the basic informatics knowledge of year 6 AHS pupils has been examined [5]. After completion of the one-year compulsory informatics class most pupils could handle the desktop, however approximately a quarter of all pupils had difficulties in copying files. Also, text editing was not accomplished without difficulties by approximately half of the pupils. Only a quarter of the pupils were able to handle standard presentation software without problems.

In a special issue of "CD Austria" [3] which is supported by the ministry (bm:ukk), it was remarked that although the range of informatics classes increased in the AHS within the last few years, the trend was more towards combined classes (like CAD) and the integration of industry certificates (like ECDL, Cisco, MS IT-Academy). Half of the investigated schools in the state Niederösterreich offered informatics classes.

Micheuz [7] evaluated the situation of the Austrian informatics education in year 9 via online questionnaires. The results show that due to the increasing school autonomy the range of informatics classes is in permanent development, leading to diverse school profiles. This results in a mismatch of knowledge among pupils of different schools. Even within the same school the pupils' informatics knowledge is very uneven, more than a third of all pupils reach the end of the lower secondary level without basic informatics training.

Another evaluation [8] deals with the autonomous informatics offers of the AHS. It is "based on elementary statistical data as well as the results of a national online survey" and comes to the conclusion: Although there are some offers for informatics classes at the lower and upper secondary level, these offers are highly unstructured, unclear and very heterogeneous. Due to the missing education standards and the unsatisfactory rooting in the curriculum, informatics education in Austria lacks clear structures.

3 Legal Framework

In Austria the ministry (bm:ukk) issues curricula and sets the legal frame for possible informatics classes. All following information is taken from the ministry's web site [2].

There are basically two possibilities for a school to establish informatics classes: on the one hand, a few weekly hours (max. 5) can be dedicated to a freely eligible school focus according to the "autonomous curriculum design" which the ministry grants its schools since the school year 2003/2004. These school autonomous subjects are obligatory for all pupils of this school (or this school branch), in this paper referred to as "*obligatory offers*". On the other hand, "*electives*" can be offered by the school in a limited range (they are financed by the ministry). These electives are in addition to the mandatory subjects and can be freely chosen (or not) by the pupils. Electives are not restricted to informatics topics of course, but rather cover a wide range of subjects (such as music, language, sports and many more).

The regulation of this design space for schools is found in the curricula [6] and reads: The curriculum can be adapted through school autonomous subjects "under consideration of the spatial, equipment and personnel conditions of the school" – in other words, at no extra costs. Furthermore it is explicitly noted that for the lower secondary level an informatics focus can be chosen "in the context of school autonomous curriculum regulations".

In Austria there are two types of schools at the lower secondary level: the AHS ("Gymnasien") runs from year 5 to 12, thus comprising both, the lower and upper secondary level. The HS ("Hauptschulen") runs from year 5 to 8 and only comprises the lower secondary level. Both operate with the same curriculum.

4 Data Collection and Evaluation

For data collection, firstly e-mail messages were sent out to all schools with a lower secondary level, the HS and the AHS. Secondly, since the number of replies to the e-mail messages was – as expected – very low, the web sites of all schools were searched for information on informatics classes in form of obligatory offers and electives. This method delivered by far the most data; there were hardly any schools without an appropriate web site. Thirdly, some schools were contacted by phone to further complete the data. In total, 1338 schools (271 AHS and 1067 HS) were investigated [11].

4.1 Results for obligatory offers

The number of schools with obligatory offers of informatics classes (in the context of the school autonomy) is already considerable in some states, although it varies widely over the states. Fig. 1 shows the percentage of schools with obligatory classes. The data was split up according to the two type of school of the lower secondary level the “Gymnasien” (left bar) and the “Hauptschulen” (middle bar). The average for the whole lower secondary (“Sek 1”, both school types put together) is indicated by the right bar. It should be noted, that fig. 1 contains no information on the weekly hours of the classes, nor on the year(s) in which the classes are held.

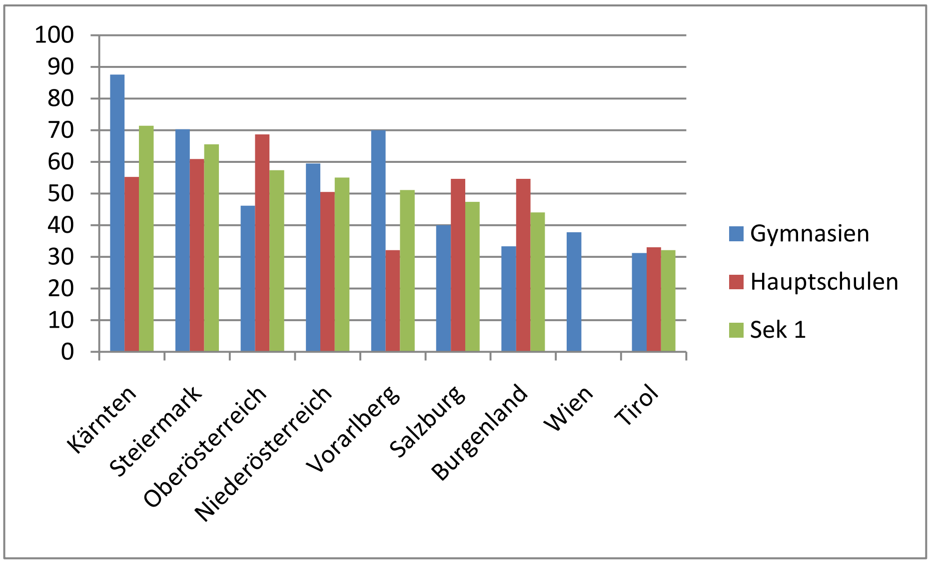


Fig. 1. Obligatory offers in percent of the schools.
“Gymnasien” and “Hauptschulen” in comparison, split up according to states.

With respect to the single states, a clear descent can be seen from leader Kärnten with a commendable average of about 71% of the schools with obligatory informatics offers in the complete lower secondary down to the bottom of the table marked by Tirol with about 32% of the schools which offer obligatory informatics classes. Especially with respect to the “Gymnasien”, Kärnten is in the fore with approx. 87%, and again Tirol forms the end with only just more than 31%. As for the “Hauptschulen”, Oberösterreich is in the lead with 68%, while Vorarlberg marks the end with about 32%.

Also, there is a visible difference between numbers for the “Gymnasien” and the “Hauptschulen” in most states, with no clear tendency.

4.2 Results for electives

Besides the obligatory offers there is a multitude of informatics electives, which are not attended by all pupils but rather only chosen by some of them. It is not obvious from the collected data as to how many pupils opt for the classes. The offer nevertheless is considerable and indicates that also the pupils show a strong interest in informatics topics. Fig. 2 depicts the range of electives in a tag cloud: Electives that are offered more often are represented in larger letters, more rare classes in smaller letters.

Fig. 2. Tag cloud of electives

Fig.3 depicts the percentage of schools which offer electives in informatics. The data was split up again into the “Gymnasien” (blue) and the “Hauptschulen” (red). Both school types put together are indicated in green. It should be noted, that fig. 3 contains no information on the weekly hours of the classes, or on the year(s) in which the classes are held, or on how many pupils chose those classes.

The overall numbers for the electives (roughly ranging between 35% and 55%) are slightly lower than those for the obligatory offers (between 32% and 71%).

As compared to the obligatory classes, a slightly different picture for the electives is revealed: Burgenland seems to prefer electives because it is located at the lead here. Three of the four leaders from the obligatory offers (Kärnten, Oberösterreich, Niederösterreich) are found on the front places again, while Tirol makes to the midfield. The capital Wien ranks at the back for both, the obligatory offers and the electives.

In some states (Oberösterreich, Vorarlberg) there is a huge difference (over 30%, over 20%) between the “Gymnasien” with a rather low average and the “Hauptschulen” with a quite high average, while in other states (Niederösterreich, Kärnten, Salzburg) numbers are almost equal. Again, no clear tendency can be seen with respect to school types.

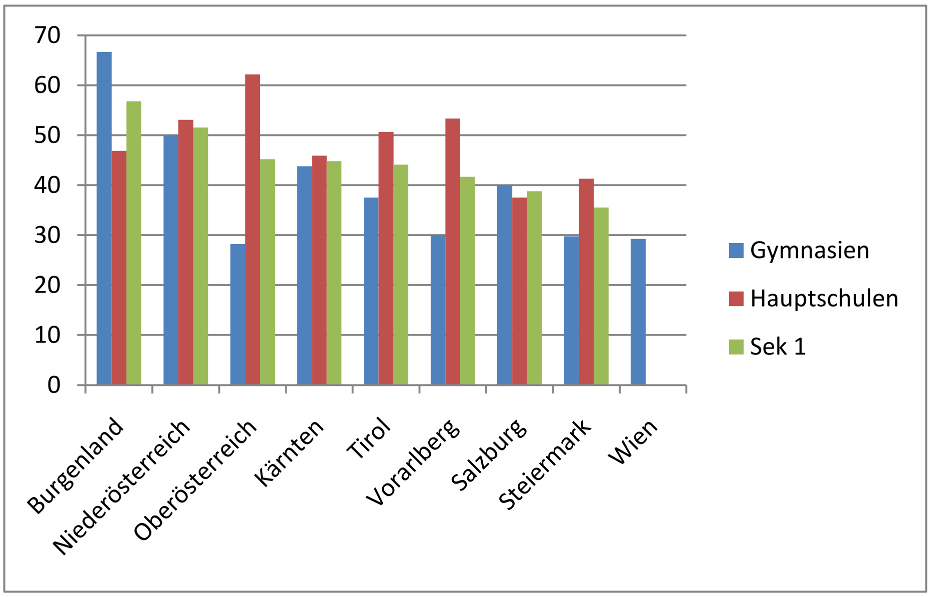


Fig. 3. Offers for "electives" in percent of the schools

4.3 Results for e-Learning

Another interesting aspect is the offer of e-learning, on the one hand to learn about its spread and gaining acceptance, on the other hand to find out any peculiar differences.

Fig. 4 depicts the percentage of schools that offer e-learning possibilities. Again, the data was split up into the “Gymnasien” (blue) and the “Hauptschulen” (red). Both school types put together are indicated in green.

The results are astonishing mainly by the fact that a serious difference was found regarding the school type which extends over (almost) all states: e-learning possibilities in “Gymnasien” can be found in 30% to 70%, while ”Hauptschulen” offer e-learning in less than 20%. They only sad exception is Wien where the “Gymnasien” approach the low average of the “Hauptschulen”.

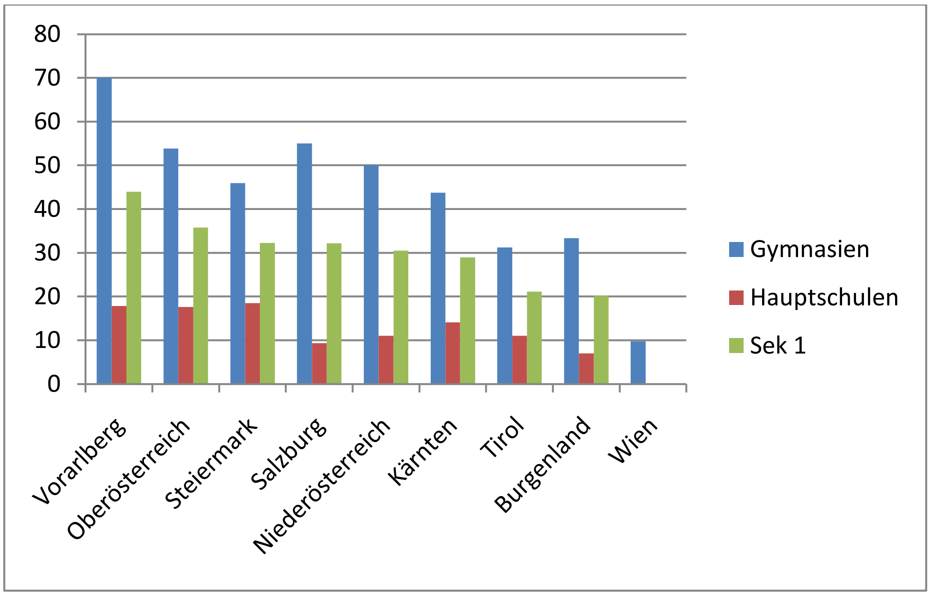


Fig. 4. Schools with e-learning possibilities (in percent)

A possible explanation for this data situation could be the circumstance that "Gymnasien" also offer the upper secondary level, while "Hauptschulen" are covering only the lower secondary level. One could therefore suspect that the effort to provide e-learning is mostly justified for the upper secondary level. In this respect it should be noted that the collected data does not indicate whether pupils from the lower secondary level actually use e-learning.

4.4 Frequent electives

The most frequent electives are in a clear agreement in (almost) all states: "informatics" (by far the most common elective and therefore basically equal with data from fig.3), "typewriting" (fig.5), and "word processing" (fig.6).

Fig. 5 depicts the percentage of schools that offer typewriting as an elective. Fig. 6 depicts the percentage of schools that offer word processing as an elective. Again, the data was split up into the "Gymnasien" (left bar) and the "Hauptschulen" (middle bar). Both school types put together are indicated by the right bar.

There is a clear trend, supported by all states, that typewriting is offered more often in the "Hauptschulen" than in the "Gymnasien", roughly twice as often on the average.

Word processing seems to show no clear tendency whatsoever, neither with respect to the school types, nor with respect to the states. Overall numbers are less than half of those from typewriting.

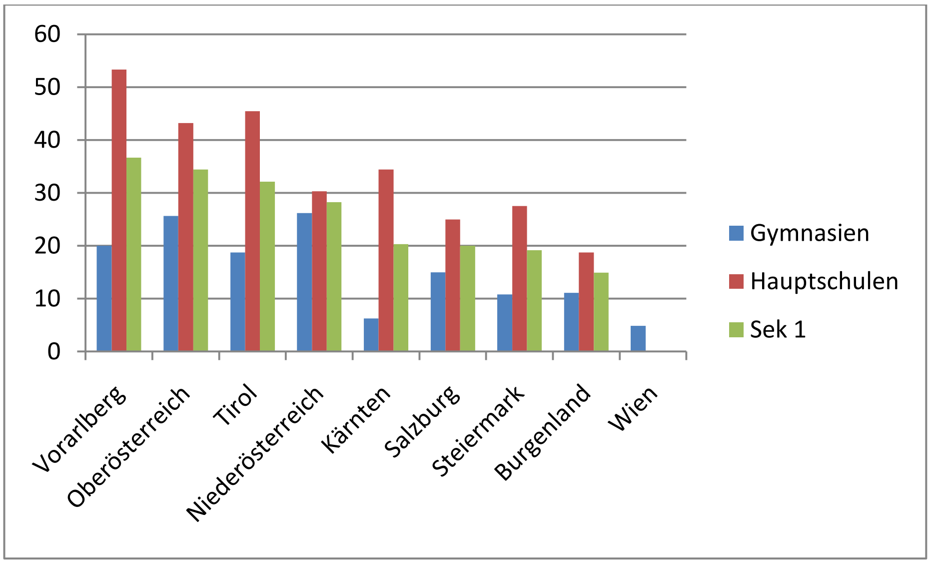


Fig. 5. Elective "typewriting" in all of Austria (in percent)

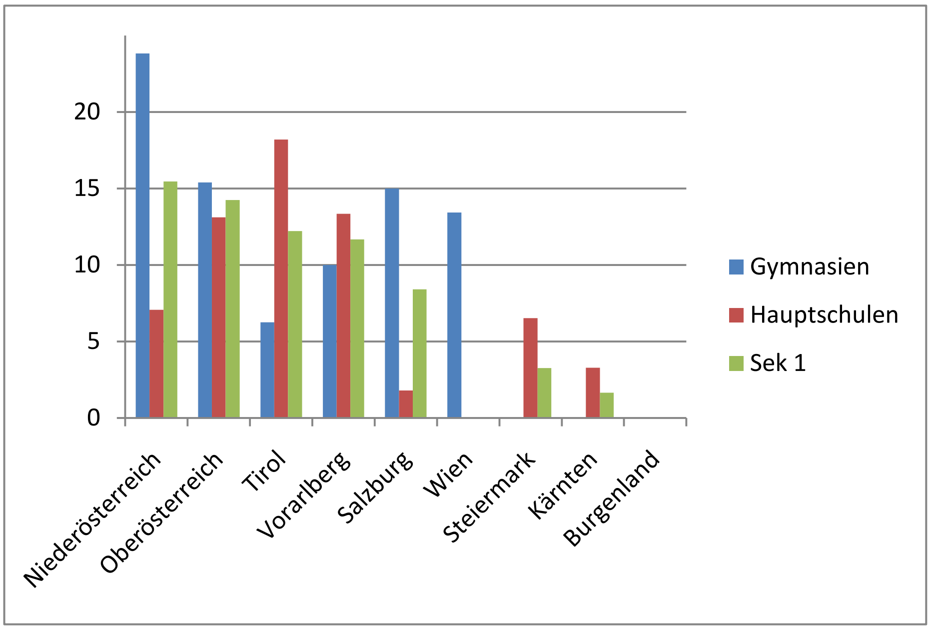


Fig. 6. Elective "word processing" in all of Austria (in percent)

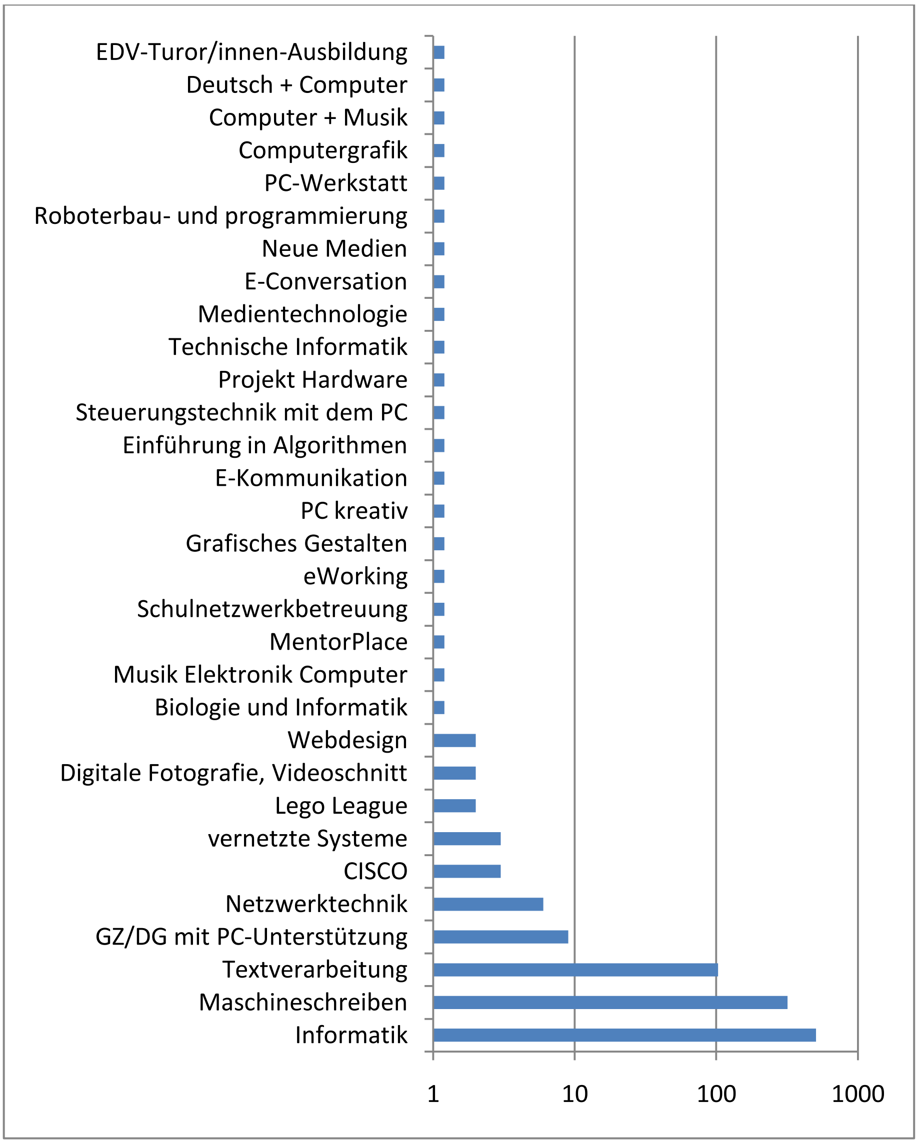


Fig. 7. List of most common electives in all of Austria (in order of frequency)

Fig. 7 lists the most common electives with their absolute numbers of occurrence in all of Austria. As can be seen, many classes are held in only a few schools. Some of them cover similar topics, some present content that overlaps with other classes.

“Informatik” (informatics) is by far the most common class, but that does not imply, that the same contents are presented in different schools. Every school is free to teach whatever they deem suitable. So, “Informatik” might actually cover contents of all other electives. This diversity with respect to the topics and contents of electives might seem like a wide choice, but actually rather reflects the lack of structure in informatics education.

5 Conclusion

Results show that in fact a large number of schools at the lower secondary level use their autonomous curriculum design space for informatics classes, be it as obligatory offers or electives. There were only a few schools which did not offer any informatics classes at all (8.9% Gymnasien, 25.2% Hauptschulen). Averaging over all of Austria, obligatory informatics classes can be found at 53.4% of the lower secondary schools, while informatics electives are offered at 47.3% of them.

These school initiatives are very laudable and welcome as they try to fill the gap between society's needs and the ministry's (non-)regulation. Still, this cannot guarantee a sound informatics education. On the lower secondary level, rooting of computer skills in the curriculum is still missing, binding informatics education standards are lacking. Despite all efforts, the current situation leads to a disastrous heterogeneity in the pupils' informatics knowledge at the end of the lower secondary level, which the upper secondary schools have to fight, inevitably.

References

1. Arbeitsgruppe „Bildungsstandards in Angewandter Informatik“: Angewandte Informatik: Berufsbildende Höhere Schulen – Das Kompetenzmodell (Version 1.18), 2008.
2. bm:ukk Bildung und Schulen (abgefragt am 21.1.2011) URL: http://www.bmukk.gv.at/schulen/index.xml
3. CD Austria Sonderheft 5/2005, CDA-Verlag, Linz, 2005.
4. Eder C.: Bildungsstandards in Informatik. Salzburg, 2007.
5. Egger H.: Grundkompetenzen in Informatik an Vorarlberger Gymnasien – IKT-Grundkenntnisse auf Schulstufe 6. In MNI-Fonds für Unterrichts- und Schulentwicklung S2 "Grundbildung und Standards", Frastanz, 2005.
6. bm:ukk Lehrpläne (abgefragt am 21.01.2011) URL: http://www.bmukk.gv.at/schulen/unterricht/lp/lp_abs.xml
7. Micheuz P.: Some Findings on Informatics Education in Austrian Academic Secondary Schools. Informatics in Education, 7(2), 221–236. 2008.
8. Micheuz P.: Zahlen, Daten und Fakten zum Informatikunterricht an den Gymnasien Österreichs, In INFOS 2009, 13. GI-Fachtagung Informatik und Schule 21.-24. September 2009 an der Freien Universität Berlin, S243-254, Klagenfurt, 2009.
9. Gesellschaft für Informatik: Grundsätze und Standards für die Informatik in der Schule. Beilage zu LOG IN, 28 (150/151), 2008.
10. Rohrer M., Micheuz P.: Evaluation des Informatikunterrichts in den 1./2. Klassen der AHS in Kärnten. In MNI-Fonds für Unterrichts- und Schulentwicklung S2 „Grundbildung und Standards“, Villach, 2006.
11. Smejkal P.: Informatik an Österreichs Schulen in der Sekundarstufe I. Diplomarbeit, TU Wien, 2010.

On Misconceptions and Implementing 'A Class Defines a Data Type'

Jan Vahrenhold

Faculty of Computer Science, Technische Universität Dortmund, 44227 Dortmund, Germany*

In the light of recent research on misconception related to the object-oriented concepts of 'object' and 'class', we discuss an approach to implement Hu's definition "A class defines a data type" and comment on how this approach can help avoiding the formation of misconceptions. Our approach is based upon bootstrapping Hu's definition, and the resulting teaching sequence follows Bruner's spiral model by covering entity-relationship models, abstract data types, and finally objects and classes. Also, the approach ties in with our concept of *braided teaching* that allows for contextualized teaching of computer science in (lower) secondary education.

Keywords: Object-orientation, misconceptions, ER-model, abstract data type

1 Introduction

In his keynote address at the 2007 *German Conference on Computer Science in Primary and Secondary Education*, Böstler [Böstler 2007] observes the dire state of first-year programming courses even though students leaving secondary schools tend to have had a broader exposure to computer science than in previous years. After a broad survey of methods taken to improve the situation, he concludes by saying "Are we teaching object-orientation in the wrong way? – Apparently, the answer is 'yes', but we don't know what we are doing wrong." [Böstler 2007, p. 9]. Among his observations is that novice computer science students are prone to misconceptions regarding object-orientation and that despite a surprisingly large body of literature on misconceptions, there are only "isolated are rare efforts" to avoid their formation [Böstler 2007, p. 18]. As a central challenge for computer science education, he identifies preparing high-school teachers with respect to teaching object-orientation.

Böstler's observations are confirmed by a study presented by Kohl and Romeike [Kohl and Romeike 2006]. The authors surveyed in-service teachers after a training workshop on object-orientation and report that a surprisingly high percentage of those willing to participate in the survey were unable to properly define central concepts in object-orientation (including the concepts of 'object' and 'class').

A large (if not the overwhelming) part of this can be attributed to the lack of proper (textbook) material. Hu presents his concern about this in an article aptly named "Just

* This work was supported in part by the Deutsche Telekom Stiftung (project *dort*MINT).

Say: ‘A Class Defines A Data Type’” [Hu 2008] where he pointed out “A class (as in an object-oriented programming language like Java) defines a data type. Yet this fact is typically not mentioned in CS1 textbooks.“ [Hu 2008, p. 19] A survey of high school textbooks (none listed in the references) similarly shows that the concepts of “object” and “class” are often defined only informally, in a circular way, or not at all.

The purpose of this article is to outline a (sequence within a) curriculum for secondary education targeted toward gradually introducing the concepts of ‘object’ and ‘class’. This sequence is designed to incorporate research on misconceptions and can serve as a *strand* in a *braided teaching* curriculum [Pasternak and Vahrenhold 2010].

1.1 Teaching Object-Orientation

The ever-ongoing debate of (whether and) when to teach object-orientation in introductory computer science courses (see, e.g., the SIGCSE 2005 special session “Resolved: Objects Early Has Failed” [Astrachan *et al.* 2005] for the case of introductory courses on college level) has led to a number of research findings and experience reports that is too large to be summarized in an appropriate manner. What is noticeable, however, is the fact that, to the best of the author’s knowledge, all courses considered on college level are oriented towards teaching object-oriented *programming* in the context of a traditional introductory computer science course sequence. As one consequence, the existing literature on how to introduce object-orientation and on how to avoid misconceptions is limited to possibly restructuring and rearranging the order in which the canonical subject matters are treated – see, e.g., [Adams 1996, Alphonce and Ventura 2002, Buck and Stucki 2000] – or to using tailored programming environments (e.g., [Dann *et al.* 2009, Kölling 2008], to name two prominent examples) or contexts [Forte and Guzdial 2005] to achieve the goals.

For the case of introductory courses in secondary education, there is a broad literature on (whether and) how to teach object-orientation (the reader is referred to the theses of Brinda [Brinda 2003] and Diethelm [Diethelm 2007] and the references therein), but there is no study on misconceptions or on how to address them.

A recent paper by Pedroni and Meyer [Pedroni and Meyer 2010] investigates the “high interrelatedness of concepts” in object-orientation and determines a “core group” of concepts starting from which object-oriented programming can be taught. This core group contains (among seven other concepts) the concepts of ‘object’ and ‘class’, so the understanding of (the difference) of these concepts is of utmost importance for any course teaching object-orientation.

1.2 Concepts and Misconceptions

Despite the above-mentioned debate of “Objects Early” versus “Objects Late” there seems to be a broad consensus about what constitutes the central concepts in introductory-level object-orientation – regardless on whether it is taught in secondary or tertiary education. Ragonis and Ben-Ari [Ragonis and Ben-Ari 2005] propose as the three “central meta-objectives” of teaching object-oriented programming that students should come to understand *modularity*, *encapsulation*, and *information hid-*

ing. Referring to Stroustrup's classification of approaches to software design, Adams names "classes" and "inheritance" as the "defining characteristics of object-oriented design" [Adams 1996]. Since Eckerdal and Thuné observe a variety of misconceptions even for the fundamental concepts of 'class' and 'object' [Eckerdal 2006, Eckerdal and Thuné 2005] and since these concepts have been identified to be a crucial starting point for any course teaching object-orientation [Pedroni and Meyer 2010], we restrict the following discussion to these concepts.

Several findings about misconceptions related to object-orientation and to the concepts of 'class' and 'object' have been reported [Culwin 1999, Eckerdal 2006, Eckerdal and Thuné 2005, Holland *et al.* 1997, Ragonis and Ben-Ari 2005, Sanders *et al.* 2008, Thuné and Eckerdal 2009].[1] Among the misconceptions encountered in early stages of the introduction to object-oriented design, the following three misconceptions are of particular importance (see, e.g., Culwin [Culwin 1999], Holland *et al.* [Holland *et al.* 1997], and Ragonis and Ben-Ari [Ragonis and Ben-Ari 2005]):

- Misconception A: Students may conflate 'object' and 'class' if only a single instance of a class is used.
- Misconception B: Students may confuse the identity with the value of a "name" instance attribute.
- Misconception C: Students may consider two objects to be identical if the values of each of their instance attributes are equal (or, vice versa, two objects of the same class are not allowed to have equal values for their instance attributes).

The importance of focusing on object-oriented concepts instead of the syntax of a particular programming language has been stressed by Alphonce and Ventura [Alphonce and Ventura 2002], and Bennedsen and Casparsen [Bennedsen and Casparsen 2004] follow up on this observation and suggest interlacing (UML-based) modeling and implementation in a spiral fashion.

Kölling and Rosenberg [Kölling and Rosenberg 2001] list the following set of "very important concepts" in object-orientation (labels added by the author):

- Concept A: "Objects are created from classes."
- Concept B: "All objects of the same class have the same structures; objects of other classes have different structure."
- Concept C: "Many objects can be created from a single class."
- Concept D: "The state (variable values) of each object is different."
- Concept E: "Objects have operations (methods) which can be invoked."
- Concept F: "Methods may have parameters and results."

Concepts A, B, and C are only suited to address Misconception A if there is a properly defined class, and this in turn requires the teacher to present the students with such a class. Unfortunately – as noted by Hu [Hu 2008] – many textbooks lack a proper definition of 'class' and this in turn can lead to the conflation of 'object' and

[1] As mentioned, research on misconceptions has been restricted to college-level courses so far. Thus, no results related to secondary education can be built upon, but it seems safe to assume that similar misconceptions may be encountered at least in upper secondary education.

'class' as observed by Kohl and Romeike [Kohl and Romeike 2006] even on teacher training level. Thus, we propose a course sequence that implements Hu's definition:

Definition 1 [Hu 2008] A *class* defines a data type.

2 Bootstrapping the Definition

In this section, we bootstrap Hu's definition (Definition 1) and show how the resulting sequence of concepts can be implemented in secondary computer science education. Hu's concise, yet appealing definition recurs to the concept of a data type:

Definition 2 [Chang *et al.* 2001, p. 116] A *data type* [is a] […] set of values with [a] set of operations.

In the light of the concerns raised by Buck and Stucki [Buck and Stucki 2000] regarding the students' exposure to complexity and structure, we bootstrap this definition in a two-fold way by separately considering "values" and "operations". For the first part, we resort to entity-relationship diagrams and the relational data model in databases, and for the second part we consider the specification of abstract data types.

As a guideline for the bootstrapping process, we aim at obtaining an understanding of the following definition:

Definition 3 An (correctly modeled) *object* models an intellectually delimited entity with a state and (possibly state-dependent) behavior.[2]

The concept of an 'entity' has been defined by Chen in the context of the entity-relationship model:

Definition 4 [Chen 1976, p. 10f.] An *entity* is a "thing" which can be distinctly identified. […] Entities are classified into different *entity sets*. […] If we know that an entity is in [some particular] entity set […], then we know that it has the properties common to the other entities in [this] entity set.

As we will outline in Section 3.2, we can lay the groundwork for addressing Misconceptions B and C (identity versus state) using a straightforward sequence of discussing entity-relationship diagrams and the relational model. To complete the preparations for Hu's definition, the sequence then covers abstract data types:

Definition 5 An *abstract data type* is a data type for which neither the representation of the values nor the implementation of the operations is prescribed.

We note in passing that a recent paper by Ragonis [Ragonis 2010] also connects abstract data types with object-orientation. Whereas we discuss abstract data types as *preparation* for introducing object-orientation, Ragonis' approach is to use the design and implementation of the abstract data type `SET` as an example and exercise for *summarizing* object-oriented principles taught before.

---

[2] This definition builds upon Echtle and Goedicke's definition "An (correctly modeled) object models an intellectually delimited entity with all its properties." [Echtle and Goedicke, p. 84]

To summarize, we propose to first teach *entities* that have a state but no state-dependent behavior, then to teach *abstract data types* that describe behavior but do not have an explicit representation of the state, and finally to converge on *objects* that have both a state and a (possibly state-dependent) behavior. Each of these topics is sufficiently unique such that the sequence can be taught in three separate steps (possibly interleaved by other topics), and this allows an embedding into a spiral curriculum [Bruner 1960] or a curriculum based upon braided teaching (see Section 3.1).

3 Design of the Sequence of Subjects Taught

The bootstrapping of Hu's definition (see Section 2) resulted in three major topics, namely "Entity-relationship diagrams", "Abstract data types", and "Objects and classes". These topics are discussed in the Section 3.2 with a focus on the preparation of the concepts of 'class' and 'object' and on how to avoid misconceptions.

3.1 Proof of Feasibility: Integration into the Curriculum

As an important prerequisite, we have to argue that it is indeed possible, i.e., allowed for by curricular restrictions, to teach the above subjects in secondary education. Since a recent report [Ericson *et al.* 2008] shows the diversity (in terms of both breadth and depth) of (curricula and thus) teacher certification across the world, we restrict ourselves to three examples. In the author's country of residence (Germany), the curricula for upper secondary education in most federal states allow for covering all three topics. In the case of lower secondary education in Germany, both information systems and object-orientation are explicitly mentioned in the "Educational Standards for Computer Science in Lower Secondary Education" [Brinda *et al.* 2009]. Unfortunately, computer science is actually taught with full credit in only a few federal states, but the approved textbooks for these states contain material for teaching both information systems and object-orientation. Admittedly, the topic of abstract data types (when taught in breadth *and* depth) is too demanding for lower secondary education, but – as outlined in the following – the sequence does not rely on a (minimal) set of equations describing the semantics. The only characteristics required are the interface defined and the absence of a concrete representation of the state.

It is worthwhile noting that all curricula (and the corresponding textbooks) follow an "Object-Orientation First" or "Programming First" approach and thus cover object-orientation prior to modeling in a database context. Ironically, one textbook links these two subjects by stating that database systems can used to model relationships between *objects* (and not: *entities*); this textbook, however, starts out presenting objects that are simply containers for attributes and do not have any behavior other than accessing these attributes. In this case, students can fall for yet another misconception "if the data aspect of objects is overemphasized at the expense of the behavioral aspect" [Eckerdal and Thuné 2005, p. 92] – see also Concepts E and F (Section 1.2).

Secondly, ACM's K-12 curriculum for computer science [Tucker *et al.* 2004] has a very strong focus on algorithmics and computational thinking and, conversely, a relatively minor component dedicated to object-oriented modeling and programming. Nonetheless, Level I covers (using) databases, Level II includes "data types" and "tools for expressing design", and Level III includes "objects and classes".

Finally, the Israeli curriculum for secondary education computer science is very much oriented towards algorithmics as well and follows an "Objects Second" approach – see, e.g., [Hazzan *et al.* 2008]. This curriculum explicitly mentions abstract data types as a mandatory component and offers an elective component that can be used to cover (management) information systems "both in theory and practice".

We conclude with a more general didactic comment: In a previous paper, we developed the *braided teaching* approach to organizing items to be taught in secondary education computer science. This approach simultaneously allows for following a spiral curriculum and for teaching contextualized computer science by working along what we call *strands*. Our proposed sequence satisfies the criteria for being a *strand*:

Definition 6 [Pasternak and Vahrenhold 2010] A *strand* is a sequence of items addressed in class that satisfies the following criteria:

a) The items can be assigned to a well-defined subject matter (by their structure or their content).
b) The subject matter is identifiable and recognizable to the students throughout the sequence.
c) The subject matter is being presented from more than one point of view or embedded in more than one context.
d) The sequence of items is addressed in more than one teaching unit.

3.2 Proof of Concept: A Course for Non-Majors

In this section, we detail the teaching units to be taught in our proposed sequence. As a proof of concept, we do so in terms of (part of) an introductory computer science course for college students majoring in humanities and social sciences; this course has been taught by the author over the past three years. On one hand, the student population can be seen as not too different (in terms of interest, previous knowledge, and possibly also aptitude) from upper-level high-school students (assuming that computer science is taught as part of general education and not in a special-focus course), on the other hand, the classroom setting in high school and college is rather different. This, and the fact that the courses taught had an audience of approximately ten students each, precludes a quantitative analysis and any attempt to transfer the results. What can be done, though, is to outline the teaching units and their relative lengths.

3.2.1 Information Systems (7 double sessions)

Information systems are so ubiquitous in students' everyday lives that they serve as a well-motivated starting point for the teaching sequence.[3] After a brief introduction

[3] Due to this didactic aspect and the fact that teaching information systems in secondary education computer science additionally helps avoiding the misconception that information sys-

of databases versus file-based data organization and the layered architecture, the entity-relationship (ER) model is introduced with a strong focus on precise terminology on diagram level (i.e. "a member of the entity set *X* can be in a *Y*-relation with a member of the entity set *Z*" instead of "an *X* is in a *Y*-relation with a *Z*"). This motivates the introduction of cardinalities using both Chen's notation and the (*min,max*)-notation. Only then, role names and (key) attributes are added to the diagram. At this point, we mention that there may be different entity-relationship models, e.g. using attributes instead of relations, seemingly representing one and the same real-world situation and thus addressing analysis, discussion, and interpretation competencies.

In the second unit of the information systems part of the sequence, we discuss the relational model. This unit introduces a concrete representation of the entities modeled; furthermore, the relational algebra serves as a starting point for designing algorithms that use the state of an object for making decisions and for revisiting the Boolean logic. The third unit of the information systems part of the sequence deals with SQL and includes hands-on lab sessions. While the transition from the relational algebra to SQL is straightforward, it still raises two issues crucial for introducing objects and classes: object identity (in contrast to the relational algebra, a SQL expression does not result in the removal of tuples with identical attributes unless explicitly stated otherwise) and the need for defining and using (elementary) data types.

Table 1 summarizes how concepts in object orientation are prepared for by introducing concept from entity-relationship diagrams. The table can be extended in an obvious way by adding multivalued attributes or `is-a` relations, but we propose to defer these topics until objects and classes have been introduced and implemented. This not only facilitates the introduction of (simple) objects but also allows for revisiting entity-relationship diagrams in a spiral manner after concepts from object-orientation have been introduced.

ER Concept	OO Concept
Entity set	Class
Entity	Object
Key attribute(s)	Object identity
An entity is defined by the values of its attributes.	The state of an object is defined by the values of its attributes.
Relations (with cardinalities)	Associations (with cardinalities)
Relation versus attribute	Association versus composition

Table 1. Preparing OO concepts by ER concepts (note that these concepts are not identical).

Addressing Misconceptions

The most important ingredient for avoiding misconceptions is to use precise terminology. For example, conflating entities and entity sets, e.g. by saying "an *X*" instead of "a member of entity set *X*", needs to be avoided at all costs. By introducing multi-

tems are "some part of ICT" and that it is sufficient to have learned to use some spreadsheet software, Rohland [Rohland 2009] proposes to intensify teacher training in this area.

attribute keys such as (*name*, *given_name*) we imply that a “name” attribute usually is not sufficient to identify an object. One of the common misconceptions noted by Holland *et al.* [Holland *et al.* 1997] is the object/variable conflation that occurs if a class has either exactly one instance variable or only instance variables of exactly one type. This misconception is avoided from scratch since the real-world examples commonly used for entity-relationship diagrams (e.g. student—teacher—class / customer—company—product) feature a variety of attributes of different types. Finally, we lay the groundwork for (compiler-given) artificial keys by pointing out that the social security number is unique (thus a key) by design but has nothing to do with almost any aspect (“real-world attribute”) of the particular person being modeled.

3.2.2 Abstract Data Types (3 double session)

A rather short section of the sequence is devoted to introducing the concept of abstract data types. This section is motivated by the observation that entity-relationship diagrams are suited to model attributes and relations but lack the power to describe behavior. The syllabus covers the canonical abstract data types `PAIR`, `LIST`, `STACK`, `QUEUE`, `DICTIONARY` and `PRIORITY_QUEUE` but in order to not to overload this section, a formal specification of the semantics is given for the first four abstract data types only. The main focus of this section is on the functional description of the signature that lays the groundwork for the message-passing paradigm of communication between objects. Also, the concept of a constructor (reported to be the source of several types of misconceptions [Ragonis and Ben-Ari 2005]) can be prepared for.

Addressing Misconceptions

In the light of Misconception B (identity versus “name” attribute), a special emphasis is put on the fact that the name of an abstract data type operation is completely unrelated to the semantics of that operation. At this point, it is instructive to ask an international student to translate, say, the name of the `enqueue` operation provided by the abstract data type `QUEUE` into a *random* term from his or her native language and to consistently use this term throughout the remainder of this class session.

3.2.3 Objects and Classes (6 double sessions)

We start by summarizing what the course has introduced so far: the description of an entity by its attributes and the description of behavior in the form of abstract data types. After revisiting Definition 3 (“An (correctly modeled) *object* models a intellectually delimited entity with a state and (possibly state-dependent) behavior.”) the YACHT class is used to introduce state and state-dependent behavior.

Excursion: The YACHT Class

During the “Nifty Objects for CS0 and CS1” special session of SIGCSE 2008 [Hummel *et al.* 2008], Hummel proposed to use the YACHT class as an introductory example. Since the author has been unable to find a published description of this example, the class is described for the readers' convenience: A (sailing) yacht is a watercraft that, in a simplified form, can be modeled by its name, the number of cabins, the number of crewmembers, and its waterline length. An additional beauty of this example lies in the fact that different types of yachts are referred to as "classes".[4]

Since all yachts in the same class have the same waterline length, this attribute – even if introduced as an instance attribute – can be explained easily as a class attribute. The number of crewmembers and the number of cabins are related and upper- and lower-bounded in an obvious way, and this relation leads to the introduction of state-dependent behavior, e.g. for a "boarding" method. Finally, the waterline length is an attribute that is – at least for the owner – of little interest *per se* but serves as the basis for computing the maximum hull speed, i.e., a derived attribute.

Object-Oriented Concepts

After the initial discussion of the YACHT class, the course covers message passing (building upon the concepts from abstract data type) and puts emphasis on the concept of encapsulation and information hiding. This motivates the distinction between public and private visibility for methods and attributes and leads to the introduction of UML class and object diagrams. Only at this point, the first contact with an object-oriented programming language is made. We found that – for the reasons given by Kölling and Rosenberg [Kölling and Rosenberg 2001] – the BlueJ environment is well suited for this task. Topics covered in the programming-related part of this section are expressions, variables, control structures, simple data types, functions, methods, and loops (which may or may not have been taught before). At the end of the course, abstract classes and inheritance (realizing the `is-a` concept) are introduced.

Addressing Misconceptions

Students – assuming that they have ever been to a lake or a port – are so familiar with seeing more than one yacht that it is virtually impossible to fall for Misconception A (conflating 'object' and 'class'). To address Misconception C, it is sufficient to consider the following tiny example: Assume there are two yachts (see Table 2), one of which sails in the Baltic Sea and one of which sails in the Adriatic Sea. Since both yachts belong to different owners, none of the owners know about the other yacht. What happens if the owner of the 'Flavia' decides to re-christen her 'Octavia'?

Name	Cabins	Crewmembers	Waterline length
Flavia	4	4	8
Octavia	4	4	8

Table 2. Yachts represented by their attributes.

According to the relational model (note that the resemblance of the display of the attributes to a relation is by design) multiple tuples with identical attribute values are

[4] Unfortunately, the beauty also lies in the eye of the beholder who is required to have at least a vague understanding of sailing boats to fully appreciate the example.

not allowed. Nobody, however, expects one of the yachts to vanish (or both yachts to merge into one) in the very moment a new name is painted on the hull. The only way out as to understand that even though the yachts now have the same state (as induced by their attribute values) yet still different identities.

We note that the YACHT class can be augmented to also illustrate more advanced concepts in object-orientation, such as inheritance (generalizing to watercraft, specializing to different classes of keelboats), aggregation (changing the overall sail area by setting different types of sails), interfaces/multiple inheritance (adding an inboard motor), the *decorator* design pattern (adding a outboard motor), and so on.

4 Conclusions

In this paper, we have proposed on a sequence of subject matters in secondary computer science education with a focus on conceptual modeling. Our approach is in line with a “Modeling before Programming”-like approach advocated by, e.g., Diethelm [Diethelm 2007], but differs significantly from previous approaches by a deliberately slow progression towards concepts in object-orientation and a strong focus on avoiding the formation of misconceptions. The defining characteristic of this sequence is to use concepts from (relational) databases and abstract data types to facilitate the introduction of the object-oriented concepts of ‘class’ and ‘object’. The separation of these steps allows for integration into a spiral curriculum resp. a curriculum based on braided teaching. It remains an open research question to investigate the long-term effects using qualitative or mixed methods and to investigate which impact (if any) such a course design might have on the students’ programming proficiency.

5 References

[Adams 1996] J. C. Adams. Object-centered design: A five-phase introduction to object-oriented programming in CS1-2. In J. Impagliazzo, E. S. Adams, and K. J. Klee, eds., *Proc. 27th SIGCSE Techn. Symp. on Computer Science Education*, pp. 78–82, 1996.

[Alphonce and Ventura 2002] C. Alphonce and P. Ventura. Object orientation in CS1-CS2 by design. In M. E. Caspersen, D. Joyce, D. Goelman, and I. Utting, eds., *Proc. 7th SIGCSE Conf. on Innovation and Technology in Computer Science Education,* pp. 70–74, 2002.

[Astrachan *et al.* 2005] O. Astrachan, K. Bruce, E. Koffman, M. Kölling, and S. Reges. Resolved: Objects Early has failed. In W. Dann, T. Naps, P. Tymann, D. Baldwin, and J. D. Dougherty, eds., *Proc. 36th SIGCSE Techn. Symp. on Computer Science Education*, pp. 451–452, 2005.

[Bennedsen and Caspersen 2004] J. Bennedsen and M. E. Caspersen. Programming in context – a model-first approach to CS1. In D. Joyce, D. Knox, W. Dann, and T. L. Naps, eds., *Proc. 35th SIGCSE Techn. Symp. on Computer Science Education*, pp. 477–481, 2004.

[Bennedsen *et al.* 2008] J. Bennedsen, M. E. Caspersen, and M. Kölling, eds.. *Reflections of the Teaching of Programming*, LNCS 4821, Springer, Berlin, 2008.

[Böstler 2007] J. Böstler. Objektorientiertes Programmieren – Machen wir irgendwas falsch? (*Object-Oriented Programming – Are we doing anything the wrong way?*). In: S. Schubert, ed., *Didaktik der Informatik in Theorie und Praxis* (INFOS 2007), pp. 9–20, Gesellschaft für Informatik, Bonn, 2007. In German.

[Brinda 2003] T. Brinda. *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht in der Sekundarstufe II* (*A didactic system for object-oriented modeling in upper secondary computer science education*), PhD Thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2003. In German.

[Brinda *et al.* 2009] T. Brinda, H. Puhlmann, and C. Schulte. Bridging ICT and CS: Educational standards for computer science in lower secondary education. In: *Proc.14th SIGCSE Conf. on Innovation and Technology in Computer Science Education*, pp. 288–292, 2009.

[Bruner 1960] J. S. Bruner. *The Process of Education*, Cambridge, MA, 1960.

[Buck and Stucki 2000] D. Buck and D. J. Stucki. Design early considered harmful: Graduated exposure to complexity and structure based on levels of cognitive development. In L. B. Cassel, N. B. Dale, H. M. Walker, and S. M. Haller, eds., *Proc.31st SIGCSE Techn. Symp. on Computer Science Education*, pp. 75–79, 2000.

[Chang *et al.* 2001] C. Chang, P. J. Denning, J. J. Cross II, G. Engel, R. Sloan, D. Carver, R. Eckhouse, W. King, F. Lau, S. Mengel, P. Srimani, E. Roberts, R. Shackelford, R. Austing, C. F. Cover, G. Davies, A. McGettrick, G. M. Schneider, and U. Wolz. Computing Curricula 2001: Computer Science. IEEE Computer Society/Association for Computing Machinery, 2001. Online: `http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/cc2001.pdf` [Accessed: 2007-11-09].

[Chen 1976] P. P. Chen. The entity-relationship-model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976.

[Culwin 1999] F. Culwin. Object imperatives! In J. Prey and R. E. Noonan, eds., *Proc.30th SIGCSE Techn. Symp. on Computer Science Education*, pp. 31–36, 1999.

[Dann *et al.* 2009] W. P. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Pearson Education, London, UK, second edition, 2009.

[Diethelm 2007] I. Diethelm. "*Strictly models and objects first" – Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht* ("*Strictly models and objects first" – Design and methodology for teaching object-oriented modeling in secondary computer science education*), PhD Thesis, Department of Electrical Engineering and Computer Science, University of Kassel, Germany, 2007. In German.

[Echtle and Goedicke 2000] K. Echtle and M. Goedicke. *Lehrbuch der Programmierung mit Java* (*Teaching Programming in Java*), dpunkt.verlag, 2000. In German.

[Eckerdal 2006] A. Eckerdal. *Novice Programming Students' Learning of Concepts and Practise*. PhD thesis, Faculty of Science and Technology, University of Uppsala, 2006.

[Eckerdal and Thuné 2005] A. Eckerdal and M. Thuné. Novice java programmers' conceptions of "object" and "class", and variation theory. In J. Cunha, W. Fleischman, V. K. Proulx, and J. Lourenco, eds., *Proc. 10th SIGCSE Conf. on Innovation and Technology in Computer Science Education*, pp. 89–93, 2005.

[Ericson *et al.* 2008] B. Ericson, M. Arnomi, J. Gal-Ezer, D. Seehorn, C. Stephenson, and F. Trees. *Ensuring exemplary teaching in an essential discipline: Addressing the crisis in computer science teacher certification. Final report of the CSTA teacher certification task force*, Computer Science Teachers Association, New York, 2008.

[Forte and Guzdial 2005] A. Forte and M. Guzdial. Motivation and nonmajors in computer science: Identifying discrete audiences for introductory courses. *IEEE Transactions on Education*, 48(2):248–253, May 2005.

[Hazzan *et al.* 2008] O. Hazzan, J. Gal-Ezer, and L. Blum. A model for high school computer science education: The four key elements that make it! In J. D. Dougherty, S. H. Rodger, S.

Fitzgerald, and M. Guzdial, eds., *Proc. 39th SIGCSE Techn. Symp. on Computer Science Education*, pp. 281–285, 2008.

[Holland *et al.* 1997] S. Holland, R. Griffiths, and M. Woodmann. Avoiding object misconceptions. In C. M. White, C. Erickson, B. J. Klein, and J. E. Miller, eds., *Proc.28th SIGCSE Techn. Symp. on Computer Science Education*, pp. 131–134, 1997.

[Hu 2008] C. Hu. Just say 'A class defines a data type'. *Communications of the ACM*, 51(3):19–21, Mar. 2008.

[Hummel *et al.* 2008] J. Hummel, C. Alphonce, J. Bergin, M. E. Caspersen, S. Hansen, J. E. Heliotis, and M. Kölling. Nifty objects for CS0 and CS1. In J. D. Dougherty, S. H. Rodger, S. Fitzgerald, and M. Guzdial, eds., *Proc. 39th SIGCSE Techn. Symp. on Computer Science Education*, pp. 437–438, 2008.

[Kohl and Romeike 2006] L. Kohl and R. Romeike. Aktueller Stand der Objektorientierung bei Informatiklehrerinnen und -lehrern (*State of object-orientation today for Computer Science high school teachers*). In: P. Forbrig, G. Siegel, and M. Schneider, eds.: *HDI 2006: Hochschuldidaktik der Informatik*, pp. 63–75, Gesellschaft für Informatik, Bonn, 2006. In German.

[Kölling 2008] M. Kölling. Using BlueJ to introduce programming. In [Bennedsen *et al*. 2008], pp. 98–115.

[Kölling and Rosenberg 2001] M. Kölling and J. Rosenberg. Guidelines for teaching object orientation. In S. Fincher, B. Klein, F. Culwin, and M. McCracken, eds., *Proc.6th SIGCSE Conf. on Innovation and Technology in Computer Science Education*, pp. 33–36, 2001.

[Pasternak and Vahrenhold 2010] A. Pasternak and J. Vahrenhold. Braided teaching in secondary CS education: Contexts, continuity, and the role of programming. In: T. Cortina and E. Walker, eds.: *Proc. 41st SIGCSE Techn. Symp. on Computer Science Education*, pp. 204–208, 2010.

[Pedroni and Meyer 2010] M. Pedroni and B. Meyer. Object-oriented modeling of object-oriented concepts. In: J. Hromkovič, R. Královič, and J. Vahrenhold, eds.: *Proc. 4th International Conference on Informatics in Secondary Schools – Evolution and Perspective* (ISSEP 2010), LNCS 5941, pp. 155–169, Springer, Berlin, 2010.

[Ragonis 2010] N. Ragonis. A pedagogical approach to discussing fundamental object-oriented programming principles using the ADT SET. *ACM Inroads*, 1(2):42–52, June 2010.

[Ragonis and Ben-Ari 2005] N. Ragonis and M. Ben-Ari. A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education*, 15(3):203–221, Sept. 2005.

[Rohland 2009] H. Rohland. Von der ersten Entitätsklasse zum Webshop – Datenbanken in allgemein bildenden Schulen (*From the first entity class* [sic!] *to a web-based shop – databases in seondary education*). In: B. Koerber, ed., *Zukunft braucht Herkunft, 25 Jahre »INFOS – Informatik und Schule«* (INFOS 2007), pp. 9–20, Gesellschaft für Informatik, Bonn, 2007. In German.

[Sanders *et al.* 2008] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, L. Thomas, and C. Zander. Student understanding of object-oriented programming as expressed in concept maps. In J. D. Dougherty, S. H. Rodger, S. Fitzgerald, and M. Guzdial, eds., *Proc. 39th SIGCSE Techn. Symp. on Computer Science Education*, pp. 332–336, 2008.

[Thuné and Eckerdal 2009] M. Thuné and A. Eckerdal. Variation theory applied to students' conceptions of computer programming. *European Journal of Engineering Education*, 34(4):339–347, Sept. 2009.

[Tucker *et al.* 2004] A. Tucker, F. Deek, J. Jones, D. McCowan, C. Stephenson, and A. Verno. *A model curriculum for K-12 computer science: Final report of the ACM K-12 task force curriculum committee*, Compter Science Teachers Association, New York, 2004.

The formation of ICT-competence in primary school in the new Federal State standard of primary education

Valery Vardanyan, Tatiana Rudchenko

Dorodnicyn Computing Centre of Russian Academy of Sciences, 40 Vavilova, Moscow, Russia

rudchenko1@yandex.ru , vardanyan47@yandex.ru

Abstract. The article focuses on the ways of forming ICT-competence in primary school as part of the "Program of the Universal Educational Activities Formation" in accordance with the new Federal State standard of primary education. The authors discuss in detail the elements of ICT-competence which should preferably be developed in each subject sphere of the curricculum. Of the four main forms of students' work with ICT means application the authors emphasise project activities and give some examples of how this form can be applied in various subject spheres.

Key words: ICT-competence formation, computer literacy, ICT means application, project activities, Federal State standard of primary education.

In accordance with a new Federal State Education Standard (FSES) in the sphere of primary schooling the main emphasis is laid upon metasubject educational results. The formation of the universal educational activities is the main result of the Standard realization. ICT-competence basics (not only basic skills of using a wide spectrum of information and communication technologies (ICT), but formation of a conscientious and competent approach to the choice and implementation of ICT means) are part of these metasubject results in mastering the program of primary education and an obligatory component of the program aimed at formation of the universal educational activities. The FSES "Program of the Universal Educational Activities Formation" contains the "Subprogram of ICT-competence Formation". This subprogram describes ICT-competence elements that are a part of any universal educational activities and of the corresponding technological skills, which are being shaped in the course of different subjects' acquisition. Therefore, the students' ICT-competence formation must penetrate *al subject spheres* in a primary schooling period.

Thus in the subject sphere "Philology" it's preferable to form elements of ICT-competence concerning lingual, reading and speechmaking competence. In the sphere of "Mathematics and Informatics" it's preferable to form ICT-elements concerning logical, iconic and symbolic competences, and, also, to learn to master interdisciplinary informational concepts. In the subject sphere "Environment" it's preferable to form ICT-competences concerning different methods of natural and social studies (in particular, obtaining, recording, presenting, and generalization of data). In the spheres "Art" and "Music" – ICT-competence elements concerning creation and transformation of sound and graphic objects, and in "Technology" –

making and use of information milieu for practical, creative and instructional task solution.

The new FSES ascribes the acquisition of a primary idea of computer literacy to the subject sphere “Mathematics and Informatics”, but initial skills of the ICT means usage belong to the sphere “Technology” for they are classified as intellectual work technologies. The acquired skills training must go on in all subject spheres while the leading integrative role is to be played by “Mathematics an Informatics” one.

Today, there exist four main forms of students’ work with ICT means application: working in adapted learning program media, project activities, working with object simulators, learner’s site communication.

Project making involves kids in solving practical, interesting and meaningful tasks using information and communication technologies including ICT means. Very often, projects look like integral activity, encompassing several subject spheres. In the first form, the projects made with ICT means use mainly special educational environments and adapted software, but, little by little, in the following forms kids start using non-adapted, universal programs.

Within the framework of subject sphere “Mathematics and Informatics” it seems convenient to work with program environment that includes subject (mathematic and informatics) task complexes, solving of which promotes subject matter mastering and main computer literacy skills formation. Mathematic simulators help kids secure the necessary skills (computational, for instance) within a rather modest class time span. Projects at Mathematics and Informatics lessons allow to be distracted from purely mathematic tasks to see, how obtained knowledge may be implemented in solving practical tasks in other spheres. While tackling such a project students learn to work in a group, distribute responsibilities in solving the task (thus, the communicative elements of ICT-competence are formed).

Touch-typing skill formation is going on mainly in subject sphere of Philology. The simulators of the ten finger method of text input must be used already in the 1st form; it’s good to learn handwriting and keyboard writing at the same time. Keyboard writing skills give child an unprecedented freedom to formulate thoughts that is much less while handwriting. Solving subject problems, students gradually get used to apply text editing programs (first, adapted and, thereupon, universal ones), and by the time of secondary schooling they become confident users. Project activity in the sphere “Philology” must also include communicative tasks (correspondence with peers in Russian and in foreign languages, for instance), which help kids learn to use different communication media (e-mail, chat rooms, IP-phone etc.). Each project solves subject tasks (writing a letter properly, for instance) and forms necessary elements of ICT-competence (using of appropriate environments and equipment, and also mastering rules of written and oral communication). For the sake of such work organization it’s necessary to create school and subject sites where pupils can communicate with each other and the teacher in different forms. Besides improving computer literacy such activity leads to a higher informational and communication culture level, which, in its turn, constitutes a part of general educational culture.

At “Environment” lessons ICT-competence most naturally forms within the framework of the project making activity. In the course of project making the use of ICT means (digital microscope, for instance) will enable kids to do independent research. As an important part of a project the research or experiment report writing and design (tests, graphs, snapshots, presentations) become a great opportunity for a kid to share personal discoveries and observations with others.

At “Music” and “Art” lessons ICT means can be used in projects and in solving class tasks for activity results recording. For instance, kids can record a performance fragment and save it in the form of audio or video file. While making projects including picture creation students may use computer graphics and appliqué. So, kids may design and make beautiful post cards including those with music and animation.

ICT-competence formation mustn’t confine itself to work with ICT means. Information and communication technologies are not mandatorily bound to any ICT means. In particular, students must get to know different communication and information activity technologies, which are not directly connected with technical means: intellectual, objective, symbolic etc. Therefore, the ICT-competence improvement must constantly take place, when kids work with data or when they communicate, i.e. both at the lesson and in extramural activities.

References

1. Bulin-Sokolova E.I., Rudchenko T.A., Semenov A.L., Khokhlova E.N.: The Formation of Primary School Students’ ICT-competence. Manual for Teachers of General Educational Institutions (Формирование ИКТ-компетентности младших школьников. Пособие для учителей общеобразоват. учреждений). Prosveshcheniye publishers, Moscow (2011)
2. The Federal State Standard of Basic General Education (Федеральный государственный образовательный стандарт основного общего образования). Prosveshcheniye publishers, Moscow (2011)
3. The Exemplary Basic Educational Program for Educational Institutions. Primary School (Примерная основная образовательная программа образовательного учреждения. Начальная школа). Prosveshcheniye publishers, Moscow (2011)

On Abstraction and Informatics

Tom Verhoeff

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
T.Verhoeff@tue.nl

Abstract. One often hears, and less often reads, the claim that informatics and its application is so difficult because it involves demanding abstractions. Abstractions in informatics supposedly are even harder than those in mathematics and the physical sciences. If abstraction is so important in informatics, then you would expect that we have good ways of dealing with it and communicating about it. To some extent, this is indeed the case, but unfortunately these ways are not widespread. It is our duty to get to grips with abstraction, and especially to address it in the teaching of informatics.

In this article, I will not solve the problems posed by abstraction, and certainly not the problem of teaching abstraction. But I would like to put it more prominently on the agenda. In order to deal with abstraction, you will have to investigate it, dissect it, analyze it, establish terminology, etc. I will give a, somewhat personal, overview of abstraction, showing that it is not a single, atomic concept, but a diverse complex of interrelated concepts. It is my hope that this will help in embedding abstraction more explicitly in the informatics curriculum.

1 Introduction

Abstraction is often mentioned as explanation for why informatics[1] is so difficult. Let me quote a few of the more explicit authors. Bucci et al. start their paper [5] as follows.

> "Abstraction is one of the cornerstones of software development and is recognized as a fundamental and essential principle to be taught as early as CS1/CS2. Abstraction supposedly can enhance students' ability to reason and think. Yet we often hear complaints about the inability of CS undergraduates to do that."

In 2006, Kramer and Hazzan organized an ICSE workshop on *The Role of Abstraction in Software Engineering*. Their summary [18] explains:

[1] I use the term *informatics* as synonymous with *computer science*.

> "The rational for the workshop stems from the observation that some software engineers seem to be able to produce clear, elegant designs and programs, while others cannot. Abstraction is suggested as the source for this phenomenon, i.e., that some engineers lack the ability to think abstractly and to exhibit abstraction skills ..."

Kramer later elaborated on this theme in [19]. Wing states squarely in [27]:

> "The essence of computational thinking is *abstraction*. In computing, we abstract notions beyond the physical dimensions of time and space. Our abstractions are extremely general because they are symbolic, where numeric abstractions are just a special case. ... [O]ur abstractions tend to be richer and more complex than those in the mathematical and physical sciences."

The IEEE-ACM computing curriculum recommendations [1] also recognize the importance of abstraction in informatics. In [2, p.8], the task force writes:

> "Software engineering differs from traditional engineering because of the special nature of software, which places a greater emphasis on abstraction, modeling, information organization and representation, and the management of change. ... An important aspect ... is that the supporting process must be applied at multiple levels of abstraction."

Saying that abstraction is important is one thing. Explaining what abstraction is and how to teach it are quite other matters. Bucci et al. express this even more strongly in their paper [5], titled 'Do We Really Teach Abstraction?':

> "It is the central claim of this paper that, despite broad agreement about their fundamental importance, both information hiding and abstraction are severely shortchanged by current CS1/CS2 pedagogy. More specifically, we claim that standard pedagogical practices tend to compromise the human dimension of information hiding and fall well short of helping student learners to realize an increased ability to reason and think *carefully* and *rigorously* as a benefit of using abstraction."

1.1 Overview

In this article, I make an attempt at dissecting the notion of abstraction, especially, the way in which it plays a role in informatics. Section 2 considers abstraction from a philosophic-linguistic-psychologic point of view. Sections 3 and 4 address abstraction in mathematics and physics. In Section 5, I elaborate on the use of abstraction in informatics, and Section 6 focuses on the issues of teaching abstraction. Section 7 concludes this article.

Let me hasten to say that this article is not intended to serve as an example of how to incorporate abstraction into the informatics curriculum. The intention is to stimulate and organize the discussion about abstraction and its teaching.

2 Abstraction in Abstracto

Abstraction is itself a highly abstract notion; try explaining it to a six-year old child. That child can already handle some pretty abstract concepts: counting numbers, colors, relationships like brother and sister, but usually not abstractness itself. Let us first look at the various word usages involving abstraction, as it concerns us here (there are other meanings, not relevant for our discussion).

to abstract (intransitive verb; used with 'from X') to ignore or suppress aspects (X) that are considered irrelevant for the purpose at hand
abstraction (noun) the act of abstracting; also, the result of abstracting
abstract (adjective) the quality of being the result of abstracting; opposite of concrete

When speaking about abstraction, it is always important to ask: abstracting *from what*? For example, when counting people in a village, you '*abstract from* the identity of the persons counted'. By the way, when counting, you abstract from more than just identity; you abstract from everything but numerosity (cardinality). It is noteworthy that with this verb we typically can and do say what is suppressed (X, 'identity' in the example), but with this verb it is not so easy to say what is (intended to be) preserved ('number' in the example). You could try to do so by adding 'obtaining Y', although that does not sound so well: 'abstract from the identities obtaining (just) a number'. Also note that this verb concerns an action, transforming an initial 'concrete' 'thing' (before abstracting; like a group of people) into a resulting 'abstract' 'thing' (after abstracting; like a number).

The example of counting as a form of abstraction is relatively straightforward. Abstraction, however, also concerns the creation of new notions from existing notions, by ignoring certain features that are deemed irrelevant in the new notion. The case of counting is relatively straightforward when the notion of cardinality is already known. But what about a situation where you want to introduce a new notion, something that is not yet known? You can try to say what to ignore, but not what it is that you want to obtain, because that is precisely what is being defined. For example, how to define the notion of *probability* when this notion did not yet exist? I will get back to this in Section 3, when dealing with abstraction in mathematics.

Note that the invention of new notions is a creative act, and an important aspect of doing science (also see Section 4 on physics). Epistemology is the branch of philosophy concerned with knowledge and its creation. But the creation of new notions is important in teaching too, because people are not born with the latest notions from science embedded in their brains. The situation of a student resembles that of the initial inventors of those concepts. So, the process of creating new notions is not a one-time thing, but is repeated every time a person tries to understand an unfamiliar notion. Since abstraction is one of the most effective ways to define new notions, teachers should be well aware of this mechanism, so that they can properly guide their students.

Why is abstraction needed at all? The main reason derives from cognitive limitations of the human mind. We, human beings, have very limited processing powers. Our short-term memory can store only a few 'chunks' of information at a time. Traditionally, this limit was believed to be 'seven plus or minus two' [24], but recent research has decreased that limit to approximately four [6]. By applying appropriate abstractions, we can concentrate on relevant information and ignore everything else. Without abstraction, our minds would be overwhelmed with information and we would not be able to function as effectively as we do.

It is good to point out some misconceptions about abstraction. Abstraction is not the same as *vagueness* or *imprecision*, although these also concern ways of limiting information content. Unfortunately, half-baked attempts at teaching abstraction get no further than just vague and imprecise bluffing.

Dijkstra expresses the need and the misconception crisply in his Turing Award Lecture [13]:

> "We all know that the only mental tool by means of which a very finite piece of reasoning can cover a myriad cases is called "abstraction"; as a result the effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer. In this connection it might be worth-while to point out that the purpose of abstracting is not to be *vague*, but to create a new semantic level in which one can be absolutely precise."

Another misconception is that abstract–concrete is a dichotomy, a black-and-white thing. In linguistics this seems to be the case, where you have concrete entities (people, rocks, paper, scissors) and abstract notions (number, color, relationships). But in scientific usage, and especially in informatics, abstraction concerns a gradual scale. You can abstract from several aspects, one by one, in separate abstraction steps, yielding a sequence of intermediate concepts of increasing abstraction. Thus, abstract–concrete is a *relative* relationship: B can be more abstract than C, and A can in turn be more abstract than B. This gives rise to multiple levels or layers of abstraction.

All of this shows that abstraction is not an easy concept, both from a linguistic and a philosophic point of view. Still, that is no reason to shy away from it. On the contrary, it poses an important challenge to be tackled.

3 Abstraction in Mathematics

Mathematics is a good place to start digging into the notion of abstraction, because it is itself abstract (a purely mental construction), and a rigorous discipline, which developed precise ways of defining abstractions. As such, it is also an important foundation for informatics. Devlin argues this in [12]:

> "The main benefit of learning and doing mathematics is that it develops the ability to reason about formally defined abstract structures like those in computer science and its applications."

How are mathematical concepts defined? Basically, there are only two methods:

– axiomatically, by postulating a collection of defining properties
– in terms of existing concepts, possibly through abstraction

I do not want to get sidetracked into the philosophy of mathematics, more particularly, into the ontology of mathematical objects. So, the following presentation is necessarily simplified. A well-known example of the *axiomatic method* is the introduction of the concepts of Euclidean geometry. The definitions of *point*, *line*, and *incidence* do not tell what these concepts *are* in themselves. Instead, they postulate relationships, that together sufficiently define those concepts to do mathematics, that is, to formulate and prove interesting theorems. For instance, Euclid's first axiom states that two distinct points determine a unique line incident on these points. Similarly, the *Dedekind–Peano Axioms* define the natural (counting) numbers, and the *Zermelo–Fraenkel Axioms* define sets. The biggest danger of axiomatic definitions is *inconsistency*, i.e., that the concepts intended to be defined actually cannot exist, because the axioms contradict each other. This is typically a hard problem, or even impossible to resolve.

In the second method, new concepts are defined in terms of existing concepts. Therefore, such definitions are more easily shown to be valid (consistent). For instance, a (mathematical) *graph* is defined as a pair (V, E) where V is a set (whose elements are called *vertices*) and $E \subseteq V \times V$ is a set of vertex pairs (called *edges*).

3.1 Definition by Abstraction

An important variant of this second method is the *definition by (logical) abstraction*. The *integer numbers* can be defined as 'signed' natural numbers, i.e., as pairs of a plus or a minus sign, and a natural number, where plus zero and minus zero are the same thing. More elegant, however, is the following definition involving the set $P = N \times N$ (pairs of natural numbers) and the relation $\sim$ on P defined by

$$(a, b) \sim (c, d) \iff a + d = b + c \tag{1}$$

This definition is expressed in terms of existing concepts, viz. addition and equality of natural numbers and well defined for all a, b, c, and d. If you already know the integer numbers, then I can reveal to you that the pair $(a, b) \in P$ is intended to 'correspond' to the integer $a - b$ (like the way bookkeepers avoid negative numbers by working in two columns). Note, however, that $a - b$ is not (yet) defined for natural numbers a and b when $a < b$. Relation $(a, b) \sim (c, d)$ intends to capture $a - b = c - d$, which is equivalent to the right-hand side of (1). But you do not need to know all that to read the definition.

It turns out that relation $\sim$ is an *equivalence relation*, i.e., it is *reflexive* ($p \sim p$), *symmetric* ($p \sim q$ implies $q \sim p$), and *transitive* ($p \sim q$ and $q \sim r$ implies $p \sim r$). An equivalence relation on a set P partitions P into disjoint nonempty subsets, such that equivalent elements are in the same part, and non-equivalent elements are in distinct parts. The set Z of integer numbers is now defined as the set of *equivalence classes* in P under $\sim$. That is, each equivalence class, i.e., set of $\sim$-equivalent pairs of natural numbers, is an integer number in Z. This is also written as $Z = P/{\sim}$, the 'quotient' of P and $\sim$. The equivalence relation is 'divided out'; the distinctions between equivalent pairs are 'erased' by uniting them into a single new 'concept', called integer number. The integers are, thus, obtained as pairs of natural numbers where we abstract from the distinction between pairs satisfying (1).

At first sight, this looks like a horribly complicated definition, because an integer now 'is' a set of $\sim$-equivalent pairs of natural numbers. For these integers, one can define operations like addition, subtraction, and multiplication, and prove their basic properties. Further analysis shows that every pair in P is $\sim$-equivalent either to a pair of the form $(a, 0)$, which can be identified with the natural number a, or to a pair of the form $(0, a)$ with $a > 0$, which is usually written as $-a$. Once, this is done, you no longer need to worry about the fact that an integer was defined as such a horribly complicated object. You can simply use the properties, and think of integers as 'atomic' objects. The quotient construction merely proves the consistency of the properties. It is an 'internal' concern for foundationalists that need not concern mere users of the integers.

Once one is familiar with the method of definition by abstraction, it becomes a powerful tool. In fact, every abstraction can be can be viewed as abstracting from an appropriate equivalence relation. For instance, the *rational numbers* can be defined by abstracting from the equivalence relation $\sim'$ defined by

$$(a, b) \sim' (c, d) \iff a \times d = b \times c \tag{2}$$

on pairs of integer numbers, where the second element is nonzero. Defined in this way, rational numbers involve two layers of abstraction based on equivalence relations (1) and (2).

3.2 How to Work with Definitions

How one handles a definition, depends on its nature. Axiomatic definitions are eliminated not by applying them to a single concept, such as a geometric point, but by applying them to a suitable *combination* of concepts that are related through an axiom, such as two distinct points that yield a line incident on those points.

Definitions in terms of existing concepts can be eliminated by *substitution*. For instance, the number $M(a, b)$ halfway between numbers a and b can be defined as $(a+b)/2$. When you want to prove something about $M(x+y, x-y)$, you can simply eliminate M by substitution, obtaining $((x + y) + (x - y))/2$, which incidentally can be further reduced to x.

There is one exception to this elimination: *recursive definitions*. These involve a new mechanism, where something is, partly, defined in terms of itself. For instance, $n!$ (n factorial) can be defined for natural numbers $n \geq 0$ by

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n \geq 1 \end{cases} \tag{3}$$

When you want to prove something about an expression of the form $(a + b)!$, you cannot simply eliminate ! by substitution. You need to resort to *induction*. In teaching, the notions of recursion and induction are always challenging [29].

Category Theory can be viewed as the summum of abstraction in mathematics. It is a theory about mathematical structure in very general terms, with *functions* as fundamental notion. I am not saying that you need to study Category Theory before teaching about abstraction. But it does serve to illustrate how the suppression of internal details makes for new ways of defining things. The traditional definition of an injective function (injection or one-to-one function) is as follows. Function $f : A \to B$ is called *injective* when

$$f(x) = f(y) \text{ implies } x = y, \text{ for all } x, y \in A \tag{4}$$

This definition involves the application of the function to arguments, thereby relying on 'internal details' of functions. In Category Theory, functions are usually referred to as *morphisms*, and an injection is called a *monomorphism*. Morphism $f : A \to B$ is called a monomorphism when

$$f \circ g = f \circ h \text{ implies } g = h, \text{ for all } g, h : Z \to A \tag{5}$$

where $\circ$ denotes morphism composition; in traditional terms: $(g \circ f)(x) = f(g(x))$. Definition (5) is *point-free*, i.e., only relies on 'external features' of morphisms, without applying them to specific arguments. The definition of an *epimorphism* (surjective function) is a completely analogous dual of (5):

$$g \circ f = h \circ f \text{ implies } g = h, \text{ for all } g, h : B \to Z \tag{6}$$

Compare this to the traditional definition, which looks very different from (4):

$$(\forall\, y \in B : (\exists\, x \in A : f(x) = y)) \tag{7}$$

Thus, (5) and (6) are more abstract than (4) and (7). This also relates to functional programming (see Section 6).

On one hand, mathematical concepts are abstract and people often express having difficulty with mathematics. On the other hand, recent research [10,25] has revealed that the human mind is born with powerful innate abstraction abilities. For instance, even newly born babies already have an innate ability to handle small numbers 'in the abstract'. This partly overthrows the traditional cognitive development sequence as described by Piaget, where abstract thinking would only appear in the formal operational stage at the age of twelve or so.

In [11], Devlin makes the case that the human brain has evolved to deal with precisely those abstract structures that are mathematically important. His argument runs as follows. Human beings are relatively weak, both on attack and defense, compared to most (other) animals. In order to survive, they have had to evolve brain structures to cooperate in larger groups. Such cooperation requires that the brain keeps track of relationships among people in groups. It is precisely such networks of relationships that are important in mathematics as well. All human brains are basically equipped to deal with mathematical structures. Why not everybody is a math whiz, is another story (see [11]).

4 Abstraction in Physics

Abstraction in the physical sciences serves a similar purpose as in mathematics, viz. to define concepts. These concepts are used to describe (laws of) nature. They are abstractions created by the human mind. Think of concepts like mass, force, energy, and temperature. In contrast to mathematics, such concepts are intended to have a link to the concrete physical world. Here we have a clear jump from concrete to abstract. After abstraction, we end up in the mathematical world.

One particularly important abstraction step in physics concerns the way a system's behavior over time is modeled. A dynamical system is difficult to analyze when you try to 'think it through', tracing all changes over time. Your mind is soon overwhelmed. The key idea is to capture such behavior by considering the system's state *as a function of time*. This function is a single entity, containing all information that is relevant. Such functions can be handled at a higher level of abstraction. For instance, a law can be expressed as a differential equation involving the system's state function. All of time is handled in one swoop.

Under the heading "The Simplification of Science and the Science of Simplification" in [26, pp.8–12], Weinberg discusses Newton's Law of Universal Gravitation (by Feynman called "the greatest generalization achieved by the human mind"). This law expresses the attractive force between two bodies as function of their mass and distance. First of all, he notes that this law implicitly says that the attractive force does not depend on anything other than the two masses and their distance. One can abstract from everything else, a huge abstraction. When Newton used his law to analyze our solar system, he had to apply further abstractions. In particular, to calculate a planet's orbit, he focused on the sun and that planet only, ignoring all other masses and (non-gravitational) influences. Weinberg writes:

> "Newton was a genius, but not because of the superior computational power of his brain. Newton's genius was, on the contrary, his ability to simplify, idealize, and streamline the world so that it became, in some measure, tractable to the brains of perfectly ordinary men."

5 Abstraction in Informatics

In informatics and its application, both the mathematical and the physical kind of abstractions play a role. When automating the processes in a car, the software will somehow have to be related to various (physical) car parts. In the software, these parts exist as abstractions. This kind of physical abstraction is encountered when modeling the problem domain. As with physics, information processing systems involve behavior over time, where the notions of state and state change in a state space are important. Also see [9], [16, Ch.1], [3], and [28].

The definition of models, both for the problem domain and the solution domain, involve the mathematical kind of abstractions. The main abstractions in theoretical informatics concern the notions of *algorithm* and *information*.

The reason that abstraction plays a more important role in informatics than in other disciplines, is that the description of every program involves the creation of new abstractions to define the data and the operations. To aid the definition of data and operations, many predefined generic abstractions are available, and must have been mastered.

Programming tools offer various layers of abstractions on top of the hardware that executes the programs. Compilers, interpreters, operating systems, and run-time libraries translate high-level descriptions into low-level instructions. Furthermore, modern programming languages provide abstraction mechanisms to aid in the description of data structures and algorithms. Let us briefly look at some of these abstraction mechanisms (see e.g. [20]):

procedural abstraction: mechanism to define parameterized routines (functions, procedures, methods) usable as an action

data abstraction: mechanism to define parameterized (generic) data types (sets of values with associated operations)

iteration abstraction: mechanism to iterate over a collection, processing each element once; there are also many loop patterns

Note that non-recursive routine and data type definitions can be eliminated by a double substitution: substitute actual for formal parameters, and substitute the body or representation for the invocation or occurrence.

These mechanisms share a number of features:

- abstraction from operational and representational implementation details, as seen from the user's perspective;
- abstraction from the identity (in case of procedural abstraction) or type (in case of data abstraction) of data involved, via parameters;
- abstraction from context of usage[2], as seen from the provider's perspective.

They involve two sides: a *user* or *client* side and a *provider* or *server* side. These sides are completely separate, except for a two-sided contract that specifies the defined entity, thereby serving as an *interface*. In reasoning about these abstractions, it is mandatory (for otherwise the abstraction loses its value) to avoid reasoning that relates the user context and the provider context directly to each other (basically, by carrying out the substitution mentioned above). Instead, both the user context and the provider context should solely be related to the agreed contract. For instance, in the case of routines, the contract is expressed in terms of a *precondition* and a *postcondition* (also see Fig.1):

- The user takes care that the precondition holds.
- The provider exploits this precondition and takes care of the postcondition.
- The user exploits this postcondition.
- The user need not know how the routine does its work.
- The provider need not know how the parameters and result are used.

[2] This abstraction is often not mentioned, but equally important as the first.

		Two-sided Contract		
		Precondition		**Postcondition**
	User	concern		benefit
Party		$\downarrow$		$\uparrow$
	Provider	benefit	$\rightarrow$	concern

Fig. 1. The relationships in two-sided contracts for routines

We see here that abstraction appears in many disguises, going by such names as *divide and conquer, separation of concerns, modularization, generalization, Design By Contract, encapsulation, information hiding, modeling and design patterns*, etc. These techniques aim at managing complexity through abstraction.

In software development, the path from specification to implementation involves multiple refinement steps, going from abstract models to more concrete code. In the analysis of existing systems, models need to be constructed for those systems, thus going from concrete to more abstract.

6 Teaching Abstraction

Many textbooks present various abstraction mechanisms in programming languages, but most do so implicitly. One of the more explicit textbooks is [20]. Others textbooks present modeling and design techniques, again with limited attention for abstraction. The pedagogy of abstraction in informatics has not been studied deeply. Useful articles are [5,14,17,19]. The teaching of abstraction must be based on a long-term plan. Abstraction cannot be taught in a few lessons; it must be infused over many years. That requires a carefully tuned curriculum. Offering curricular advice is beyond the scope of this article. In summary, [17] advises to start early; to teach abstraction consciously; to stress the benefits of using abstraction. I would like to add: teach it in small steps.

The question remains: How to do so? Consider the case of teaching abstraction mechanisms like parameterized procedures and Abstract Data Types (ADTs). In a traditional approach, you teach such mechanisms by explaining and practicing the steps involved in the order of the development process; see Fig. 2 (left). The student then always works toward the unknown. An alternative approach is *backward chaining*, which I learned from [8, p.38] (also see [4]). Here, a process consisting of several steps is taught in reverse order, by starting with the *last* step, and successively adding *preceding* steps, one by one; see Fig. 2 (right). In backward chaining, the student always works toward familiar steps. Backward chaining is also applied successfully in sports, music, and military training. It resembles Meyer's *inverted curriculum* and *outside-in method* [21,23]. It would be interesting to see whether abstraction in informatics can be captured in a set of Meyer's TRUCs [22], i.e, testable and reusable units of cognition.

Unfortunately, many modern programming languages, like C, C++, C#, and Java do not provide sufficient support for abstraction. For instance, only the syntactic part of contracts can be expressed as part of the program, while the

In the development process, you	In backward-chaining, you learn to
(1) Draw up a contract for an abstraction	(1) Use an abstraction, given its contract
(2) Validate the contract	(2) Test an abstraction, given its contract
(3) Design/implement it (can be split)	(3) Review a given design/implementation
(4) Review the design/implementation	(4) Design/implement a given contract
(5) Test the implementation	(5) Validate contracts
(6) Use the abstraction	(6) Draw up a contract for an abstraction

Fig. 2. Forward and backward chaining

semantic part must be expressed as comment. The contract for a function cannot be named separate from its implementation. There are some add-on features for these languages, but they are not appropriate for teaching. Eiffel [23], Perfect Developer [7], and RESOLVE [5] are a considerable step in the right direction.

The role of efficient and effective notation should not be underestimated [15, Ch.16]. In this respect, *functional* and *declarative* languages are better alternatives than *imperative* languages. But they are —unrightfully— too often avoided in teaching, probably because of the higher level of abstraction. I agree with [5], in that we need to teach students about abstract mathematical notions, like sets, mappings, relations, bags, and sequences, before they start to develop implementations. They need these notions to read proper contracts and to express models without thinking in terms of implementations. But it is also important that model-oriented specifications are complemented by property-oriented specifications (resembling the axiomatic definitions in mathematics). In particular, reasoning about abstractions is underexposed.

7 Conclusion

We have looked at abstraction from various angles, especially as it plays a role in informatics. It is important that abstraction is properly integrated into informatics teaching. Our hope is that this overview will organize and stimulate the discussion on how to teach abstraction in the informatics curriculum.

References

1. ACM–IEEE Joint Task Force on Computing Curricula. *Computing Curricula 2001 Computer Science: Final Report.* Dec. 2001. `www.acm.org/education/curric_vols/cc2001.pdf` (accessed Apr. 2011)
2. ACM–IEEE Joint Task Force on Computing Curricula. *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.* Aug. 2004. `sites.computer.org/ccse/SE2004Volume.pdf` (accessed Apr. 2011)
3. Dines Bjørner. *Software Engineering 1: Abstraction and Modelling.* Springer, 2006.
4. B. Brandon. "Last Things First: The Power of Backward Chaining", *The eLearning Developers' Journal*, Oct. 2003.
5. P. Bucci, T. J. Long, B. W. Weide. "Do We Really Teach Abstraction?", *SIGCSE Bull.*, **33**(1):26–30 (Feb. 2001).

6. Nelson Cowan. “The Magical Number 4 in Short-Term Memory: A Reconsideration of Mental Storage Capacity”, *Behavioral and Brain Sciences*, **24**(1):87-114 (2001).
7. D. Crocker, J. Carlton. “Perfect Developer: what it is and what it does”, *FACS Facts*, newsletter of the BCS Formal Aspects of CS special interest group, Nov. 2004. `www.eschertech.com/papers/introduction_to_perfect_developer.pdf` (accessed Apr. 2011)
8. E. de Bono. *Teach Your Child How to Think*. Penguin, 1994.
9. O.-J. Dahl, E. Dijkstra, C. Hoare, *Structured Programming*, Academic Press, 1972.
10. S. Dehaene. *The Number Sense: How the Mind Creates Mathematics*. Oxford Univ. Press, 1997.
11. K. Devlin. *The Math Gene: How Mathematical Thinking Evolved and Why Numbers Are Like Gossip*. Basic Books, 2000.
12. K. Devlin. “Why universities require computer science students to take math”, *Communications of the ACM*, **46**(9):36–39 (Sep. 2003).
13. E. W. Dijkstra. “The Humble Programmer” (Turing Award Lecture 1972). `www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD340.html` (accessed Apr. 2011)
14. P. Frorer, O. Hazzan, M. Manes. “Revealing the Faces of Abstraction”, *International J. Computers for Mathematical Learning*. Kluwer Academic Publishers, **2**(3):217–228 (Oct. 1997).
15. A. J. M. van Gasteren. *On the shape of mathematical arguments*. LNCS **445**, Springer, 1990.
16. Hans Jonkers. *Abstraction, Specification and implementation techniques: with an application to garbage collection*. Dissertation, Eindhoven Univ. of Technology, 1983.
17. H. Koppelman, B. van Dijk. “Teaching abstraction in introductory courses”, *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education ITiCSE ’10*, pp.174–178 (2010).
18. J. Kramer, O. Hazzan. “Summary of an ICSE 2006 Workshop: The Role of Abstraction in Software Engineering”, *ACM SIGSOFT Software Engineering Notes*, **31**(6):37–42 (Nov. 2006).
19. J. Kramer. “Is Abstraction the Key to Computing?” *CACM* **50**(4):36–42 (Apr. 2007).
20. B. Liskov, J. Guttag. *Program Development in Java: Abstraction, Specification, and Object-Oriented Design*. Addison-Wesley, 2001.
21. Bertrand Meyer. “The Outside-in Method of Teaching Introductory Programming.” In M. Broy and A. V. Zamulin (Eds.), *Ershov Memorial Conference*, LNCS **2890**:pp.66–78. Springer, 2003.
22. B. Meyer. “Testable, Reusable Units of Cognition”, *IEEE Computer*, **3**(4):20–24 (Apr. 2006)
23. B. Meyer. *Touch of Class: Learning to Program Well with Objects and Contracts*. Springer, 2009.
24. George A. Miller. “The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information”, *Psychological Review*, **63**:81-97 (1956).
25. D. A. Sousa. *How the Brain Learns Mathematics*. Corwin Press, 2008.
26. G. M. Weinberg. *An Introduction to General Systems Thinking*. Wiley, 1975.
27. J. Wing. “Computational thinking and thinking about computing”, *Phil. Trans. R. Soc. A*, **366**:3717–3725 (Jul. 2008).
28. Niklaus Wirth. “A Brief History of Software Engineering”, *IEEE Annals of the History of Computing*, **30**(3)32–39 (July–Sept. 2008).
29. D. R. Woodal. “Inductio ad Absurdum?”, *Mathematical Gazette*, **59**(408):64–70 (Jun. 1975).

Metaphorical Geometry

Michael Weigend[1,2] and Vaiva Grabauskiene[3,4]

[1] Holzkamp-Gesamtschule Witten, Willy-Brandt-Str. 2, 58452 Witten, Germany,
[2] University of Münster, Fliednerstr. 21, 48149 Münster, Germany,
michael.weigend@uni-muenster.de
[3] Vilnius University, Akademijos Str. 4 LT-08663 Vilnius, Lithuania
[4] Vilnius Pedagogical University, Studentų Str. 39, LT-08106, Vilnius, Lithuania
vaiva.grabauskiene@vpu.lt

Abstract. Interpreting and creating metaphorical algorithms can be considered as a facet of computational thinking. This paper presents the results of a study that was performed with 97 students from German schools in the age of nine to thirteen years. The children had to interpret metaphorical algorithms that explained how to change images composed of simple geometric shapes. The results were analyzed with respect to typical geometric mistakes and levels of abstraction.

Keywords: algorithm, metaphor, geometry, abstraction, computational thinking.

1 Introduction

Broadly speaking, an algorithm is an effective method for calculating a function or solving a problem. It consists of a limited number of instructions, which describe operations precisely enough to be executed automatically by a machine. Algorithms can be expressed in many ways. They may be written in natural language for human readers. The point is that it must be possible to refine and formalize the algorithm to a program that can be interpreted by a computer.

Sometimes a sophisticated sequence of abstract geometrical operations can be described metaphorically by referring to an act of manipulating real things. For example, peeling an orange may be considered as a metaphor for subtracting two concentric spheres. For most people, "peeling an orange" is a holistic concept of activity. It is intuitive, subjectively certain knowledge. Referring to this familiar concept in a metaphorical way might make a geometrical algorithm easier to comprehend.

In cognitive linguistics metaphors are considered as cross-domain conceptual mappings [1]. They project knowledge from a well known source domain onto a target domain. In contrast to rhetoric, in cognitive linguistics metaphors are primarily not a matter of language but of thought. People use metaphors to understand the world and also to communicate this understanding.

Metaphors are used while creating computer programs on several levels of abstraction:

In Extreme Programming a software development starts with a “project metaphor” describing the idea of a whole software system in one holistic concept. Example: A metaphor for an agent-based information retrieval system is a “hive of bees, going out for pollen and bringing it back to the hive.”[2] Well known high level metaphors are “World Wide Web” or the “scan line” in algorithmic geometry.

Data structures or standard classes like stacks and queues are metaphors projecting knowledge about physical collections onto the domain of data processing. For example, a physical stack of plates supports certain access-operations: You can only remove the top plate (pop) and put a new plate on top of the stack (push). This structure is adopted in the data type called “Stack”.

An assignment like `a = b` is often interpreted adopting one of these metaphors: (1) *Names are labels*. “Put a new label ‘a’ at the object already labeled with ‘b’”. (2) *Names are containers*. “Put a copy of the content of container ‘b’ into container ‘a’.

Using a high level programming language a programmer has a lot of freedom to structure and phrase an algorithm. By choosing appropriate names, a program can be written in a metaphorical style. The primary goal is not to make the text more interesting but to increase its comprehensiveness. Thus, metaphorical thinking can be considered as a facet of programming competence.

2 Metaphors and Computer Programming

In this section we discuss three cognitive operations connected to metaphorical thinking during program development:

- Finding metaphors (searching for appropriate source domains)
- Switching between different metaphors (changing related sources)
- Adapting program code to metaphors (changing the target).

Consider the following Python program. It consists of a class definition, the definition of a function processing instances of the new class, and a few lines of code, creating instances .of the class, calling the function and printing the result.

```
class Thing():
    def __init__(self, name, weight):
        self.name=name
        self.weight = weight

def collectSmallThings (maxWeight, box):
   bag = []                                          #1
   for thing in box:                                 #2
       if thing.weight <= maxWeight:
           bag.append(thing)                         #3
   return bag

stationery = [Thing("paper", 1), Thing("stapler", 0.2),
       Thing("pen", 0.1)]                            #4
```

```
smallThings = collectSmallThings (0.2, stationery)    #5
print([item.name for item in smallThings])
```

Instances of class `Thing` represent things with a name and a weight. In line #4 a list consisting of three instances of `Thing` (representing stationery items) is created. In line #5 a new list representing items weighing less than 0.2 kg is created by calling the function. Finally a list with the names of the small items is printed. The output is:

```
['stapler', 'pen']
```

We are going to discuss the function definition. The programmer has the freedom to choose arbitrary names for all kinds of objects including functions and parameters. The names of the function (`collectSmallThings`), its parameters (`maxWeight`, `box`) and the local variable (`bag`) induce a coherent meaningful story, metaphorically explaining how the function works: The agent (executing the function) gets a box with things in it and a maximal weight. He takes an empty bag (line #2). Then he weighs all things in the box. If the weight of a thing is equal or less than the maximal weight, a copy of it is put into the bag. At the end the agent returns the bag with the small things to the client, who has called the function.

While choosing names, the author of this program had to invent a story like this. This implies finding appropriate metaphors for activity (operation 1): taking a box, collecting things in a bag and returning the bag. The box and the bag are implemented by Python lists, instances of an already existing (built-in) class modeling sequences of objects. This technical detail is irrelevant for phrasing the iteration `for thing in box ....` A problem occurs in line #3. A new item is added to the list named bag. The name of the operation is `append()`. This name only makes sense in connection with a sequence not with a bag. You put things into a bag but you cannot append something to a bag. Thus, in this line a metaphorical shift takes place (operation 2). The program text induces the idea that you have a chain of things that can be extended by appending new items.

One can amend this inconsistency by introducing a new class named `Bag`, modeling bags for collecting things.

```
class Bag ():
    def __init__ (self):
        self.content = []
    def collect (self,thing):
        self.content.append(thing)
```

Instances of the class Bag have the attribute content, which is a standard list. The method `collect()` appends an object to this list. No functionality is added by the new class. The only effect is that `Bag`-object fits to the metaphor “collecting things in a container “. That means, the program code is adapted to a metaphor (operation 3) that describes the algorithmic idea on a higher level of abstraction. In real life software projects such operations take place in a refactoring phase, when the naming of objects is changed in order to increase the readability of the code.

3 Metaphors in Geometry

In the study which is presented later we have tried to investigate to what extend students in the age of 10 to 12 are able to understand and create metaphorical instructions that describe how to transform an image consisting of two-dimensional geometrical shapes (straight lines, rectangles, triangles and circles) to another image.

Metaphorical interpretations of geometric shapes are common in everyday life. A simple geometric image with a funny metaphorical meaning is called a droodle. Figure 1 depicts a few examples.

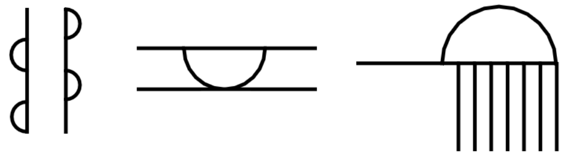


Fig. 1. Droodles: (1) Bear climbing up a tree (seen from the other side of the tree), (2) a fried egg making a head stand, (3) an octopus signaling that he is going to turn left.

Torreano et al. [4] suggest that a metaphorical term is a prototype of a category. Therefore the metaphorical meaning is more abstract (less specific) than the literal meaning. Based on this idea they define levels of abstraction. The more abstract an expression the more facets of the literal meaning are missing in the more abstract metaphoric meaning. Consider the two statements

1) "The dog flew across the yard."

2) "The rumor flew across town."

The second usage of the verb "to fly" is more abstract than the first. In its context free meaning "to fly" refers to swift physical movement through the air. That implies high speed, a physical entity and airborne. In the first statement the metaphorical meaning abstracts from "airborne" and in the second statement the facets "airborne" and "physical entity" are omitted. Torreano et al. found out that most people consider more abstract metaphors as "more metaphorical".

Torreano's language-related principle for defining levels of abstraction can be transferred to conceptual metaphors for geometric transformations. Consider the following two algorithms (figure 2). They are from tasks we used in workshops with fourth- and sixth-graders (see below).

1) "There are two trees standing side by side on a meadow. A couple of years later the trees have been grown. Tina connects the trunks with a string. How does it look like now?" (figure 2, first picture)

2) "There are six horses on the meadow and an empty feeding trough. Someone puts oat into the trough. The horses walk to the trough on the shortest way in order to eat the oat. How does it look like now?" (figure 2, second picture)

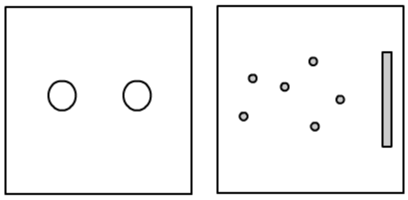


Fig. 2. Two geometrical images to be transformed

These little text documents can – in a broad sense – be considered as simple algorithms processing visual data. The given image is the input and the new image that is to be drawn is the output.

The first algorithm uses a tree as a metaphor for a circle and the activity of growing as a metaphor for increasing the radius. In this case the only abstraction is the reduction of the tree's shape to a simplified projection from above. The activity "growing" applied to a circle – as a two-dimensional projection of a tree– has no other meaning than "increasing the radius" (see figure 3, left image).

In the second algorithm the metaphorical meaning of "horse" completely omits the shape of the horse and the activity of eating at the end (see figure 3, right image). Furthermore the motive for moving to the rectangle (trough) is ignored. Therefore, the metaphorical phrasing of the second algorithm is more abstract that the phrasing of the first.

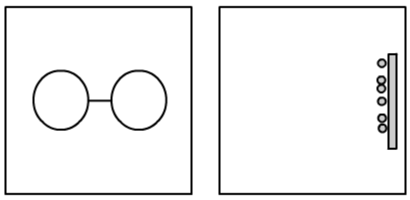


Fig. 3. Expected results of the image transformations

Beside these metaphorical abstraction levels of the instructions there are further dimensions of abstraction to be considered. When students execute an algorithm transforming a geometric image they must be able to perform certain geometric operations. According to the van Hiele model of geometric thinking there is an increase of abstraction from simple visual thinking based on manipulating things to strict and formal geometric thinking, independent from reality [5, 6].

4 A workshop on metaphorical geometry

In April 2011 we performed four workshops with 44 fourth-graders at an elementary school and 53 sixth-graders at a comprehensive school in Germany. All workshops had the same structure. They took approximately 50 minutes at the elementary school and 45 minutes at the comprehensive school. The workshops consisted of four parts:

Introduction. At the beginning, the children performed a physical exercise that was supposed to give an idea about metaphorical algorithms. The moderator explained a sequence of actions in two different ways: explicitly and metaphorically.

Explicit instruction: “Please stand up. Look to the ceiling. Raise your right hand. Open your right hand and stretch your fingers in a way that they point to the ceiling. Close your fingers. Take your arm down. Now do it again a few times: Raise your right hand. Open your right hand ...”

Metaphorical instruction: “Imagine you are standing under a cherry tree. Above your head there are all these wonderful cherries. Pick a few cherries with your right hand. “

In the consequent discussion it was clarified that there are different ways to describe activity. One girl (grade 6) put it like that “In the second explanation you just told us about the cherry tree and we *automatically* did the right movements.” The term “metaphor” was never used during the discussion. Instead we used the phrase “describing by telling a story”.

Interpreting algorithms. In the second phase the students had to draw pictures according to given metaphorical algorithms, written like stories. Each student got a working sheet with four tasks. They served as models for the next step, when the participants wrote algorithms on their own. Sometimes the students asked for help. The moderators’ scaffolding was limited to making sure that the students read the text of the algorithm to the end and that they understand the meaning of every word. Typical scaffolding phrases were “Read the first sentence again.”, “Read the whole story from the beginning to the end”. The moderators never judged a student’s solution and never told whether or not a solution was correct. The questions on each task were counted. The number of questions can be considered as an indicator of the level of difficulty. This article focuses on this phase of the workshop.

Creating algorithms. In the next step the students got working sheets with two corresponding images. They had to write a metaphorical algorithm that described how to transform the first image to the second.

Quiz show. At the end of the workshop, the comprehensibility of some student-made algorithms was tested following this pattern: A kid read a story aloud and three class-mates (who did not know this specific problem) tried to draw a sketch on the blackboard according to this.

5 Interpreting Metaphorical Geometrical Algorithm

In phase 2 of the workshop each student got a working sheet with four tasks. A task consists of an image, an empty frame and a story (metaphorical algorithm) that describes how to change the picture. Each story is structured like this (the examples refer to the first task in figure 4).

- The first sentences interpret the given picture metaphorically. Example: “Round paper balls and angular blocks made of iron are lying on the table.”

- Then the text describes some activity using the introduced metaphors. Example: “Somebody has opened the window. The wind has blown away all paper balls. But the iron blocks remain where they are.”
- The last sentence is “How does it look like?”

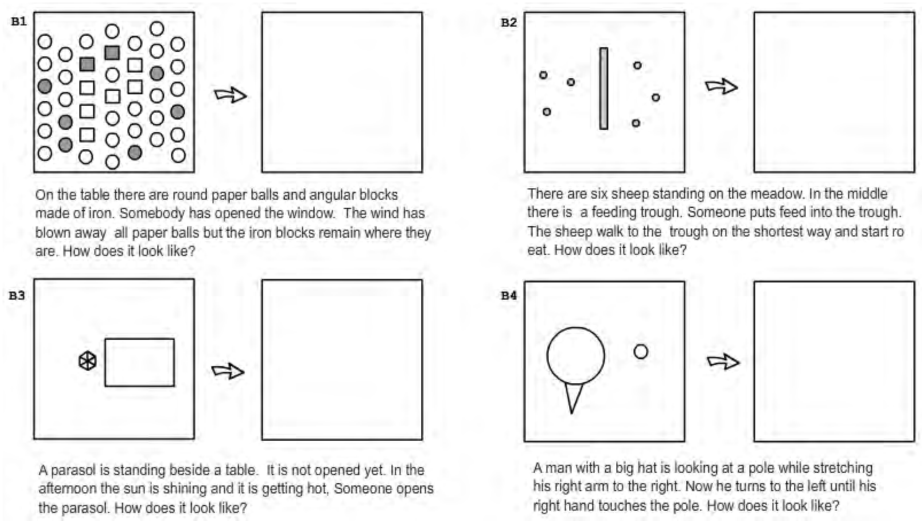


Fig. 4. Tasks from one of the two working sheets

The students had to draw the second picture in the empty frame. The intellectual challenge was to understand the metaphorical algorithm and to perform geometrical operations. At the end they had to judge the difficulty of the tasks by drawing an X at the most difficult and a circle at the easiest task.

We used two different working sheets (A and B) with slightly different tasks, in order to encourage independent working instead of copying the neighbor’s solution. Therefore there were altogether eight different tasks forming four groups with different transformation principles. The examples in the following list are shortened algorithms and refer to working sheet B, which is depicted in figure 4.

- Deleting. Example: The Wind blows away all round paper balls (B1).
- Moving. Example: Animals move to a rectangular feeding trough. (B2).
- Enlarging. Example: Someone opens the parasol standing beside a table. (B3)
  Rotation. Example: A man with a big hat turns to the left until he touches the pole with his right hand (B4).

On the basis of the metaphorical abstraction concept discussed in section 3 one can allocate the tasks to two abstraction levels (abstraction level of task). The algorithms related to deleting and enlarging are on a lower level than the algorithms related to moving and rotation.

6 Results

Altogether 97 students participated at the workshops, including 46 girls and 50 boys (one person did not tell the gender). The average age was 10.8 years.

6.1 Difficulty of the tasks

Two indicators for difficulty were assessed: (1) The students' judging of the tasks by marking the easiest and the most difficult task and (2) the number of students' questions related to a task.

Table 1. The difficulty of the tasks – self-estimation and number of questions

Task theme	Abstraction level of task	easy	difficult	questions
Deleting	low	18	32	8
Moving	high	5	29	9
Enlarging	low	35	8	8
Rotation	high	18	19	8

6.2 Levels of Metaphoric Understanding

Interpreting the given algorithms, the students drew pictures. These pictures document how well the algorithm was understood. On the basis of a qualitative analysis the solutions we introduce five levels of metaphorical understanding:

Incomprehension. A complete misunderstanding of the algorithm results in a meaningless drawing which is not related to the task. An empty picture suggests that the student did not understand the algorithm and did not know what to do. Both cases belong to the incomprehension level.

Literal understanding. On this level the student understands the algorithm in a non-metaphorical way without abstraction. The drawn picture is not geometric but naturalistic. It depicts entities from the source domain of the adopted metaphor (e.g. trees instead of circles; see figure 5, first image).

Basic metaphoric understanding. The student understands principally that the task is to transform (change) a geometric image. Thus the drawing is geometric and not literate regarding the algorithm. But the construction is not accurate. In many cases the accuracy is restricted to one dominating property of the transformed entities (for example the size or shape). Other features (for example position or color) are ignored. This level corresponds to van Hiele's analytic/descriptive level.

Fundamental understanding. The student is able to interpret the metaphorical algorithm in detail. No facet is omitted. The drawing depicts the geometric structure after the transformation.

Integral understanding. The drawing depicts the geometric structure after the transformation and contains additional information about the algorithm. In some cases the paths of moving entities were depicted (figure 5, last image).

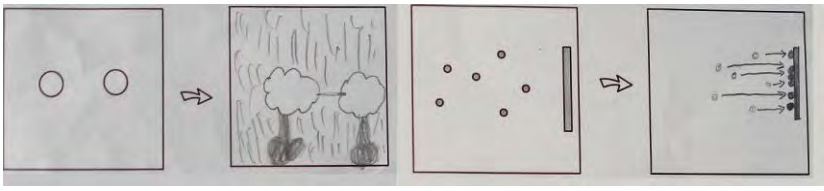


Fig. 5: Examples of algorithm interpretations on different comprehension level. Left: Literal understanding. Right: Integral understanding with additional visualization of the paths.

Table 2. Students' comprehension of metaphorical instructions. ALT means the abstraction level of the task. Comprehension levels are: incomprehension, literal, basic metaphorical, metaphorical and integral understanding

Theme	ALT	Incompre-hension	Literal	Basic metaphor.	Metaphor.	Integral
Deleting	low	9	11	55	20	2
Moving	high	7	13	58	15	3
Enlarging	low	2	10	59	25	0
Rotation	high	16	35	29	17	0
Sum	-	34 (9%)	69 (18%)	201 (52%)	77 (20%)	5(1%)

Some results of the evaluation are displayed in table 2. Obviously the algorithms involving rotation were interpreted more often on a literal level of understanding than the other algorithms. Altogether one might state that most tasks were done on the basic metaphorical understanding level. Considering the age of the students, this result corresponds to the findings by [5], [6] and [7] on the development of geometric thinking.

Table 3 displays the person-related distribution of algorithmic comprehension levels, which was rated on the basis of the overall-performance in four tasks. Comparing 4th- and 6th-grade-students' comprehension levels, it turns out that literal understanding is replaced by basic metaphorical understanding.

Table 3. Students' comprehension of metaphorical algorithms. Comprehension levels: incomprehension, literal, basic metaphorical, metaphorical and integral understanding

Grade	Incompre-hension	Literal	Basic metaphorical	Metaphorical
Grade 4 (n=44)	2%	27%	55%	16%
Grade 6 (n=53)	0%	11%	74%	15%

6.3 Geometrical Accuracy

In psychological and pedagogical literature non-metric (topological, projection-based) and metric aspects of geometrical images have been examined [8], [9].

The *topological* image information is related to the conception of shape. It constitutes the basis of geometric image ontogenesis [8]. Topological errors are shape deformations that are incompatible with the algorithm (figure 6 first image).

Projection-based aspects are the relations between shapes. Projection-related errors are incorrect changes of positions, orientation and layout. Such errors usually show the lack of practical experience related to the context (figure 6 second image).

Metric image information refers to the sizes of shapes or the number of parts a segmented entity consists of. Metric images errors comprise all quantity and size inaccuracies (figure 6 third image).

Fig. 6. Examples of geometric accuracy errors. (1) Topological: shape is not a regular hexagon, (2) projection-based: circles ("animals") should not be exactly at these positions (3) metric: circles are too small.

To realize the visual quantitative properties of a geometric entity it is necessary to identify and summarize its basic properties, taking into account the contribution made by each element. Therefore, the successful formation of metric image information requires a deep understanding of the shape under measurement [9]. Indeed, the results in table 4 show that the metric aspects of the algorithmic image transformation were the most difficult. Not much more than a third of all solutions were correct in this respect.

Table 4. Percentage of correct image formation (388 task solutions)

	Topological	Projection-based	Metric
Correct image formation	78%	62%	37%

6.4 Algorithmic Centration

Jean Piaget [10] had observed that young children (in the preoperational stadium) focus on one aspect of a situation and neglect the others. He calls this tendency *centration*. Geometric inaccuracy might be interpreted as a special form of centration. The students just sketch an idea focusing on the most important and neglecting details. Although being far beyond the preoperational stadium (which is typically in the kindergarden age) many children observed in this study showed a tendency to implement just one aspect of an algorithm while ignoring others. This "algorithmic centration" can be observed comparing the solutions of these two tasks:

1) "There are two trees standing side by side on a meadow. A couple of years later the trees have been grown. Tina connects the trunks with a string. How does it look like?"

2) "There is a parasol standing on the left side of the table. It is it not opened yet. In the afternoon the sun is shining and it gets hot. Somebody opens the parasol. How does it look like?"

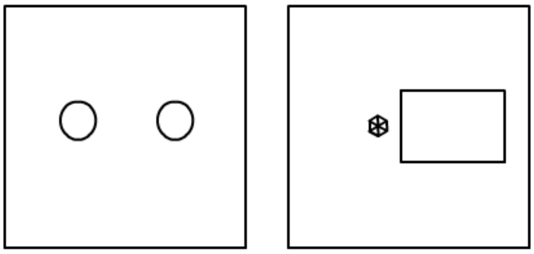


Fig. 7. Starting images for enlargement-related tasks

The algorithms use two different metaphors for enlargement: Growing and opening a parasol. Beside the enlargement of the circles (projections of trees), the first algorithm additionally describes a horizontal line. 33 out of 65 students ignored the enlargement and just drew the line between two circles of the original size (or even smaller). Furthermore, the majority of the students who solved this task (42/65) considered this task to be the easiest of all tasks. In case of the second algorithm the only required transformation was the enlargement of a hexagon. Only two out of 32 students failed to model this.

7 Summary

Software developers use all kinds of metaphors for modeling digital systems. Transferring knowledge from one domain to another is typical for computer science. Therefore, the ability of understanding metaphorical algorithms can be considered as facet of computational thinking. Young students at the age of 10 are able to understand metaphorical algorithms for transforming simple geometric images at least on a basic level. About one quarter of the fourth-graders, who participated in our study, interpreted the given algorithms literally instead of metaphorically. The fraction of literal comprehension dropped to 6/53 in grade six. Students tend to metrical inaccuracy and algorithmic centration. Both can be seen as the ugly side of abstraction. The results underline the expressive power of metaphors. Since even young children are able to build bridges between worlds, this seems to be very natural to human thinking.

References

1. Lakoff, G. & Núnez, R. E.: The Metaphorical Structure of Mathematics: Sketching Out Cognitive Foundations for a Mind-Based Mathematics. In: English, Lyn D. (ed.): Mathematical Reasoning. Analogies, Metaphors, and Images. Lawrence Erlbaum Associates, Publishers, Mahwah, London pp. 21--92 (1997)
2. Lindstrom, L. & Jeffries, R.: Extreme Programming and Agile Software Development Methodologies. In: Information Systems Management Volume 21, Issue 3, pp. 41--52 (2004)
3. Beck, K.: Extreme Programming Explained. Addison Wesley, Boston San Francisco New York Toronto Montreal London Munich Paris Madrid Capetown Sydney Tokyo Singapore Mexico City (1999)
4. Torreano, L. A., Cacciari, C. & Glucksberg, S.: When Dogs Can Fly: Level of Abstraction as a Cue to Metaphorical Use of Verbs. In: METAPHOR AND SYMBOL, *20*(4), 259--274 (2005)
5. Van Hiele, P.M.: Developing Geometric Thinking through Activities that begin with Play. In: Teaching Children Mathematics 6 (February 1999), pp. 310--316 (1999)
6. Fuys D., Geddes D., Tischler R.: The van Hiele Model of Thinking in Geometry among Adolescens. In: Journal for Research in Mathematics Education. Monograph, Vol. 3, pp. i-196. Published by: National Council of Teachers of Mathematics, http://www.jstor.org/stable/749957 (1988).
7. Clements D.H., Sarama J.: Early childhood teacher education: the case of geometry. In: Journal of Mathematics Teacher Education, Vol. 14, Number 2, pp. 133--148, Springer (2011)
8. Jakimanskaja, I.S.: Развитие пространственного мышления школьников. [Development of Pupils' Spatial Thinking], Moscow (1980)
9. Grabauskiene V.: Formation of Geometric Images at Primary School Stages. Summary of Doctoral Dissertation. Vilnius Pedagogical University (2005)
10. Piaget, J. & Inhelder, B.: The Child's Conception of Space. Rutledge & Kegan Paul, New York (1956)

Note

Vaiwa Grabauskiene's fellowship is being funded by European Union Structural Funds project "Postdoctoral Fellowship Implementation in Lithuania" within the framework of the Measure for Enhancing Mobility of Scholars and Other Researchers and the Promotion of Student Research (VP1-3.1-ŠMM-01) of the Program of Human Resources Development Action Plan.

Teaching and Learning Styles in Informatics

Gabriela Andrejková, *gabriela.andrejkova@upjs.sk*
Institute of Informatics, Faculty of Science, P. J. Šafárik University in Košice

František Galčík, *frantisek.galcik@upjs.sk*
Institute of Informatics, Faculty of Science, P. J. Šafárik University in Košice

Ľubomír Šnajder, *lubomir.snajder@upjs.sk*
Institute of Informatics, Faculty of Science, P. J. Šafárik University in Košice

Abstract

At the university we can observe some differences how students prefer to learn. The individual preferences to the learning are called the learning styles. When the students study in some ways that match their preferences, they learn more effectively. The teachers should prepare their teaching styles in the cooperation to student preferences and to do teaching and learning process more effective. The workshop is dedicated primarily for informatics teacher trainers and also informatics teachers.

Main aims of the workshop are:

- to point out of importance of consideration of teaching and learning styles in informatics education,
- to show our experience how to apply theories about teaching and learning styles to teaching programming at university level, and
- to discuss with workshop participants about ways of properly adapting their teaching styles to learning styles of students or pupils.

The workshop consists of three parts. In first part we give some short description of teaching and learning styles and overview of the theories about teaching and learning styles (by Fleming, Kolb, Dunn&Dunn).

In second part workshop participants can fulfil learning and teaching styles questionnaires to know more about themselves and then we discuss about ways of adaptation teaching styles to various learning styles of students. We will analyse the system of questions in the used questionnaire.

In last part we show our experiences with teaching programming at university level with consideration of various learning styles of students and teaching styles of teachers. The main goal is to call some discussion of participants about problems in the area.

Keywords

Informatics education, learning styles, teaching styles, VARK model, university level

Model Checking with Uppaal in a High School Computer Science Course

Nataša Grgurina, *n.grgurina@rug.nl*
University Center for Learning and Teaching, University of Groningen, the Netherlands

Abstract

In traditional Computer Science courses in high school, a lot of time and effort is spent on teaching programming, whereas in industry and academia there is a clear trend showing the increasing importance of modeling. Models can describe system requirements, behavior and functionality of a particular system and can contribute to better understanding of its design, code generation, implementation, and maintenance; as well as aid in the communication with clients and other participants involved in a project. High schools should follow this trend and pay (more) attention to modeling, and in particular, to the construction of models of systems and analysis of these models. [1]

Uppaal is a software tool (free for use in academia) that can model a system describing it as networks of automata:

> *Uppaal is an integrated tool environment for modeling, simulation and verification of real-time systems, developed jointly by Basic Research in Computer Science at Aalborg University in Denmark [3] and the Department of Information Technology at Uppsala University in Sweden [4]. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables. Uppaal consists of three main parts: a description language, a simulator and a model-checker. [2]*

During the workshop the author intends to give a short demonstration of Uppaal, followed by few examples of systems that are suitable for modeling in a high school CS course. She plans to discuss her experiences with teaching modeling with Uppaal to her 11th and 12th grade students; the difficulties they experience when abstract thinking is required (for example for Wolf, Goat and Cabbage riddle) and their enthusiasm when they succeed in modeling an occurrence such as vending machine or having model checker prove that there does (not) exist a solution for a riddle.

Keywords

high school CS course, modeling, automata, state diagrams, Uppaal

References

1. Vaandrager, F., Jansen, D. N., Koopmans, E.: Een Module over Model Checking voor het VWO. In: Vodegel, F., Loots, M. (eds.). NIOC 2009, pp. 135-137. Hogeschool Utrecht, Utrecht (2009)
2. Uppaal, http://www.uppaal.org/
3. Basic Research in Computer Science, http://www.brics.dk/
4. Department of Information Technology, http://www.it.uu.se/

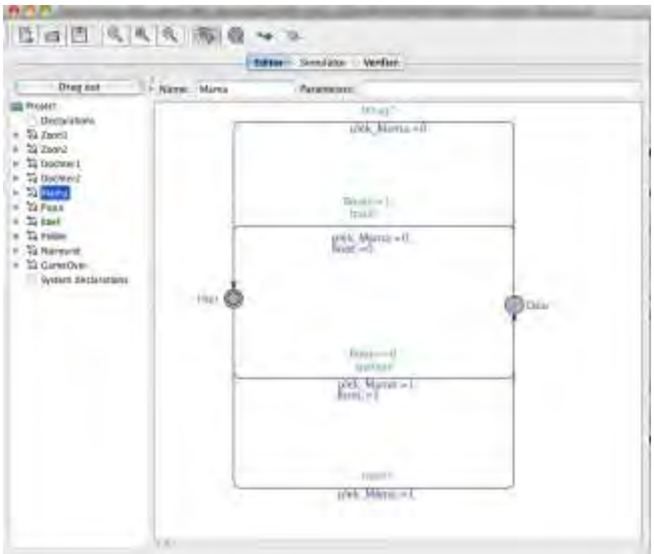


Figure 1.Uppaal editor

Fostering creativity through programming - Scratch workshop

Martina Kabátová, *kabatova@fmph.uniba.sk*
Dept of Informatics Education, Comenius University, Bratislava

Katarína Mikolajová, *mikolajova@fmph.uniba.sk*
Dept of Informatics Education, Comenius University, Bratislava

Abstract

Scratch provides pupils with an easy way to create multimedia projects. If it's properly used, students learn computer programming concepts. The purpose of our workshop is to explore activities and teacher approaches that enable pupils to express their ideas through programming and develop their creativity during this process. We will focus on multimedia projects and games that provide pupils with an opportunity to engage their imagination and at the same time give them some freedom to express themselves in a unique way. We will try to introduce some divergent activities with open-ended results and various approaches for solving problems.

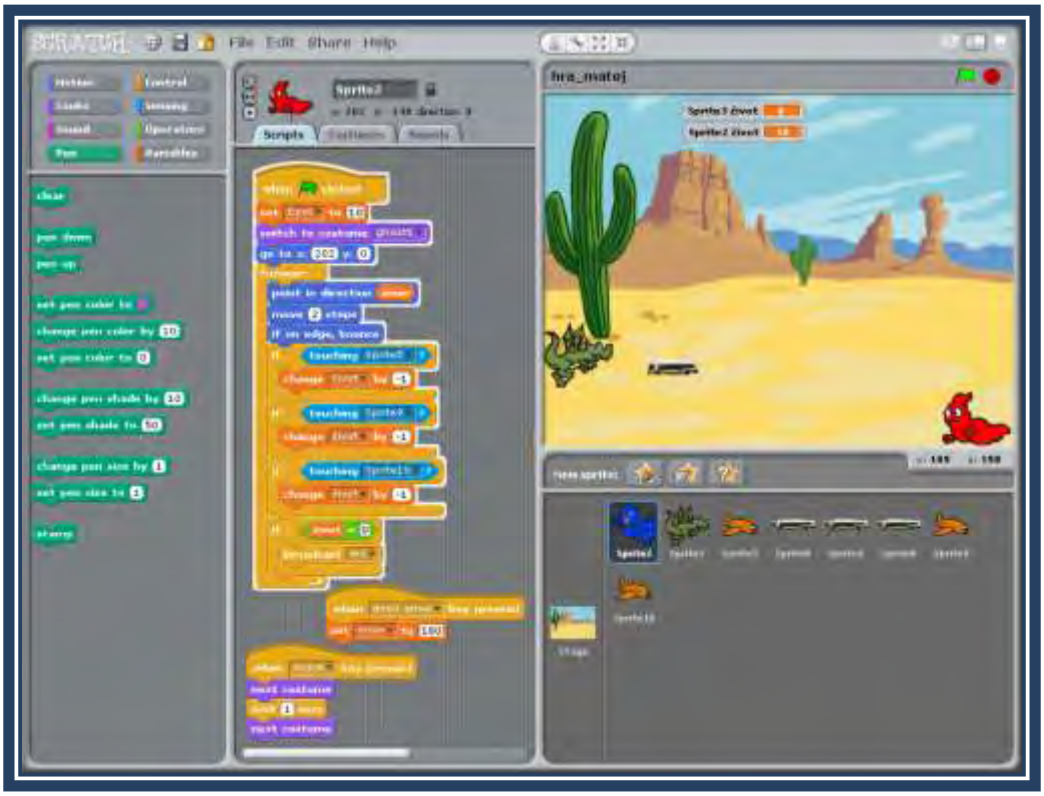


Figure 1. Scratch – programming tool for pupils

During the workshop we will discuss creative approaches to solving selected tasks and projects.

Keywords

programming, creativity, multimedia projects

Music structure and computer science concepts

Illustrating CS ideas by creating and modifying music

Erich Neuwirth, *erich.neuwirth@univie.ac.at*
University of Vienna, Computer Science Didactics and Learning Research Center

Abstract

When teaching computer science concepts, it is always difficult to find sets of problems attracting the students' attention. The workshop will give the participants the chance of experimenting with musical ideas (and, more generally, ideas about creating sounds) with a set of easy to use tools to create and manipulate structure. Writing a score for a musical piece can be seen as writing a program, and identifying musical structures can be seen as identifying structures in programs. Representing music in different notations (musical score, piano roll, numerical representations, MIDI) also naturally leads to discussions about the nature of different representations.

We use a set of two toolkits. The first one is spreadsheet based. Building on the ubiquitous spreadsheet paradigm students can immediately create their own musical pieces and listen to them. The toolkit allows musical operations like repeating, transposing or changing the tempo of musical phrases.

As the musical ideas get more complex, the spreadsheet paradigm becomes too narrow, and therefore we can introduce a more powerful LOGO based toolkit. With this toolkit, the computer science concepts like data structures and even unit testing can be introduced.

The toolkit has been tested with high school and undergraduate students quite successfully. It seems that music the theme of a CS introductory course is able to attract the attention of students in a natural way.

The combination of the spreadsheet toolkit and the more abstract LOGO toolkit allows designing activities according to the ideas of low floor - high ceiling. Students create music they like within the first 15 minutes of activities, and there is almost no limit for the complexity of the music ideas they can implement.

The toolkits are open source and available for download.

Keywords

Computer science, didactics, music, notation

TurtleArt

Artemis Papert, *artemis@turtleart.org*
Independent artist, Montreal, Canada

Abstract

TurtleArt is a microworld for exploring art through turtle geometry. It brings programming and art together. The main focus of TurtleArt is static artistic images.

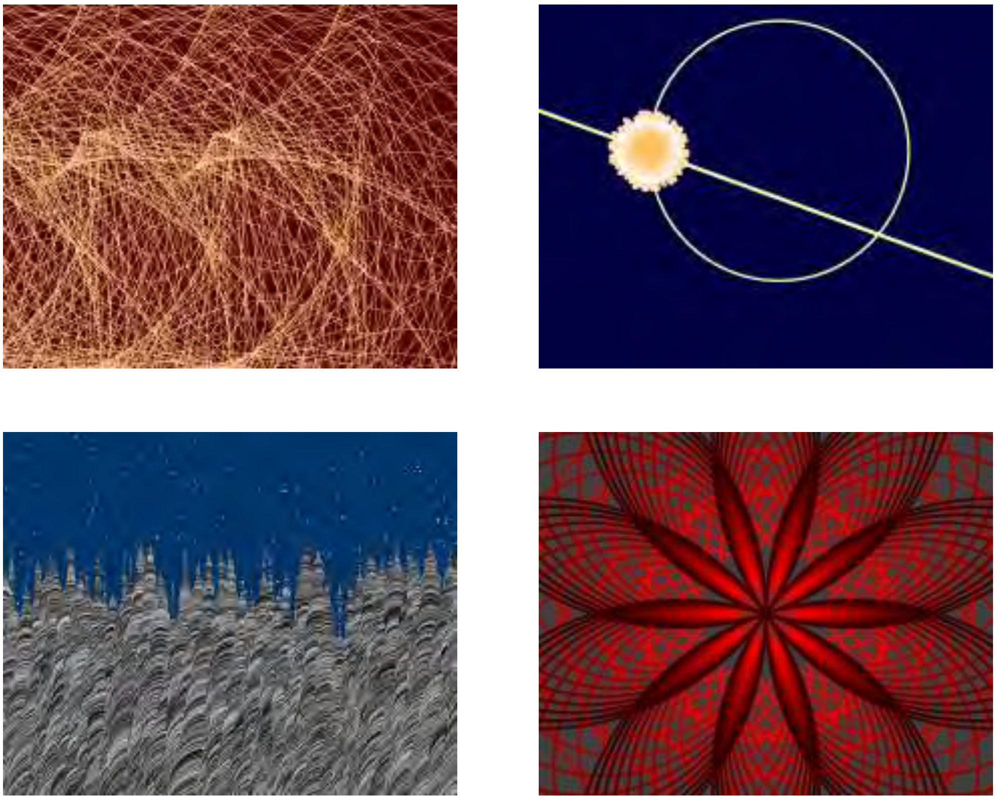


Figure 1. Some TurtleArt images.

TurtleArt programmes are created by snapping together blocks. It borrows from the earliest versions of LOGO by having a vocabulary centred around Turtle Geometry. The vocabulary of TurtleArt is small, therefore fluency can be reached fairly quickly.

In this workshop you will get an introduction to programming in TurtleArt but mainly you will be given lots of hands on time to do your own exploration and create your own images.

Bring your laptop along.

Keywords

TurtleArt, turtle geometry, LOGO, constructionism, art, programming, fluency.

Program by Design

Viera Krňanová Proulx, *vkp@ccs.neu.edu*
College of Computer & Information Science, Northeastern University, Boston, MA, USA

Abstract (style: Abstract title)

The goal of *Program By Design* methodology is to teach children how to solve problems by following a systematic design process, regardless of the programming environment. The *Design Recipes* organize the design process into small steps that allow both the learner and the teacher to monitor the progress and to verify the learner's understanding of the problem.

The workshop will present the key ideas of the *Program By Design* methodology illustrated on the curriculum used with students from 6th grade of elementary school all the way though the introductory university.

Children and students design the behaviour of interactive games: the functions that represent the movement of the game objects, the predicates that check for collision of objects, or whether an object has reached the desired destination, the functions that update the game score or other measure of the game progress, and the functions that place the game objects onto the game display. The design of these functions is a simple application (or practice) of algebra and geometry knowledge.

Figure 1. Frogger game and a Mario-like game designed by students.

The *Bootstrap* level is designed for children in grades 6 through 8, the *How to Design Programs* level targets the secondary school students as well as introductory university level, the *How to Design Classes* extends the curriculum to reach advances secondary school students as well as university students.

We will first present the general framework for *Program by Design*, show the power of the *Design Recipes*, and follow with a hands-on session individualized for the participants. The participants can work on that segment that is most applicable to the level of their students.

Keywords

programming; didactics of design

Creating interactive board games with Easy Logo

Maria Skiadelli, *skiadelli@gmail.com*
Deptof Electrical and Computer Engineering, National Technical University of Athens

Abstract

EasyLogo is an environment based on common Logo principles which mainly focuses on teaching basic programming techniquesto novices and young children. This workshop is about creating interactive board games with the EasyLogo environment.

Interactive board games are not very similar to the typicalelectronic games where there is some action that takes place throughout the game, whilst the player (human or the computer itself) can control this action to achieve certain goals. They are more like the common board games played by several human players (1-2 or more) whocompete with each other following certain rules. We claim that when implemented as software products, these gamesmay become more interesting since they acquire some degree of interactivity that is missing from the classical board gamesfound on a singlepiece of paper.Moreover, these games not only can be implemented by programming but they can also be played by programming, which is a very good practice, especially for the very beginners of the programming field. EasyLogo, although simple to use, processes some interesting features that allow the implementation of severalinteresting board game ideas.

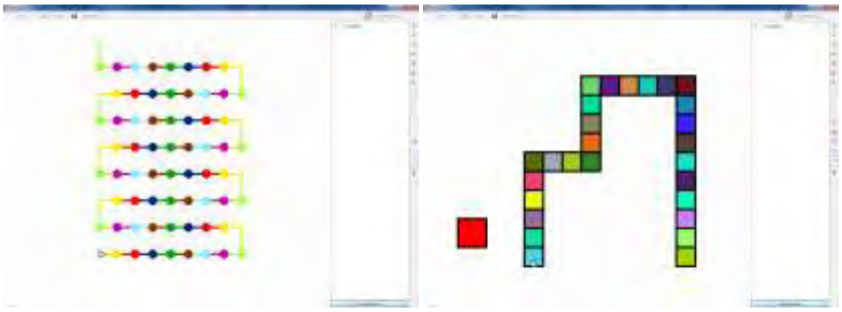


Figure 1.Screenshots of interactive board games created with EasyLogo

In this workshop we will first showsome pre-made examples of interactive board games in order to explain the basic idea and philosophy behind the creation of such games and to give inspiration to the participants. Technical details of the environment will also be explained. Then the participants will form groups (of 2 or 3 members) and each group will have to think of its own game idea andimplement itwithEasyLogo. While creating such a game,the groupswill have to take decisionsthat need critical and creative thinking: the rules of the game, how many players can play, who wins and when, how to provide (if needed) randomness and many others, concerning design, aesthetical and structural issues. A worksheet will help them not to miss any of these important factors. The implementation of the game is an iterative process;the members of each group have to play it while implementing it, by using programming again, in order totestits functionality, to see if it is interesting enough, etc. When finished, they will share it with other groups. Playing other people'sgamesis not only good for feedback, butit can also begreat fun.

Keywords

board games, logo, programming, ICT teaching