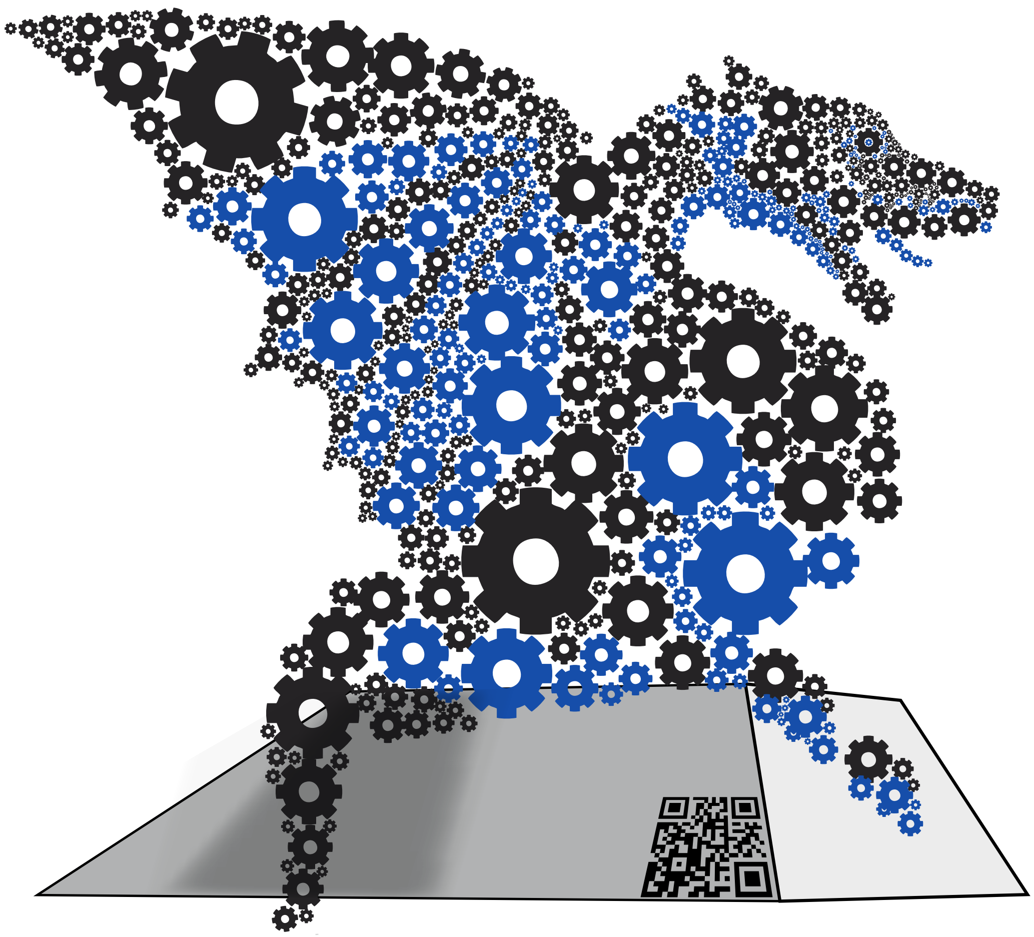
University *of Ljubljana*
Faculty *of Computer and Information Science*

International Conference on Informatics in Schools

ISSEP 2015

September 28 - October 1, Ljubljana, Slovenia

Proceedings



http://issep15.fri.uni-lj.si

Title:
The Proceedings of International Conference on Informatics in Schools: Situation, Evolution and Perspectives — ISSEP 2015.

Publisher:
University of Ljubljana, Faculty of Computer and Information Science

Editor:
Matevž Jekovec

Co-editors:
Regular Papers: Andrej Brodnik and Jan Vahrenhold
Poster Session: Matija Lokar
Workshops: Peter Micheuz
International Teacher's Conference: Peter Micheuz, Barbara Demo, Claudio Mirolo

Web URL:
`issep15.fri.uni-lj.si`

Conference Organizer:
University of Ljubljana, Faculty of Computer and Information Science

Publisher:
Založba FRI, Večna pot 113, Ljubljana

This publication is free of charge.

Ljubljana, October 2015

CIP — Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004:37(082)(0.034.2)
659.2:004:37(082)(0.034.2)
37.091.3:004(082)(0.034.2)

INTERNATIONAL Conference on Informatics in Schools (8 ; 2015 ; Ljubljana)
Proceedings [Elektronski vir] / International Conference on Informatics in Schools - ISSEP 2015, September 28 - October 1, Ljubljana, Slovenia ; [editor Matevž Jekovec ; organizer University of Ljubljana, Faculty of Computer and Information Science]. - El. knjiga. - Ljubljana : Založba FRI, 2015

Način dostopa (URL): `https://issep15.fri.uni-lj.si/files/issep2015-proceedings.pdf`. - Nasl. z nasl. zaslona. - Opis vira z dne 9. 10. 2015

ISBN 978-961-6209-87-8 (pdf)
1. Jekovec, Matevž, 1986- 2. Fakulteta za računalništvo in informatiko (Ljubljana) 1536545987

Foreword to Information Society 2015

In its 18th year, the Information Society Multiconference (http://is.ijs.si) remains one of the leading conferences in Central Europe devoted to information society, computer science and informatics. In 2015 it is extended over three weeks located at Faculty of computer science and informatics and at the Institute "Jožef Stefan".

The pace of progress of information society, knowledge and artificial intelligence is speeding up. Several countries allow autonomous cars in regular use, major car companies sell cars with lane assist and other intelligent functions. It seems that humanity is approaching another civilization stage. At the same time, society conflicts are growing in numbers and length.

The Multiconference is running in parallel sessions with 300 presentations of scientific papers at twelve conferences, round tables, workshops and award ceremonies. The papers are published in the conference proceedings, and in special issues of two journals. One of them is Informatica with its 38 years of tradition in excellent research publications.

The Information Society 2015 Multiconference consists of the following conferences:

- Intelligent Systems
- Cognitive Science
- Data Mining and Data Warehouses
- Collaboration, Software and Services in Information Society
- Education in Information Society
- Facing Demographic Challenges
- Cognitonics
- SPS EM-Health Workshop
- Workshop "Smart Cities and Communities as a Development Opportunity for Slovenia"
- 2nd Computer Science Student Conference, PhD Students
- 2nd Computer Science Student Conference, Students
- ISSEP15 – 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspective.

The Multiconference is co-organized and supported by several major research institutions and societies, among them ACM Slovenia, i.e. the Slovenian chapter of the ACM, SLAIS and the Slovenian Engineering Academy. In the name of the conference organizers we thank all societies and institutions, all participants for their valuable contribution and their interest in this event, and the reviewers for their thorough reviews.

For 2013 and further, the award for life-long outstanding contributions will be delivered in memory of Donald Michie and Alan Turing. The life-long outstanding contribution to development and promotion of information society in our country is awarded to Dr. Jurij Tasič. In addition, a reward for current achievements was pronounced to Dr. Domnu Mungosu. The information strawberry is pronounced to the web application "Supervizor, while the information lemon goes to lack of informatization in the national judicial system. Congratulations!

28th September, 2015

Niko Zimic, Programme Committee Chair
Matjaž Gams, Organizing Committee Chair

Predgovor multikonferenci Informacijska družba 2015

Multikonferenca Informacijska družba (http://is.ijs.si) je z osemnajsto zaporedno prireditvijo osrednji srednjeevropski dogodek na področju informacijske družbe, računalništva in informatike. Letošnja prireditev traja tri tedne in poteka na Fakulteti za računalništvo in informatiko in Institutu "Jožef Stefan".

Informacijska družba, znanje in umetna inteligenca se razvijajo čedalje hitreje. V vse več državah je dovoljena samostojna vožnja inteligentnih avtomobilov, na trgu je moč dobiti čedalje več pogosto prodajanih avtomobilov z avtonomnimi funkcijami kot "lane assist". Čedalje več pokazateljev kaže, da prehajamo v naslednje civilizacijsko obdobje, hkrati pa so konflikti sodobne družbe čedalje težje razumljivi.

Letos smo v multikonferenco povezali dvanajst odličnih neodvisnih konferenc. Predstavljenih bo okoli 300 referatov v okviru samostojnih konferenc in delavnic, prireditev bodo spremljale okrogle mize in razprave ter posebni dogodki kot svečana podelitev nagrad. Referati so objavljeni v zbornikih multikonference, izbrani prispevki pa bodo izšli tudi v posebnih številkah dveh znanstvenih revij, od katerih je ena Informatica, ki se ponaša z 38-letno tradicijo odlične znanstvene revije.

Multikonferenco Informacijska družba 2015 sestavljajo naslednje samostojne konference:

- Inteligentni sistemi
- Kognitivna znanost
- Izkopavanje znanja in podatkovna skladišča
- Sodelovanje, programska oprema in storitve v informacijski družbi
- Vzgoja in izobraževanje v informacijski družbi
- Soočanje z demografskimi izzivi
- Kognitonika
- Delavnica "SPS EM-zdravje"
- Delavnica "Pametna mesta in skupnosti kot razvojna priložnost Slovenije"
- Druga študentska konferenca s področja računalništva in informatike za doktorske študente
- Druga študentska konferenca s področja računalništva in informatike za vse študente
- ISSEP15 – Osma mednarodna konferenca o informatiki v šolah: razmere, evolucija in perspektiva.

Soorganizatorji in podporniki konference so različne raziskovalne institucije in združenja, med njimi tudi ACM Slovenija, SLAIS in Inženirska akademija Slovenije. V imenu organizatorjev konference se zahvaljujemo združenjem in inštitucijam, še posebej pa udeležencem za njihove dragocene prispevke in priložnost, da z nami delijo svoje izkušnje o informacijski družbi. Zahvaljujemo se tudi recenzentom za njihovo pomoč pri recenziranju.

V 2015 bomo tretjič podelili nagrado za življenjske dosežke v čast Donalda Michija in Alana Turinga. Nagrado Michie-Turing za izjemen življenjski prispevek k razvoju in promociji informacijske družbe bo prejel prof. dr. Jurij Tasič. Priznanje za dosežek leta je pripadlo dr. Domnu Mungosu. Že petič podeljujemo nagradi "informacijska limona" in "informacijska jagoda" za najbolj (ne)uspešne poteze v zvezi z informacijsko družbo. Limono je dobilo počasno uvajanje informatizacije v slovensko pravosodje, jagodo pa spletna aplikacija "Supervizor". Čestitke nagrajencem!

28. september 2015

Niko Zimic, predsednik programskega odbora
Matjaž Gams, predsednik organizacijskega odbora

Table of Contents

Regular papers

Preface 1

List of accepted Papers to LNCS 9378 3

Computer Science Competences in Italian Secondary Schools: a Preliminary Study 4
Silvio Giaffredo, Luisa Mich, Marco Ronchetti1

A language independent assessment of programming concepts knowledge 13
Franc Jakoš, Matija Lokar

Aspects of Quality in the Presentation of Informatics Challenge Tasks 21
Wolfgang Pohl, Hans-Werner Hein

Tasks Classification and Age Differences in Task Perception. Case Study of International On-line Competition "Beaver" 33
Ekaterina Yagunova, Sergey Podznyakov, Nina Ryzhova, Evgenia Razumovskaia, Nikolay Korovkin

Posters

Preface 44

Informatics Teacher Education in Turkey 45
Okan ARSLAN, Selcan KİLİS

YASAAPE – Yet Another System for Automatic Assessment of Programming Exercises 47
Gregor Jerše, Sonja Jerše, Matija Lokar, Matija Pretnar

Learning to Program: from VPL to C 49
Greg C Lee, Ling-Chian Chang

Publication of Learning Resources: Central or Interoperable? 51
Paul Libbrecht

Concurrent programming basics through Snap! 53
Michele Moro, Luigino Calvi

Can I do that? Scenario Feasibility as an Enabler of ICT Usage 55
Wolfgang Müller, Paul Libbrecht

From Paper to Web – Some Help from PirateBox 57
Martina Palazzolo, Paolo Mauri

Computer Science for All in Swiss High Schools: Current State, Issues, and Perspectives 59
Jean-Philippe Pellet, Gabriel Parriaux, Morgane Chevalier

Content Categories for Informatics Tasks 61
Wolfgang Pohl, Jörg Westmeyer

The use of Nao, a humanoid robot, in teaching computer programming 63
Boštjan Resinovič

The influence of teaching methods during technical e-safety instruction 65
Václav Šimandl, Václav Dobiáš, Michal Šerý

Teaching Programming Indirectly with "Paint" 67
Zsuzsanna Szalayné Tahy

Workshops

Preface ... 69

Teaching Software Engineering in Primary and Secondary Schools ... 70
Peter Antonitsch, Andreas Bollin, Stefan Pasterk, Barbara Sabitzer

Learning Computational Thinking through Bebras Tasks ... 71
Valentina Dagiene, Gerald Futschek

A web service for teaching programming ... 72
Gregor Jerše, Sonja Jerše, Matija Lokar, Matija Pretnar

The International Teacher's Conference

Preface ... 73

A cautious look at coding in primary education ... 74
Peter K. Antonitsch

The cat, the turtle, the snake and GCD ... 81
Agnieszka Borowiecka, Katarzyna Olędzka

Roboval: Robot contest and education with Arduino in high school ... 92
Maurizio Boscaini, Alberto Valente

Blended learning environments, flipped class and collaborative activities to teach databases in a secondary technical school ... 99
Maria Concetta Brocato

Active learning in a "Introduction to networks" course ... 111
Sophia Danesino

Introducing recursion with LOGO in upper primary school ... 121
Ágnes Erdősné Németh

Experiences of the T4T group in primary schools ... 130
Fabrizio Ferrari, Alessandro Rabbone, Sandro Ruggiero

Preparing programming exercises with efficient automated validation tests ... 142
Gregor Jerše, Sonja Jerše, Matija Lokar, Matija Pretnar

Selected spotlights on informatics education in Austrian schools ... 153
Peter Micheuz, Barbara Sabitzer

Algorithms and well formatted texts: Introducing computer science activities in lower secondary schools .. 164
Martina Palazzolo

ZaznajSpoznaj – a modifiable platform for accessibility and inclusion of visually-impaired elementary school children ... 173
Matevž Pesek, Daniel Kuhl, Matevž Baloh, Matija Marolt

8th International Conference on Informatics in Schools — ISSEP 2015

This year, the 8th International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP 2015) held at the University of Ljubljana, Slovenia, between September 28th and October 1st, 2015, was rather special. It was federated with a very successful Teacher Conference (VIVID) and therefore represented a wider forum to deliver fresh knowledge, ideas, and reports in direct classroom experiences.

ISSEP is on its own a forum for researchers and practitioners in the area of Informatics education, in both primary and secondary schools (K-12 education). The topics discussed at the conference are various and span from curriculum design and country reports to issues rather special on how to approach teaching programming and/or bring computational thinking to classroom. This colourfulness is also reflected in papers presented at this year's conference. The ISSEP series started in 2005 in Klagenfurt (Celovec) and was followed by meetings in Vilnius (2006), Toruń (2008), Zürich (2010), Bratislava (2011), Oldenburg (2013), Istanbul (2014) and this year at the University of Ljubljana, Faculty of Computer and Information Science.

We received 36 submissions for the conference and each submission was reviewed by up to four reviewers and evaluated on quality, originality, and relevance. Slightly less than half of the papers (17) were accepted for the presentation at the conference. Out of 17 accepted submissions, 14 were chosen to be published in a volume of Lecture Notes on Computer Science (LNCS 9378), Informatics in Schools – Curricula, Competences, and Competitions, published by Springer.

We want to thank once more to the Programme committee members for their diligent and great work that made this conference possible. We hope you will enjoy the conference and papers at least as much as we did.

25th September, 2015

Andrej Brodnik, General Chair
University of Ljubljana

Jan Vahrenhold, Program Chair
Westfälische Wilhelms-Universität Münster

Program Committee:

- Erik Barendsen, Radboud University Nijmegen and Open Universiteit
- Andrej Brodnik (chair), University of Ljubljana and University of Primorska
- Michael Caspersen, Aarhus University
- Valentina Dagiene, Vilnius University
- Barbara Demo, Universitá di Torino
- Ira Diethelm, Carl-von-Ossietzky-Universität Oldenburg
- Kathi Fisler, Worcester Polytechnic Institute
- Yasemin Gülbahar, Ankara University
- Juraj Hromkovič, ETH Zürich
- Peter Hubwieser, Technische Universität München
- Peter Micheuz, Alpen-Adria-Universität Klagenfurt

Springer LNCS 9378: Informatics in Schools

Accepted papers to the LNCS 9378: Informatics in Schools, Curricula, Competences, and Competitions, published in 2015 by Springer, are:

1. Surprising Computer Science (invited talk)
   Tim Bell

2. The Theory Behind Theory - Computer Science Education Research Through the Lenses of Situated Learning (invited talk)
   Maria Knobelsdorf

3. Robotics Activities–Is the Investment Worthwhile?
   Ronit Ben-Bassat Levy and Mordechai (Moti) Ben-Ari

4. Dimensions of Programming Knowledge
   Andreas Mühling, Peter Hubwieser, and Marc Berges

5. Defining Proficiency Levels of High School Students in Computer Science by an Empirical Task Analysis Results of the MoKoM Project
   Jonas Neugebauer, Johannes Magenheim, Laura Ohrndorf, Niclas Schaper, and Sigrid Schubert

6. Classification of Programming Tasks According to Required Skills and Knowledge Representation
   Alexander Ruf, Marc Berges, and Peter Hubwieser

7. Online vs Face-To-Face Engagement of Computing Teachers for their Professional Development Needs
   Sue Sentance and Simon Humphreys

8. Programming in Scratch Using Inquiry-Based Approach
   Jiří Vaníček

9. Olympiad in Computer Science and Discrete Mathematics
   Athit Maytarattanakhon, Vasiliy Akimushkin, and Sergei Pozdniakov

10. CS Unplugged: Experiences and Extensions
    Irena Demšar and Janez Demšar

11. Computing at School in Sweden – Experiences from Introducing Computer Science within Existing Subjects
    Fredrik Heintz, Linda Mannila, Karin Nygårds, Peter Parnes, and Björn Regnell

12. A Snapshot of the First Implementation of Bebras International Informatics Contest in Turkey
    Filiz Kalelioğlu, Yasemin Gülbahar, and Orçun Madran

13. Introducing a New Computer Science Curriculum for All School Levels in Poland
    Maciej M. Sysło and Anna Beata Kwiatkowska

14. Analyzing the Twitter Data Stream Using the Snap! Learning Environment
    Andreas Grillenberger and Ralf Romeike

15. Is Coding the Way to Go?
    Violetta Lonati, Dario Malchiodi, Mattia Monga, and Anna Morpurgo

16. Visual Literacy in Introductory Informatics Problems
    Françoise Tort and Béatrice Drot-Delange

Computer Science Competences in Italian Secondary Schools: a Preliminary Study

Silvio Giaffredo[1], Luisa Mich[2] , Marco Ronchetti[1]

[1] DISI – University of Trento

[2] DII – University of Trento

(silvio.giaffredo, luisa.mich, marco.ronchetti)@unitn.it

Abstract. To enable a more effective process of learning and teaching, the pedagogic research and many educational institutions suggest an approach by competence. This approach is not yet widespread in the classes of Computer Science. This paper describes a study on Computer Science competences in Italian secondary schools. The study is the first step of an ongoing research project, whose goal is to develop an environment for the support of teaching by competences. To this end, a survey was run to gather data about the adoption of the competence approach in Italian secondary schools, among Computer Science teachers of 11 to 13 grade classes. The survey results are illustrated in the paper, along with the work-plan for developing the further steps of the research.

Keywords. Computer science education, informatics education, secondary schools, education by competence.

1 Introduction

Many aspects of the competence-based approach have been investigated in recent years. Competence is defined as "a combination of knowledge, skills and attitudes appropriate to the context" [16]. The competence-based approach is widely used in the professional training activities, and its extension to the educational systems has been recommended by the European Institutions [16]. Sets of competences have been defined, mainly concerning general "top" level competences, and also some examples of discipline-specific competences. This paper describes the preliminary results of a research in the field of Computer Science Education, at the level of the upper secondary school, grade 11 to 12 (in Italy 13), with the focus on the technical schools. In a preliminary survey, illustrated here, we tried to understand if other approaches, as the content-driven approach, are still prevalent in the Computer Science (CS for short) teaching activity, in spite of the recommendation towards the competences. In order to deal with this problem, we propose to build a teaching environment and a plan for teachers training.

This paper is organised in several sections. Section 2 summarises the state of the art in education by competence. In Section 3, the problem statement has been detailed. The study goals have been presented in Section 4. Section 5 shows some preliminary results. Conclusions and future steps of the research are then described.

2 State of the art in CS education by competence

In order to build a structure of reference for the research, we introduce several but relevant issues from the vast field of the pedagogy, even though we will have just the space to list the most important pedagogical theories in a very short summary. Further on, we will also study the specific concept of competence deeper.

Many theoretical pedagogical approaches have been developed. We will try, following the track depicted in [2], to focus on the most relevant theories of learning. For the learning process the three main theories are behaviourism, cognitivism and constructivism. According to the behaviourism, the reality is objective, external to the individuals and we can only know the facts of the reality through the experience, which changes the learners' behaviour; then, the learning is measurable, observing the changes in behaviour of the learner. The focus is on the content, and nobody can say anything about "what is going on in the learner's head" (on p. 19 in [2]). The cognitivism, on the opposite, claims that the learning outcomes depend also on an interpretation of the reality. Personal characteristics of the individuals influence their internal process of learning, which involves also the meta-cognition of the learners, that is "their knowledge and cognition about cognitive phenomena" (in [8] p. 906). According to the constructivist approach, learning is a process able to build the knowledge inside the individuals, who are trying to understand the world, in a certain social and cultural environment. The construction of the personal knowledge depends on the individual situation, and follows these three steps: observation, processing of the information collected, and interpretation, that is the re-construction of the previous knowledge of the learners. The constructivist teachers maintain the role of leadership and guide, but in order to "encouraging and orienting the students' constructive effort" (p. 26 in [19]). The principles of the constructivist approach to the learning process are the basis for an educational system addressing the development of the competences.

The competence approach to education is promoted as an effective way to learning and teaching, in many countries of the world and at different levels. At the end of the 1950s the term "competence" was used by White in [20], with the broad meaning, derived by biology and then relevant also to a human subject, of "capacity to interact effectively with its environment" (by [20] p. 297). Learning can help to preserve the results of reciprocal effects of interaction between the individuals and the environment: in this way, their competence to deal with the environment might increase.

Narrowing the focus on the European countries, the competence-based approach started to be adopted after the year 1980. A useful summary has been recently proposed by Le Deist and Winterton [11], as reported in [14]. The authors include - with the three traditional competences also famous as KSA (knowledge, skills, attitudes) – the

meta-competence, that is the personal competence which makes individuals aware of their own competence. The multidimensional model has been also called holistic model of competence (see [21] p. 691) and represented as a tetrahedron, to highlight that meta-competence has to be based on the other three traditional competences.
In the year 2006, the European institutions defined a set of competences for European citizens [16]. The recommendation of the key competences, also known as general competences, has been generally acquired by the individual member states and included into the curricula of European countries, covering the mandatory levels of education, up to grade 10. Among the eight key competences, the digital competence is considered a priority to ensure the digital inclusion into the knowledge society, for the European citizens. The DIGCOMP project proposed a digital competence framework in [15], with the goal of supporting the development of this competence by all the European citizens.
In contrast to the key competences, for the competences of the single subject disciplines, also named the subject-specific or domain-specific competences, a commonly accepted definition has not yet been pointed out for all disciplines. The TUNING project [18] is concerned with the subject-specific competences, but limited to the higher level of education, the University level. One of the parts of the TUNING project is the FETCH initiative [7], specialised in CS competence, at the level of tertiary education. As long as the digital competence is one of the key competences, the CS competences are subject-specific, directly related to the CS discipline. Also ACM and IEEE have released important guidelines to define a CS curriculum [1]; again in this case, the issues addressed by the work are both University-oriented and content-driven.
In Italy, the Ministry for Education listed the competences for the secondary schools in the New Organisation Structure (see [10]). This contains a set of documents, in which the different branches of the secondary schools – liceo (general education), technical school and vocational school - are described. Particularly for the range from 11 to 13 grade, there are definitions of competences for the disciplines, in some cases detailed for the different grades. Abilities and knowledge are also included, but they are not directly linked with the defined competences.

3 Adoption of CS education by competence in Italian secondary schools

The focus of this study is mainly on the technical schools, since they offer a meaningful range of competences in informatics, both in quality and in quantity terms. Two different sectors are defined: technical-economic and technical-technological schools. The competences, abilities and knowledge defined for the two kinds of schools are not homogeneous, and several disciplines are included in the CS group.
We are focusing on the problem of the adoption of the competence approach, in CS classes of the target schools. There are no quantitative analysis, related to the use of this approach. There are no available official survey, describing how spread the adoption is. Qualitative indicators may help to understand the situation. To collect meaningful data, we worked with different kind of sources: literature references, direct

contacts with institutional subjects, and qualitative researches among the teachers. First of all, some evidences from Italian literature describe a general sense of uncertainness for the teachers, in adopting the competence-based approach. According to Bottani [5], the teachers don't have suggestions on how to teach. And Pellerey claims that the institutional concept of competence lacks an adequate semantic and operative framework [13]. The educational institutions of some countries have stated an official definition of subject-specific competences. But in many cases, the schools and the teachers don't seem so eager in adopting this definition. Often, the competences are depicted through generic descriptions. The teachers have the hard work of interpreting the competences in the different, real situations. Also in the cases when the competences have been defined, there are no suggestions, how to use them for the teaching activity. Teachers are free to choice. Or, in other words, they are alone. Furthermore, the traditions and the habits of the school systems are mainly content-driven: then, 'competence' is considered just a word with no real meaning and no practical consequences. Thus, educators are experiencing in the daily work how difficult it is to keep their activities in the track of a planned, competence-based learning design.
The second source of data is a local institutional research institute, in charge for the biggest part of the teachers training in our small geographical area. The research institute didn't launch any official survey, in order to test the teaching effects of its wide training action in the field of competences. Nevertheless, the involved researchers are ready to extend the number of interventions on the territory, not only to enlarge the number of teachers to train, but also to give an extra support to trained teachers. The schools are now able to release to their pupils the competence certification, settled for the grade 10. But, according to a general sensation of “not completed” training action by the research centre, the process of defining the levels for every competence of every pupils is not so clear and easy to manage for the teachers.
As our third kind of data, we started also a phase of qualitative research among the teachers, using two different techniques: focus-groups and interviews. The focus-groups involved around sixty teachers of different disciplines, attending a meeting held in Trento last November to show the intermediate results of the eSchooling project. This is an industrial project (more details in [6]), run by public and private entities: Telecom Italia SpA, the publisher Edizioni Centro Studi Erickson SpA, two SMEs (Memetic Srl and ForTeam Studio Srl) and the University of Trento. The project is co-funded by the Autonomous Province of Trento and will last till July 2015. The data collected by the focus groups confirm that the teachers suffer for lack or for fuzzines of the guidelines, declaring also a low level of trust on the competence proponents. And if we could expect an understandable and common resistance to change, it is remarkable the teachers belief of more work and efforts required by the different approach. In order to understand what is the use of the competence-based approach to teaching, particularly of the CS competences among secondary teachers in Italian technical schools, we interviewed seven CS teachers. Again, the collected data reconfirm a certain distance from the competences defined by the institutions, complaining also for the time-consuming, bureaucratic mission. Along with critical words, we also recorded the acknowledgement that competence-based approach is relevant for developing students' knowledge, skills, attitudes.

Now we are able to explicit our research question: how can we help the CS teachers to move towards the competence-based education? In order to find out an answer, we divide the main question into two sub-questions: which repertory of competences is suitable for CS teachers? And which methods can support and spread the adoption of the competence-based CS teaching?

4 Study Goals

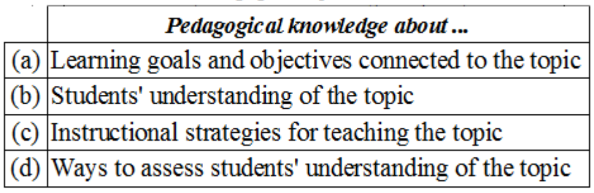
The research project will try to point out a practical solution, in order to reduce the gap between an institutional definition of competences and a limited adoption in teaching, aiming at supporting the day-by-day activities of the teachers.
The competence-based approach implies an attitude, shared among all the teachers of a student. With a collaborative work, every teacher should support the competences development of their learners. The collaborative work makes easier and more meaningful for the teachers to follow the evolution of their students competences, compared to simply continuing to measure the amount of disciplinary contents, collected by the students. The competence-based approach seems to lead towards a collective action for groups of teachers. Nevertheless, when the Learning Units of a specific discipline are designed, built, and assigned with the competences in mind, rather than with regard to the pure contents, this approach can offer a meaningful advantage to the learning process, also for the individual teachers and for their individual disciplines. For this reason, the objective of our research can have a positive impact, even if it is restricted only to the subject specific of Informatics and to the Informatics teachers.
Related to the two sub-questions of the research, we will pursue two distinct goals. Firstly, a teaching environment will be create, to support the teachers in adopting the competence approach. Starting from real situations, we will define a useful and practical repertory of tools and guideline, available for the teaching/learning activities, according to the competence approach. Secondly, we will suggest a teachers training plan. The study illustrated by this paper is the first step of the research project. The main goal of the study is the definition of the work-plan for developing the further steps of the research, divided into two parts according to the two research goals.

4.1 Creating a teaching by-competence support environment

The work-plan for the first goal of the research provides two products: the tool-box and the supporting software system. The research project, recently started, will manage a process of gradual revision to define the contents of the tool-box. The CS tool-box will include different instruments into three main parts: the Learning Units examples, that is activities to develop or previously developed in classes, along with a synthesis of the students' evaluations collected; the competence repertory, where knowledge, skills, and attitudes are defined; methodologies, which include guidelines and examples to use the learning units, according to the competence repertory. We will start collecting the Learning Units, which in our case will be learning projects de-

signed by the teachers and assigned to the students. The project will require the development of a real or realistic software application, as commonly used in CS education, in which the project-based learning approach [4] is often adopted. We will promote the design of such learning projects, asking to the teachers to create a complete set of materials, to be reused and assigned by the other teachers to the students of different classes and groups. The applications produced by the students will be firstly assessed by the teachers of the related classes; then, small groups of teachers will produce common evaluations of all the learning outcomes, with the aim of refining the materials of the learning unit. Further activities with the teachers will build up the competence repertory. With an incremental process of subsequent refining, we will progressively adapt: the choice of the competences to develop and their definitions; the relations between competences and activities, with worthy alternatives; the definition of the activities, detailing the learning tools chosen for the activities, possibly with a range of options, fitting the different learning styles.
The software application, called "Co4CS" (Competences for CS), will support the teachers, in their cyclic process of revisions to define the contents of the tool-box. We will also apply some tools of the Content Representation (CoRe) method, explained in [12]. CoRe is a method which was initially used to describe the professional knowledge of the science teachers, in terms of pedagogical content knowledge (PCK) (see [17]). Usually, PCK is an element really difficult to investigate, since teachers tend to "focus more on doing teaching rather than explicating the associated pedagogical reasoning" ([12], page 371). The CoRe method suggests a structured analysis to elicit the most important content, or "Big Ideas", of the discipline to teach. For every "Big Idea", the teachers will fill a simple table with their answers to eight questions, as explained soon. The CoRe table was applied in a research on Lithuanian CS teachers [3], in which the eight questions were aggregated in four, different pedagogical aspects, referred to the aforementioned PCK, as listed in Table 1.

Table 1. Pedagogical aspects (listed in [3])



	Pedagogical knowledge about ...
(a)	Learning goals and objectives connected to the topic
(b)	Students' understanding of the topic
(c)	Instructional strategies for teaching the topic
(d)	Ways to assess students' understanding of the topic

In our research, we will start to use the CoRe method to collect some "Big Ideas" from the teachers. This means that in the first round, the teachers will be involved to define some of the CS contents. But we note here that the aspect (a) of Table 1 is related to learning goals and objectives, and these can bring to the competence. So, in the second turn we will work with the group of teachers to the elicitation of the CS competences. This will be the critical and crucial point, in which the research will try to suggest a clear and shared connection between the contents (the "Big Ideas") and the competences or, better, to build a bridge towards the competences, starting by the contents. With the aim to offer the opportunity for the teachers to build such a connection, we will refer to every "Big Ideas", the most important CS contents, to apply the question: "Why is it important for the students to know this". Then, the answers will be combined to find the first list of competences. Table 2, derived from page 73

Table 2. CoRe questions and pedagogical aspects (adapted by [3])

(a)	1. What do you intend the students to learn about this Big Idea?
	2. Why is it important for the students to know this Big Idea?
	3. What else do you know about this Big Idea (and you don't intend students to know yet)?
(b)	4. What are the difficulties/limitations connected with the teaching of this Big Idea?
	5. Which knowledge about students' thinking influences your teaching of this Big Idea?
(c)	6. Which factors influence your teaching of this Big Idea?
	7. What are your teaching methods (any particular reasons for using these to engage with this Big Idea)?
(d)	8. What are your specific ways of assessing students' understanding or confusion around this Big Idea?

of [3], shows the complete list of the eight questions, linked to the four aspects.

4.2 Towards a teachers training plan

In order to apply the results of the research to a wider area and to more teachers, we will try to develop an idea for a possible training plan, with a proposal toward a competence-based teaching. The plan will be build upon a training course template for CS teachers. The outline of the course template will be detailed, according to the results of the first research goal.

5 Preliminary results

The results of the first survey activities have been included in previous sections. This section adds another preliminary outcome of the research, showing one of the use-case diagrams representing the system requirements for the Co4CS sw tool. The diagram is shown in Fig. 1 and represents the most general functions of the system. One

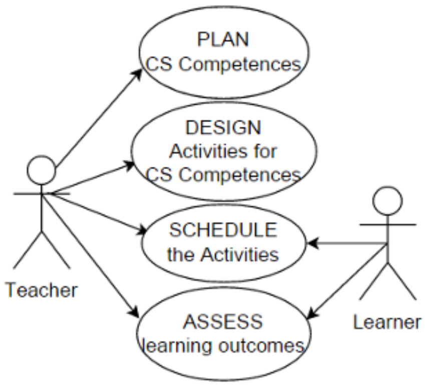


Fig 1. Top level use case diagram for CS SW tool

of the most relevant goals of the competence-driven approach is to make learners conscious of their own learning process. As said in [9], the students should be “cognitively activated” and have the opportunity to “self-regulate their work”. In order to support the achievement of this result, the learner's role has been included here, even though the research started from the teacher-side.

6 Conclusion and future steps

The research will be developed along the rest of the current school-year, lasting the next two school-years. The work will involve various CS teachers of different schools in province of Trento, mainly teachers of the technical schools. The critical factor of the project is the capability of involving the teachers in a positive and effective way. For this reason, we decided to approach the teachers gradually, starting with a small number of enthusiastic teachers. At the moment, we met four teachers, who will form the small sample group. With this group, we started to define a set of preliminary works and to collect materials and documents.

Acknowledgements. Our thanks to all the eSchooling partners, for allowing us to be involved in the project. A special thank to the teachers engaged in the first steps of the research, and to the teachers who accepted to be interviewed.

References

1. ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013. ACM Press and IEEE Computer Society Press.
2. Anderson, T.:. The theory and practice of online learning. Athabasca University Press (2008)
3. Barendsen, E., Dagiene, V., Saeli, M., Schulte, C.: Eliciting Computing Science Teachers’ PCK using the Content Representation Format Experiences and Future Directions. In: 7th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. Heidelberg: Springer International Publishing, pp. 29-40 (2014)
4. Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A.: Motivating project-based learning: Sustaining the doing, supporting the learning. Educational psychologist. 26(3-4), 369-398 (1991)
5. Bottani, N.: L’istruzione scolastica a un bivio di fronte alla voga travolgente e stravolgente delle competenze. (in Italian) in D.S. Rychen, L. Hersh Salganik. Agire le competenze chiave. Scenari e strategie per il benessere consapevole. Ed. Franco Angeli (2007)
6. Chiozzi, G., Giaffredo, S., Gris, R., Ronchetti, M.: Helping educators to teach competences with the support of technology in the Italian context. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications. pp. 106-113 Chesapeake, VA: AACE (2014)
7. FETCH Project (Future Education and Training in Computing): How to support learning at anytime anywhere. Retrieved 3 November, 2014 on http://fetch.ecs.uni-ruse.bg/?cmd=gsIndex
8. Flavell, J. H.: Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. American Psychologist. 34(10). 906-911 (1979)

9. Köller, O., Parchmann, I.: Competences: The German Notion of Learning Outcomes. in Bernholt, S., Neumann, K., Nentwig, P. (eds.), Making it Tangible: Learning Outcomes in Science Education. pp. 151–168. Waxmann (2012)
10. La Riforma della Scuola Secondaria Superiore (in Italian). Retrieved 20 October, 2014 on http://archivio.pubblica. istruzione.it/riforma_superiori/nuovesuperiori/index.html
11. Le Deist, F. D., Winterton, J.: What Is Competence? Human Resource Development International. vol. 8, No. 1, 27–46 (2005)
12. Loughran, J., Mulhall, P., Berry, A.: In search of pedagogical content knowledge in science: Developing ways of articulating and documenting professional practice. Journal of Research in Science Teaching. 41(4), 370–391 (2004).
13. Pellerey, M.: Ripensare le competenze e la loro identità nel mondo della scuola e della formazione. Seconda parte: l'approccio per competenze nei processi educativi e formativi (in Italian). Orientamenti Pedagogici. vol.57, n.3, 379-400 (2010)
14. Pikkarainen, E.: Competence as a key concept of educational theory - a semiotic point of view. Journal of Philosophy of Education. Vol. 48 No. 4, 621-636 (2014)
15. Punie, Y., Brečko, B.N.: A framework for developing and understanding digital competence in Europe. eds. DIGCOMP Publications Office (2013)
16. Recommendation of the European Parliament and of the Council of 18 December 2006 on key competences for lifelong learning. Official Journal of the European Union 06.05.2008 (2008/C111/01) 10-18. Retrieved 4 November, 2014 on http://eur-lex.europa.eu/LexUriServ/ LexUriServ.do?uri=OJ:L:2006:394:0010:0018:en:PDF
17. Shulman, L. S.: Knowledge and teaching: Foundations of the new reform. Harvard Educational Review. 57, 1-22 (1987)
18. TUNING Project - Educational Structure in Europe. Retrieved 3 November, 2014 on http://www.unideusto.org/tuningeu/ home.html
19. von Glasersfeld, E.. An exposition of constructivism: Why some like it radical. Journal for Research In Mathematics Education. Monograph 4: 19–29 & 195–210 (1990)
20. White, R.H.: Motivation reconsidered: the concept of competence. Psychological Review Vol. 66 (5), 279-333 (1959)
21. Winterton, J.: Competence across Europe: highest common factor or lowest common denominator? Journal of European Industrial Training Vol. 33 Issue 8/9, 681 – 700 (2009)

A language independent assessment of programming concepts knowledge

Franc Jakoš[1] and Matija Lokar[2]

[1] Janka Glazerja Ruše Primary School
Ruše, Slovenia
franc.jakos@glazer.si

[2] Faculty of Mathematics and Physics
University of Ljubljana, Slovenia
matija.lokar@fmf-uni-lj.si

Abstract. At the end of any educational process there is usually an assessment of the newly acquired knowledge. There are pedagogical goals and examples of tests, but they are all usually bound to a particular educational environment.

The article describes knowledge assessment created for grading the students' knowledge in the basic concepts of programming at the entry level of education. The tasks are simple, easily understood and do not assume any additional pre-existing knowledge. They are based on minimal teacher involvement and in most cases, students do not need further explanation. Such assessment requires the students to be familiar with the concept, as they have to transfer the knowledge pattern from the learning environment into the testing environment.

Keywords: teaching, programming, pseudo code, knowledge assessment

1 Introduction

In Slovene primary education a new elective subject Computer Science was introduced into year 4 in the 2014/15 school year. The subject will be gradually introduced into years 5 and 6 as well. Its three-year curriculum goals are based on computer science as a field of science and is not not merely computer literacy oriented. All that encourages the teachers to use additional content in other elective computer related subjects, the curricula of which and especially their execution is mostly computer literacy oriented now.

The introduction of computer science content into lessons is a novelty for the teachers and as such it is approached differently; with various tools, educational environments, criteria. The article [7] shows several options available for the introduction into the world of programming and algorithmic thought processes. Students eventually get bored with any educational tool and the initial motivation abates quickly as the problems become more demanding. Thus, it makes sense to vary the tools and methods, both because this enables the acquisition of appropriate goals and because it increases the students' interest for the subject.

However, variety can cause problems during knowledge assessment if it is bound to a certain level of familiarity with a specific environment, e.g. Scratch[3].

In addition, teachers are often invited to participate in numerous studies. At the moment the hot topic is whether the students in year 4 are capable of abstraction that is required for the acquisition of the subject goals and programming tasks, and what method of teaching is most successfully employed for year 4 students. Thus, one of the authors of this article was asked to take part in a research with the hypothesis that the students who first encountered conditionals in CS Unplugged and then started exercises in Scratch were more successful than the students who did not have the CS experience. However, the author teaches in a slightly different way. Students are initially motivated with a virtual reality educational game and only then do they start using Scratch. The questions in the research study were dependent on Scratch environment, so it is highly questionable whether these students were a good control group in this research.

Therefore, it would be practical to create methods of testing which would contain tasks that are not bound to particular environment or depend on the rules of a certain programming language syntax.

The article describes the tasks meant to assess the knowledge that does not depend on pre-existing knowledge, educational environment, or the teacher. This set of tasks can be useful for the teachers during knowledge assessment. It could also be used to pre-test and post-test the students in order to determine the level of knowledge when different educational environments or approaches to teaching the basics of computer programming languages are analysed.

The proposed model of the task is based on activities that are often used during entry level courses in programming. The object uses a single command to move across the field and leaves a trail behind it. The easily understood activity allows the students to focus on the meaning of the programming concepts.

2 Research and knowledge assessments

Assessment is always the most difficult part of the learning process, both for the teachers and the students as they both receive a mark of value. Therefore when creating a test, the value of the knowledge acquired has to be taken into account. It can be claimed that thinking about assessment leads to a better understanding of the subject goals.

Our research as well as the practical teaching goal was to create methods of testing which would contain tasks that are not bound to particular environment or depend on the rules of a certain programming language syntax.

Checking the literature for existing tests with the features mentioned above and following suitable taxonomy levels of knowledge shows that the web offers many programming skills tests (see[4] for example) that are purposely programming language bound. In addition, many authors discuss automatic program

[3] http://scratch.mit.edu
[4] http://www.testdome.com

assessment. The solution offered by the article [6] involves the translation of the program into pseudo code and then determining the presence of the elements necessary for the correct solution of the problem. Numerous articles ([1, 3, 8, 10, 12]) analyse the manner of thought and describe tools or methods of testing, but all are either strictly bound to very short periods of teaching (computer camps) or limited to a specific learning environment (making games, or a concrete programming language). Tew and Guzdial in [11] analyse the comparison of testing results in pseudo code and in the programming languages taught. They determine that though the results vary at the beginning of the educational courses, the final outcome is more uniform, showing but a slight difference. Their tests are somehow language independent; however, we could not get access to practical examples of their tasks, so we could not use them.

We have also encountered some scientific contributions that aimed to use pre and post assessment to analyse new educational methods or environments (see [4] for example). In order to achieve that they created tests and criteria, all dependent on a particular environment, though.

There is also the question what taxonomy levels of knowledge best describe a certain level of difficulty ([5]). The determination of the level of acquired knowledge where two taxonomies (Structure of Observed Learning Outcomes - SOLO [2] and Bloom) were joined, as described in [9], seemed the most appropriate to us.

Of course, when computational knowledge is to be assessed, a kind of programming syntax and a certain enivironment must be used. During the assessment itself we wished to exclude the influence of the teachers on the execution of the assessment. Therefore, we tried to construct the assessment in such a manner that does not require additional explanation from the teachers. Teachers are namely the least predictable element in the research, despite training all the teachers in the same way and giving all of them the same instructions. Thus, we wish to involve the teachers to the smallest extent possible, and in order to achieve this minimal involvement, the students are provided with short instructions at the beginning.

We tried to use the approach where the syntax as well as the tasks themselves are as obvious as possible. Therefore, a rectangular field was used containing an object that moves around the field leaving a trail behind it. Such tasks can be met in many different environments, e.g. Logo[5] (turtle graphics), Scratch[6] (drop the pen), virtual reality educational environment[7] (trace the trail activity). So when post-tests are performed, results from groups using different environments could be compared.

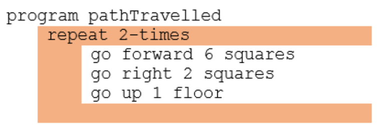
The tasks are constructed with the desire to first check the single-structure use and then the multi-structure or relational use for each separate concept of programming. According to the Bloom taxonomy, applying signifies the students' ability to read written code that according to SOLO taxonomy only contains one

[5] https://logo.codeplex.com/
[6] http://scratch.mit.edu
[7] http://opensimulator.glazer.si

concept and then in the multi-structure or in the relational structure contains several concept that can be co-dependent. (Figure 1) gives an example of a task, categorised into a single-structure (SOLO) / apply (Bloom), and Figure 2 is an example of a task that belongs to a multi-structure (SOLO) / apply (Bloom).

```
program pathTravelled
      go forward 5 squares
      go right 3 squares
      go forward 10 squares
      go left 3 squares
```

Fig. 1: single-structure (SOLO) / apply (Bloom) .

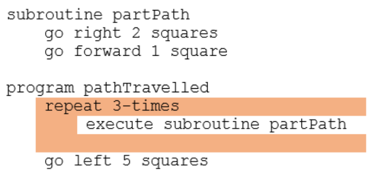


```
program pathTravelled
   repeat 2-times
      go forward 6 squares
      go right 2 squares
      go up 1 floor
```

Fig. 2: multi-structure (SOLO) / apply (Bloom)

The tasks should incorporate clear criteria regarding the testing itself and the goals. They should provide clear information about the students' achievement of each of the goals. An additional requirement is that the test should be formulated in such a way that minimal teacher involvement is required; the teachers thus do not need to offer additional explanations or help the students during the task solving.

We did not want to construct lower level tasks, as the questions would then be too bound to particular environment. Therefore, we included some tasks that according to Bloom belong into the category of creativity, where students need to read the task and complete or correct it. The final task tests the students' ability to write three different programs for a known solution. It was assumed that students might try to write commands without using concepts, so add a rule that does not enable that was added. (Figure 3) shows an example of a task categorised as relational (SOLO) / apply (Bloom), and(Figure 4) shows an example of a task categorised as relational (SOLO) / creative (Bloom).

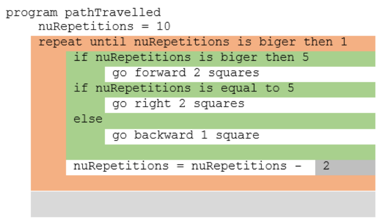


```
subroutine partPath
    go right 2 squares
    go forward 1 square

program pathTravelled
   repeat 3-times
       execute subroutine partPath

    go left 5 squares
```

Fig. 3: relational (SOLO) / apply (Bloom)



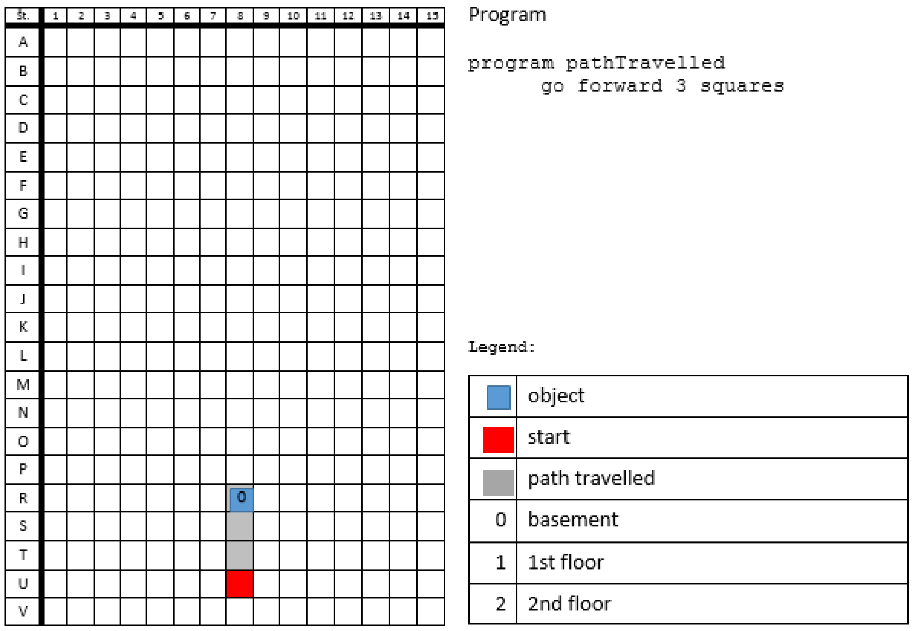
```
program pathTravelled
    nuRepetitions = 10
   repeat until nuRepetitions is biger then 1
       if nuRepetitions is biger then 5
           go forward 2 squares
       if nuRepetitions is equal to 5
           go right 2 squares
       else
           go backward 1 square

       nuRepetitions = nuRepetitions -  2
```

Fig. 4: relational (SOLO) / creative (Bloom)

3 Suggested tasks

As explained above the environment for solving the tasks themselves is simple. There is a rectangular field containing an object that moves according to the program instructions. Using a grid with marked columns and rows enables the students to easily understand and imagine where exactly the object is at any given moment in the course of the program. As only the knowledge of the concepts is to be assesed, we wanted to exclude any pre-existing knowledge necessary for a successful solution of the problem. For example: as the students do not encounter the coordinate system until year 8, we decided to use a grid with the system of letter and number marked rows and columns instead. Figure 5 shows the first of the three examples illustrating the rules of the tasks following.



Program

```
program pathTravelled
      go forward 3 squares
```

Legend:

🟦	object
🟥	start
▨	path travelled
0	basement
1	1st floor
2	2nd floor

Fig. 5: Task 1: example 1

Each example consists of the task (field and program) and the legend. The 3D field consists of a network of cells with a uniform address, e.g. A/2 and the number on the object that states the final floor. The addressing was kept in this form because conditionals and loops can refer to it. It turns out that this kind of notation is more easily understood by the students than the coordinate system. ”The programming language” only consists of the command go in all directions (up, down, left, right, forward, back). The legend describes the object, the starting point of the trail, the trail travelled, and numbers 0, 1, or 2 symbolize the height at which the object is found at the end of the trail.

The commands in the program named ”pathTravelled” direct the object on its way to the final destination. At the start, the object is in the red field. Student must read the program, colour the trail and determine the final field.

The instructions and the three examples at the beginning of the assessment provide enough information for most students to start solving the problem immediately. Very few of them require additional explanation.

Figures 6 and 7 show the increasing complexity of the variable concept. At first, the students should be able to read and understand the program and the value of the variable. The next task uses two variables and a loop that changes the value of one variable. We assume that a student who has never encountered programming will solve the task in Figure 6 successfully, whereas it will be interesting to monitor the age influence on the absolute beginners’ success in solving the task shown in Figure 7.

```
program pathTravelled
        nuSquares = 5
        go forward nuSquares squares
        go up 2 squares
        go right nuSquares-1 squares
```

Fig. 6: Simple program

```
program pathTraveled
    nuSquares = 5
    nuRepetitions = 3
    repeat nuRepetitions-times
        Go forward nuSqares squares
        nuSqares = nuSquares - 1
```

Fig. 7: More complex program

A similar increase of complexity for the condition concept is shown in Figures 8 and 9.

```
program pathTravelled
    go forward 2 squares
    go up 2 floors
    go right 5 squares
    if located in 12. column
        go right 2 squares
    else
        go left 5 squares
```

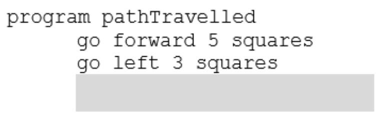
Fig. 8: Simple program

```
program pathTravelled
    repeat
        go left 1 square
        if located in 3. column
            stop traveling
```

Fig. 9: More complex program

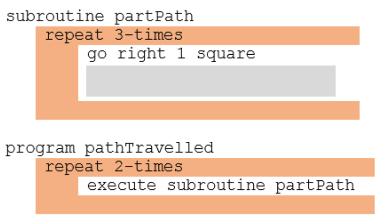
Additional instructions were included before the tasks where the students are expected to correct, complete or write a program. Namely, those tasks expect the exact opposite from the students. They are to “translate” the picture of a trail into a program.

Another group of tasks is slightly different again. The trail had already been coloured and the object is at the end of the trail. The program is either wrong or is incomplete. The students’ task is to correct or complete the program so that the solution matches the picture. Figures 10 and 11 show tasks where the students need to correct or add code in the grey fields.



```
program pathTravelled
      go forward 5 squares
      go left 3 squares
```

Fig. 10: Simple example



```
subroutine partPath
   repeat 3-times
      go right 1 square

program pathTravelled
   repeat 2-times
      execute subroutine partPath
```

Fig. 11: More complex example

4 Findings, further research and conclusions

At the moment, the preparation of the assessment is in its pre-pilot version. The assessment has already been used with a small group of students. It transpires that most students complete the "simple" tasks successfully. Tasks that are more complex are successfully solved by those students that have previously encountered programming. Thus, the year 8 students who have already programmed in Scratch solved most tasks. The problems mostly arise during the last stage where they need to correct, write, or complete the program.

We wish to include a larger number of students and teachers into the research at the end of the school year. That is namely the time when most curricula for Computer Science propose the introduction of extra content connected to programming. We are interested in the responses and results of both the complete beginner students and the students who had previously encountered programming. The progress of individual groups will be followed, regardless of the environment used and an attempt will be made to determine the correlation between the proposed assessment model and the environment bound assessment. We wish to offer a model of assessment that would provide the teachers/researchers with a tool that would enable them to claim that the knowledge acquired in any environment matches the goals and is transferrable to real life problems. Such a tool would also enable reverse assessment where the effectiveness of methods and learning environments could be evaluated.

The proposed model of assessment comprises several positive factors. It is a learning environment free test and matches the goals of the subject. That in turn enables teachers to switch environments if the motivation of the students begins to abate. What is more, this type of assessment enables comparison on a wider level; between schools, learning environments, and pedagogical methods. Many different experts can be involved in the construction of such assessment, which further increases the credibility of the assessment.

References

1. Basawapatna A., Han Koh K., Repenning A., C. Webb D., Sekeres Marshall K.: Recognizing computational thinking patterns In: SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education, pp 245 - 250, ACM New York, NY, USA, 2011

2. Biggs J., Collis K.: Evaluating the Quality of Learning: the SOLO taxonomy New York, Academic Press, 1982
3. Franklin D., Conrad P., Boe B., Nilsen K., Hill C., Len M., Dreschler G., Aldana G., Almeida-Tanaka P., Kiefer B., Laird C., Lopez F., Pham C., Suarez J., Waite R.: Assessment of Computer Science Learning in a Scratch-Based Outreach Program In: SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education, pp 371 - 376, ACM New York, NY, USA, 2013
4. Howland K., Good J.: Learning to communicate computationally with Flip: A bi-modal programming language for game creation In: Elsevier, Computers and Education, Volume 80, January 2015, Pages 224–240
5. Johnson C., Fuller U.: Is Bloom's taxonomy appropriate for computer science? In: Baltic Sea '06 Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, pp 120 - 123, ACM New York, NY, USA, 2006
6. Khirulnizam Abd R., Syarbaini A., Nordin J.: The Design of an Automated C Programming Assessment Using Pseudo-code Comparison Technique In: National Conference on Software Engineering and Computer Systems 2007, University Malaysia Pahang, Pahang, Malaysia, 2007
7. Lokar M.: Prvi koraki v programiranje – številne poti in možnosti (First steps into programming – numerous paths and possibilities) in: Uporabna Informatika, Ljubljana (to appear) (2015)
8. McCracken M., Almstrum V., Diaz D., Guzdial M., Hagan D., Kolikant Y., Laxer C., Thomas L., Utting I., Wilusz T.: A multi-national, multi-institutional study of assessment of programming skills of first-year CS students In: Proceeding ITiCSE-WGR '01 Working group reports from ITiCSE on Innovation and technology in computer science education, pp 125 - 180, ACM New York, NY, USA, 2001
9. Meerbaum-Salant O., Armoni M., Ben-Ari M.: Learning computer science concepts with scratch In: ICER '10 Proceedings of the Sixth international workshop on Computing education research, pp 69 - 76 ACM New York, NY, USA, 2010
10. Seiter L., Foreman B.: Modeling the Learning Progressions of Computational Thinking of Primary Grade Students In: ICER '13 Proceedings of the ninth annual international ACM conference on International computing education research, pp 59-66, University of Glasgow, UK, 2013
11. Tew E. A., Guzdial M.: The FCS1: a language independent assessment of CS1 knowledge In: SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education, pp 111 - 116, ACM New York, NY, USA, 2011
12. Werner L., Denner J., Campe S., Kawamoto Chizuru D.: The Fairy Performance Assessment: Measuring Computational Thinking in Middle School In: SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education, pp 215-220, ACM New York, NY, USA, 2012

Aspects of Quality in the Presentation of Informatics Challenge Tasks

Wolfgang Pohl and Hans-Werner Hein

BWINF / Bundesweite Informatikwettbewerbe
Wachsbleiche 7, 53111 Bonn
{pohl,hein}@bwinf.de

Abstract. So far, there has not been much scientific discussion about the quality of informatics tasks. The international community that is concerned with competitions like olympiads in informatics and the Bebras contest, however, has seen significant internal debate about even very detailed aspects of task quality. We describe the mechanics of developing Bebras tasks and formulate a central quality guideline for the development of task presentations. As an example, we demonstrate the critical steps in the development of one specific task and show that the modifications comply with the guideline. The guideline certainly refers to the circumstances of running a Bebras contest. Nevertheless, the guideline, and the recommendations we formulate on how to comply with it, are applicable to tasks in other settings – like exams or unsupervised learning scenarios – as well.

1 Introduction

Task quality is an important matter when quality of school education is discussed. In the area of mathematics, for instance, a significant amount of work on task quality can be found. In an internet search via google.com on March 12, 2015, we looked for sources that are relevant to our work, i.e. are concerned with the quality of tasks that are used in an educational context. We obtained 80,900,000 results for the search term "mathematics quality tasks"[1], with only relevant sources among the top ten results.

In the area of informatics (and its didactics in particular), the discussion about task quality is much less abundant. In our search, the term "informatics quality tasks" yielded 9,570,000 results, with four relevant sources among the top ten, while "computer science quality tasks" had more results (46,000,000), but no relevant top ten source. In Germany, in particular, work on task quality in informatics is rare. An online resource[2] documents the results of a workshop on "Aufgabenkultur Informatik" (engl.: task culture in informatics) and admits: "In Informatics, the debate about the culture of tasks is just at the beginning."

[1] The search term "quality tasks" turned out to be more successful than "task quality".
[2] http://bildungsserver.berlin-brandenburg.de/fortbildung_aufgabenkultur_informatik_htw.html (last accessed 03/24/2015)

Task quality should be a main concern among the organizers of task-based competitions (see [16] for a taxonomy of competitions) in informatics and other subjects. Hence, it is no surprise that most of the relevant contributions to this discussion are coming from the competition community (see next section). But even in this work, one important facet of (competition) task quality has been neglected so far: the presentation of a task, which is comprised of task material like text and images plus the composition of this material.

In this work, we introduce aspects of task quality, with a focus on task presentation, that are motivated by the specific circumstances of running the informatics contest Bebras [5, 1]. First, we discuss previous work on quality of competition tasks; we will see that task presentation has not been a predominant topic there. The following section describes the Bebras task development process, and then discusses quality guidelines for (Bebras) tasks. Section 4 presents one example task and illustrates how the task was modified during several steps of the task development process. The modifications are then related to our quality guideline. Finally, we discuss directions of future work and the applicability of Bebras-related task quality guidelines or measures to other areas.

2 Quality of Competition Tasks

While task quality is not a big issue in informatics education in general, there has been quite a discussion about task development and task quality among organizers or scientific committees of informatics competitions. In 1992 already, Hein [8] identifies constraints for competition tasks, suggests task types and presents "features" of good tasks.

The following topics seem to be most important in this discussion:

task types Specific types of tasks might be used in competitions to require collaboration within teams [13], to stimulate students' creativity [11], or in general make a competition better achieve its goals (as defined by the contest organizers) [9].

task attractiveness Tasks that involve graphics [18] or programming of games [12] might increase attractiveness and appeal of competitions.

task development Task quality can be supported by well-designed task development processes [7] and task management systems [10]. Specific techniques can be used to find new ideas for tasks and to refine them into suitable competition tasks [4].

Task presentation, however, is not much covered in that discussion. Mainly, quite general requirements can be found, like "a [task] formulation must be clear, comprehensive and not too long" [7], "[task] text must be clear and complete" [10], and "[. . .] a problem statement [should be] (relatively) short and easy to understand" [4]. But what does that mean: easy to understand? There is only few work that approaches this question. Pankov states that ". . . a good task should create a particular image in the mind of the contestant" [15]. Similarly, van der Vegt relates the problem of estimating task difficulty to the process of searching the mental representation of a text [20], which is part of the general reading comprehension process, as discussed in [3]. Our recommendations (as presented in Section 4.4) on how to arrive at high-quality task presentations essentially aim at enabling the contestant to easily find a good mental representation of the problem to be solved.

3 Tasks in the Bebras Contest

3.1 Development of Tasks for Bebras

Bebras is an international contest initiative that started in Lithuania in 2004. As of now, Bebras involves more than 30 nations world-wide, each of which organize their own national Bebras contests. In Germany, for instance, the national Bebras contest is known as "Informatik-Biber" [17]. All Bebras contests draw their tasks from an international task pool. This task pool is filled at the annual International Bebras Workshop. Each national contest (namely its organizers) contributes a bunch of task proposals in English. At the workshop, all task proposals are reviewed, perhaps modified and finally either *recommended* (added to the task pool) or not. Up to the year 2013, some tasks were selected to be *mandatory tasks* (i.e., to be used in all national Bebras contests). In 2014 there were no mandatory tasks, but some tasks in the pool were recommended higher than others.

After the international workshop, the organizers of each national Bebras contest choose tasks for their contests from the task pool. In order to use them in their contest, each national organizer needs to transfer the chosen tasks into the language(s) that are spoken in their country. This transfer process may require not only translation, but also modification of the English version that was output by the workshop. First, that version may still not be ready to be used in a contest. Second, transferring a task into another language may require linguistic (i.e., semantic, idiomatic, or metaphoric) adjustments. Last but not least, preparing a task to be used in a national context also requires to consider cultural aspects: "political correctness", "common sense" or even cultural taboos may vary significantly from country to country; and it is probably impossible to consider all these aspects for all the potential users of a task during the international workshop.

3.2 A Quality Guideline for Bebras Tasks

Quality of tasks, including task presentation, has been discussed quite early in Bebras history, and several publications have presented aspects of this discussion [14, 6, 19]. But also Bebras-related work mainly contains vague recommendations like "good tasks have easy understandable problem statements" [6, p. 23] but do not make these recommendations operational (i.e. do not tell how a task is made "easy understandable").

In general, "quality" is a relative notion that needs a measure or benchmarks. This holds for task quality as well: "Tasks may not be considered good in an absolute sense; in order to be able to talk about good tasks, a quality measure is needed"[3] [21]. If quality is to play a role in the development of (Bebras) tasks, then quality measures or, at least, guidelines or criteria that tasks should meet are needed.

Dagiene and Futschek [6] present early results of discussion in the Bebras community, among them a large table of "Criteria for Good Bebras Tasks". Finally, they arrive at a short list of so-called mandatory criteria:

- The task can be solved within 3 minutes.
- The problem statement is easily understandable.
- The task is presentable at a single screen page.
- The task is solvable at a PC without use of other SW[4] or paper and pencil.
- The task is independent from specific systems.

Some of these criteria may be criticized:

1. The 3-minutes-constraint depends on the overall amount of time given and the number of tasks presented to the participant of a national Bebras contest. Within a contest, tasks should be of different difficulty levels. Hence, it may be acceptable to have a 4-minute-task, if there are enough other tasks that can be solved quickly.
2. The size of "a single screen page" varies wildly among the hardware that can be used to participate in a Bebras contest; so this criterion is not very precise and may be considered obsolete.
3. Pen and paper are not needed to solve a task if the task allows the participant to explore the solution space through interactive means (within the online contest environment). However, it may be considered an important part of a participant's informatics competence to develop their own model of the solution space via pen and paper; in the German "Informatik-Biber", participants are therefore explicitly encouraged to use pen and paper.

To us, the second criterion in the list above is crucial: The problem statement is easily understandable. But what does "easily understandable" mean? Let us take a closer look at the circumstances of a Bebras contest. Each participant is asked to solve a relatively large number of tasks within limited time. In the German "Informatik-Biber", for instance, 18 tasks need to be solved within 40 minutes. This immediately leads to the following requirement: A task needs to be understood *quickly*. Furthermore, a contest must be fair; all participants must find similar conditions. This leads to a second requirement: A task needs to be understood *correctly* by everyone; with "correctly" meaning "in the way the task setters wants it to be understood". Third, in most Bebras countries Bebras is not a knowledge test; this leads to the third requirement: A task needs to be understood without prior knowledge (in the contest's subject: informatics). We summarize these three requirements into one central task quality guideline:

[3] Originally in German: "Aufgaben an sich sind nicht in einem absoluten Sinn gut; um von guten Aufgaben reden zu können, bedarf es eines Qualitätsmaßstabes."
[4] We assume SW to stand for "software".

A task needs to be understood quickly, correctly,
and without prior knowledge.

In other exam contexts, prior knowledge may be required or even tested; in such cases the guideline abbreviates to: *A task needs to be understood quickly and correctly.*

4 Developing Bebras Tasks with Quality in Mind: An Example

In this section, we will present one example task and how it got modified within the task development process. As example, we have chosen the task "Verlorene _nf_rmat_on" (*engl.:* Lost Information; its initial English title was "Missing Piece"). It was presented at the International Bebras Workshop of 2011. The German "Informatik-Biber" incorporated this task in both 2011 and 2012 [2, p. 40].

Like many Bebras tasks, the presentation of this task consists of the following components: (a) an introductory text, (b) one central image, (c) a question, and (d) a set of choice answers (with only one answer being correct; Bebras choice tasks typically are single-choice). In this case, the choice answers are images. Further on, we will refer to all text components of a task presentation (introductory text and question) as *task text.* These four components are presented at participation time, i.e. *in-contest.* In addition, there are further task components that are presented *post-contest*, i.e. after the contest: (a) a solution text that explains which solution is correct and why, and why other solutions are not correct; (b) a background text (titled: It's Informatics!) that explains why the question relates to informatics and which of its topics it does refer to.

4.1 Task Version Produced by the International Bebras Workshop

Figure 1 displays the task presentation as produced by the International Bebras Workshop. Concerning the task's subject matter, it is about error-correcting codes: A binary code given by a 5×5 matrix is enhanced by an error-correcting row and column.

4.2 Transfer into German

After the international workshop, the task was transferred into German. We present the results of this transfer in English, though, trying to stick to the German formulation as close as possible. Figure 2 shows the text components of this version; the images were not modified in this transfer process. The task text significantly differs from the German original, as follows:

1. The introductory text of the task (often referred to as "story" of a task) was modified. The "message" turned into a "label". The latter notion much better fits the two-dimensional nature of the black-and-white-matrix code.

Missing Piece
Beaver John has received a secret message. Unfortunately a part of the message has been destroyed by a spill of red colour.
This case was foreseen and there are additional squares in the message. Each square in the rightmost column (column 6) or the lowest row (row 6) is coloured such that the number of black squares in a row, respectively in a column is even.

	1	2	3	4	5	6
1	■	■	□	□	■	■
2	□	□	□	□	□	□
3	□	■	🟥	🟥	■	□
4	■	■	🟥	🟥	□	■
5	□	□	□	□	■	■
6	□	■	□	■	■	■

John considers there are sixteen different possible messages. Only four of them make sense to him.
What is the pattern of the red piece?

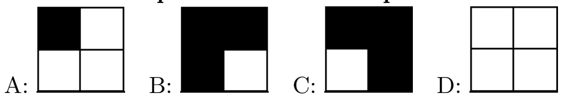


Fig. 1. Task version as output by the International Bebras Workshop

2. In the first version, the parity effect of an additional row and column had to be described quite laboriously. The new version more concisely mentions a 6×6 matrix the rows and column of which obey a parity condition.
3. The first version tried to motivate why just four choice answers are presented ("... sixteen different possible messages. Only four of them make sense ..."). This part is not needed in order to understand the task correctly. Hence, it can be dismissed.
4. In the first version, the defective part of the code is referred to in multiple ways: It is called either "part of the message" or "piece" (this is unclear: piece of what?). The new version consistently uses the phrase "the four red fields". Please note that the question fully repeats this phrase, instead of abbreviating it (which might seem more elegant) to "the four fields".

All modifications comply with our quality guideline: Modifications 1 and 4 help to understand the task correctly by using appropriate and consistent wording. Modifications 2 and 3 shorten the text and therefore help to understand the task quickly. The first version already had not required prior knowledge of informatics, and neither does the new version. It introduces the technical notion of "barcode", though, and we cannot safely assume that this notion is known to all potential

Lost Information
The beavers label cut trees using a kind of barcode in a 6×6 square with the particularity that in each row and each column, the numbers of black fields must be even.
In the following barcode, the four red fields were damaged.
Image as in Figure 1
How did the four red fields look?
Choice answers as in Figure 1

Fig. 2. Task version resulting from the transfer into German (re-translated to English)

Bebras participants. This may be considered a weakness of the new version. But note that the new version of the task mentions barcodes, but does not require knowledge about barcodes to be solved.

4.3 Further Modifications

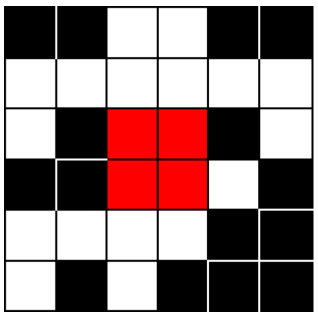
Figure 3 shows a third version of the task, including the post-contest "It's Informatics!" background text. This version is the result of further modifications by task editors. At first it can be seen that the central image was modified:

1. The upper row and leftmost column got dismissed. There are two reasons for this: (a) The fields of this row and column contained numbers that were meaningful in the first version only, but not in the second and third. (b) The white fields with the numbers were hard to distinguish from the black, white, and red fields that are genuine elements of the code.
2. There are white separation lines between black fields now, consistent with the black separation lines between white fields. Before, adjacent black fields had formed a big black block, hiding the matrix structure of the code. (Unfortunately, the editors missed to apply this idea to the choice answer images as well.)

The task text was modified either:

1. In the second version, the label was described as "a kind of barcode". We assume that this analogy was introduced to relate the matrix code of the labels to something known. That may make sense, but may be dangerous as well: In case such an analogy is inappropriate, it may cause connotations that are not only unnecessary to understand the task but may even prevent the participant from doing so. In this task, the barcode analogy does not fit; a barcode is one-dimensional, while the matrix code of the tree labels is two-dimensional.
2. The barcode analogy is replaced by the noun "label". Thus, a uniquely identifying word is used to refer to the code, which even matches the verb "label" used in the sentence before.

Lost _nf_rmat_on
The beavers label the trees they cut.
A label consists of a matrix of 6 times 6 fields that may be black or white.
In each row and column of a label, the number of black fields is even.
Thus, in the rough environment the label is more robust.
This label got dirty in a tree transport:



How did the four red fields look before?
Choice answers as in Figure 1
It's Informatics!
In everyday life, there are many situations where communication is disturbed. Informatics knows a lot of methods to ensure that information will be preserved along a communication channel, in spite of eventual disturbances.
In the theory around these methods, "redundancy" is a central notion. The rule is: The more redundant the information source (here: design of the labels), the more robust the information (the label coloring).

Fig. 3. Final version of the task presentation

3. The first sentence of the second version had been quite long and complex. This sentence is split into three much shorter sentences. By repeatedly using the noun and verb "label" in all three sentences, the semantical coherence of the former long sentence does not get lost.
4. The vague notion of a "6×6 square" from the second version is now specified as "matrix of 6 times 6 fields that may be black or white". This phrase precisely defines the structure of the code. Moreover, the colors black and white are defined to be the only colors permitted in a code; thus, the reader is prepared to understand the red coloring of fields as error situation.
5. In the second version, the problem with the code was described in a misleading way. It said "the four red fields were damaged"; this left unclear whether the red coloring was the damage (this is the intended meaning) or the fields were initially red and then damaged. The new version now simply says "This label got dirty". Together with priming the reader to understand

the red coloring as error (see previous item), this terse wording is absolutely clear.

All these modifications comply with quality guideline and are meant to make the reader quickly and correctly understand the task. The resulting version of the task was considered final and used in the contest.

4.4 Recommendations for High-Quality Task Presentation

Above, we described specific modifications that were applied within the development process of one particular task. From these modifications, we try to derive and generalize a few recommendations for action. In order to comply with the proposed quality guideline, task authors should use:

... short sentences: Short sentences are easier to read, and they facilitate quick understanding. (See also [3]: "Students are more likely to misunderstand a complex question, especially if complex language is used.")

... words or phrases repeatedly: Repetitions allow for splitting long sentences without losing semantical coherence. Furthermore, repetitions consolidate correct understanding.

... clear definitions: A clear definition of a notion facilitates correct understanding.

... a one-to-one relationship between words and objects: using only one linguistic term for the same object facilitates quick and correct understanding. Do not use synonyms; it's a task, not an essay.

... appropriate analogies: Inappropriate analogies interfere with both quick and correct understanding.

... unambiguous wording: A wording that is ambiguous or misleading interferes with correct understanding.

4.5 No Version is Perfect

Two peculiarities with the final version of the task may have been noticed:

1. In the title, a few letters were replaced by underscores. This "lost information" nicely corresponds with the task title. We do not consider the task title crucial for task understanding. The task title mainly serves to attract attention, which it certainly does in this case. Moreover, this specific "title gimmick" helps to introduce the problem and guides the reader into thinking about completion of missing information.
2. A full sentence was introduced into the final version, namely: "Thus, in the rough environment the label is more robust." This sentence is not needed for task understanding. An unnecessary sentence seems not to comply with our guideline, because it unnecessarily consumes resources needed for task understanding. However, this sentence plays an important part: not in relationship to the in-contest components of task presentation, but among the post-contest components. It introduces the notion of "robust information",

which is the main keyword in the "It's Informatics!" background text associated with this task. We consider this background text to be of vital importance to the overall quality of a task (if quality is considered over the task's whole life-time). Therefore we find it legitimate to create a strong relationship between task text and background text by inserting a sentence that is not essential to the in-contest task presentation.

Furthermore, we see the following two potential shortcomings in the final version of the task:

1. The mathematical term "matrix" is used in the task text. In the contest, the task was presented to students of grades 7 and 8 (12-14 years old), which usually are not familiar with that term. We argue that it is not necessary to understand the meaning of "matrix"; the phrase "6 times 6 fields" together with the image will yield a clear mental model of the problem to be solved. Hence, the term could or even should be omitted ("A label consists of 6 times 6 fields ...") or be replaced by a non-technical term ("A label is an arrangement of 6 times 6 fields ...").
2. In the question, the phrase "the four red fields" is used to refer to the code elements the content of which is to be determined. This phrase is not completely in line with the definition that the fields of a label must be black or white only. So, the wording could be more precise: "the four fields colored in red". However, we argue that the shorter wording will be understood more quickly, and that – together with the image – the reference to the four inner fields of the label is significantly clear.

5 Summary and Future Work

We have presented a guideline for developing high-quality presentations of informatics tasks. This guideline has been inspired and influenced by the specific circumstances of the Bebras contest. We have demonstrated the development steps of an example Bebras task and described in detail the modifications that were applied to this task in order to improve the quality of its presentation. Furthermore, we have shown that the modifications comply with the presented guideline. From these specific modifications, we have derived a short list of generalized recommendations on how to produce task presentations of high quality with respect to the presented guideline.

We argue that these recommendations are useful for task authors in Bebras as well as in other contexts where quick and correct understanding is crucial. Such contexts may be other competitions, but also unsupervised computer-aided learning environments where it is usually not possible to clarify the correct understanding of a task by interacting with a (human) tutor.

The concepts presented in this work need to be investigated further. First, task presentation may (and in Bebras perhaps should) involve images. There are many details to be considered as far as images are concerned: Where to place an image in relation to the text and other material? What information should be given by images, what information by text? If colors in images are used to encode information: Is the code unambiguous, appropriate, and common sense (e.g. in a European context it is common to use red for "stop" and green for "go"). Second, there are certainly other aspects of quality that are important to informatics tasks. Producing a high-quality task presentation is just one part of task development. What makes the core problem of a task interesting or even fascinating? What is a good story, i.e. a sound and culturally acceptable outfit for the core problem? Third, we have argued that following our recommendations will lead to task presentations that can be understood quickly and correctly. But we still need to prove our claim by empirical investigations.

References

1. Bebras: International Contest on Informatics and Computer Fluency, `http://www.bebras.org/`
2. Informatik-Biber: Aufgaben und Lösungen 2012. BWINF (Februar 2013), `http://informatik-biber.de/fileadmin/user_upload/archiv/2012/Informatik-Biber\_2012\_Web\_01032013\_mitLoesungen.pdf`
3. Ahmed, A., Pollitt, A.: Curriculum demands and question difficulty. In: IAEA Conference. International Association for Educational Assessment, Bled, Slovenia (May 1999)
4. Burton, B., Hiron, M.: Creating informatics olympiad tasks: Exploring the black art. Olympiads in Informatics 2, 16–36 (2008)
5. Dagiene, V.: The BEBRAS contest on informatics and computer literacy – students' drive to science education. In: Joint Open and Working IFIP Conference. ICT and Learning for the Net Generation. pp. 214–223. Kuala Lumpur (2008)
6. Dagiene, V., Futschek, G.: Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R., Syslo, M. (eds.) Informatics Education – Supporting Computational Thinking. pp. 19–30. LNCS 5090, Springer-Verlag, Berlin Heidelberg (2008)
7. Diks, K., Kubica, M., Radoszewski, J., Stencel, K.: A proposal for a task preparation process. Olympiads in Informatics 2, 64–74 (2008)
8. Hein, H.W.: International olympiads in informatics: What is a proper programming contest task? In: 12th IFIP World Computer Congress. Madrid, Spain (September 1992)
9. Kemkes, G., Cormack, G., Munro, I., Vasiga, T.: New task types at the Canadian Computing Competition. Olympiads in Informatics 1, 79–89 (2007)
10. Kolstad, R.: Infrastructure for contest task development. Olympiads in Informatics 3, 38–59 (2009)
11. Kulczyński, T., Łacki, J., Radoszewski, J.: Stimulating students' creativity with tasks solved using precomputation and visualization. Olympiads in Informatics 5, 71–81 (2011)
12. Ninka, I.: The role of reactive and game tasks in competitions. Olympiads in Informatics 3, 74–79 (2009)
13. Opmanis, M.: Team competition in mathematics and informatics "Ugāle" – finding new task types. Olympiads in Informatics 3, 80–100 (2009)
14. Opmanis, M., Dagiene, V., Truu, A.: Task types at "beaver" contests. In: Dagiene, V., Mittermeir, R. (eds.) Information Technologies at School: Proceedings of the

2nd International Conference “Informatics in Secondary Schools: Evolution and Perspectives”. pp. 509–519. Institute of Mathematics and Informatics, Vilnius (2006)

15. Pankov, P.S.: Real processes as sources for tasks in informatics. Olympiads in Informatics 4, 95–103 (2010)
16. Pohl, W.: Computer science contests for secondary school students: Approaches to classification. Informatics in Education 5(1), 125–132 (2006)
17. Pohl, W., Schlüter, K., Hein, H.W.: Informatik-Biber: Informatik-Einstieg und mehr. In: Koerber, B. (ed.) Zukunft braucht Herkunft: 25 Jahre INFOS – Informatik und Schule. pp. 38–49. Gesellschaft für Informatik, Bonn (2009)
18. Ribeiro, P., Guerreiro, P.: Increasing the appeal of programming contests with tasks involving graphical user interfaces and computer graphics. Olympiads in Informatics 1, 149–164 (2007)
19. Vaníček, J.: Bebras informatics contest: Criteria for good tasks revised. In: Gülbahar, Y., Karataş, E. (eds.) ISSEP 2014. pp. 17–28. LNCS 8730, Springer International Publishing, Switzerland (2014)
20. van der Vegt, W.: Predicting the difficulty level of a Bebras task. Olympiads in Informatics 7, 132–139 (2013)
21. Walther, G.: Modul 1: Gute und andere Aufgaben (Arbeitsversion), Mathematikmodul 1 des Programms “Sinus-Transfer Grundschule”, `http://sinus-transfer-grundschule.de/fileadmin/Materialien/Modu1.pdf`

Tasks Classification and Age Differences in Task Perception. Case Study of International On-line Competition "Beaver"

Ekaterina Yagunova[1], Sergey Podznyakov[2], Nina Ryzhova[3], Evgenia Razumovskaia[4], Nikolay Korovkin[5]

[1] Saint-Petersburg State Electrotechnical Institution LETI after V.I.Ulianov (Lenin), ul. Professora Popova 5, St. Petersburg, Russia
St Petersburg Academic University, "Physical-Technical School" Lyceum, 8/3 Khlopina Str, St Petersburg, Russia
katrin.home@mail.ru

[2] Saint-Petersburg State Electro technical Institution LETI after V.I.Ulianov (Lenin), ul. Professora Popova 5, St. Petersburg, Russia
pozdnkov@gmail.com

[3] State Corporation 'Institution of Training – ARB Pro', Kaluzsky per, 3, St Petersburg, Russia
ryzhova.nina@gmail.com

[4] The University of Edinburgh, Old College, South Bridge, Edinburgh, United Kingdom EH89YL
evgeniar@yahoo.com

[5] St. Petersburg State Polytechnic University, 29 Polytechnicheskaya st., St. Petersburg, Russia
nikolay.korovkin@gmail.com

Abstract

Complexity is objective characteristic of a task. **Difficulty** defines the relationships between the task and the person solving it. We can evaluate task **complexity** a posteriori - by the portion of the participants who solved task correctly. Task **difficulty** is hard to evaluate.
We offered and compared several approaches to evaluation of **difficulty** and **complexity** of tasks of the international informatics competition "Beaver".
We found that a priori evaluation of problems by the organizers does not correspond to the **difficulty** of the task for the participants. The organizers underestimated the **complexity** for younger pupils and overestimated the **difficulty** for older. The pupils, especially primary school kids, frequently underestimate the **complexity** of the tasks.
We clustered the tasks by their **difficulty** and **complexity** into 4 clusters. One of them had tasks with significantly underestimated **difficulty**. We showed that one year age difference results in differing evaluations of task **difficulty** and **complexity**.

Keywords

complexity of tasks • difficulty of tasks • on-line competitions • educational tests • typology of tasks • competition "Beaver"

1 INTRODUCTION

1.1 Methods for estimating task complexity and difficulty

A task which is easy for one participant may be difficult for another one. The task **difficulty** reflects the relationships between a task and an individual who performs it. To underline this feature many authors separate the notions of "**complexity**" and "**difficulty**" ([2], [11], [18], [23]). **Complexity** means a certain objective feature of a task while the **difficulty** is understood as a subjective feature, i.e. how a participant interprets a task. While speaking about the **difficulty** the authors focus on the individual's activity to perform a task – to analyse and to process the information, to design and to make decisions, to forecast consequences of their own decisions and to build operation images and frameworks ([16], [21]).

The task **complexity** may be measured upon competition results by counting a share of participants who got right answers. A measurement or at least evaluation of task **difficulty** requires serious efforts.

The **difficulty** of a task for a subject is formed of their mental workload (cognitive, informational, emotional, attentional loads) and expenditures for their own state control ([23]).

The most accurate methods of workload estimation suggest measuring of various human factors ([4]). A diagnostic procedure may be accomplished only during "live" competitions with limited number of participants. The procedure itself may be an additional stress for participants. This may lead to the increase of effort made to control their state ([13]). Meanwhile, it is only possible to assess the state of participants of remote competitions using self-reflection tests during task performance. The results of such questionnaires shall be adequate only for senior school students because it will be difficult for primary school children to make an objective evaluation of their state and abilities ([25]). According to Piaget ([20]), a primary school child is at the stage of concrete operational intellectual development. Typically for this age, thinking restrictions affect not only the cognition of outside world but also the manner of children to perceive themselves. It is fair to start talking about conceptual thinking only by the age of 11-12 years.

The level of the development of thought process is associated with the age as well as the appropriateness of self-assessment. That is why primary school children may not cope with solution of tasks that require the operation with abstract notions (and this is natural!) ([20]). During the period from 8 to 10 years the capacity of memory is rapidly increasing, attention can be switched much better. So, even minor age difference in this period may cause significant differences in results when solving same tasks.

The results of "human system" activity exhibit the relationship between the quality of operation information (quality and quantity of stimuli, coding, distribution etc.) and the capacity of resources available ([18]). In subject competitions the attention load, the processes of short-term and operative memory may be assessed with the **difficulty** of the text of problem statement. Among numerous ways of assessment of the **difficulty** of text, the most straight forward is the length of the statement, i. e. number of stimuli to be processed for solution ([3]). The workload may be a function of the level of **difficulty** and the number of tasks to be performed within a unit of time ([22]).

Online informatics competition "Beaver" gives us a unique opportunity not only to compare the results of the participants, but also to analyse the process of their work on the tasks. Thus we can estimate both **complexity** and **difficulty** on a large sample of schoolchildren.

Using the protocols of on-line competitions one can evaluate the workload of participants taking into account the time spent by them to solve tasks ([10]). The rules of "The Beaver" competition presume the possibility by participants to complete only a part of tasks. In this case the participant's refusal to solve the task is to be considered as their assessment of task **difficulty** upon binary rating scale ("difficult" – "not difficult"). Solved tasks are assessed by a participant as "not difficult", while those that are not presented – as "difficult". A share of participants who found the task "difficult" shall determine the **difficulty** of a task for the whole body of competition participants.

There is a huge number of ways to determine task weights (SoČan, 2009, [5]). The organisers of the 'Beaver' competition assign weights to tasks prior to the beginning of the competition. Previously, it was shown ([27]) that the distribution of the results is far from normal. A prior estimation of tasks by experts (weighting) shall be correct once both the task **complexity** and **difficulty** are taken into account for participants of each age group. Only analysis of the results of competition can establish whether the **difficulty** has been really considered at weighting, whether a priori estimate is in agreement with an objective **complexity**.

1.2 Research objectives

1. To evaluate the **difficulty** and **complexity** of the tasks of online informatics competition "Beaver"
2. To compare different evaluations of task **complexity** and **difficulty** gained using different methods
3. To classify tasks upon their **complexity** and **difficulty**.
4. To estimate the differences in the perception of the tasks by the school children from different age groups

It should be noted that some authors use the notions of "**complexity**" and "**difficulty**" as synonyms because not the content of mental processes is fundamental for them, but the execution – whether a schoolchild can or cannot solve the task ([15]). Further on we shall differ the terms "**complexity**" and "**difficulty**" by high lightening them in bold. If the matter is a complex evaluation with account made for both parameters, we shall use the term "**complexity**" without high lightening it.

2 MATERIAL AND METHODS

2.1 "The Beaver" competition: organization, tasks selection

"The Beaver" international on-line competition on informatics started in 2003 ([6]). The task pool is prepared by representatives of participating countries. The statements of the problem should meet certain requirements ([19], [7]). The statements are discussed and formulated in English. Each country chooses the problems for the competition out of the task pool. Some tasks are "compulsory", i.e. they should be included by all participating countries. The organizing committee of each country is responsible for translation of the statements to the national language of each country.

Russia took part in this competition for the first time in 2012. The competition in Russia is organized for six age groups. The participants are proposed to solve the tasks of three levels of **complexity**: for schoolchildren of 1 and 2 grades – 4 tasks for each level, for schoolchildren of 3-10 grades – 5 tasks for each level. 8 simple and 7 complicated tasks are proposed to senior schoolchildren. The tasks of 2012 can be found at http://ipo.spb.ru/bebras-files/beaver-2012-rus.zip (in Russian) or http://ipo.spb.ru/bebras-files/beaver-2012-eng.zip (in English). 40 minutes are given for task completion. Every wrong answer is fined the penalty rate which makes one-third of task value.

Competition tasks are numbered, simple tasks are of smaller order and complicated tasks have larger numbers. The tasks of the same **complexity** are randomly numbered (the order is different for each participant). Participants may solve the tasks arbitrarily, they may return to solved tasks.

Participants are to choose from multiple answers, three of them are wrong and one is correct. A participant can choose the option "no answer". In this case they will receive neither score, nor penalty. Certain tasks prepared for students of 1-2 grades are dynamic – certain actions with the mouse are required to be made. These tasks will be scored in the same way as others. A sum of scores and penalties gained shall make the result of competition for each participant. Participants gained the highest results shall become winners in each age group.

2.2 Scales and analysis

The protocol of competition where all participants' actions are recorded shall be taken as a basic data set ([26]). If a participant introduced successively several answers the last one is to be taken into consideration when counting the results. The total number of competition participants in 2012 was 6602 schoolchildren. The distribution of the participants by age is given in table 1.

Table 1. Number of students in each age group, who took part in the competition

level of contest	Grade	number of pupils	
1	1	297	733
	2	436	
2	3	661	1483
	4	822	
3	5	793	1603
	6	810	
4	7	672	1329
	8	657	
5	9	615	1148
	10	533	
6	11	306	6602

We used several scales to evaluate the **complexity** and **difficulty** of competition tasks.

Scale 1 – scale of expert estimation. It is performed upon three-point scale (1 – simple, 3 - complicated) during the meeting of international steering committee. In order to consider the estimate as correct both the objective **complexity** and **difficulty** of tasks should be taken into account.

Scale 2 – share of participants who chose the "no answer" option for particular task. When the answer is arbitrarily chosen out of 4 proposed options, the probability (p) to choose a correct answer makes ¼. In this case a mathematical expectation of scores gained is p*x+ (1-p)*(-x/3) =x (4p-1)/3=0 (where x – task value), that coincides with total scores received when choosing "no answer" option. If one of the answers proposed is rejected as certainly wrong, the probability of arbitrary choice of a correct answer out of remained options is greater than ¼ and a mathematical expectation of score gained for solving a task becomes positive. Therefore, a participant has no reasonable motive to choose "no answer" button. The use of this button may only be due to psychological reasons – for example, a fear of failure (and penalty for it) or an extreme lack of self-confidence, a fear of task statement. As statistical expectation of score may be hardly calculated by schoolchildren, the choice of "no answer" option reflects their feelings about the relation between winning and failing probabilities. In all cases the choice of "no answer" option is the result of an interaction between the schoolchild and the task, i.e. it features the **difficulty** of a task for a participant.

Scale 3 – share of participants who gave a correct answer among those who decided to solve the problem. It shall be determined upon completion of a competition and shall contribute to proper evaluation of the task **complexity**. We must underline that this is just a task **complexity** because it is calculated only taking into consideration those who decided to solve it, i.e. among those who understood it as "not difficult" upon scale 2.

Scale 4 – number of symbols in problem statement. This is an indirect assessment of task **difficulty** because it relates to memory and attention loads.

Because scales 1-3 are ordinal, the comparison of scale 1 with scales 2 and 3 has been made using Spearman's correlation coefficient.

Clustering of tasks has been made by Ward's method while using Euclidean metric.

3 RESULTS

3.1 Task complexity and difficulty

The distribution of the number of correctly solved tasks is shown in fig. 1. It appeared that average schoolchild solves less than a half of all the offered tasks in all the grades. Moreover, 75% of the participants both from primary and secondary schools solved less than a half of the offered tasks. The majority of schoolchildren showed low results. It means that **difficulty** and/or **complexity** of the tasks were underestimated by the organisers. Extremely few students solved all the tasks of the competition.

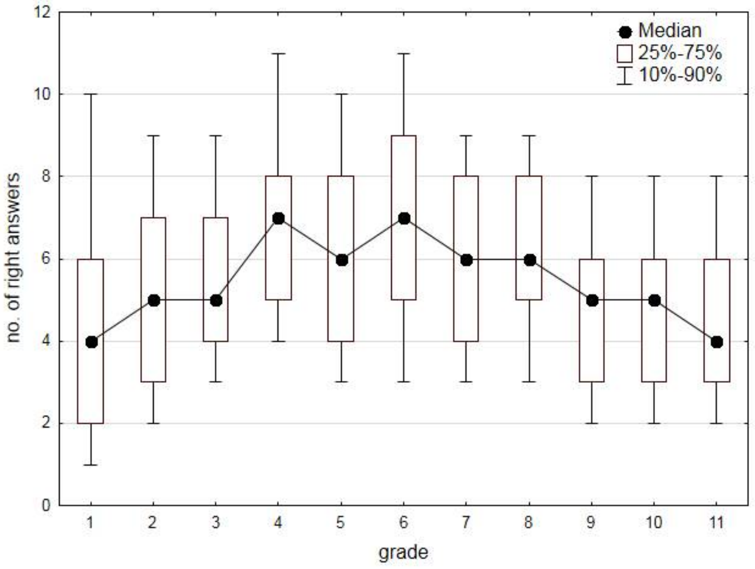


Fig. 1. Distribution of the number of tasks solved by the students from different years.

Task assignments upon **difficulty** (scale 2) and **complexity** (scale 3) are far from normal (Figure 2). The skew of task **complexity** distribution is positive while that of task **difficulty** distribution is negative, i.e. tasks of low

difficulty prevail in competition but the most part of tasks are of high **complexity**. There are tasks with difficulty of less than 10% among the tasks for each grade. The task of the lowest **difficulty** was found in a test for the 8th grade (1, 1%) and of the highest **difficulty** - in a test for the 10th grade (47, 6%). The task **complexity** within test versions for each grade varies from 10-20 to 80-90 %. The lowest **complexity** task was found in the 2nd grade version (7, 7%) and the one of the highest **complexity** – in the 7th grade task set (89, 8%).

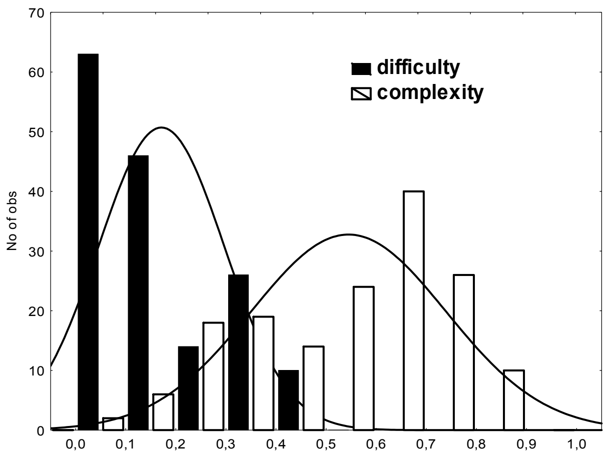


Fig. 2. Distribution of competition tasks upon their **difficulty** and **complexity**

Spearman's rank correlation coefficients of expert tasks evaluation (scale 1) with their **complexity** (scale 3) and **difficulty** (scale 2) for participants calculated for all competition tasks are significantly positive ($p<0.01$) and equal to 0.56 and 0.60 respectively (table 2). There was no correlation of expert evaluation of **complexity** with task **difficulty** revealed for junior grades (1, 2, 3). There is a significantly positive correlation for grades starting from the 4th. The best agreement of expert evaluation with task **complexity** was found for 1-2 grades. As far as it concerns the versions for 7-8 and 11 grades the evaluation of task **complexity** by organizers did not correspond to real **complexity** of tasks for school students.

Table 2. Spearman's rank correlation coefficients of expert evaluation of tasks **complexity** for each grade with their **complexity** and **difficulty**. Correlation coefficients significantly at $p<0.05$ are highlighted in bold.

Grade	1	2	3	4	5	6	7	8	9	10	11
Tasks **difficulty**	0,41	0,46	0,51	**0,67**	**0,53**	**0,76**	**0,76**	**0,76**	**0,63**	**0,79**	**0,84**
Tasks **complexity**	**0,74**	**0,8**	**0,62**	**0,64**	**0,59**	**0,57**	0,28	0,25	**0,62**	**0,66**	0,49

Table 3 shows correlation coefficients between the number of characters in task statement (scale 4) and its **complexity** and **difficulty** (scales 2 and 3). In junior grades (from the 1st to the 4th) a significantly positive link ($p<0.01$) has been found between task statement length and its **difficulty** (number of "no answer" responses). Moreover, the link between the task statement length and its objective **complexity** was revealed only for tasks of the 3rd - 4th grades.

Table 3. Spierman's rank correlation coefficients of task statement length and its **difficulty** and **complexity.** Correlation coefficients significant at $p<0.05$ are bolded.

Grade	1	2	3	4	5	6	7	8	9	10	11
Task **difficulty**	**0,90**	**0,83**	**0,77**	**0,78**	0,17	0,13	-0,04	-0,06	0,10	0,04	-0,14
Task **complexity**	0,37	0,36	**0,77**	**0,78**	0,29	0,32	-0,18	-0,20	-0,13	-0,01	-0,23

3.2 Tasks classification and age differences in task perception

The results of the clustering are shown in fig. 3. Two clusters were distinguished. One of them includes the tasks with high **complexity** and the other includes all the other tasks. The first cluster is composed of 2 sub-clusters of tasks which differ by **difficulty**. The second cluster is also composed of 2 sub-clusters of tasks. The tasks for the second cluster differ by **complexity** while the **difficulty** of all these tasks is low. Examples of tasks from each cluster can be found on fig. 4. , i.e. the objective characteristic of the task becomes the most important. The division of tasks by the subjective evaluation by students (by **difficulty**) is made only for the tasks of the first cluster(tasks with high **complexity**). The perception of task **difficulty** by students is not important for clustering of the tasks from the second cluster , tasks with low **complexity**.

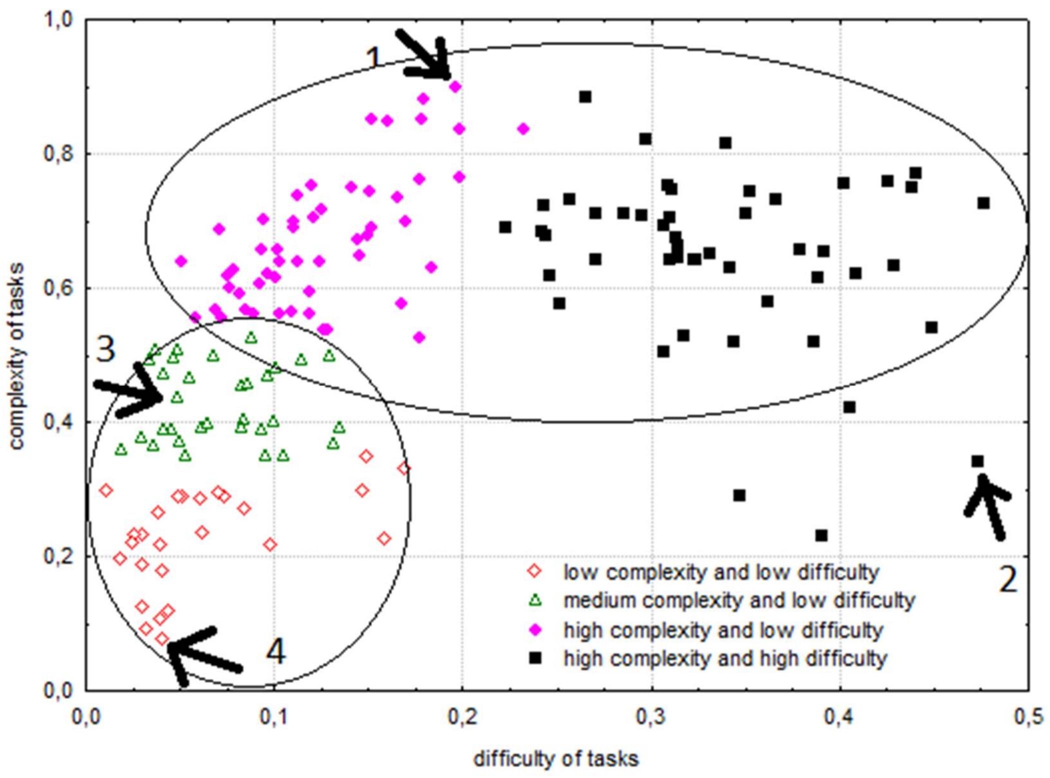


Fig. 3. Clusters and subclusters on the basis of comparison of their **difficulty** and **complexity. The statements of the tasks shown with arrows are in fig. 4.**

If the same task is offered to the children of different ages, its **complexity** and **difficulty** will be different. Schoolchildren of two grades took part in each of the first five competition levels. The results of comparison of junior and senior schoolchildren in each pair of grades are shown in Figure 5. At each level (i.e. among schoolchildren who were solving the same tasks) the results of younger participants were lower (Fig. 5A). Junior schoolchildren of levels 1 & 2 assessed task **difficulty** as higher (i.e. they chose “no answer” response, Fig. 5B) while those who were in the 4th and 5th age group assessed it as lower. Tasks were complicated (number of wrong answers, Fig. 5C) for junior children at all competition levels except the first one. We have to note that it is not correct to compare the results of children of different levels because the number and sets of tasks differed.

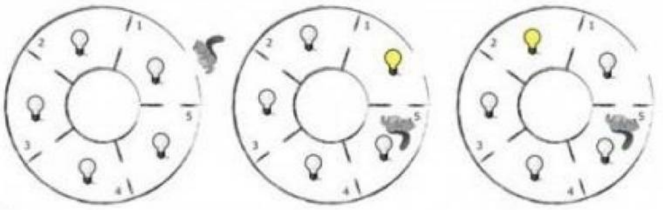
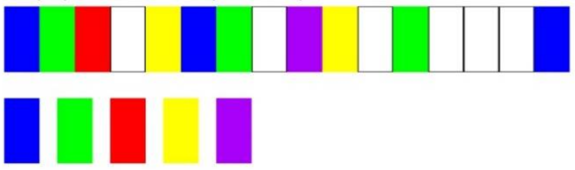
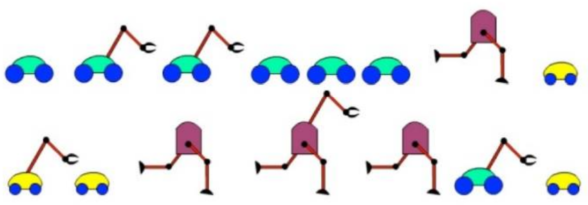
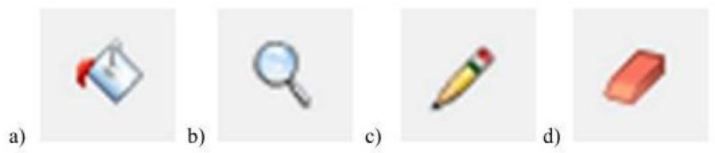
Task 1 «Restless Martin» 7nd grade Difficulty 0.2 Complexity 0.9	Every evening the beaver Martin runs around his unusual circular house: • He enters room 1 and them goes to rooms 2,3 etc. • When he starts, all the lights are swtiched off. • Every time he enters room 1, he switches on or off the light(depending on whether it was on or off). • If, in the course of going round, he switches off the light in any room, he has to change the lighting in the next room. When all the lights are finally off again, he finishes his running around.  In the picture we can see the situation before going in, after the first circle and after the second one. How many circles must Martin run every evening? A)5 B) 6 C)31 D) 32 E) no answer
Task 2 «Wallpapers» (interactive task) 1st grade Difficulty 0.47 Complexity 0.34	Beaver Charles wanted to decorate his lodge by wallpapers. He has bought five colors of wallpapers. He has fixed them on his wall so that they formed a repetitive pattern. But some of the wallpapers have fallen down. Help him to restore them back correctly. Drag the wallpapers from the bottom row to the white places on the wall so that all wallpapers form a repetitive pattern. 
Task 3 «Selecting Robots» 8th grade Difficulty 0.05 Compexity 0.44	The Beaver Robotics Company owns 15 robots of different kinds. They can listen to commands and execute them.  For a special task a team of several robots has to be selected. Somebody calls the following sequence of commands: 1. All robots with small wheels stop listening! 2. All walking robots with arms go to the meeting point! 3. All robots with arms or legs stop listening! 4. All robots go the meeting point! How many robots go to the meeting point? A) 5 B) 0 C) 8 D) 15 E) no answer
Task 4 «Graphics tools» 2nd grade Difficulty 0.04 Complexity 0.08	Choose the right tool, which is used to fill an area by a color.  e) no answer

Fig. 4. Examples of tasks from different clusters. The position of the tasks in 'difficulty-complexity' space is shown in fig. 3

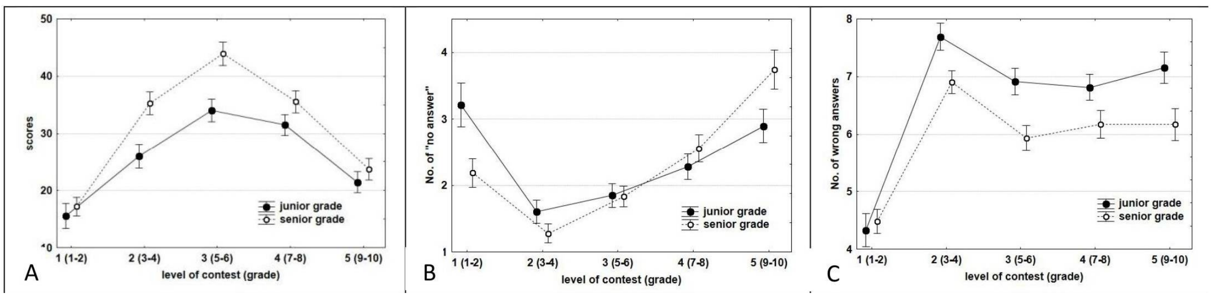
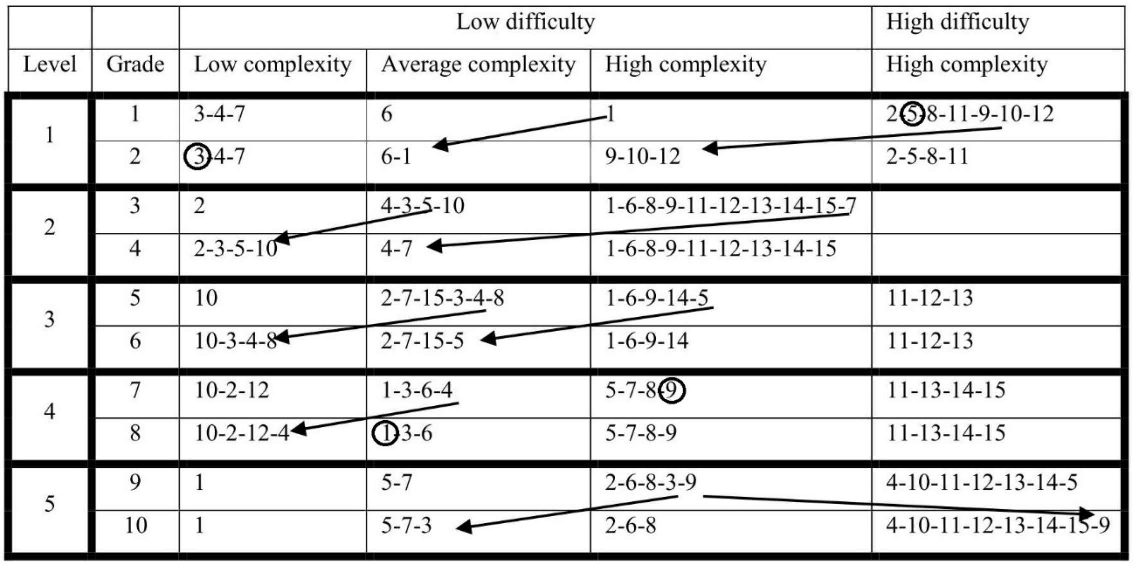


Fig. 5. Differences of results of junior and senior schoolchildren at each competition level. A. Differences of average score B. Differences in assessment of task **difficulty**. C. Differences in task **complexity**. White circles – senior grade, full circles – junior grade Averages are given with 95%-confidence intervals.

Figure 6 presents the task clustering by grades. When comparing junior and senior grades within competition levels we note that all tasks for senior schoolchildren with a single exception are of the same or lower **complexity** and **difficulty**. Only ninth task of fifth competition level is of low **difficulty** for juniors (9th grade) and of high **difficulty** for seniors (10th grade). The task **complexity** for all participants is the same, and it is high.



		Low difficulty			High difficulty
Level	Grade	Low complexity	Average complexity	High complexity	High complexity
1	1	3-4-7	6	1	2-5-8-11-9-10-12
	2	3-4-7	6-1	9-10-12	2-5-8-11
2	3	2	4-3-5-10	1-6-8-9-11-12-13-14-15-7	
	4	2-3-5-10	4-7	1-6-8-9-11-12-13-14-15	
3	5	10	2-7-15-3-4-8	1-6-9-14-5	11-12-13
	6	10-3-4-8	2-7-15-5	1-6-9-14	11-12-13
4	7	10-2-12	1-3-6-4	5-7-8-9	11-13-14-15
	8	10-2-12-4	1-3-6	5-7-8-9	11-13-14-15
5	9	1	5-7	2-6-8-3-9	4-10-11-12-13-14-5
	10	1	5-7-3	2-6-8	4-10-11-12-13-14-15-9

Fig. 6. Task clustering for each grade. Numbers of tasks are given in cells. Arrows show changes of **complexity** or **difficulty**. The circled tasks are the tasks from fig. 3. Their statements can be found in fig. 4.

4 Discussion

4.1 Difficulty and complexity of tasks

Due to a significant right skewness of distribution of total scores, the selection of winners and ranking of the strongest participants run at high point. Ranking of the main body of participants is rough. A set of tasks used for this competition could be more appropriate if its goal was to select the best students. Taking into account that the competition is aimed at the general public in order to heighten the interest in the subject and tailored for students of general education schools, the set of tasks must be admitted to be far from successful.

As it was mentioned above, most of the tasks proved to be complicated for the majority of participants. There is a possibility to make some assumptions about the reasons for that. The assumptions were made based on the results of comparison of expert estimation of tasks and their **complexity** and **difficulty**. As far as it concerns the tasks proposed to junior pupils, the correlation between expert estimation and task **complexity** is high while the one related to the **difficulty** is insignificant. The elder the pupils were, the closer was the task assessment by organizers to its **difficulty** set up in the protocol of competition. However, the correlation between expert opinion and task **complexity** was not always established in tasks designed for senior students. That is to say, the experts do not evaluate accurately the **difficulty** of tasks for juniors and their **complexity** for seniors.

Our results confirm to some extent the opinion of Navon and Gopher ([18]) that one of the components contributing to the task **difficulty** is its statement length. In junior classes the tasks became significantly more

difficult with the increase in statement length. Perhaps, this factor was not taken into account by experts when estimating tasks for junior grades which resulted in making the task excessively complicated. We suppose that the task length was the factor which determined a large number of refusals in junior classes, even though the tasks were not complex, in fact. This must have skewed the results of measurement of knowledge and skills of junior pupils. Because of lower level of development of mental processes junior pupils misunderstand long texts. It was proved that **complexity** of text provokes loss of interest to its content ([9]), therefore, tasks with too intricate statements should be avoided.
Another complicating factor is interface peculiarities of a competition. Tasks containing long texts may have not enough space to be presented on one screen. In this case to read it from the screen some skills related to computer-literacy will be needed (to know what "vertical scroller" means and how to use it), as well as fine motor skills shaped in a certain manner (to know to use a mouse).
For senior students the length of text is not an extra complicating factor. Moreover, during educative process a personal experience to assess the **difficulty** of a task by eye is being gained as well as schemata to differ "difficult" and "simple" tasks. That is why the tasks estimated by experts as difficult prove to be such for senior participants.
Having made mistake with assessment of task **difficulty** for junior schoolchildren, the experts evaluated best of all the **complexity** of tasks for them. As far as it concerns senior pupils the objective **complexity** of tasks did not coincide with an a priori opinion of organizers. It probably means that an objective **complexity** of a task is due to students' knowledge to a large extent. The knowledge of senior students participating in the competition on informatics was overestimated by organizers. The impression is that the experts were oriented to select and estimate competition tasks for a standard student of secondary school.
When considering the causes of incorrect a priori evaluation of task **complexity**, we should bear in mind that the competition is international. It is well-known fact that same tasks of TIMMS and PISA can have different weights for children from different countries. This is due to varied linguistic and cultural backgrounds ([12], [1]). Appropriate **complexity** evaluation by the experts is essential for adaptation of the tasks of international competition for the specialties of each country. It would be interesting to compare the results gained by us with the data from other countries.

4.2 Tasks categorization and age differences with regard to task perception

Tasks of low **difficulty** prevail in the competition. Pupils are more disposed to solve tasks than to make "no answer" choice. As shown above, the selection of "no answer" version gives no advantage in scores over a simple guessing.
The tasks where a large part of pupils refused to find a solution are of higher interest – these are tasks of second cluster. Most of them were found in tests designed for junior and senior students. We suppose that this cluster contains nonstandard tasks that frighten pupils by their presentation. The probability of giving a right answer is instinctively assessed as extremely low (this is not consistent with the reality – this probability is not less than ¼). However, as stated by Sočan([24]), the choice between the omission of answer, i.e. 'no answer' option, and guessing an answer even in the conditions where 'no answer' gives you an advantage in scores over guessing, the choice students make are determined by the personality, instructions given for the exam and other not linearly connected factors.
Three other clusters contain more intelligible and/or ordinary tasks. The simplest (tasks of low **difficulty** and **complexity**) are those called in competition slang "consolation tasks". Their solution is possible for practically each participant. The existence of such tasks in competition gives a positive emotion even to those who solved few tasks. The number of such tasks proposed for competition was negligible.
The above results show that a mere one year gap makes significant differences with regard to the perception of the same tasks. Junior participants of elementary schools are prone to assess tasks as more difficult and they are ready to choose "no answer" version more often than senior pupils. Elder pupils of elementary schools are prone in the contrary to assess tasks as less difficult (Tab. 3). The **complexity** of tasks for younger pupils of elementary school is in fact higher which becomes apparent in larger number of wrong answers and it is not surprising that it leads to lower results.

5 CONCLUSION

1. The evaluation of **complexity** and **difficulty** of the tasks of the competition "Beaver" was conducted based on the analysis of the protocols.

2. It was shown that the organisers underestimated the **complexity** for younger pupils and overestimated the **difficulty** for older.
3. The length of the statement of the task was demonstrated to be one of the factors determining the **difficulty** of the task for primary school children.
4. The cluster of non-standard tasks was clustered out.
5. It was shown that the difference of one year in the age of the school children results in significant differences in their perception of **difficulty** and **complexity** of the same tasks.

REFERENCES

1. ARFFMAN, I. (2012). TRANSLATING INTERNATIONAL ACHIEVEMENT TESTS: TRANSLATORS' VIEW. FINNISH INSTITUTE FOR EDUCATIONAL RESEARCH. REPORTS 44. 92P.
2. Ball, G.A. (1990). Theory of training tasks: Psychological and educational aspect. Moscow, Russia: Pedagogy.
3. Benjamin, R. G. (2012). Reconstructing Readability: Recent Developments and Recommendations in the Analysis of Text Difficulty: Educational Psychology. Vol. 24, pp. 63 – 88.
4. Cain, B. (2007). A review of the mental workload literature. Toronto, ON, Canada: Defense Research and Development Canada.
5. Chang, S.-W. (2009). Choice of Weighting Scheme in Forming the Composite: Bulletin of Educational Psychology. Vol. 40, N 3, pp. 489–510.
6. Dagienė, V. (2006). Information Technology Contests--Introduction to Computer Science in an Attractive Way: Informatics in Education-An International Journal. Vol. 5, N1, pp. 37-46.
7. Dagienė, V., Futschek G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. s. In R.T. Mittermeir and M.M. Sysło (Eds.), Informatics Education – Supporting Computational Thinking, Vol. 5090 of Lecture Notes in Comput. Sci.. Springer, Berlin, 19–30.
8. Delandshere, G. & Petrovsky, A. R. (1998). Assessment of Complex Performances: Limitations of Key Measurement Assumptions: Educational Researcher. Vol. 23(2), pp. 14 – 24.
9. Fulmera, S. M. & Tulis, M. (2013). Changes in interest and affect during a difficult reading task: Relationships with perceived difficulty and reading fluency: Learning and Istruction. Vol. 5, pp. 11 – 20.
10. Gibson, D. & Clarke-Midura, J. (2013). Some Psychometric and Design Implications of Game-Based Learning Analytics. Perth, Australia: Curtin University.
11. Golikov, Yu. Ya. & Kostin, A. N. (1996). Psychology of facilities management automation. Moscow, Russia: RAS Institute of Psychology.
12. Grisay, A., de Jong, J., Gebhardt, E., Berezner, A., & Halleux-Monseur, B. (2007). Translation equivalence across PISA countries. Journal of Applied Measurement, 8 (3), 249–266.
13. Kantovits, B. & Sotokin, R. (1991). Human factor: Mir. Vol. 4, pp. 85 – 113.
14. Kiriukhin, V. M. (2012). Methodological recommendations on elaboration of tasks for school and municipal stages of All-Russian competition on informatics for students in 2012/2013 academic year. Moscow
15. Krotov, V. M. (1999). On complexity of problems on physics: Physics: teaching problems. Vol. 3, pp. 69 – 74.
16. Leontiev A. N. (1975). Activity. Comsciousness. Personality. Moscow, Russia: Politizdat.
17. Lord, F. M. (1952). Theory of Test Scores: Psychometric monograph. Vol. 7.
18. Navon, D. & Gopher, D. (1979). On the economy of human information processing systems: Psychology Review. Vol. 86, pp. 214 – 255.
19. Opmanis, M., Dagiene, V., Truu, A. (2006). Task Types at Beaver Contests Standards. In: Dagienė, V., Mittermeir, R. (eds.) Proc. of the 2nd Int. Conference Informatics in Secondary Schools: Evolution and Perspectives, Vilnius, pp. 509–519.
20. Piaget, J. (1951). Psychology of intelligence. London, Great Britain: Routledge and Kegan Paul.
21. Ponomarenko, V.A. Cherniakov, G. M. & Kostritsa, V.G. (2013). Operator's mental states as the object of engineering and psychological researches: Cybernetic issues.
22. Riley, V. Lyall, E. & Wiener, E. (1994). Analytic workload model for flight deck design and evaluation: Proceedings of the Human Factors and Ergonomic Society. Vol. 38, pp. 81 – 84.
23. Sammer, G. (1997). Concepts of mental workload in psychological research: Proceedings of the 13th Triennial Congress of the International Ergonomic Association. Vol. 5, pp. 368 – 370.
24. Sočan, G. (2009). Scoring of multiple choice items by means of internal linear weighting: Review of Psychology. Vol. 16, N2, pp. 77-85.
25. Towler, L. & Broadfoot, P. (1992). Self-Assessment in the primary school: Educational Review. Vol. 4(2), pp. 137 – 155.

26. Yagunova, E., Ryzhova, N. (2013). Use of protocols of online competitions for an assessment of complexity of tasks and increase of a validity of measuring procedure. Komp'ûternye instrumenty v obrazovanii. Vol. 6, pp.33-44.
27. Yagunova, E., Pozdnyakov, S., Ryzhova, N., Razumovskaia, E., Korovkin, N. (2015). Analyses of difficulty and complexity of items in international on-line competition "Beaver". IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations". Preliminary Proceedings. July 1 st - 3 rd , 2015, Vilnius University, Lithuania. Andrej Brodnik, Cathy Lewin (Eds). pp. 242-254

ISSEP 2015 — The Poster Session

Poster presentations are an integral part of this conference including a session with a fast-forward presentation of the poster's summary to all conference attendees. Twelve posters are presented, with various interesting topics. In addition, an extended abstract of each poster is published in the Local Proceedings. Topics covered are quite diverse ranging from describing the situation with computer science in different countries to various approaches in learning and teaching programming. Altogether, 24 authors coming from nine different countries are authors of these posters.

The overview of current state of computer science in Swiss high schools is reported by Jean-Philippe Pellet, Gabriel Parriaux, and Morgane Chevalier, contrasting the presentation by Okan Arslan and Selcan Kilis on Informatics Teacher Education in Turkey. The most represented topic is teaching computer programming, presented in various posters. Greg C Lee and Ling-Chian Chang talk about transition from visual programming language to C, Zsuzsanna Szalayne Tahy is approaching teaching programming indirectly with the use of "Paint" programme and Boštjan Resinovič uses a humanoid robot in teaching computer programming. Michele Moro and Luigino Calvi discuss concurrent programming basics through Snap! Gregor Jerše, Sonja Jerše, Matija Lokar and Matija Pretnar present their YASAAPE – a system for automatic assessment of programming exercises. Paul Libbrecht and Wolfgang Muller describe a vision of supporting the teachers towards the choice and adoption of ICT-based learning scenarios. The influence of teaching methods during technical e-safety instruction is analysed by Vaclav Šimandl, Vaclav Dobiaš, and Michal Šery. Martina Palazzolo and Paolo Mauri report how they used PirateBox to teach how to create simple web pages. Wolfgang Pohl and Jorg Westmeyer propose content categories for Informatics Tasks while Paul Libbrecht discusses alternatives for publishing open educatinal resources (OERs) and how they can be found using regular tools on the web.

16th September, 2015

Matija Lokar, Poster Session Chair
University of Ljubljana

Informatics Teacher Education in Turkey

Okan ARSLAN[1] and Selcan KİLİS[2]

[1] Middle East Technical University
Ankara, Turkey
okana@metu.edu.tr
[2] Middle East Technical University
Ankara, Turkey
skilis@metu.edu.tr

Abstract. The purpose of the study is to examine the informatics teacher education programme and ICT curriculum in Turkey. The study also compares national ICT teacher education programme in public universities with national ICT curriculum in public schools. This study applied systematic review as a quantitative method. Descriptives and frequencies were analysed in order to explore, define and interpret the data. There were 67 public universities in Turkey which have Computer Education and Instructional Technology (CEIT) department. Curriculums of all universities were investigated. After excluding missing data, there were 34 universities. Results indicated that there were 4 main course categories: Domain Knowledge (38 percentage), Pedagogy (22 percentage), General (29 percentage), and Elective courses (11 percentage). The scope of Informatics teacher education programme was found much higher than ICT curriculum in schools that Informatics teachers teach in schools.

Keywords: informatics, teacher education, ICT curriculum in Turkey, CEIT

1 Introduction

In information age, information and communications technology (ICT) and tools has advanced. With the purpose of meeting with the demands of this age, a new ICT program was developed based on standards-based curriculum approach [2].

The name of the course was changed as Information and Communication Technologies and Software from Information Technologies in 2012 to match with new topics covered in new program. In new ICT program, learning domains covered are digital literacy, communication, knowledge sharing and self-expression via ICT, research, knowledge construction and collaboration, problem solving, programming and development of authentic materials.

This new ICT program, based on some taxonomies and levels [1, 3, 4] includes 3 main levels namely basic, intermediate, and advanced with 2 dimensions which are I and II. In new ICT program, evaluation methods are Portfolio, Rubric, Peer evaluation, Self-evaluation, and Performance evaluation.

2 Research

The purpose of this study is to introduce and examine national ICT teacher education programme and ICT curriculum in Turkey. This study compares also national ICT teacher education programme in Turkey' public universities with national ICT curriculum in public schools in Turkey.

Systematic review as a quantitative method was used as a research methodology. Descriptives and frequencies were analysed in order to explore, define and interpret the data. There were 67 public universities in Turkey which have Computer Education and Instructional Technology (CEIT) department.

Curriculums of all universities were investigated. In the organisation and preparation process, data sets that had missing data were excluded. After this step, the number of universities were reduced to 34. Thus the sample of this study is 34 university CEIT curriculums. As a final step descriptive data was analysed and the results interpreted. The course requirements are formed of the courses: Domain Knowledge (38 percentage), Pedagogy (22 percentage), General (29 percentage), and Elective courses (11 percentage).

3 Conclusion and further research

Matching the courses taught in ICT in schools with teacher education programme indicates that only a small part of the courses in teacher education programme are enough for the whole national ICT curriculum in public schools. National ICT curriculum includes mainly algorithm, programming, ICT literacy, ethics and privacy, and project development and applicaiton. These are included in seven courses in domain knowledge courses in teacher education programme in the universities. So, teachers cannot teach and use all their knowledge that offered themselves in the universities. Therefore, it is suggested that either the scope of course Information and Communication Technologies and Software should be enhanced or scope of teacher education programme in the universities should be reduced.

References

1. Fraillon, J., Ainley, J.: The IEA International Study of Computer and Information Literacy (ICILS). International Association for Evaluation of Educational Achievement. 2011
2. Gülbahar, Y., Ilkhan, M., Kilis, S., Arslan, O.: Informatics education in Turkey: national ICT curriculum and teacher training at elementary level. In: Informatics in Schools: Local Proceedings of the 6th International Conference ISSEP 2013–Selected Papers (p. 77)., Oldenburg, Germany, 2013
3. Tomei, L.A.: Taxonomy for the Technology Domain. In: Information Science Publishing, USA, 2005
4. UNESCO: Strategy Framework for Promoting ICT Literacy in the Asia-Pacific Region. In: UNESCO Bangkok Communication and Information Unit, Bangkok, 2008

YASAAPE – Yet Another System for Automatic Assessment of Programming Exercises

Gregor Jerše, Sonja Jerše, Matija Lokar, and Matija Pretnar

Faculty of Mathematics and Physics
University of Ljubljana, Slovenia
`Matija.Pretnar@fmf.uni-lj.si`

Abstract. Programming is a skill where teachers are required to both encourage students by exposing them to numerous problems and supervise their attempts to solve them. To support this teaching approach we developed a web service *Projekt Tomo*, presented in this poster.
The service is designed in such a way that it requires little or no additional work from students and teachers, enabling them to focus on the content. Furthermore, the service can be used in almost all teaching environments, as it can be adapted to most programming languages and has minor technical requirements.

Keywords: programming, teaching, web service

1 Introduction

In introductory programming courses groups are usually large and heterogeneous. Assessment of programming exercises by manual inspection of code on paper is notoriously inefficient and error-prone. So several different automatic assessment systems have been proposed. The obvious benefits of automatic assessment are the objectivity, consistency and speed of assessment, as well as constant availability.

Tools for automated assessment of programming assignments are a well-researched approach to support teaching programming. There are several surveys of such systems (among others [1–3]) and there are numerous such systems available with different features such as programming languages supported, connection to learning management systems, possibilities of defining tests, ways resubmissions are tackled, possibility of manual assessment, security issues, distribution and availability, and others. For a review of possible features see for example [2, 3].

2 Main objectives of the service

As all the known solutions failed to provide proper support to our education process for various reasons, we started developing our own *Projekt Tomo*[1] service in 2010. Our main objectives were:

[1] tomo.fmf.uni-lj.si

Low overhead interaction We wanted to keep the interaction with the system (uploading solutions, waiting for feedback. . .) to the minimum.

Possibility of remote work A service was required that would be available 24 hours a day and both at school and at home.

Local execution Due to security issues a solution was required that executes programs on the student's computer. This approach also allows usage where the internet connection can be unreliable (at home, in dormitories. . .).

Ability to choose a preferred coding environment Students should use a coding environment of their choice.

Independence of a programming language Different courses use different programming languages, each suited for its particular task, and ideally, the proposed service should be adaptable to offer support to any one of them.

Flexibility in administering tests In addition to a direct string comparison, much richer possibilities for testing have been foreseen.

Open source development

The fundamental idea of the resulting system is as follows. A student downloads the file containing problem instructions (in form of comments) from the server and enters his own solutions into the file.

When a student executes a program, his attempts are evaluated and a feedback is given, even when his computer is offline. When connected to the internet, his attempts are also automatically saved on the server.

3 Conclusion

The *Projekt Tomo* offers many features that are expected from a service for teaching programming like low overhead interaction, remote work, local execution, preventing various malicious attacks and allowing the service to have small technical requirements, independence of both a coding environment and a programming language and open source development. So far, this service has been successfully used in several different courses. In the 2013/14 school year, this service was used by 10 teaching assistants and about 500 students. Altogether, they solved 40.000 problems in 600.000 attempts.

References

1. Ala-Mutka, K.: Survey of Automated Assessment Approaches for Programming Assignments, in: Computer Science Education, Volume 15 Issue 2, pp 83–102, 2005
2. Caiza, J.C., Del Alamo, J.M.: Programming assignments automatic grading: review of tools and implementations, In: INTED2013 Proceedings, 7th International Technology, Education and Development Conference, pp 5691–5700, 2013
3. Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O.: Review of recent systems for automatic assessment of programming assignments, In: Proceedings of the 10th Koli Calling Conference on Computing Education Research, pp 86–93, 2010

Learning to Program: from VPL to C

Greg C Lee[1] and Ling-Chian Chang[2]

[1] National Taiwan Normal University
Taipei, Taiwan
leeg@csie.ntnu.edu.tw
[2] Hsin-Tien Senior High School
New Taipei City, Taiwan
ninachang@htsh.ntpc.edu.tw

Abstract. In recent years, visual programming langauge (VPL) has been very popular for introducing programming to K-12 students. Although students may enjoy the fun of visual programming, it is still a challenge to leap into programming in C/C++/Java. In this study, we aim to find ways to bridge the gap between VPL and C programming. Curriculums have being developed and pilot study is underway. Results on students' development of computational thinking skills as well as programming skills are to be reported at the conference.

Keywords: Visual Programming Language, Computational Thinking, Computer Programming

1 Introduction

The need for problem solving techniques is inherent in engineering, as well as scientific and other disciplines [1]. Improvements in computing have made it possible for all fields to incorporate computation as part of their problem solving and research process. Programming skills also have impacts on the development of thinking skills; the connection between programming and problem solving has long been established [3]. Therefore, students who are not going to pursue computer science further can still benefit from the process of learning to program. But which language should be taught and learned by high school students? According to the Tiobe programming community index [2], the most widely used programming languages both in industry and education today are Java and C, and having been so for many years. However, there have been much debate about the suitability of these programming languages for education, especially when introducing programming to novices [4, 5]. These languages are criticized for the verboseness in syntax and overhead in notation so that students have little practice in thinking algorithmically and writing structured programs. Over the past several years, new visual programming languages have been developed targeting K-12 students to learn about programming with a focus to develop computational thinking skills. Programming languages and environment such as Scratch and Greenfoot have since attracted own group of teachers. Although visual programming learning environment might be easier for novice programmers to handle, can those programmers make the necessary leap into programming in C/C++/Java in later courses? In this ongoing research, we proposed a learning progression for going from visual programming to line-based programming for high school students. In year one of this three year research, curriculum has been developed and a pilot study is been conducted.

2 Research

There are four phases in the planned programming learning progression:

1. Learn to think computationally through VPL (e.g. Code.org).
2. Learn to solve simple problems computationally through VPL (e.g. in-house Blockly tasks).
3. Learn to devise algorithm and solve problems computationally through VPL (e.g. in-house Blockly tasks).
4. Learn the syntax of C and solve the same problems in (3) in C/C++.

The in-house Blockly tasks and programming environment have been fully developed. Pilot study is to be conducted during the summer session (July and Auguest) for 20+ incoming freshman (10th grader) at a local high school. Students will be pre- and post-tested with Bebras tasks to assess changes in their computational thinking skills. Furthermore, learning difficulties during each step of the learning progression will be monitored and reported.

3 Conclusion and further research

The pilot teaching experiment is ongoing. However, the small sample size test conducted during the task development showed that students were able to learn to program in VPL during the first three phases. We are confident that students will be able to build on their VPL programming skills to solve problems with the C programming language.

References

1. Wing, J.M.: Computational Thinking. In: Communications of the ACM, 49(3), 33-35, ACM New York, NY, USA, 2006
2. http://www.tiobe.com
3. Soloway, E.: Should we teach students to program? In: Communications of the ACM 36(10), 21-24, ACM New York, NY, USA, 2006
4. Gupta, D.: What is a good first programming language? ACM Crossroads. 10(4): 7–7, ACM New York, NY, USA, 2004. doi: 10.1145/1027313.1027320
5. Kölling, M.: This much I know: thoughts on the past, present and future of educational programming tools. In: Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE ' 13): 5–6, ACM New York, NY, USA, 2013. doi: 10.1145/2445196.2445200

Publication of Learning Resources: Central or Interoperable?

Paul Libbrecht

Weingarten University of Education, Germany
(Supported partially by the EU project OpenDiscoverySpace)
`libbrecht@ph-weingarten`

Abstract. This poster discusses alternatives for publishing open educatinal resources (OERs) and how they can be found using regular tools on the web. This discussion attempts to propose solutions to the recurring problem of low endorsement of OERs.

Keywords: open educational resources, sharing, web-authoring

1 Introduction

Activities around Open Educational Resources exist since more than a decade. Expectations such as the ease of re-purposing because of modularity are at least as far as 2000, e.g. in [4]. Repositories that collect Open Educational Resources (OERs) are numerous with broad and specialized servers. However, the penetration of OERs in the daily life of teachers is not yet broad.

A common reason is the *spread* of learning resources in a way that make them hard to find again. Whereas the early OERs history proposed sharing platforms where one could have the impression that everyone of a community will converge on a single platform, specialized tools (such as encoding, preview, or evaluation services) and specialized vocabularies have been developed and many specialized platforms have emerged contributing to a broader spread. Even for a small field such as that of mathematics education with dynamic geometry, there exists a generic server (i2geo), and several specialized servers (e.g. Sketchpad Exchange, GeoGebraTube), and many author-publishers which display on their website.

How can regular teachers find across this collection? Only individual OER platforms provide search functions and each using their vocabulary. Global web search engines offer reasonable services for well-identified concepts which can be expressed using simple words but fail very quickly as soon as refined concepts of the disciplines to be learned are used or when pedagogical concepts are expected (often, synonyms' suggestions are not possible, and concepts expressed in multiple words yield results with unrelated words in the documents).

2 Possible Solutions

One of the solution is that learning resources travel among the servers: using processes such as the OAI-PMH harvesting allows the information about a learning resource to be collected so as to be displayed in other repositories. Harvesting has been applied with success to many OER repositories including OERcommons, LreForSchools, and Gooru which collect from multiple sites and offer an enhanced access to resources. In Europe, the platform OpenDiscoverySpace.eu has been designed with this strategy: harvesting as many learning resources as possible from dozens of repositories, overall about a million, providing a taxonomy vocabulary that is adjusted to Europe, and supporting the further enrichment of resources' metadata on the portal by tools for scenarios and communities. Such a large server provides an answer to a unified search and collaboration facility for OERs. However, small author repositories are not harvestable and are thus excluded. Moreover, harvesting often looses precious information.

Generic search engines could be refined to take more information in account from web-pages so that learning-resources information is faithfully searched. The micro-data markup LRMI [3] contributes to it and a prototype to employ Google's search for this has been made [1]. However, contrary also to the expectations in [2], there does not seem to be an interest yet from the big search engines to offer a mature service of this sort, the *educational domain vertical* is not yet a feature.

Other strategies support the transfer of some information carried in a web page to other servers which are in current use. E.g. Social networks witness the content of a link when sending message that mention one: they show an *excerpt*.

3 Conclusion and Future Work

The options to publish learning resources that we have presented leave current authors perplex: Should I simply use a repository or should I maintain my web-page and become a web-site designer to stage my creations? (a standard competency in secondary schools). Is the example of medien-in-die-schule.de to be followed? How will others talk about my learning resource and at which URL?

Displaying key information of the resource within communication, and ensuring that this information is sufficient for search engines, social networks and repositories, might offer a scalable way for all teachers to host their learning resources while still allowing them to be easily found. Approaches to this solution might be to turn an LRMI web-crawler into an OAI-PMH server, or to develop such custom search engines as [1] to understand many discipline concepts.

References

1. Phil Barker, *Building a Google custom search engine for LRMI-tagged pages*, `http://blogs.cetis.ac.uk/philb/?p=976`.
2. EdReNe Project, *D3.4 Repository Strategies*, Available from `http://edrene.org`.
3. Creative Commons and Dublin Core, *Learning Resources Metadata Initiative 1.1*, `http://www.lrmi.net/the-specification`.
4. Wiley D. A. (2001) Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In Wiley D. A. (editor), *The Instructional Use of Learning Objects*, pp. 1-35. AECT, 2001. See `http://reusability.org/`.

Concurrent programming basics through Snap!

Michele Moro[1] and Luigino Calvi[2]

[1] Dipartimento di Ingegneria dell'Informazione, University of Padova, Italy
michele.moro@unipd.it
[2] Istituto di Istruzione Superiore Negrelli-Forcellini, Feltre (BL), Italy
luigino.calvi@negrelliforcellini.gov.it

Abstract. Snap! can be effectively used for introducing fundamentals of concurrent programming at secondary school level. Some basic synchronization tools have been implemented and tested.

Keywords: Concurrent programming, Snap!, Constructionism

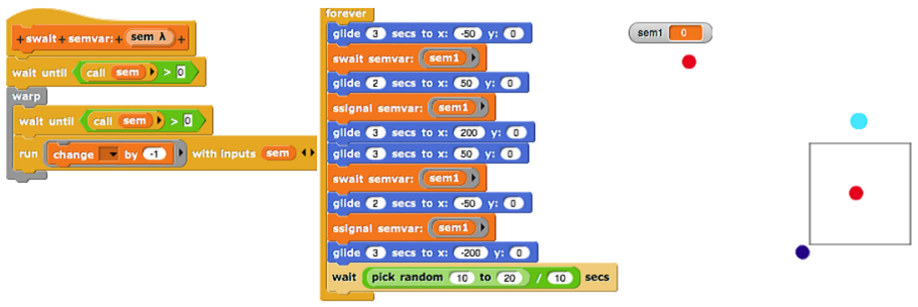
1 Introduction

Concurrent programming (CP) has been recently introduced in some curricula of secondary schools in Italy [1]. Its learning requires a special attention to subtle effects and wrong forms of synchronization: gaining a sufficient experience in CP is therefore not a trivial task. Snap! [2] is a programming environment which extends the Logo-heir Scratch, both being inspired by the constructionist Papertian pedagogy [3]. Exploiting recursion, lists and procedures as first class data, its simple forms of object-oriented and functional programming, the interpretation of its active item (*sprite*) as a simple sensorized robot on a 2D scenario [4], and its intrinsic multithreading, our proposal shows how effective Snap! is for teaching CP at this level. A first experimentation was done this year in the second biennium (16-17 years old students) of a technical secondary school: some preliminary evaluations are presented in the poster.

2 Research and proposal

Snap! concurrency is based on a time-slice, round-robin scheduling. Some basic commands of the language allow to implement several forms of synchronization between threads and to realize a variety of examples of CP in a pleasant and affordable way. First of all it is necessary to intuitively introduce multitasking in Snap! through some simple examples, like: concurrent execution of scripts on the same sprite; concurrent execution of a script with several sprites; interpretation of a script executed on an event as an event listener or an interrupt task; the evidence of race conditions when setting concurrently global variables.

The first construct we propose is the counting semaphore whose state is represented by a normal global integer variable. Its primitives are realized by suitable user blocks where the *warp* command ensures atomicity executing critical code without yielding to other threads. Figure 1 shows the implementation of the semaphore and an example of mutual exclusion (at any time not more than one sprite, represented by a colored circle, can be inside the square).



```
+swait+semvar:+ (sem λ) +
wait until < (call (sem) ▸) > 0 >
warp
    wait until < (call (sem) ▸) > 0 >
    run (change [ ▾] by (-1) ▸) with inputs (sem) ◂▸
```

```
forever
    glide (3) secs to x: (-50) y: (0)
    swait semvar: (sem1) ▸
    glide (2) secs to x: (50) y: (0)
    ssignal semvar: (sem1) ▸
    glide (3) secs to x: (200) y: (0)
    glide (3) secs to x: (50) y: (0)
    swait semvar: (sem1) ▸
    glide (2) secs to x: (-50) y: (0)
    ssignal semvar: (sem1) ▸
    glide (3) secs to x: (-200) y: (0)
    wait ((pick random (10) to (20)) / (10)) secs
```

Fig. 1: semaphore wait and mutual exclusion

Snap! messages provide an elementary form of synchronized message-passing, but they are not first class data, they are only broadcasted and cannot bring additional information, they are not enqueued and are essentially asynchronous. Due to these limitations, we realized a synchronized FIFO message queue using Snap! generic lists, together with one testing producer/consumer example.

The Object-oriented approach of Snap! can be fully exploited to implement an advanced synchronization structure like the Hoare's Monitor [5]. Following this approach, we have realized a generic constructor block returning a selector script which must be subsequently run in order to execute the requested access method. Due to its relative complexity, this constructor can be initially provided by the teacher as a black box, and afterwards examined in detail to improve the students' skills. We adopted a similar approach also to implement the semantics of the simplified Monitor of Java [6] with its *wait*, *notify*, *notifyAll* primitives.

We are convinced that our proposal smoothly introduce CP in preparation of other professional languages and environments.

References

1. Guide to the reform of technical and vocational schools (in Italian), Mondadori Education, Italy, 2012, pag. 143
2. http://snap.berkeley.edu
3. Papert S.: Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc., 1980.
4. Arlegui J., Moro M., Pina A.: Simulation of Robotic Sensors in BYOB In: Proceedings of 3rd Robotics in Education Conf. 2012, pp. 25-32 Prague, CZ, 2012
5. Hoare C.A.R: Monitors: an Operating System Structuring Concept Comm. ACM 17(10)(1974) pp. 549-557.
6. Goetz B., et al.: Java concurrency in practice Pearson Education, 2006.

Can I do that?
Scenario Feasibility as an Enabler of ICT Usage

Wolfgang Müller[1] and Paul Libbrecht[2]

[1]Media Education and Visualization (MEVIS), [2]Informatics Education
Weingarten University of Education, Germany
`muellerw|libbrecht@ph-weingarten.de`

Abstract. In this poster, we describe a vision of supporting the teachers towards the choice and adoption of ICT-based learning scenarios by means of mappings to the school infrastructure. The vision proposes the selection and curation of didactical design patterns, as repeatable solutions to problems found in such works as learning scenarios, and their mapping to each school's infrastructure. This collection of patterns, linked to experience reports and scenarios, will offer the regular teachers a way to plan for their applications with a trust of realizability.

Keywords: computers at school, course planning, teacher training

1 Adopting Scenarios: Barriers and Opportunities

Today, schools are expected to support students in building media competence and ICT literacy. This requires an comprehensive integration of media and IT into the classroom, allowing for frequent, unobstructed, and wise use of technology in all subject fields and situations. However, today's teachers often lack appropriate knowledge and experience to apply media and IT effectively in the classroom, and they require therefore adequate support in this field.

Descriptions of approaches and lesson plans that can use the information and communication technologies of the schools can be found in great amounts on the world wide web. The forms include didactical scenarios, simple re-usable material, complete interactive games, or semi-formal descriptions of best practices in terms of didactical design patterns, such as [**?**]). They are generally attractive for teachers to apply provided that some conditions are met.

Technical feasibility relates to the question whether the infrastructure at the teacher's school is sufficient to implement a selected scenario. Without a sufficient degree of confidence that this can be answered positively, teachers can only plan for an attempt, having one or two *plan Bs* in their pockets. Assessing technical feasibility requires technical skills which teachers often do not possess: Questions such as "Is the version of the plugin sufficient?" are typical sources of uncertainty.

Adequacy of content to the instructional goals and the learning environment represents another essential aspect to be met. While corresponding reports and discussions of teachers, who applied this content, can sometimes be found in rating and comment sections of supplying portals, teachers need to be able to assess relevance to the context of their classroom using their professional knowledge by simulating as much as possible.

Furthermore, resources need to provide sufficient *adaptability* to allow for the application in the teachers own classroom, if the two above criteria fail a bit.

Few approaches aim at supporting the teachers' in assessing these values. There aspects may be addressed in some teacher trainings and in peer discussions, but corresponding knowledge typically remains undocumented and therefore not at hand when required. As a result, teachers often drop ideas to adopt new approach and scenarios for using IT in the classroom.

2 The Proposed Approach

In our approach we target to bridge teachers' uncertainties in the possible application of educational scenarios and contents and to provide adequate support on demand by the means of *rich instructional patterns* and a platform providing access to these patterns and allowing for extension and commenting. *Rich instructional patterns* represent an extension classical design patterns [?] to contain not only abstract descriptions of general solutions for recurring problems in the educational field, but also to provide links to required infrastructure and possible variants of the scenario conforming to variations in the infrastructure.

Our *rich instructional patterns* will represent a compact set of guidelines for several types of applications, which focus on the essential aspects of the learning processes. They will be based on examples, but will be sufficiently abstract so that they can address a wide range of situations, and bridge the gap between the technological (what is feasible?) and the pedagogical aspects (what is intended?) of learning scenarios. These patterns will be obtained via interviews aimed at understanding what current ICT practices are followed by each participating school, and also from existing literature.

Our approach will be applied and evaluated in a collaboration project with schools and academic partners from three different countries called eSIT4SIP (Empowering the School IT infrastructures for the implementation of Sustainable Instructional Patterns). In this project we shall disseminate the rich instructional patterns found and thus contribute to the increase of efficient technology enhanced learning practices in identified schools where we shall map them to the infrastructure information.

References

1. Bergin, J. Fourteen pedagogical patterns. In *Proceedings of the 5th European Conference on Pattern Languages of Programs (EuroPLoP)* (Konstanz, 2000), M. Devos and A. Rüping, Eds., Universitätsverlag Konstanz, pp. 1–40.
2. Johnson C., Fuller U.: Is Bloom's taxonomy appropriate for computer science? In: Baltic Sea '06 Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, pp 120 - 123, ACM New York, NY, USA, 2006
3. Mor, Y., Mellar, H., Warburton, S., and Winters, N., Eds. *Practical design patterns for teaching and learning with technology.* Springer, 2014.

From Paper to Web - Some Help from PirateBox

Martina Palazzolo[1] and Paolo Mauri[2]

[1] Istituto Comprensivo Ilaria Alpi
Milan, Italy
martina.palazzolo.5@gmail.com
[2] Istituto Comprensivo I.Calvino
Lecco, Italy
paolo@paolomauri.it

Abstract. We used PirateBox in a 7th grade class to teach how to create simple web pages. The first approach was to make pupils understand the concept of digitally formatted text using an unplugged activity developed by the Aladdin team. PirateBox allowed us to maintain a strong motivation in learning HTML.

Keywords: mark-up languages, HTML, PirateBox

1 Introduction

Last year we joined the project "PirateBox a scuola" from LOPTIS (Laboratorio Online Permanente di Tecnologie Internet per la Scuola -[1]) to test PirateBox in our teaching activity. PirateBox [2] is a portable electronic device that creates a wireles network allowing the connected users to share files. We used a TPK-Link Wi - Fi router and a 16 Gb USB pen drive [3]. To organize your contents in PirateBox some knowledge of HTML is required. Aladdin's Wikipasta workshop [4] helps pupils in developing their mental model of formatted text, allowing us to introduce the use of abstrac digital tags [5].

2 Research

We first prepared our PirateBox by loading HTML pages, videos or pictures to be shared. Pupils brought their own devices endowed with a Wi-Fi antenna (notebooks, tablets or cell phones) and began learning how to connect to PirateBox. Once they got familiar with sharing by using PirateBox, we moved on to constructing web pages. We realized that pupils were lacking an important concept: in digital documents, content and formatting are two pieces of information that browsers manage in completely different ways, whereas on paper they are handled together. Aladdin's Wikipasta workshop [4] tackles this misunderstanding by replacing the symbolic manipulation of tags by a physical activity. In Figure 1 we can see a text where pupils are invited to use different kinds of pasta and colored buttons to format it. Following this activity, the association of the function of pasta or colored buttons and mark-up, like wiki or HTML tags, was mostly evident. Figure 2 shows an example of a simple HTML page that pupils wrote using the Text Editor at the end of the project.

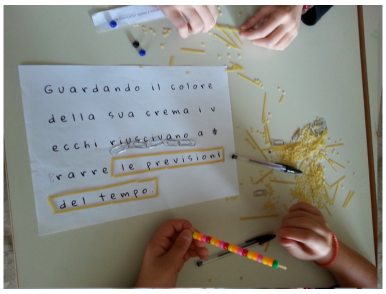
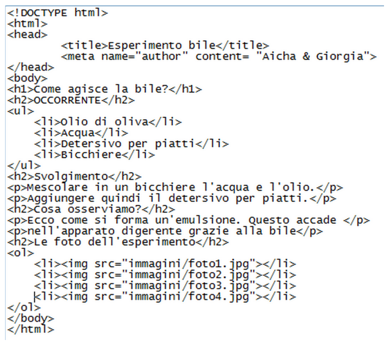


Fig. 1: wikipasta



```
<!DOCTYPE html>
<html>
<head>
        <title>Esperimento bile</title>
        <meta name="author" content= "Aicha & Giorgia">
</head>
<body>
<h1>Come agisce la bile?</h1>
<h2>OCCORRENTE</h2>
<ul>
    <li>Olio di oliva</li>
    <li>Acqua</li>
    <li>Detersivo per piatti</li>
    <li>Bicchiere</li>
</ul>
<h2>Svolgimento</h2>
<p>Mescolare in un bicchiere l'acqua e l'olio.</p>
<p>Aggiungere quindi il detersivo per piatti.</p>
<h2>Cosa osserviamo?</h2>
<p>Ecco come si forma un'emulsione. Questo accade </p>
<p>nell'apparato digerente grazie alla bile</p>
<h2>Le foto dell'esperimento</h2>
<ol>
    <li><img src="immagini/foto1.jpg"></li>
    <li><img src="immagini/foto2.jpg"></li>
    <li><img src="immagini/foto3.jpg"></li>
    <li><img src="immagini/foto4.jpg"></li>
</ol>
</body>
</html>
```

Fig. 2: HTML with Text Editor

3 Conclusion and further research

A text editor, basic HTML tags, some pictures and PirateBox allowed pupils to write and share well formatted documents. Pupils seem to have developed the mental model in order to deal with other mark-up languages, for example wiki. Besides, and indeed unexpectedly, PirateBox allowed them to build a first mental representation of the Internet: they realized that their documents sent over the intranet connection are collected in a specific physical place, the teacher's USB pen drive, and were then lead to generalize the same concept for the Internet, where these details are not easy to grasp.

References

1. Andreas Formiconi: http://iamarf.org/portfolio/piratebox/
2. PirateBox:https://en.wikipedia.org/wiki/PirateBox
3. Roberto Marcolin: https://nilocram.wordpress.com/2014/06/30/e-arrivata-la-piratebox-1-0/).
4. Bellettini T., Lonati, V., Malchiodi D., Monga M., Morburgo A.: Exploring the Processing of formatted texts by a kynesthetic approach. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education: WiPSCE '12, pp 143 - 144 ACM New York, NY, USA, 2012
5. Bellettini T., Lonati, V., Malchiodi D., Monga M., Morburgo A., and Torelli M.: What you see is what you have in mind: constructing mental models for formatted text processing. In: Proceedings of ISSEP 2013 n.6 in commentarii informaticae didacticae. pp 139 - 147

Computer Science for All in Swiss High Schools: Current State, Issues, and Perspectives

Jean-Philippe Pellet, Gabriel Parriaux, and Morgane Chevalier

Lausanne University of Teacher Education
Teaching & Research Unit for Media & ICT
Lausanne, Switzerland
`firstname.lastname@hepl.ch`

Abstract. This poster discusses the main issues at stake in the task of determining a "computer science for all" curriculum in Swiss high schools. Such a task raises fundamental questions such as: what is CS exactly; what are its subtopics and its fundamental concepts; how should it be classified with respect to other sciences; who should teach it and with which required background; etc. In the poster, we graphically depict the discussion points and conclusions we have come to on such issues.

Keywords: computer science curriculum, computational thinking, field definition, concept map, high school

1 Context

There is an international trend to shift K–12 curricula towards a more technical education [1, 4]. The targeted topics are often referred to with the STEM acronym: **s**cience, **t**echnology, **e**ngineering, and **m**athematics. Switzerland is no exception, although it was certainly no pioneer. One of the prominent members of the STEM topics is computer science (CS). The teaching of CS in Swiss high schools is the focus of this poster.

In Switzerland, about 20% of all students complete the version of high school known as "maturity school" (MS), which is the one that traditionally leads to university [3]. Global, detailed nation-wide curricula do not exist: each of the 26 cantons is responsible for its own high school. However, a national reference document known as Framework for High School Curricula (FHSC) exists, which determines the basic list of topics and sets the bounds in which the cantons are free to operate.

This poster describes the current state of the FHSC and the ongoing efforts, to which the authors are contributing as members of SSIE (Swiss Association of CS Teachers), to make it change and include CS as a full-fledged topic for everyone.

2 Main Poster Points

When the current version of the FHSC document was published in 1995, it specifically mentioned CS as not being a field *per se*, but a collection of transdisciplinary topics. In 2008, an addendum to the 1995 FHSC was published, introducing CS as an actual (but optional) field. Currently, the committee editing the FHSC is considering recommending adding CS as a mandatory field for every student in MS. This is clearly a disciplinarization process, which the authors adhere to.

The issues we are discussing in the poster are ones the authors have needed to deal with while writing a proposal for a CS curriculum for MS. They include:

– **What is CS?** We detail the three-pillar breakdown proposed by the Hasler Foundation (unaffiliated with the authors) [2]:
  1. *Fundamental CS* (FCS) as both an applied and formal science;
  2. *Digital literacy*, or the ability to adequately use CS-based software and hardware tools;
  3. *Media education* as a social science.

  We argue that FCS should be the main focus of a CS-for-all curriculum.
– **What are the main subtopics of FCS?** We present a high-level cartography of CS as an applied and formal science and a concept map derived therefrom.
– **Where should FCS be classified in curricula?** The current Swiss curricula tend to classify sciences as either experimental sciences or human sciences. We argue that it is vain to try to make FCS as an applied science fit in either category and that applied sciences deserve a place of their own.
– **How should FCS for all be taught?** Rather than delve into mainly theoretical or abstract concepts for their own sake, we argue that FCS for all should start from very applied problem-solving tasks.
– **Who should teach FCS?** Current laws mandate teachers to have the equivalent of a Master's degree in the field they teach. We foresee that this condition may not be fulfilled for the first years where FCS for all would be taught, as current CS teachers are mostly non-specialists who teach concepts closer to digital literacy than to FCS.
– **When should we start teaching FCS concepts at school?** We consider it possible and desirable to get acquainted by FCS concepts much earlier than at the MS level (16–19 years old). There, these concepts could be seen as belonging to *computational thinking* and be understood in a wider context than strictly CS.

Finally, we show how each of these questions has had a direct or indirect impact on the format of our modification proposal of the FHSC document and mention potential further issues.

References

1. R. W. Bybee. Advancing STEM education: A 2020 vision. *Technology & Engineering Teacher*, 70(1):30–35, 2010.
2. P. Kleiner. *Was ist Informatik?* Schriftenreihe der Hasler Stiftung, 2014.
3. Swiss Federal Statistical Office. *Indicateurs de la formation suisse*, 2014.
4. US Congress Joint Economic Committee. *STEM Education: Preparing for the Jobs of the Future*, 2012.

Content Categories for Informatics Tasks

Wolfgang Pohl[1] and Jörg Westmeyer[2]

[1] BWINF / Bundesweite Informatikwettbewerbe
Bonn, Germany
pohl@bwinf.de
[2] Rheinische Friedrich-Wilhelms-Universität
Bonn, Germany

Abstract. Organizers of task-based informatics competitions aim at composing task sets which cover diverse areas of the field. To achieve this goal, a system of categories is needed for classifying tasks according to the content area they cover. We identified and analyzed several category systems for informatics content. From that, we derived a new system that allows for task classification along both abstract and specific concepts of informatics.

Keywords: competitions, informatics areas, content categories

1 Introduction

Competitions have proven to be popular and effective in motivating K-12 students to discover or demonstrate their talents. Informatics is no exception: from motivational contests like Bebras [3] to high-level competitions for the most brilliant talents like the International Olympiad in Informatics, there is a large portfolio of very different school-level competitions in informatics.

Usually, in a competition involving tasks, organizers aim at composing a well-balanced task set. In particular, tasks should nicely distribute content-wise, i.e. should cover a large part of the competition subject (informatics in our case). To assess this requirement, a system of content categories is needed. Then, tasks can be assigned to categories to finally show how the task set is distributed over the categories. We analyzed existing content category systems for informatics, and now we are proposing a new system for content classification of informatics (competition) tasks.

2 Research

For the Bebras challenge, a set of content categories was presented in [3]. This set contains categories like ALG for tasks about algorithms and basic algorithmic thinking, or INF for tasks about representation of information. In practice, this category system has shown to be too coarse to be useful for task set assessment.

In order to develop a more fine-grained system, we considered many proposals to characterize the essential aspects and concepts of informatics; for instance:

– The computational thinking (CT) vocabulary [2] lists the following basic computational thinking skills: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms & procedures, automation, simulation, and parallelization.
– The fundamental ideas of computer science [4] are based on three "master ideas": algorithmization, structured dissection, and language (later extended to formalization). Each of these master ideas is the root node of a tree of concepts; each concept is required to be fundamental according to a set of well-defined criteria.
– Areas of "content competences" used to define educational standards for CS education in middle school: information and data; algorithms; languages and automata; informatics systems; and informatics, man, and society [1].

Based on our experience with competitions, we looked for a content category system that (a) is rich enough to allow for fine-grained task classification; (b) is organized hierarchically to allow for categorization on different levels of detail as well as for flexibly introducing subcategories if more detail is needed; and (c) allows for categorization along abstract notions (like those of the CT vocabulary) as well as specific or even application-oriented terms.

Only the fundamental ideas [4] show a hierarchical approach. Therefore, we propose a set of category hierarchies, the root nodes of which mainly correspond to Schwill's master ideas (algorithmization, structuring, formalization). A fourth, application-oriented hierarchy has the root node "informatics systems".

3 Conclusion and further research

We have made a first step towards a content category system for informatics competition tasks. Now, our proposal needs to be tested and evaluated. We will apply the system to the tasks of competitions like Informatik-Biber 2015, this year's German implementation of Bebras.

References

1. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS – educational standards for computer science in lower secondary education. In: ITICSE 2009 proceedings (2009)
2. CSTA, ISTE: Computational thinking teacher resources (second edition). http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf. Accessed 12-08-2015.
3. Dagiene, V., Futschek, G.: Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R., Syslo, M. (eds.) Informatics Education – Supporting Computational Thinking, pp. 19–30. LNCS 5090, Springer-Verlag, Berlin Heidelberg (2008)
4. Schwill, A.: Fundamental ideas of computer science. EATCS-Bulletin (53), pp. 274–295 (1994)

The use of Nao, a humanoid robot, in teaching computer programming

Boštjan Resinovič

Šolski center Celje, Srednja šola za kemijo, elektrotehniko in računalništvo
Celje, Slovenia
`bostjan.resinovic@guest.arnes.si`

Abstract. Visual programming languages can reduce a novice programmer's problems in mastering the syntax of a language and developing computational thinking. But when the intrinsic switch to a traditional programming language is made, the same problems, along with new ones, arise. Students are faced with a different language, IDE, platform and often programming paradigm. To help overcome all of the above mentioned problems we propose using Nao, a humanoid robot, and its programming tools.

Keywords: computer programming, visual programming language, Choregraphe, Nao, humanoid robot

1 Introduction

Students learning programming using traditional languages like C, C++, C#, Java or Python are faced with two considerable problems: they must learn a complicated syntax and develop computational thinking. In addition they are typically instructed to write simple but useless programs which can lead to a lack of motivation. To counter these problems teachers often choose a visual programming language like Scratch or App Inventor which offer clickable items representing programming constructs thus minimizing the syntax learning effort. Some of these items represent high level operations helping students concentrate on writing interesting, useful programs. This maintains their motivation and by eliminating many of the details lessens the complexity of computational thinking needed.

2 Nao, Choregraphe, and Python

But to become employable, students must master a traditional language at some point and previously mentioned problems, along with new ones arise. Students have to learn a completely new language and IDE; they often use a different platform and/or programming paradigm.

Trying to introduce programming with traditional languages and later with App Inventor, teachers at our schools have observed all of the above problems, so we researched other options and decided to introduce Nao, a humanoid robot. We believe this will help our students acquire robotics related skills very likely needed in the near future and at the same time help them master programming in an easier way. Nao can be programmed with Choregraphe, a dedicated visual programming language, which offers all expected basic controls along with high level controls for speech, movement, vision, etc. Controls are represented by boxes and connected with lines (Figure 1). Each box is actually a part of a program written in Python (Figure 2) and can be completely reprogrammed using the same already familiar IDE, the code is object oriented and allows for sequential, parallel and event driven approach. [1, 2]

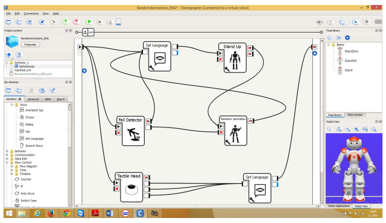
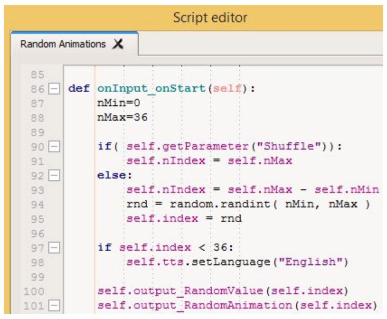


Fig. 1: Choregraphe window



```
def onInput_onStart(self):
    nMin=0
    nMax=36

    if( self.getParameter("Shuffle")):
        self.nIndex = self.nMax
    else:
        self.nIndex = self.nMax - self.nMin
        rnd = random.randint( nMin, nMax )
        self.index = rnd

    if self.index < 36:
        self.tts.setLanguage("English")

    self.output_RandomValue(self.index)
    self.output_RandomAnimation(self.index)
```

Fig. 2: Python code of a box

3 Conclusion

Nao has proven to be excellent for student motivation and for mastering programming with re-duced effort and less problems compared to previous approaches since it offers all the benefits of the visual languages and allows for an easy transition from a visual language (Choregraphe) to a traditional programming language (Python) using the same IDE, platform and programming paradigms. So far this approach has only been applied on a small group of students so it remains to be tested with the whole class.

References

1. Beitter, M., Coltin B., Liemhetcharat S.: An introduction to robotics with Nao. Aldebaran Robotics (2012)
2. Suh, Ki-sung: Using Nao: Introduction to interactive humanoid robots. Aldebaran Robotics (2013)

The influence of teaching methods during technical e-safety instruction

Václav Šimandl[1], Václav Dobiáš[2], and Michal Šerý[3]

[1] Faculty of Education, University of South Bohemia
Budweis, Czech Republic
simandl@pf.jcu.cz

[2] The Institute of Technology and Business
Budweis, Czech Republic
dobias@mail.vstecb.cz

[3] Faculty of Education, University of South Bohemia
Budweis, Czech Republic
kyklop@pf.jcu.cz

Abstract. The article looks at the influence of various teaching methods on the perception of technical e-safety issues as taught in the university curriculum. We have proposed four lesson scenarios for the teaching of this topic. To measure the influence of the lessons, we have used pairs of semantic differential questionnaires, one pre-lesson and one post-lesson.

Keywords: E-safety, teaching method, semantic differential

1 Introduction

E-safety involves protecting the user and his ICT from the risks that occur during the use of ICT [1]. This paper focuses on the narrower topic of so-called technical e-safety, which involves the problems of malware, sharing personal data, identity theft, spam, hoax, phishing and computer crashes. Becta claims there is a clear need to educate children and young people about the e-safety issues and risks [2]. The lack of accessible studies on the influence of various teaching methods in technical e-safety lessons led us to propose, test and analyse the influence of several lesson scenarios for the teaching of this topic at the Institute of Technology and Business in Budweis.

2 Research

2.1 Teaching methods

As the technical e-safety issues are part of the Informatics 1 subject curriculum at the institute, four lesson scenarios were proposed for teaching these issues, with an estimated lesson time of 90 minutes.

- *Group method*: Students work on chosen technical e-safety issues in groups and go on to present their results to the other students.

– *Experience method*: Students are faced with demonstrative e-safety threats which they have to cope with. The threats are discussed and a lecture follows.
– *Inviting an expert*: An expert gives a talk on technical e-safety, with an emphasis on examples from everyday life.
– *Explanatory method*: The teacher presents lesson content in the form of a lecture where the students are passive. The teacher tries to appeal by making the presentation interesting.

2.2 Research method

The semantic differential was chosen for measuring the influence of the lessons. This enables connotative psychological meanings of terms to be measured in individual probands [3]. Consequently, their perception of technical e-safety can be captured via selected terms. The use of a pair of pre-lesson and post-lesson questionnaires of an identical semantic differential can identify changes in the probands' perception of the problem.

The semantic differential questionnaire contained 15 nouns and 12 pairs of bipolar adjectives (4 expressing activity, 4 potency and 4 evaluation). The nouns included 8 key words from the area of technical e-safety (Back-up, Ulož.to (the Czech version of RapidShare.com), Password, Facebook, Privacy, Loss, Virus, Email) and 7 anchor words (Knowledge, Fear, Teacher, Life, Work, Money, Me).

A total of 227 probands (students of the Informatics 1 subject) took part in the chosen type of lesson, pre-lesson and post-lesson semantic differential questionnaire investigations.

3 Conclusion

The acquired questionnaires are currently being analysed. The shift in perception of key words between the first and second questionnaire investigation is examined. Results of the research will be used to compare individual teaching methods and to subsequently optimise technical e-safety lessons and achieve improvement of knowledge in the most effective way.

The first outcomes of the research will be presented on our poster.

References

1. Barrow, Ch. and Heywood-Everett, G. E-safety: the experience in English educational establishments [online]. Becta, 2006 [cit. 20120715]. Available from http://dera.ioe.ac.uk/1619/1/becta_2005_esafetyaudit_report.pdf
2. Becta. Safeguarding children in a digital world: Developing a strategic approach to e-safety [online]. Coventry: British Educational Communications and Technology Agency, 2006 [cit. 20120914]. Available from http://www.wisekids.org.uk/BECTA%20Publications/safeguarding_digital_world.pdf
3. Smékalová, L., Homolová, K. Methodological argumentation of M. M. Bergman in the context of using Q-methodology and semantic differential in the mixed-methods research. Paidagogos [online]. 2013, 2013(2), p. 268–279 [cit. 20150621]. Available from http://www.paidagogos.net/issues/2013/2/article.php?id=17

Teaching Programming Indirectly with “Paint”

Zsuzsanna Szalayné Tahy

Eötvös Loránd University, Faculty of Informatics
Budapest, Hungary
sztzs@caesar.elte.hu

Abstract. In many cases IT literacy is inadequate: users do not understand the concepts of software, and consequently using applications creates more problems. Professionals suggest learning programming to improve computational thinking [1]; but this way is impractical for many. There is another efficient method to teach computational thinking and prepare for programming. By using an application such as Paint software the teaching programming can be embedded into the teaching of application usage. So the way to the high level computational thinking and programming comes through exploring the used application.

Keywords: application, programming, basics, teaching method

1 Introduction

The main task is the improvement in digital literacy for members of the digital society. The most important stage is to understand how IT tools work. As they are based on written programs, the starting point of improvement is to teach programming. This is the idea behind several national curriculums [2], where programming is set as key competence. There are students, who had learnt programming using Scratch in elementary school. Many of them learn program writing later but felt the gap: “I am not able to learn programming”. As they cannot transfer the visual solution to code, they cannot transfer the learnt algorithm to everyday practice. The suggested approach is indirect teaching: students explore the well-known software, then understand the idea behind the tools. In fact, the teachers are teaching programming skills but students do not realise this. This method will be demonstrated through the example of the Paint program.

2 Embedded Programming – Using Paint

Paint is an example of one layer handled pixel-graphics software, saving the product into bitmap (bmp) format. In the following the concept of ‘take apart, look inside, explore who, what, how, why. . . ’ is demonstrated. Students like to paint but in this case they start by exploring the software. Maybe the first question is “Where are the colour tools?” but the next question is about the model of colours and the data structures: “How many different colours are defined in the software?” “How are the codes for colours stored in the memory?”

Through teaching Paint we can teach several concepts of programming such as *set of objects*, *property* and *method*. Students should imagine an instance of array abstraction, so they can understand and practice indexing. Exploring colour schemes creates several questions for students about number systems, data representation, picture size, file size, etc. Practical solutions involve the concept of lossy and lossless compression. By working with painting examples students are able to explore methods and algorithms: students should guess “what does the little man in the computer do” when they resize the image. Depending on the level of teaching and students’ precognition, we can model the method or we can write pseudo code. The main points are:

– data structures, models and objects are explored;
– students understand the relation between view and binary sign;
– students state algorithms as hypotheses and try to prove them with tests.

The next stage, after exploring paint and other applications could be “Try to hack an image”. This task involves modifying files by writing code. The best practice for teachers is to give a frame code which reads and writes files. The frame program includes the implementation of data structure, the class of *Pixel* with properties like Red, Green and Blue components and the array of pixels. Although the goal is to modify an image, the first step is to explore the code to find relations between codes and learnt concepts. The next step is to learn how to debug and run codes. The last step is the start of explicit learning of programming putting these skills and knowledge all together: modifying an image by coding [3].

3 Summary

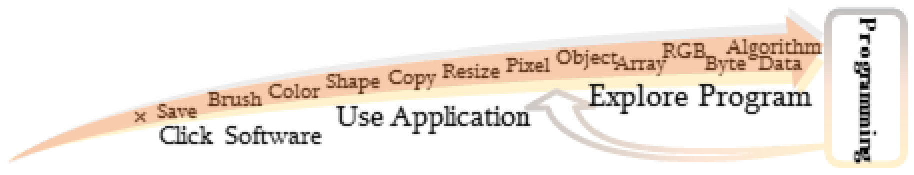


Fig. 1: . . . look inside, explore who, what, how, why. . .

References

1. J.M. Wing: Computational Thinking and CS@CMU Carnegie Mellon University, 2006.
2. Department for Education, GOV.UK: National curriculum [Online] https://www.gov.uk/government/collections/national-curriculum, 16. July 2014.
3. David J. Malan: BMP Puzzles Harvard University, [Online] http://nifty.stanford.edu/2011/malan-bmp-puzzles/, 2011.

ISSEP 2015 — Workshops

Workshops are distinguished meeting points for getting informed about the ongoing work in our relevant fields; current topics preferably in practical informatics at all school levels.

The ISSEP series started in 2005 in Klagenfurt/Austria. Five ”tutorials” as a synonym for “workshops” have been offered then, covering rather soft Informatics topics as Security and Dependability in E-Learning, a presentation of Moodle, Didactic Aspects of e-Learning Contents Development, Fundamentals of Human-Computer Interaction and last, Insights into the functionality of a model search engine.

Ten years later, within the call for the 8th ISSEP conference following rather genuine Informatics topics have been proposed:

- Maker movement (e.g. robotics)
- CS unplugged activities and informatics contests/challenges
- Development environments and programming interfaces
- Web applications, web collaboration and production tools (e.g. moocs and e-books)
- Good practice and worked out examples
- Short and long term lesson plans, reference models and special curricular issues

Finally, the call yielded the three proposals “Teaching Software Engineering in Primary and Secondary Schools” from the Informatics Didactics team at Klagenfurt University in the neighbourhood Carinthia, “A web service for teaching programming” from an Slovene team and third, “Learning Computational Thinking through Bebras Tasks” as an Lithuanian/Austrian co-production.

23rd September, 2015

Peter Micheuz, Workshops' Chair
University of Klagenfurt

Teaching Software Engineering in Primary and Secondary Schools

Peter Antonitsch, Andreas Bollin, Stefan Pasterk, and Barbara Sabitzer

Institute for Informatics and Informatics Didactics, University of Klagenfurt, Austria
Software Engineering Research Group, University of Klagenfurt, Austria

Software is everywhere — be it in mobile phones, in washing machines, or in cars. With it, the importance of software Engineering (SE) is uncontested, and it is taught all over the world: at Universities, at Colleges, and recently also at High Schools. There are international Software Engineering curricula, standards, and certificates, but there is no manifestation of SE (and related practices) in the course syllabi at primary and secondary schools. Most important, SE is not just programming.

Taking a closer look at SE, its main goal is to develop programs that are affordable and dependable for consumers without bugs or glitches. In order to do so, SE education must account for a broad spectrum of knowledge and skills that software engineers will be required to apply throughout their professional life. Covering all the topics in depth within a school setting (from primary to secondary schools) seems to be infeasible due to the previous knowledge of the pupils, the curricular constraints as well as due to the inherent differences between the school types. Similar arguments hold for the teachers, as most of them are not really trained in SE. Now, based on the authors' experiences gained in combining SE topics with school projects in a vocational high school for commerce and tourism (11th grade) in cooperation with a lower secondary school (6th grade) it turned out that, by customization of the approach, one is able to address pupils with different maturity levels, educational aims, and backgrounds.

The objective of this 90-minutes workshop is to show that it is possible to interweave SE topics with school projects and to motivate for the most important practices related to that field. Key skills and challenges are identified, mapped to the situation at hand, and, by following a stepwise approach, example settings are discussed.

Learning Computational Thinking through Bebras Tasks

Valentina Dagiene[1] and Gerald Futschek[2]

[1]Vilnius University, Lithuania
[2]Vienna University of Technology, Austria

This workshop addresses all educationists and education scientists who are interested how school students can learn informatics (computer science) concepts and Computational Thinking through a contest. The International Bebras Contest on Informatics is the world's largest contest on Computational Thinking. In the 2014 contest more than 900,000 students participated in 36 countries of all continents. The students have to solve 15 to 21 tasks within 40 to 60 minutes. To solve these tasks, students do not need specific pre-knowledge. Tasks are developed for different age groups, from primary school to upper secondary school students. The tasks contain concepts of about nearly all areas of informatics. Usually a short story introduces a task and states a problem, termini technici are not used, but to solve the task some kind of computational thinking must be applied. There are tasks about concept categories of information representation, algorithms, programming, logic, encryption and many other.

Items discussed in the workshop:

– Operational definition of computational thinking
– Why Bebras tasks can convey computational thinking?
– Which concepts of informatics can be introduced through Bebras tasks?
– How to teach computational thinking using Bebras tasks?
– Relations of Bebras contests to informatics curricula in various countries
– Formal and informal introducing informatics concepts

In the workshop the participants will learn more about the Bebras contest, how the tasks are created, which kind of tasks were produced, what are the effects on learning and teaching. We will discuss how the Bebras contest should be performed in a school context and how the teachers may use the Bebras tasks in their teaching activities. The participants will experience wow-effects while solving Bebras tasks and how thinking is directed to solving strategies that are typical for informatics and computational thinking.

A web service for teaching programming

Gregor Jerše, Sonja Jerše, Matija Lokar, and Matija Pretnar

University of Ljubljana, Faculty of Mathematics and Physics, Slovenia

Programming is a skill that can be best learned by writing as many programs as one can. So teachers are required to expose the students to numerous problems and of course supervise their attempts to solve them. To support this teaching approach, the authors developed a web service Projekt Tomo, which we aim to present at the workshop. This service has already been successfully used in various courses ranging from secondary schools to introductory courses in higher education. The service works as follows: the student first downloads the files containing problems to his computer and starts filling in the solutions, checking them locally in his favourite coding environment, while the server automatically stores and verifies the solutions. In this way, there is no need for powerful servers and the service provides instantaneous feedback to the student and an overall insight into the obtained knowledge to both the student and the teacher, all without disturbing the teaching process. This helps teacher save time which he can spend for in-depth discussion about programming and giving additional help to those who need it. An important aspect is also the fact that existing programming environment can be used by the student. The teachers can also view a student's history of attempts and download the files with the attempted solutions if they want to analyse the student's progress and provide appropriate advice. These submissions can serve as a valuable insight into efforts made by the students towards the solutions. The service can be adapted to almost all teaching environments, as it can be used with all programming languages and has low technical requirements.

Agenda:

– Introduction to the web service - understanding the motivation behind the web service, logging into the service.
– Solving problems (as a student) - downloading a problem file, submitting a correct and an incorrect solution, understanding feedback.
– Analyzing submitted solutions (as a teacher) - getting an overview of correctness of submitted solutions, looking at individual feedbacks, exploring the history of student submissions.
– Creating and editing problems (as a teacher) - modifying an existing problem, adding automated tests, creating a new problem.
– Discussion - getting feedback to steer future development.

ISSEP 2015 — The International Teacher's Conference

Up to now, for Italian teachers of Informatics it has not been customary to organize regular meetings in order to address educational, curricular and pedagogical issues of their discipline. Also the most self-motivated among them had few occasions for discussion and for sharing their experience with colleagues facing similar problems elsewhere. Thus, the idea of having international teacher sessions within the ISSEP conference is very welcome, in that it offers a valuable opportunity of professional enrichment. The contributions from Italy (5), Hungary (1) and Austria (2) encompass all levels of school education and present interesting approaches to the teaching of computing topics.

Primary and lower-secondary teachers have taken a trans-disciplinary "computational thinking" perspective and view the learning within the field as a peculiar component of scientific education. More specifically, a main concern in Ferrari, Rabbone and Ruggiero's paper is interplay between unplugged ativities and coding in order to design a sustainabile curriculum for the elementary school. A balanced mix of unplugged tasks and work with computers is also central to the experiences described by Palazzolo, who in addition points out the need of engaging a larger number of middle-school teachers in similar projects. Moreover, Erdősné Németh addresses a classical topic in computing education: how to teach recursion to young pupils. Her proposal revisits the traditional approach of exploring graphical recursive structures in Logo.

High school teachers, on the other hand, seem to focus on "active learning" with some significant technological support. Boscaini and Valente discuss the educational implications of projects aimed at participating in robot contests, in particular as to the tradeoff between theoretical knowledge and practical skills learned by students. Brocato reports on her experience of teaching database fundamentals following a flipped-classroom approach with the aid of a learning management system. Finally, Danesino describes an introductory unit where the students are encouraged to analyze and explain network-related concepts by producing learning materials themselves. Her students use specific applications that allow them to apply augmented-reality techniques.

The two Austrian contributions provide a cursory and deep insight into all levels of Informatics education.

Peter Antonitsch takes a "A Cautious Look at Coding in Primary Education" where he reports on an action research project in a primary school. He eleborates on two antagonistic viewpoints, one propagating that programming at this early stage fosters the intellectual developement of pupils, and the other pointig at developmental risks when children are exposed too early to virtual envirements.

In their contribution "Selected Spotlights on Informatics Education in Austrian Schools" Peter Micheuz and Barbara Sabitzer take a look at current developments going on in Austrian general Informatics education. They provide an overview with some insights about initiatives at primary education, insights into competence models and their impact on Informatics at secondary level, including curricula issues in the grade 9. Finally they present first results of a major reform of the final school leaving exam in Informatics (Matura).

16th September, 2015

Peter Micheuz, Chair of the Program Committee
University of Klagenfurt

Barbara Demo, Chair of the Program Committee
University of Turin

Claudio Mirolo, Chair of the Program Committee
University of Udine

A Cautious Look at Coding in Primary Education

Peter K. Antonitsch

Alpen-Adria Universität Klagenfurt
Institut für Informatikdidaktik
Peter.Antonitsch@aau.at

Abstract. New programming environments try to attract younger and younger children to computers in general and coding and programming in particular, claiming to foster their intellectual development. Others warn by pointing at developmental risks if children are exposed to virtual environments at too young an age. This article reports on a small scaled research project at primary level, indicating that both viewpoints might have their justification, but also points at coding being some special case.

Keywords: Primary education, Informatics, Coding, Programming

1 Introduction

Informatics education has always been triggered by current technical innovations. Right now, with the availability of comparably inexpensive digital devices like smartphones or tablets we witness efforts to extend teaching and learning of basic Informatics down to primary education, where, in the meantime, "basic Informatics" covers a wide spectrum ranging from skills to operate application software of various kinds to the abstract field of programming (in terms of writing pieces of code). Due to the development of new curricula ([1], [2]), the latter attracted particular attention, resulting in the development of software-enviroments that facilitate young children's first steps into coding, like the software available at code.org [3], or ScratchJr [4].

Some hail this development stating that "[ScratchJr is] almost purely graphic-driven, which makes it accessible to an age group for whom reading is sometimes still a lot of work." [5]. Others, not less enthusiastic, praise coding as "not only a way to learn about computer science but how to think and tackle problems through computational thinking skills and abstracting out details that are meaningful." [6], suggesting this also to be true for children at primary level or even before.
But there is another, yet different point of view, seeing possible drawbacks in children's development when they are exposed to computers too early. This point of view not only can be found within the educational thought of anthroposophical philosophy, [7], but is also an issue of popular scientific publications. In [8], for instance, the basis of argumentation is Piaget's theory of cognitive development, concluding that meaningful use of digital devices needs the capability for abstraction and should not take place before the formal operational stage, beginning around the age of twelve. In-

stead, early childhood should provide lots of experience within the physical world, supporting the children's development of spatial perception, or their ability to concentrate on a certain task for a longer period of time. This is believed to provide a sound basis for abstract thinking and, later on, for reflected use of digital devices and software in general, and for successful programming in particular.

The two positions sketched above appear to be rather incompatible. Nevertheless, results of a small-scaled research project to introduce principles of computational thinking into traditional education of a third grade of an Austrian primary school indicate that even programming can be taught the age between nine and twelve. However, neither the abandonment nor the execcive use of digitial devices seems to be favourable, but an intermediate course of action where computers are used as tools to support learning but play only a minor role.

2 Research

2.1 Motivation

The research project was partly motivated by observations during Informatics courses at higher secondary level (learners at the age of 15 or 16) during the 2012-13 school year. At this level, learners are introduced into formal reasoning, for instance, by expressing a solution to a posed problem by means of a formalized programming language. But while being at ease with operating digital devices or being able to work with application software quite well, some of these so called "Digital Natives"

- had poor reading comprehension,
- had problems to produce meaningful passages of text by themselves,
- had little experience with structuring aids like tables or mind maps to chunk information into pieces,
- were unable to learn by following written step-by-step instructions.
- and lacked basic problem solving strategies like dissecting a big problem into smaller solvable parts.

Obviously, these learners missed some basic skills that are necessary to model programmable solutions to a problem and to code these solutions, with the acquisition of these skills being supposed to start back in primary education. Nevertheless, as Informatics is not taught in Austrian primary schools, the concept of computational thinking had to serve as an additional motivation to look for traces of Informatics content in primary education. According to [9], [10], computational thinking is a problem-solving process that includes (selection):

- formulating problems in a way that enables us to use a computer and other tools to help solve them,
- logically organizing and analyzing data,
- (formulating and) automating solutions through a series of ordered steps,

and should be present at all levels of education, thus also at primary level.

Finally, the findings of E. Stern that children can develop the ability for abstraction and for operating within (simple) formal systems before they reach the formal operational stage [11], motivated to broaden the horizon and to look for ways how to augment primary education with aspects of coding and automation.

2.2 Research Questions

Regarding basic literacy and numeracy to be the major educational goals of primary education (in Austria), the following research questions were posed:

- How can algorithmic thinking (as a distinct aspect of computational thinking) at primary level be linked to these major educational goals?
- How can traditional educational practice at primary level be augmented by unplugged exercises that illustrate basic ideas of Informatics in general and the practice of programming, in particular?
- Can traditional educational practice at primary level be enhanced by introducing computers to allow for automation of written code?

2.3 Context and Course of Research

The research took place in a third grade of an Austrian primary school (14 girls, 8 boys, 8 to 9 years old) in the 2013-14 school year. This age group was chosen, because the learners were supposed to have basic reading- and writing-skills before getting in touch with Informatics mindsets. The concept of algorithmization should not be introduced as an add-on to traditional learning content but should smoothly be integrated into the traditional mode of primary education by being linked to basic literacy, i.e. reading and writing. Hence, and according to the research questions, there were three different preparing phases of the project with computers being introduced not before the last of these phases, and a subsequent fourth phase dedicated to work on tasks using appropriate software. These phases spanned the whole school year, considering "doing Informatics" – either due to preparatory exercises or in terms of computer-based working with software - one lesson per two weeks, in average.

The software of choice was Blinkenpaint, a simple program providing a tabular grid to create sequences of pictures that can be viewed in a flip-book manner, and Scratch, a software-environment for novice programmers. Programs in Scratch are composed by combining ready-made programming blocks that prevent from running into syntax errors and are used to animate two-dimensional sprites, which gives instant feedback about the correctness of the written code.

The first phase could be titled "understanding and creating step-by-step instructions and started at the end of the previous school year as part of the project preparation. First of all, existing educational material appropriate for preparing algorithmic thinking had to be identified. This material included step-by-step instructions for simple science experiments or tinker instructions for manufacturing a wind rotor or a flip book. All of these were used to train how to read and follow stepwise instructions, and were augmented with tasks from Computer Science unplugged [12], Informatik ErLeben [13] or the Bebras Contest [14]. These tasks also included a first introduction into structuring tools like tables or trees. Furthermore, the learners had to create storyboards to prepare for creating stepwise instructions by themselves (see [15] for a more detailed overview of the material put to work).

The second phase was dedicated to formalizing step-by-step instructions and was partly inspired by the preparation of conventional flip books during the first phase. To prepare for the use of Blinkenpaint the learners had to draw a second series of flip books provided by sheets with an 8 rows – 18 columns matrix, thus having to change their mode of drawing from "continuous" to "pixel wise". Furthermore, building upon their experience on writing text-based instructions the learners had to describe sequences of simple movements by means of enlarged, printed and cut Scratch-blocks. These programs where tested during a role play where learners had to translate each other's sequence of instructions into stepwise movement, thus providing both, a coding environment and a feedback mode very similar to the software intended to use during the subsequent phase.

To prepare the use of the software the learners from primary school paid three visits to a higher secondary school providing a sufficient number of computers so that all children could be instructed in parallel. At primary school, four laptops were available to practice and to work on tasks posed during the third and the fourth phase.

2.4 Selected Findings

To assess the intended learning progress of the primary school children, a mixed-method approach was chosen, which combined qualitative observations, mainly of the corresponding primary school teacher, process portfolios written by the learners to document their activities, the outcome of an intermediate exercise based on sequencing cards, evaluation of two written assessments focusing on algorithmization and qualitative analysis of Scratch-projects created during the final phase of the project, the latter being done by the author. Relevant results can be summarized as follows:

- The sequencing cards exercise displayed out of order pictures "telling" the story about the evolution of a chicken. The task was to "retell" the story giving the correct steps, either by cutting the pictures and rearranging them, by listing the correct order of pictures within a table or by writing a short story. While all of the learners mastered this exercise, remarkably none of them chose the swiftest possible solution based on tables.
- The first written assessment contained a selection of stepwise instructions the learners had to understand. Most of the instruction sequences used rather long textual descriptions to examine the reading comprehension of the children, only one used a semi-formal representation utilizing Scratch blocks. Scratch blocks were unknown to the learners until that moment but were chosen intentionally as they

should prepare for the use of the software later on and are generally considered very intuitive.

The children had no problems to follow written instructions, except for confusing the left and right when making turns, but did hard to write an ordered sequence of instructions by themselves, even when they had the task to rewrite a given “algorithm” by filling the lines of a table. Furthermore, only one could figure out the meaning of the algorithm coded with Scratch blocks.

- When working with the software Scratch the learners became acquainted with Scratch blocks rather soon. Nevertheless, most of them preferred to invest a lot of time into drawing elaborate backgrounds and/or characters so that little time was left for coding. Most of the learners kept this habit during the last phase of the project, where they were expected to invent and animate stories by themselves. Hence some extra programming exercises (also including loops as a means to control program flow) were provided. These exercises were included as an optional track within the art/technical crafts lesson at primary school.
- The final evaluation focused again on the ability to understand and to formulate simple (movement-) algorithms, and on trees as a structuring tool specific to Informatics, especially with file organisation. This topic turned important when it became evident that storing at or saving from a specified directory on the working USB-stick posed serious difficulties for some of the children. Unsurprisingly, the results of the corresponding tasks within the final examination were rather poor. Regarding algorithmic thinking, the learners were able to deal with sequences of instructions but did not use loops as a shortcut to describe repetition. Furthermore, many learners still confused left and right.

3 Discussion and Outlook

With regard to the three research questions, the project was only partially successful. On one hand, even traditional primary education offers many possibilities and ready-to-use learning material to link algorithmization and formalization in the sense of understanding and writing step-by-step instructions to the common course of learning and teaching. Hence, at least to some extent, formal reasoning can be grounded at primary level. On the other hand, learning how to program by means of computer software proves harder than expected. One possible reason might be the scattering of Informatics content and corresponding phases for practice across the weeks, with enough time to forget in between. In this case, a period of condensed Informatics instruction using the computer might yield better results.

But the results sketched above indicate that the reason for the learners’ problems with implementing algorithmic thoughts might be rooted deeper, as prior knowledge and skills seem to be essential for success in programming:

- Informatics uses a variety of structuring tools to represent information. Some of these tools young learners simply do not know yet, others they know but do not use frequently. Generally speaking, at primary level the ability to structure rather abstract information is under progress, if not just starting. This view is encouraged by the learner's problems when working with tables or tree-structures, by the simple structure of code written by children of that age (see [16] for further examples of that kind) as well as in most of the children's inability to transfer basic concepts from one piece of code to another. All of this seems to be in accordance with Piaget's theory of cognitive development.
- What has been trained before using the computer seems to determine what learners prefer to do with software at hand. Hence, the learner's concentration tends to shift away from what they are expected to do. This can be derived from most learners preference for drawing instead of coding and can be considered a general habit of human behaviour: We like to do what we know how to do: While the children had been drawing pictures (by hand) for at least two or three years, "creating" a sequence of instructions was rather new. Remarkably, those who were very focused during the "unplugged" tasks dealing with stepwise instructions, performed better when writing and automating programs by means of Scratch-scripts.
- Besides foregoing mental experiences, bodily real-world experiences seem to be as important. With the example of equilibrium, in [8] the authors point at the importance of bodily experiences to develop abstract thinking. The distinction between left and right seems to be similar to "exploring equilibrium", suggesting that programming needs certain body knowledge as well.

These interpretations allow to connect with the introductory thoughts: It makes sense, to diminish the learner's problems with structuring of data within computer systems, or with concepts of computer software by letting them do meaningful work with computer software at an early age, say, at primary level. But specialized tasks like programming/coding that demand a certain capability of abstraction seem better to be postponed to prevent learners from being overstrained by the tasks they have to fulfil, and to give them time to develop properly.

Of course, the results presented above do not settle the question whether a "modern", digital-devices based approach to learning at young ages can modify the learner's cognitive development or rather hinder it, for instance, by creating habits that are at odds with the desired outcome. Meerbaum-Salant et al. have pointed at this danger in [17] looking at a different age group, though. But the results indicate the value of intensified "unplugged" experiences in the field of Informatics before using the computer as the major tool in Informatics education.
Hence, beginning with the 2015–16 school year, a corresponding long term research project will be launched by the Informatics didactics group at the University of Klagenfurt, designed to evaluate the development of children during four years at primary school, where their learning will be enhanced by "mostly unplugged" Informatics content.

References

1. CSTA Standards Task Force: K–12 Computer Science Standards, revised 2011. http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf
2. National Curriculum in England: Computing Programmes of Study, published 2013. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study
3. Code Studio Website. https://studio.code.org/
4. ScratchJr Website. http://www.scratchjr.org/
5. Wohlsen M.: Finally, a Way to Teach Coding to the Touchscreen Generation. WIRED business, 2014. http://www.wired.com/2014/07/finally-a-way-to-teach-coding-to-the-touchscreen-generation/
6. Shine E.: Bringing Coding to Kindergarden, ACM news, 2015. http://cacm.acm.org /news/183337-bringing-coding-to-kindergarten/fulltext/
7. Setzer V.W., Lowell M.: An Alternative View on Why, When and How Computers Should Be Used in Education. In: Muffoletto, R. (ed.): Education and Technology: Critical and Reflective Practices, Hampton Press, 2001. http://www.ime.usp.br/~vwsetzer/comp-in-educ.html
8. Lembke G., Leipner I.: Die Lüge der digitalen Bildung. Redline Verlag, München, 2015 (in German)
9. Wing J.M.: Computational Thinking. In: Communications of the ACM, March 2006/Vol 49 No. 3. http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf
10. Barr V., Stephenson C.: Bringing Computational Thinking to K–12: What Is Involved and what Is the Role of the Computer Science Education? ACM Inroads 2011 March Vol. 2 No. 1 http://csta.acm.org/Curriculum/sub/CurrFiles/BarrStephensonInroadsArticle.pdf
11. Stern, E.: Wie abstrakt lernt das Grundschulkind? In: Petillon, H. (ed.): Individuelles und soziales Lernen in der Grundschule - Kindperspektive und pädagogische Konzeple. Leske + Budrich, Opladen; 2002 (in German)
12. Computer Science Unplugged Website. http://csunplugged.org/
13. Informatik ErLeben Website. http://informatik-erleben.uni-klu.ac.at/
14. Austrian Bebras Contest Website. http://www.ocg.at/de/biber-der-informatik
15. Antonitsch P., Hanisch L.: Aspekte von Computational Thinking im Unterricht der Primarstufe, project report, June 2014 (in German). https://www.imst.ac.at/imst-wiki/ index.php/Aspekte_von_Computational_ Thinking_im_Unterricht_der_Primarstufe
16. Northern Ireland Curriculum 2012, Using ICT Case Studies, http://www.nicurriculum.org.uk/key_stages_1_and_2/skills_and_capabilities/uict/UICT_in_practice/case_studies/scratch.asp#top
17. Meerbaum-Salant O., Armoni M., Ben-Ari M.: Habits of Programming in Scratch, Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, ACM, New York 2011

The Cat, the Turtle, the Snake and GCD

Agnieszka Borowiecka and Katarzyna Olędzka

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

Abstract. In this paper, we discuss our proposition for the lessons of algorithmic in which we work with interactive projects. By playing and programming small applications students have the opportunity to understand an algorithm and become familiar with different ideas. Everyone learns effectively when acts (learning-by-doing), but it is even better when one is programming (learning-by-programming). Different approaches to the greatest common divisor problem (GCD) are presented from pedagogical perspective.

Keywords:learning algorithmic, computational thinking, greatest common division, Euclidean algorithm

1 Introduction

Familiarizing students with the world of programming is an interesting and challenging task. By recognizing the need for continuous improvement we are asking some questions: How to encourage students to learn? What kind of tasks are interesting and suitable for them? How to help them understand difficult ideas in a friendly way? In spite of the many steps that we have taken to encourage students for learning programming, we have prepared several application which can help in developing computational thinking skills. By making different projects students broaden their knowledge and grown in their experience. In this article we will present some ideas focus on greatest common divisor problem.

The Euclidean algorithm for finding the greatest common divisor (GCD) combines ancient and modern times. It is said that this algorithm is one of the oldest algorithms in common use. It appears in Euclid's Elements in 3rd century BC. Moreover, it can be an object of interest for primary school children and mature programmers as well. In programming lessons it can be presented in such a way that students with different learning approaches could have fun – for those who prefer to understand logical formulas, and also for those who would like to see it or to point a finger on a screen. This algorithm is quite short in implementation so it is relatively easy to code. It allows us to find the searched number in a fast way. Everyone learns effectively when acts (learning-by-doing), but it is even better when one is programming (learning by programming). Teachers should combine creative approach to the problem and critical thinking. We will present different concepts how to work with students around the problem of finding the greatest common divisor. The implementations are prepared in various programming environments. They are available for students at different educational stages. We will start from visual programming, next, there will be presented an example from the Logo language, and finally – the high-level language. The last one is abstract – to be solved with ink and papers without a computer. Our ideas are organized into four lessons.

2 A cat walks on a board and shows Euclidean algorithm

On the first lesson we will invite a cat to present an algorithm. Students are observing what is going on the screen by running a previously prepared application. Then they try to write the rules that guide movement of a ball in a natural language. Next students code that algorithm using blocks in Scratch, or other environment (like Blockly).

First, a user should input two numbers so a ball starts its journey from a specific point. Next the ball is moving left or down according to distance to the bottom and to the left border of a grid. If it is closer to the bottom edge it moves left, otherwise – down. In every step the longer distance is shortened. In a shown example, two given numbers are: `25` and `15`, then the ball moves `15` to the left, `10` to the down, and finally `5` to the left. Now both numbers are equal.

This is a graphical representation of the Euclidean algorithm. First, the user specifies two numbers `a` and `b`. In each step, values `a` and `b` are presented, until the algorithm gets to the point where these two values are equal.

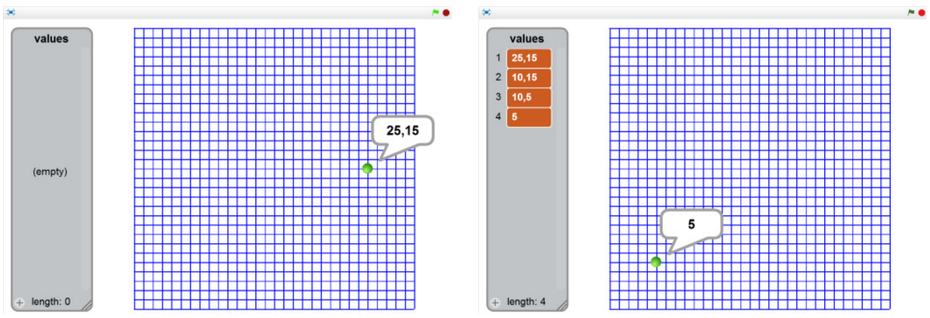
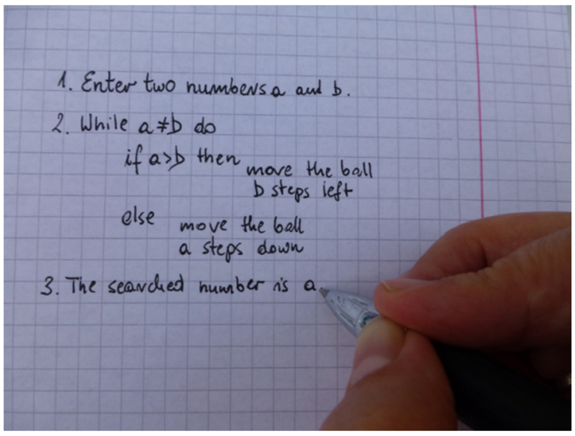


Fig. 1. The beginning and the end of the visualisation

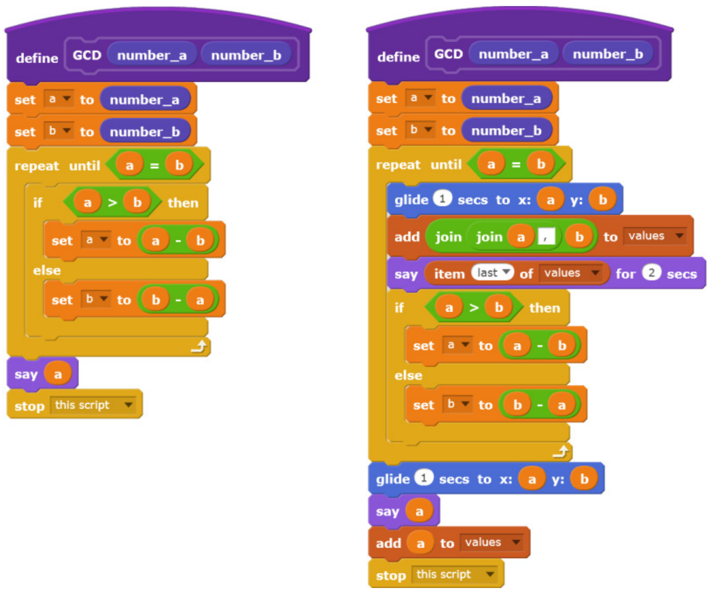
After analysing this application, a teacher can ask students some questions such as: What is the input for the program? When the ball moves left and when down? How many steps it moves? When the algorithms ends?



```
1. Enter two numbers a and b.
2. While a≠b do
      if a>b then  move the ball
                   b steps left
      else   move the ball
             a steps down
3. The searched number is a
```

Fig. 2. The algorithm written in a natural language

After that, the teacher should help students to code with blocks. The more specific the language of writing, the shorter the program, and the easier to code.



```
define GCD (number_a) (number_b)
set a to (number_a)
set b to (number_b)
repeat until <(a) = (b)>
    if <(a) > (b)> then
        set a to ((a) - (b))
    else
        set b to ((b) - (a))
say (a)
stop [this script]
```

```
define GCD (number_a) (number_b)
set a to (number_a)
set b to (number_b)
repeat until <(a) = (b)>
    glide (1) secs to x: (a) y: (b)
    add (join (join (a) [,]) (b)) to [values]
    say (item (last) of [values]) for (2) secs
    if <(a) > (b)> then
        set a to ((a) - (b))
    else
        set b to ((b) - (a))
glide (1) secs to x: (a) y: (b)
say (a)
add (a) to [values]
stop [this script]
```

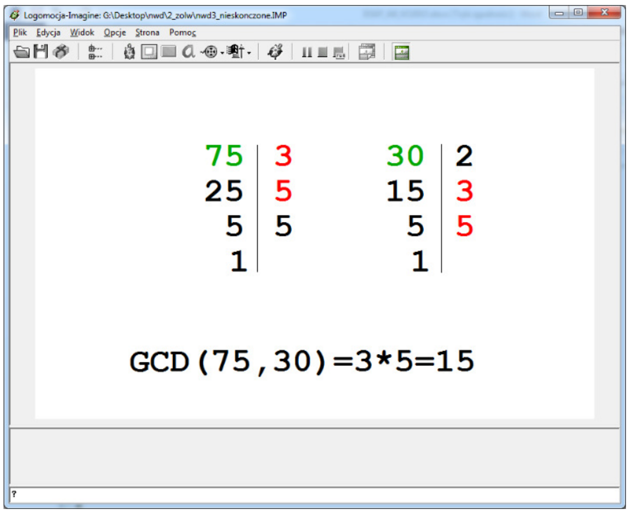
Fig. 3. Blocks for the basic algorithm and the algorithm with visualisation

Moreover, students have to prepare a grid. It can be done in a graphics editor, but more ambitious student can prepare it in Scratch by defining code for a sprite.

A visual language is chosen for this task because it is easier for students to use blocks instead of writing a code. There is a huge gap between programming in a visual language like Scratch and programming by typing commands. Beginners prefer manipulating program elements graphically rather than specified them textually. On one hand, using blocks student does not need to concentrate on the syntax but on ideas, on the other hand, when the problem is more complex code is usually long and unreadable. The advantage of learning such visual languages is that you can create many interesting applications, even you are not advanced programmer. In this task the algorithm and our visualization is simple – by adding only few commands for the ball we got the whole application.

3 School algorithm with a turtle in background

In the second lesson the main role will play a turtle. We stimulate students to improve their experience in computational thinking. We will start from presenting a well know algorithm and ask students to solve some problems based on this knowledge. In this lesson some task are more difficult like decomposing a number into primes, other are simpler, so the teacher can differ students' work.
Primary school children learn how to find the greatest common divisor of `a` and `b`. For this purpose, they list all the `a` and `b` divisors that are repeated in both lists. The product of items that are in the both lists is GCD.


Logomocja-Imagine: G:\Desktop\nwd\2_zolw\nwd3_nieskonczone.IMP
Plik Edycja Widok Opcje Strona Pomoc

```
75 | 3        30 | 2
25 | 5        15 | 3
 5 | 5         5 | 5
 1 |           1 |

GCD(75,30)=3*5=15
```

Fig. 4. An application in Imagine (Logo)

The implementation requires to code three things: first one – finding a list of all divisors, second – find an intersection of these lists and third – calculating the product of numbers that are repeated in the both list. For finding divisors of `a`, we divide given number by 2, 3, 4, … up to the square root of `a`. If the modulo of division is 0 we remembers this number and we try to do it another time, otherwise we increase value by which we divide. All remembered values build the searched list.

```
to gcd :a :b
 let "ra decom :a
 let "rb decom :b
 let "com common :ra :rb
 output apply "product :com
end

to decom :n
 let "temp []
 let "i 2
 while [:i*:i<=:n] [
  ifElse (mod :n :i)=0 [
   let "temp lput :i :temp
   let "n div :n :i ]
  [ let "i :i + 1 ]]
 output lput :n :temp
end

to common :a :b
 let "temp []
 let "nb_a 1
 let "nb_b 1
 let "a lput 1000000 :a
 let "b lput 1000000 :b
 while [:nb_a<=count :a][
   ifElse (item :nb_a :a)=(item :nb_b :b)[
     let "temp lput (item :nb_a :a) :temp
     inc "nb_a
     inc "nb_b]
     [ifElse (item :nb_a :a)<(item :nb_b :b)
        [inc "nb_a]
        [inc "nb_b]]]
 output butLast :temp
end
```

This algorithm is introduced in mathematics lessons, because it is useful for operations with ordinary fractions. However, this requires `a` and `b` to be factored, and there is no effective algorithm for this problem.
Reviving the newest Polish e-textbook for Informatics we can find and interesting task. It is worth to mention that the whole lesson is for teenagers who are familiar with this topic from mathematics lessons but they are programming beginners. The task is to prepare an application which is drawing many squares – some of them are coloured and some not. A user should point which part of the figure is filled with the colour. While coding this task student should not only concentrate on visualisation

and user-computer communication, but also an appliance of some algorithms. The most difficult part of this task is to simplify fractions.

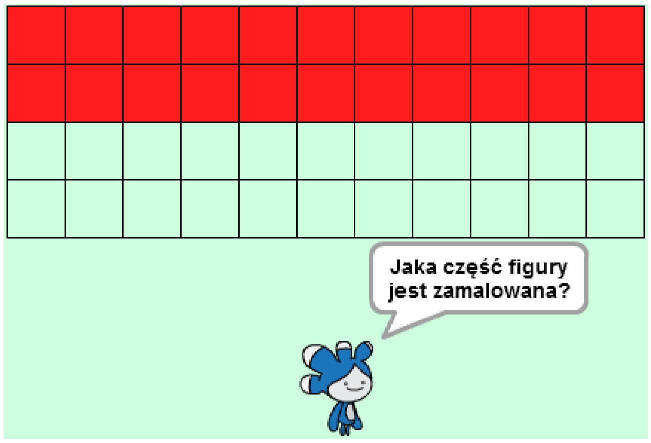
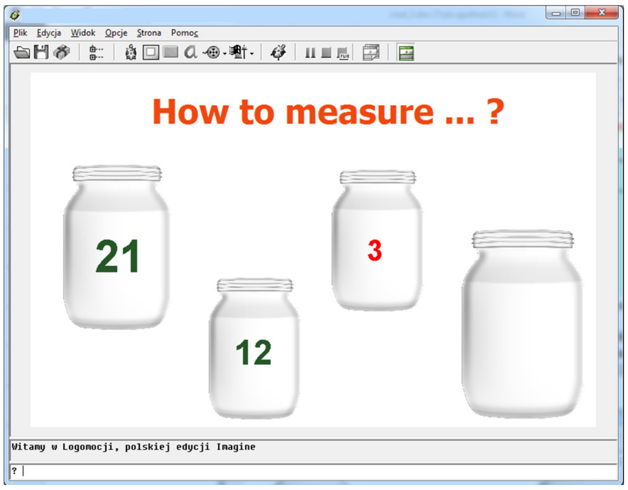


Fig. 5. From e-textbook for gymnasium students in Informatics

Another problem is also strictly connected with GCD. Suppose that you have two jar glasses with a capacity of a (in example 21 Litres) and b (12 Liters) and one more of unlimited capacity. How to measure exactly c (3 Liters) using these jars?



```
Witamy w Logomocji, polskiej edycji Imagine
? |
```

Fig. 6. Application in Imagine environment (Logo)

Students can solve this problem by trial and error method. They will gain some experience and direct their intuition. They can also try to write an algorithm for this task. A detailed discussion of this problem can be found in [3].
In conclusion, although the Logo language is created for children, but is quite powerful. In this language students can implement many algorithms using advanced con-

cepts. They can improve computational thinking and develop their programming skill. Although Scratch is becoming more and more popular, the Logo language still has its educational value. Even problems of medium complexity in visual programming are almost impossible to code whereas in the Logo language solutions are nice and readable.

4 Time for a snake

The third lesson is for older students. We prepared the Python application in the Processing environment because of its multimedia features. Implementation starts from a basic algorithm code, through simple visualization and ends with interaction.
There are a different way to code Euclidean algorithms. Although version with subtraction is less efficient, we will concentrate on it because it is easiest to visualise.

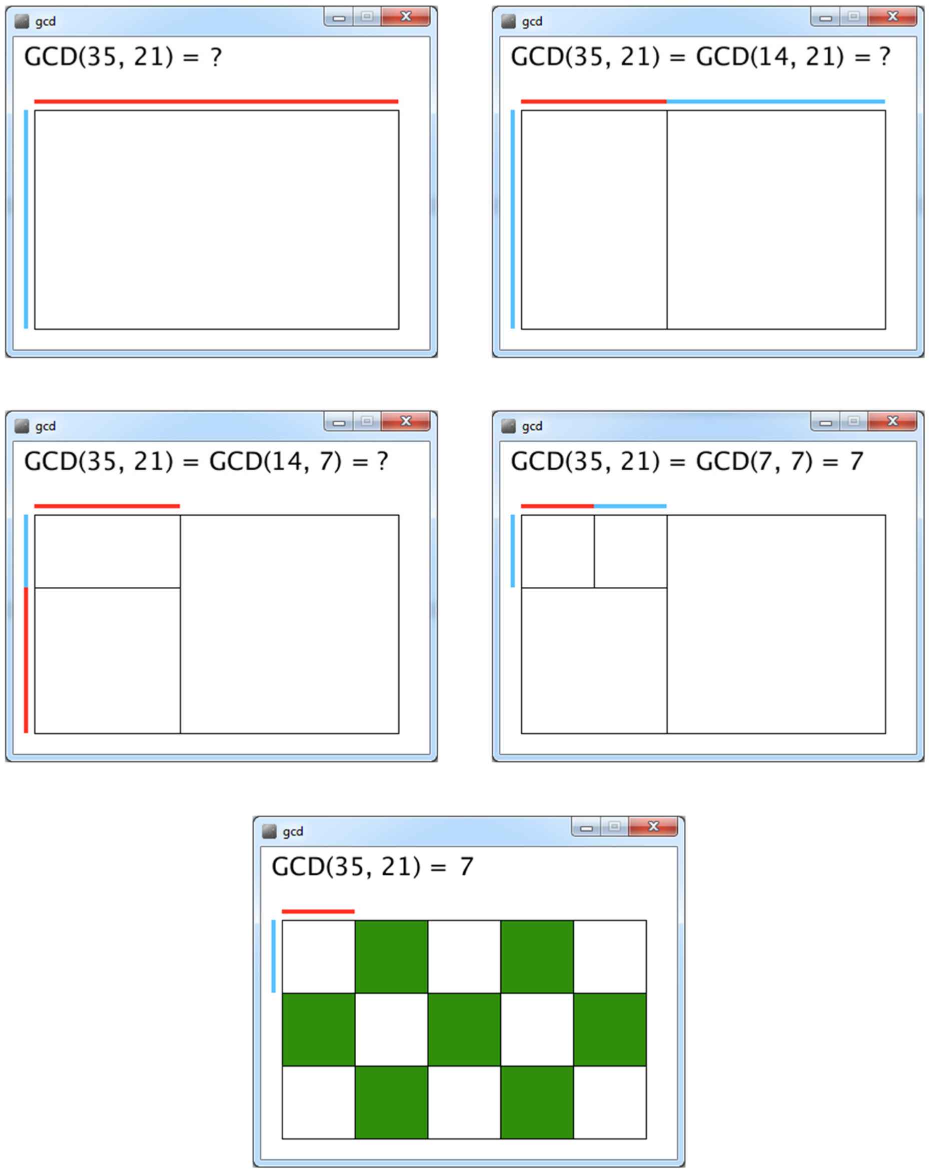
```
def gcd(a,b):
  while not (a==b):
   if (a > b):
    a = a - b
   else:
    b = b - a
  return a

def gcd(a,b):
  while (a>0):
    r = b % a
    b = a
    a = r
  return b

def gcd(m,n):
  if m>n:
    return gcd(n,m)
  else:
    if m==0:
      return n
    else:
      return gcd(n % m,m)
```

After coding the core of algorithms we can go forward to add some graphics. At the beginning, we built a rectangle with sides `a` and `b`. In each iteration a square is cut. In the presented example we calculate `GCD(35,21)`. It means that the program is drawing the rectangle `35` wide and `21` height. Next, we cut the square `21x21`, so we get the rectangle `14x21`. Now the height of the rectangle is bigger than its width, so we cut the square of side `14`. As we have rectangle `14x7`, we cut square `7x7`. Finally, we get the square `7x7`. At the end, we can draw a board with colour squares each

7x7. In addition, there are colour section shown above and next to the rectangle – a red one (horizontal) for the value of variable a and a blue one (vertical) for variable b.



```
GCD(35, 21) = ?
GCD(35, 21) = GCD(14, 21) = ?
GCD(35, 21) = GCD(14, 7) = ?
GCD(35, 21) = GCD(7, 7) = 7
GCD(35, 21) = 7
```

Fig. 7. Application in the Processing for visualisation of the Euclidean algorithm

Next we can add some interaction. Instead of automatic presentation, a user can point out the new values of a and b. If she/he does it correctly, the computer will do one step of the algorithm, otherwise nothing will happen. In this way the student is stimulated to think. In our application, ten buttons are added for numbers, and two more for a new a and new b.

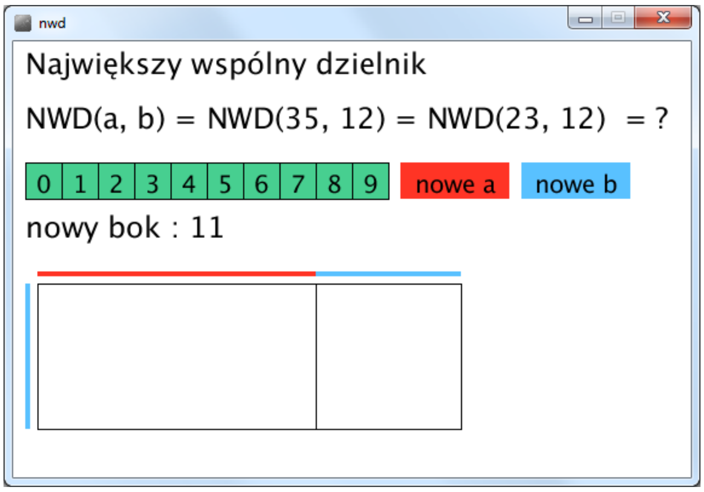

nwd
Największy wspólny dzielnik
NWD(a, b) = NWD(35, 12) = NWD(23, 12) = ?
0 1 2 3 4 5 6 7 8 9
nowe a
nowe b
nowy bok : 11

Fig. 8. Application in the Processing for visualisation of the Euclidean algorithm

In general, Python as one of high-level languages deserves interest. Firstly, the syntax usually allows programmers to express concepts in less lines of code than other languages and it emphasizes code readability. Secondly, it is developed as an Open Source project, and it is free. Thirdly, it can be programmed either by novice programmers, even primary school students, and professionals in programming as well. This language is widely used in education as well as in real-life systems. In this project, working in the Processing environment allows us to prepare multimedia applications. This environment is designed to create visually appealing graphics, interactive applications and games. Moreover, in Processing one can prepare applications for web pages and for different platforms like Android.

5 One more lesson

The older students, the more advance tasks they can solve. On the maturity exam in Poland (2015) there was the task entitled the Extended Euclidean algorithm. It was in the theoretical part of exam where students solve problems without computer. Students were given explanations and an example. Their task was to fill a form for another example and formulate the algorithm.
In the task there was a description for Euclidean algorithm. Students were given specification and short introduction to Extended Euclidean algorithm. Given example:

```
GCD(231,30) = 3 * 231 + (- 23 ) * 30
```

i – nr wywołania	*NWD* (*a*, *b*) Wartość *a* w *i*-tym wywołaniu	Wartość *b* w *i*-tym wywołaniu	Zagnieżdżanie rekurencji ←	Powrót z rekurencji →	Wynik *x*	Wynik *y*
1	231	30	↓	↑	3	–23
2	30	21	↓	↑	–2	3
3	21	9	↓	↑	1	–2
4	9	3	↓	↑	0	1
5	3	0	↓	↑	1	0

Fig. 9. The given example – calculations for a = 231, b = 30

Students have to fill a table for an example where `a`=188, `b`=12.

i – nr wywołania	Wartość ***a*** w *i*-tym wywołaniu	Wartość ***b*** w *i*-tym wywołaniu	Wynik x	Wynik y
1	188	12		
2				
3				
4		0	1	0

Fig. 10. First part of students' task

The next step was to specify algorithm written in pseudo-code. They had to formulate formulas and an output. For some students recursive algorithms are very difficult.

```
Specyfikacja:
    Dane:
        liczby całkowite $a > 0$ i $b \geq 0$
    Wynik:
        para liczb całkowitych $(x, y)$, dla których $NWD(a, b) = a \cdot x + b \cdot y$

RozszerzonyEuklides(a, b):
   Krok 1.   Jeśli $b = 0$, podaj jako wynik funkcji parę (1, 0) i zakończ jej wykonywanie.
   Krok 2.   $r \leftarrow a$ mod $b$
   Krok 3.   $(x, y) \leftarrow$ RozszerzonyEuklides(________________,________________ )

   Krok 4.   Podaj jako wynik parę (________________,________________ ).
```

Fig. 11. Second part of students' task

This tasks is interesting and instructive. Students can prove their ability to read algorithms and broaden their knowledge. The Extended Euclidean algorithm is a base for RSA algorithm that is important in the digital world. Generally, the Euclidean algorithm has many theoretical and practical applications.

6 Summary

Our proposition for the lessons on algorithmic is to work with interactive projects not only with the GCD problem but also with others. By playing and programming small applications students have the opportunity to understand an algorithm on a specific example and become familiar with the different ideas. Designing and coding their own programs they deepen and broaden algorithmic knowledge. Selecting the programming language and environment is not the main issue, but on the other hand, it cannot be said, that it is completely unimportant. The role of a teacher is to inspire students to take efforts and be creative. In algorithmic world, every student can assimilate the idea of another human being, she/he can see how it works on a specific example and even code himself. As John D. Carmack wrote: "Because of the nature of Moore's law, anything that an extremely clever graphics programmer can do at one point can be replicated by a merely competent programmer some number of years later."

References

1. Knuth, D.E., The Art of Computer Programming: Fundamental algorithms, Addison-Wesley, 1968
2. Cormen, T.H., Introduction to Algorithms, MIT Press, 2009
3. Man, Yiu-Kwong, Solving the General Two Water Jugs Problem Via an Algorithmic Approach. International Association of Engineers, 2015.
4. Scratch portal, http://scratch.mit.edu
5. The central examination commission, http://www.cke.edu.pl (Polish)
6. E-textbook, http://www.epodreczniki.pl (Polish)

Roboval: Robot Contest and Education with Arduino in High School

Maurizio Boscaini[1] [2] and Alberto Valente[3] [4]

[1] maurizio.boscaini@gmail.com
High School G. Marconi, Piazzale Guardini 1, Verona, Italy
[2] Department of Computer Science, University of Verona, Italy
[3] alberto@plumake.it
Plumake Srl, Viale del lavoro 2, Grezzana (VR), Italy
[4] co-founder Verona FabLab, Viale del lavoro, 2, Grezzana (VR), Italy

Abstract. In this paper we describe an experience of educational robotics in high school for Roboval, an annual fair organized by Verona FabLab that takes place in Verona (in the north of Italy) about robotics, innovations, and makers. Our focus will be on the learning and technical aspects. The aim is to form some student teams to participate in contests with a robot built by themselves. All the teams have to deal with software, in particular the programming of the Arduino system, and to take care of practical aspects. Some teams also have the opportunity to tinker the pieces together or to repair the hardware. The learning skills provided by this experience are varied and interesting. The theory is important but stays quietly in the background, while practice and collaboration play the main role.

Keywords: robotics, education, high school, positive challenge, learning by doing, cooperative learning, peer tutoring, teaching enzyme, active learning, computational thinking, contests and competitions in informatics.

1 Introduction

Since its first edition in 2012, Roboval has had students, schools, as well as the many fans of technology and hi-tech in a major role. Organized by Associazione Verona FabLab[1], Roboval is an annual fair of robotics, innovations and their innovators,and where creators and curious meet to share inventions and experiences. In 2015 the event moved from the town of Grezzana to the center of Verona in the former Austrian arsenal buildings.

For many students the central event is the robotic competition, attended by some high schools from the province of Verona. The team of students, guided by their teachers, begin preparing for the event a few months before with meetings, mostly in the afternoon. The whole work is based on collaboration and mutual help. The skills required are focused mainly on information technology and electronics, and the ultimate goal is to program a robot to solve a maze or to make a circular route in the shortest possible time.

Over time, the number of schools participating increased from 4-5 to 12-13, while the number of teams are now about twenty for the race labyrinth and 12 for the speed one.

Every school involves 6-10 students and every team is usually made up of 3-5 members.

In this paper we first introduce Roboval Fair activities, then we describe the hardware and software used for the competitions and in the end we focus on contents and educational methodology.

2 Kit Roboval Easy

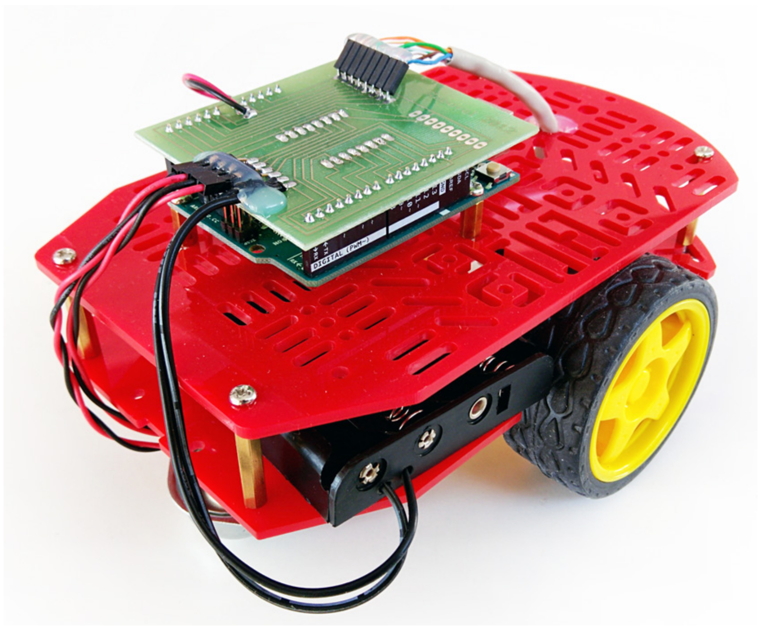


Fig. 1. The Roboval Easy Robot.

To the schools involved the Associazione Verona FabLab provides free of charge (thanks to the availability of some corporate sponsors) some robot kits based on the Arduino platform (assembled or unassembled). Other kits can be purchased at a cost of around 100 Euros.

2.1 Hardware

Roboval Easy robot is composed of an acrylic chassis equipped with two independent wheels and an omnidirectional wheel (steel sphere).

The robots movements are handled by an Arduino, which, using a custom shield developed specifically for Roboval by Plumake srl, allows the control of the two motors and of the front luminosity sensors. The assembly instructions and the photogallery are available on Roboval website.

2.2 Software

The Arduino IDE is freely downloadable from `www.arduino.cc` for Linux, Windows and Mac operating systems[2]. Some custom firmwares are available from Roboval website for testing robots behaviors:

– Pololu library for line sensor (to be installed inside Arduino libraries folder)
– Line sensor test firmware
– Motor test firmware

3 Challenges

Easy robots are used for the challenges held during Roboval Fair. Every team starts from the same initial conditions, both for hardware and for software, since the organizers provide a basic firmware which can be improved or completely rewritten.

The Roboval **Labyrinth** contest is an opportunity to challenge design and problem solving skills. The goal is to program the robot to solve a labyrinth in the shortest time.

Beginning this year a new race has been added : the Contest Roboval **Speed**. The purpose of this competition is to program a robot to independently move along a loop without crossings or sharp bends in the shortest possible time.

4 Contents and educational methodology

We put together content and teaching methodology for multiple reasons. First, the training process/design, though based on a common canvas, was developed completely autonomously by the participating schools according to availability, teaching styles, skills and knowledge of each teacher involved. Secondly, because in this case the motto "learning by doing" has been the pervasive motivation in the educational experience, making it quite difficult to separate the product from the inventing process.

The spirit of adventure and discovery is a key point of the educational approach. Even the competitive aspect is important ,in that it heightens the need to meet and discuss, in the light of **positive challenge**: first and foremost to themselves and their limits, and then to the other team.

In addition, the cooperative aspect, where possible, is heavily favored. In the first meetings of the working group cooperation is practically mandatory: the help and mutual support is basic and continuous.

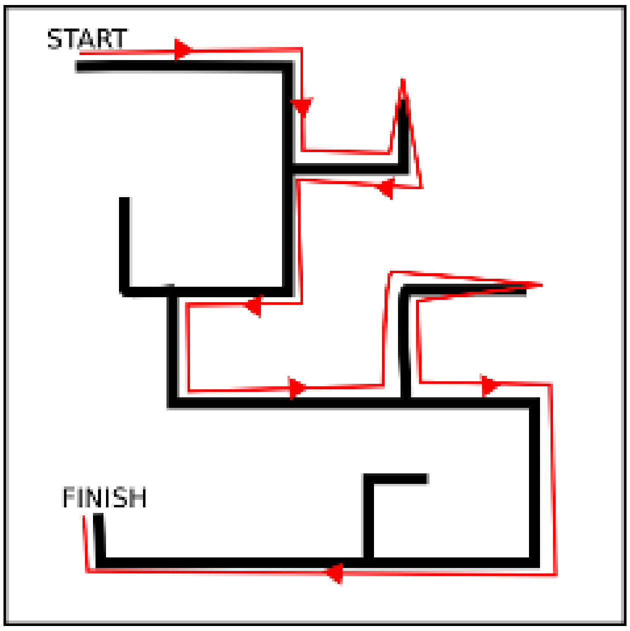


Fig. 2. A Roboval Labyrinth example.

At the ITIS Marconi in Verona this year, the number of boys involved was more than 30! From first year to fifth year students participated, some of whom had never programmed before. In the first of the afternoon meetings to begin the project, students new to the competition were introduced by the teacher to the objective to be achieved and the means to do so. In particular to define the subject of robotics and programming Arduino, within the syllabus as follows:

4.1 A simple Syllabus for the course

Knowledge Area	**Knowledge Item**
Introduction to robotics	Brief history of robotics and etymology of the word robot. Application areas of robotics.
Basic features of a robot	Environment, sensors and actuators. Degrees of autonomy.
Introduction to the Arduino platform	Brief to Arduino project. Basics. Connection to the PC.
Introduction Kit Roboval "Easy"	Basics Kit Roboval "Easy": engines, inline sensor, wheel, power source, shield connection, Arduino.
Introduction to Programming	Basic principles of programming: control structures, variables, functions.
Programming Arduino	Arduino programming environment. Basics of Arduino programming language. Functions setup() and loop().
Problem Posing and Solving	Presentation and discussion of basic problems: rotate the robot in a square pattern, ... Presentation of the challenges of the competition: "Labyrinth" and "Speed".

The canvas of a program for Arduino.

```
/* it is performed only once at the launch of the program */
void setup ()
{}

/* is performed after the setup and called cyclically
until the end of the program */
void loop ()
{}
```

In the first meeting the students could experience what they saw briefly in theory. In later meetings, they would experience for the first time, some theoretical arguments still not well understood; but most of the time was devoted to spontaneity in the laboratory in groups of two or three . In this situation, the teacher, using a sports metaphor, was also playing the role of a "coach", a catalyst (to encourage the creation of any groups that might not otherwise form), a facilitator, an organizer.

A very interesting and important aspect of "peer tutoring", is the mutual help of students. The older and/or more expert assume the task of helping the younger and/or less expert. In particular, the support takes place in two situations: in the initial phase to knowledgeably start the first tests (a **teaching enzyme**, that is, a biological catalyst of the learning process) and again if the group is faced with a problem and cannot overcome it.

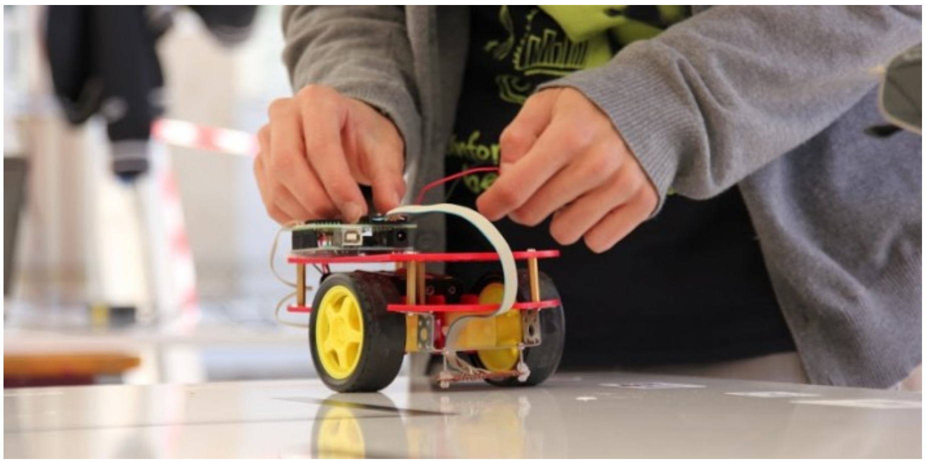


Fig. 3. Preparation phase at the Roboval contest.

The climate in the classroom is typically constructive and cooperative and this allows the development of relational skills as well as learning skills and new knowledge .

On some occasions, especially in the early stages of the competition, the students were involved in the activity of the kit assembly. The physical experiences of welding, cutting, gluing etc. were very appreciated by almost all the students. Because for many of them, it was the first time they could get the hands on experience of building an electronic device.

The team must also consider all the practicalities. Robots "Easy" uses a 9V battery to power Arduino and a battery pack of 4 AA batteries to power the motors. Participants must be autonomous with regard to replacement batteries and chargers for the conduct of the race. In addition to the main program, the teams have to deal with other aspects such as the very practical concern of the field tests. These can sometimes differ greatly from theoretical calculations, for example, the natural or artificial light detected by the sensors, or the slipperiness of the ground on which the robot moves, or by the weight and weight distribution of the robot, (in particular the electric batteries).

5 Conclusions

This experience so far has certainly proven to be valid from the point of view of motivation and direction for students, as well as for the educational aspects that were conveyed. The short syllabus presented above may be a common reference for teachers but can also be developed and extended. This is to try to overcome some practical difficulties, that more than one school reported, in being able to progress with a certain continuity with their students. The commitments of teachers especially in the final stage of the school year sometimes made this continuity difficult.

This method, to start from a challenge, create a project and make it happen, was much appreciated by the participants, who have demonstrated a steady and effective collaborative spirit. The theoretical contents are sometimes sacrificed. With older students, also the direct and inverse kinematics issues could be considered, but only if placed in a more advanced and comprehensive curriculum.

To continue this analysis, taking as reference the Bloom's taxonomy educational path[3], though this education process starts from important theoretical basis, it is somehow biased towards experiential and heuristic learning. From the beginning it focuses too much on the higher levels of the taxonomy (application, creation, analysis , evaluation) and maybe too little on the basic ones (memorization and understanding). Nevertheless our belief is that, in this experience, a theoretical-front lack is a price worth paying.

References

1. Roboval, Web. 11 Jul. 2015. http://www.roboval.it/
2. What is Arduino? Arduino. Web. 11 Jul. 2015 https://www.arduino.cc/
3. Tassonomia di Bloom, Wikimedia Foundation, n.d. Web. 12 Jul. 2015 https://it.wikipedia.org/wiki/Tassonomia_di_Bloom

Blended Learning Environments, Flipped Class and Collaborative Activities to Teach Databases in a Secondary Technical School

Maria Concetta Brocato

ISIS Arturo Malignani, Computer Science Teacher, Udine, Italy

mariaconcetta.brocato@malignani.ud.it

Abstract. The paper describes some learning and collaborative activities, some tools and an on-line environment created to teach the Databases to secondary technical school students (aged 17-18). The on-line course, created by using the Learning Management System Moodle, is a blended learning environment to support daily classroom activities. The on-line environment increases dynamically with the contribution of teacher and students and "wraps around" the class during the learning process. The course allows *"Flipped Classroom"* and collaborative activity (on-line and classroom), it supports discussion and sharing information. It also enables students to have personal/sharing areas for files and supports the main cloud services to upload/download files. Topics of the course: modelling a database using the relational model, designing Entity/Relationship diagrams, SQL language, case studies based on real life and student scenarios.

Keywords: Database, Relational Model, SQL, Blended Learning, Flipped Classroom, Moodle

1 Introduction

The "*Fundamentals of Databases*" is a course designed for Italian Secondary Technical School students (aged 17-18, specializing in Computer Science and Telecommunications). The Computer Science subject has 99 hours per year for the first, third and fourth year of the five provided. An Italian Technical School year has 32 hours per week; Computer Science occupies 3 hours per week for the whole school year. In the first year of this specialization, students, aged 14-15, have an introduction to the basic concepts of ICT and digital literacy to develop digital skills and key competencies. These goals are achieved by using Office Automation tools, searching information on the web, evaluating sources and organizing the data, which is finally presented. This also includes studying computer architecture, numbering systems and Boole's algebra. Very often the approach is problem solving based. During the second part of the year, teachers introduce algorithms, flow charts and coding. This is possible using one of two different approaches: easy and informal with icons and graphic environments (like *"Scratch"* [1] and *"Code.org"* [2]) or more formally using a common Integrated Development Environment for a high level language, e.g. C++ Language. During the entire third year, students, aged 16-17, study programming languages, algorithms, flow charts and code. At 17-18, students complete their route in Computer Science by studying Website Design and Databases; this last module contains the course described.

The course designed is composed by common daily class activities (in the classroom) and some on-line activities (in the classroom or at home); some activities are synchronous, some other asynchronous. The on-line course, created using the Learning Management System Moodle [3], supports all daily class activities, so the learning methodology is "blended" and "hybrid". The on-line environment increases dynamically with the contribution of students and teacher and *"Wraps Around"* the class during the learning process. The course allows *"Flipped Classroom"* methodology and collaborative activities (on-line and classroom); supports discussion and sharing of information from teacher to students but, mainly, through **peers who are the basic actors of the learning process**. It also enables students to have personal/sharing areas for files and supports the main cloud services to upload/download files such as Google Drive or Dropbox. It is possible to highlight three basic features of the course "Fundamentals of Database":

- the first is connected with **topics**: students are introduced to important topics of ICT such as modelling a database, designing Entity/Relationship Diagrams, building the schema in a *MySQL* database, using *SQL Language*; all the topics are connected to problem solving competence;
- the second is connected with **didactic methodology**: concepts are learnt by practical and collaborative activities (on-line and classroom), through team work and group discussions, using flipped classroom scenarios and blended learning environments; the teacher supports learning in the classroom but is also a tutor in the on-line course; this enables students to help each other in the on-line forums, to discuss in threads, as well as to share and suggest solutions; write collaboratively on a wiki page; teacher handouts and textbooks are expanded with best student's homework or conceptual maps that are very often discussed/modified and shared in the classroom;
- the third feature is connected with **students motivation**: the student's interest is enhanced using case studies taken from their daily life; this allows them to share and discuss their personal observation of reality and to easily identify a possible solution; furthermore collaborative and useful interaction in forums/wiki pages are evaluated by the teacher, at the end of each term.

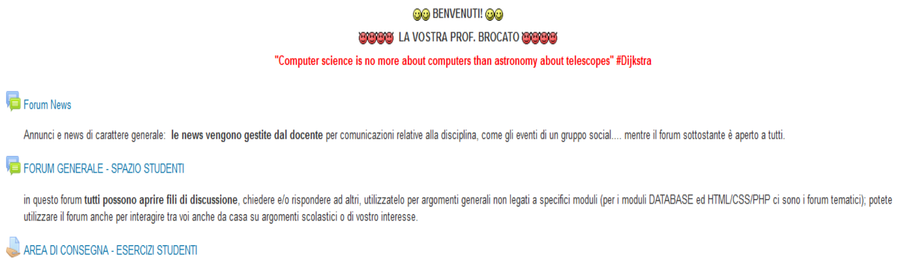
The effects of the last two points are that, each year, the course is different: enriched with the contribution of peers, not static, dynamic and open to personal styles and student's needs.

2 Topics: the Modules of the on-line course

2.1 Common Area

At the beginning of the Moodle course, there is a common area with three resources:

- "**News Forum**": managed only by the teacher for communications such as the presence of new material in the on-line course, a deadline to hand in an exercise, a date of an important conference to follow, etc.;
- **"General Purpose Forum"**: everyone (students and teacher) can open a new discussion, ask or answer; it is used for general topics not related to specific modules;
- **"Delivery Area":** each student can upload files or exercises *like a cloud personal area inside the course*, the area is also visible to the teacher but not to other peers; the upload supports the main cloud services. To help the teacher during corrections and evaluation, the name of each exercise delivered must follow the guidelines for standard file names: "*Surname_ProgressiveNumber_ShortText*".


BENVENUTI!

LA VOSTRA PROF. BROCATO

"Computer science is no more about computers than astronomy about telescopes" #Dijkstra

Forum News

Annunci e news di carattere generale: **le news vengono gestite dal docente** per comunicazioni relative alla disciplina, come gli eventi di un gruppo social.... mentre il forum sottostante è aperto a tutti.

FORUM GENERALE - SPAZIO STUDENTI

in questo forum **tutti possono aprire fili di discussione**, chiedere e/o rispondere ad altri, utilizzatelo per argomenti generali non legati a specifici moduli (per i moduli DATABASE ed HTML/CSS/PHP ci sono i forum tematici); potete utilizzare il forum anche per interagire tra voi anche da casa su argomenti scolastici o di vostro interesse.

AREA DI CONSEGNA - ESERCIZI STUDENTI

Fig.1: the icon 💬 point out Forums, the icon 📄 points out the student's cloud personal area

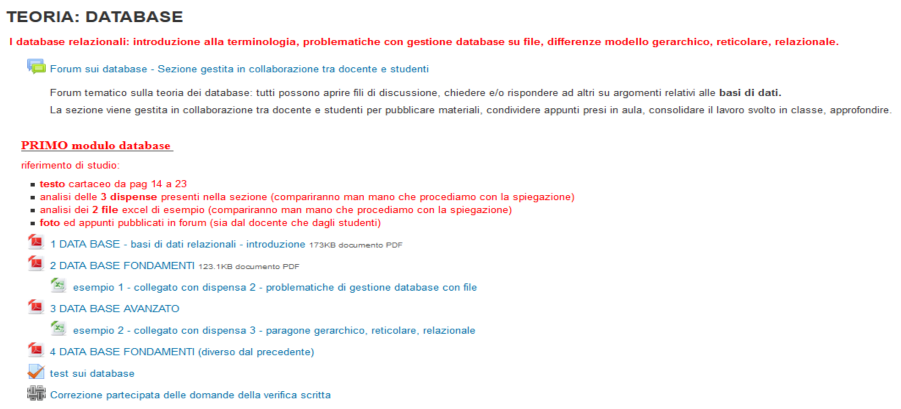
The teacher uses a communication mode related to students such as icons like *"Smileys"* or *"Red Devils"* and the hash symbol "#" to reference a main topic or the "@" to reference a student like on *"Twitter"*, in order to engage the student's attention.

2.2 Module 1 (Study and Learn): Theory of Modelling a Database

Module 1 is the theoretical part of the course: students have to study and learn basic concepts of Databases. It contains:

- **"Database Forum"**: a specific forum, everyone can open a new discussion, ask or answer about materials and lessons in the classroom, as well as sharing personal notes, maps, diagrams, summaries, links and so on;
- **Summary:** helps to study the module and to perform the final assessment: it contains reference pages from the text book, a quick index of different kinds of files and a short description on how to use materials;

- **Documents about the "Theory of Databases"**: written by the teacher to explain: basic terminology and symbols, Fundamentals of Relational Model, Conceptual, Logic and Physical Level, Entity/Relationship Diagrams (E/R), *SQL Language* for query, manipulate and create a database in a *MySql* Environment; sometimes the best student's homework, is published in this section, showing great valence to revise or to study; very often the contents written on the whiteboard are also shared in this section;
- "**On-line Test**": the final, individual, assessment; the test is composed of different types of questions (open, multiple choice, radio button, cloze, gapped text, etc.).



TEORIA: DATABASE

I database relazionali: introduzione alla terminologia, problematiche con gestione database su file, differenze modello gerarchico, reticolare, relazionale.

Forum sui database - Sezione gestita in collaborazione tra docente e studenti

Forum tematico sulla teoria dei database: tutti possono aprire fili di discussione, chiedere e/o rispondere ad altri su argomenti relativi alle **basi di dati.**
La sezione viene gestita in collaborazione tra docente e studenti per pubblicare materiali, condividere appunti presi in aula, consolidare il lavoro svolto in classe, approfondire.

PRIMO modulo database

riferimento di studio:

- **testo** cartaceo da pag 14 a 23
- analisi delle **3 dispense** presenti nella sezione (compariranno man mano che procediamo con la spiegazione)
- analisi dei **2 file** excel di esempio (compariranno man mano che procediamo con la spiegazione)
- **foto** ed appunti pubblicati in forum (sia dal docente che dagli studenti)

1 DATA BASE - basi di dati relazionali - introduzione 173KB documento PDF
2 DATA BASE FONDAMENTI 123.1KB documento PDF
esempio 1 - collegato con dispensa 2 - problematiche di gestione database con file
3 DATA BASE AVANZATO
esempio 2 - collegato con dispensa 3 - paragone gerarchico, reticolare, relazionale
4 DATA BASE FONDAMENTI (diverso dal precedente)
test sui database
Correzione partecipata delle domande della verifica scritta

Fig.2: : the icon points out thematic forums, the icon points out theoretical files, the icon points out Excel files, the icon points out a test assessment, the icon points out a wiki page to correct the assessment

The concepts are explained by real cases, using simple problematic situations, possibly related to a student's daily life. Materials contain diagrams that help students to understand historic passages in Database Theory: to motivate the Relational Model and the use of a DBMS. At the beginning of the module, the teacher highlights errors connected with types of databases not managed with a Relational DMBS. Examples of these types of databases are paper databases, electronic databases that use separate and not synchronous files. This topic is connected with "*Data Redundancy*": to manage a database without errors it is necessary to save, read and maintain information in one place and to refer to it with the definition of relations through data. This aim is introduced through a case study: "*A bank database has to memorize and to manage some current accounts, memorizing name, surname, address, account number, amount, deposits, withdrawals and bank transactions*". Redundancy errors are highlighted using Excel Sw (*"esempio 1"* file in figure 2) with different and not synchronized folders for *"bank details"*, *"clients"* and *"transactions"*. The teacher asks the students for information connected to the problem; this information is written in folder's cells including total amounts with one or two transactions. After a short talk, the teacher manually modifies the cells of the transaction folder by adding, deleting and altering data. Errors are then pointed out by reading the asynchronous folder of the current amount. Other similar examples are possible for the account number or name and surname.

Files, numbered from 1 to 4 in figure 2, contain detailed and technical contents such as: terms and definitions with connected examples, Entity/Relationship Diagrams, real life case studies, screenshots from *MySql*, queries and commands of SQL Language.

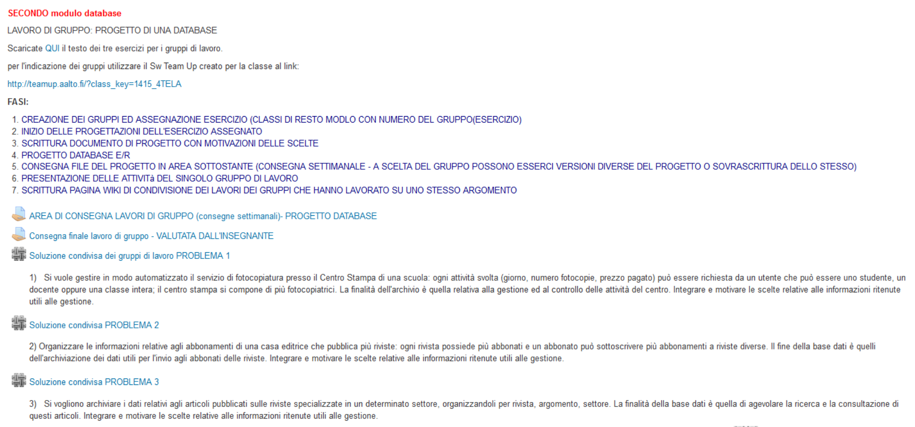
About the on-line test: the teacher manually evaluates all the open answers; the close answers are evaluated automatically by Moodle LMS; at the end of the test, the system automatically shows students feedback for each question; the feedback could be written by the teacher before the test or modified during corrections.

After the end of the on-line test a key activity is performed: as a homework task, students have to write down solutions to each question on a wiki page (last activity of figure 2). For each question, they have to work in collaboration: writing reference page numbers of the textbook or useful electronic materials (in the course or on-line, written by the teacher or students) resulting in a clear and exhaustive answer. The purpose and the relevance of this activity is connected to the third feature of the introduction section.

2.3 Module 2 (Apply and Collaborate): Design a Database and Write a Project

Module 2 is relevant because it involves the application of the theoretical concepts using an E/R Design Tool and some graphic tools for *MySql* environment. Students work in pairs, because the task "*Design a database and write a Project*" is not a simple task and needs lots of different and transversal competencies. The teamwork is a detailed project file of a relational database with the E/R diagram, logical and physical levels and a technical report file containing all the choices made and motivations. The Module contains:

- **"Case Studies"**: three different and simple case studies, chosen from an environment close to real life;
- "**Creating Groups and Logbooks**": the working groups are not chosen manually by teacher but "*Team Up"* Sw is used [4]; this is an on-line tool developed by Aalto University which allows one to create random groups and to record a simple voice logbook for each group (weekly, daily, less or more often as students desire).
- **"Steps and Final Delivery of Teamwork"**: it describes the whole module, helps each group to perform the teamwork activities, contains steps on how to use materials, to deliver the weekly activity report and the final project;
- **"Collaborate and Improve with Wiki"**: all students collaborate writing a dedicated wiki page for each of the three case studies. At home, each student discusses the two case studies not performed in pairs; students have to read all the teamwork projects connected to the case studies and identify an improved common solution. The comment section, of each wiki pages, can be used to discuss the common solution.



SECONDO modulo database

LAVORO DI GRUPPO: PROGETTO DI UNA DATABASE

Scaricate QUI il testo dei tre esercizi per i gruppi di lavoro.

per l'indicazione dei gruppi utilizzare il Sw Team Up creato per la classe al link:

http://teamup.aalto.fi/?class_key=1415_4TELA

FASI:

1. CREAZIONE DEI GRUPPI ED ASSEGNAZIONE ESERCIZIO (CLASSI DI RESTO MODLO CON NUMERO DEL GRUPPO(ESERCIZIO)
2. INIZIO DELLE PROGETTAZIONI DELL'ESERCIZIO ASSEGNATO
3. SCRITTURA DOCUMENTO DI PROGETTO CON MOTIVAZIONI DELLE SCELTE
4. PROGETTO DATABASE E/R
5. CONSEGNA FILE DEL PROGETTO IN AREA SOTTOSTANTE (CONSEGNA SETTIMANALE - A SCELTA DEL GRUPPO POSSONO ESSERCI VERSIONI DIVERSE DEL PROGETTO O SOVRASCRITTURA DELLO STESSO)
6. PRESENTAZIONE DELLE ATTIVITà DEL SINGOLO GRUPPO DI LAVORO
7. SCRITTURA PAGINA WIKI DI CONDIVISIONE DEI LAVORI DEI GRUPPI CHE HANNO LAVORATO SU UNO STESSO ARGOMENTO

AREA DI CONSEGNA LAVORI DI GRUPPO (consegne settimanali)- PROGETTO DATABASE

Consegna finale lavoro di gruppo - VALUTATA DALL'INSEGNANTE

Soluzione condivisa dei gruppi di lavoro PROBLEMA 1

1) Si vuole gestire in modo automatizzato il servizio di fotocopiatura presso il Centro Stampa di una scuola: ogni attività svolta (giorno, numero fotocopie, prezzo pagato) può essere richiesta da un utente che può essere uno studente, un docente oppure una classe intera; il centro stampa si compone di più fotocopiatrici. La finalità dell'archivio è quella relativa alla gestione ed al controllo delle attività del centro. Integrare e motivare le scelte relative alle informazioni ritenute utili alle gestione.

Soluzione condivisa PROBLEMA 2

2) Organizzare le informazioni relative agli abbonamenti di una casa editrice che pubblica più riviste: ogni rivista possiede più abbonati e un abbonato può sottoscrivere più abbonamenti a riviste diverse. Il fine della base dati è quelli dell'archiviazione dei dati utili per l'invio agli abbonati delle riviste. Integrare e motivare le scelte relative alle informazioni ritenute utili alle gestione.

Soluzione condivisa PROBLEMA 3

3) Si vogliono archiviare i dati relativi agli articoli pubblicati sulle riviste specializzate in un determinato settore, organizzandoli per rivista, argomento, settore. La finalità della base dati è quella di agevolare la ricerca e la consultazione di questi articoli. Integrare e motivare le scelte relative alle informazioni ritenute utili alle gestione.

Fig.3: the icon points out the delivery area for the group files, the icon points out the three wiki pages, one for each case study, to perform collaboration for the improved solution

Some observations about module 2:

- the first part of the module, which needs group activity, is developed during:
  (a) classroom activities: in an ICT laboratory; during this stage students collaborate together and can use the net to search for information, take notes and share documents using Google Tools [5], (Google Docs, Google Sheets, Google Slides) or other collaborative Tools; the design of the E/R schema could be performed: with a simple graphic Tool like "*Diagram Designer*" [6], or others more technical, like "*Raise Editor*" [7];
  (b) extra scholastic time: to observe a real case connected to the case study (for example to interview people connected with the case;
- the second part of the module, that also needs personal activity, is developed at home or in the ICT lab during class activities, using an asynchronous interaction to encourage autonomous reflection.

Solutions adopted are thoroughly evaluated by the teacher if supported by photos from real life or the web, connected to the problem and that confirms their choices. Students, before activity, know the evaluation scale that contains **quantitative indicators** (wiki writings, forum posts, peers support, number of real photos or interviews) and **quality indicators** (complete and clear E/R schema, correct logical and physical level, types of errors, correct technical descriptions, clear motivations).

In detail, the text of the three case studies are:

- *Organize information concerning **subscriptions of a publishing house** that publishes different magazines: every magazine has subscribers and a subscriber can subscribe to multiple subscriptions to different magazines. The purpose of the database is to archive useful data for sending to subscribers. Integrating and motivating the decisions related to the information;*

- *Organize information about a photocopying service at a* ***Press Center of a school****; any activity carried out (date, time, number of photocopies, price paid) may be requested by a user who may be a student, a teacher or a whole class. The press center has different copy machines. The purpose of the database is the management and the control of the activities of the center. Integrating and motivating the decisions related to the information considered useful;*
- *Organize the store and the search of data relating to* ***articles published in specialized magazines*** *in a given area, by organizing the layout of a magazine, topics and industries. The purpose of the database is to facilitate the search and consultation of these items. Integrating and motivating the decisions related to the information considered useful.*

2.3 Module 3 (Apply Autonomously): SQL Language

Module 3 is relevant because it involves the application of the theoretical concepts in the whole *phpMyAdmin* environment. Students work alone. The activity, in the first part, continue the teamwork of Module 2: student must write personal observations versus the common solution presented in wiki (for the two cases not analyzed in pairs). After, each student must write, in text words and not in SQL language, almost one need of information connected to the reality and post them in forum. In the final delivery, students write a personal project containing E/R diagram, observations and queries in SQL language. They also have to write some creation commands for databases, tables and fields. The Module contains:

- a **forum** about each **"Case Study"**: in the three different forums each student has to write a post that contains at least a query written in text words;
- within the **three forums** there are **different threads**, one for each text query posted by students; students must reply to, at least, one post writing the SQL query connected to the text; students can correct each other's solutions;
- A **delivery area** for final personal project file.

The activity **"collaborative writing queries in forum"** allows students to collaborate during homework activities in an on-line environment; students have to read all the threads connected to the case study and identify a good response to the solution. The teacher observes the behavior of students in forums, reads posts and corrects them; he uses a communication mode close to students: icons like *"Smileys"* or *"Red Devils"* and the "@"symbol to reference a student.

3 Methodology: "Wrap Around" environment, collaborative and group activities, flipped teaching

3.1 "Wrap Around" Environment

All the modules, materials, activities and tools used in the course are collected and available in an on-line environment which was created using the Learning Management

System (LMS) *Moodle - version 2.7.1* as well as some plugins developed by *Moodle's Community*. The on-line course is used as a "*blended learning environment"* that collects and shares all the materials used and produced by students and the teacher (links to important web sites, all kinds of documents, discussions in forums, wiki pages), so **it supports every daily class activity and it is the "core" of the whole didactical action**. The course was built to allow "*flipped classroom*", to use collaborative tools and collaborative activities, to share documents using some common cloud tools such as Dropbox and Google Drive. The environment expands dynamically during the year and "*wraps around*" the whole class.

During school activities (classroom):

- the teacher uses the interactive whiteboard and a PC, connected to the net, in order to use and discuss documents saved through the course or to immediately save the actual lesson performed;
- during all the lessons in the classroom, students can choose to use their personal devices (BYOD) in order to feel more comfortable, to add personal notes in their reserved area or to share information and relevant files;
- during ICT lab lessons, each student uses a PC in order to use uploaded materials, to execute exercises, to add personal files in their reserved area or to share information or files in dedicated areas with the rest of the class;
- there is an alternation of plenary discussion moments and group activities.

During home activities (on-line):

- the teacher publishes materials in the on-line course, uploads files, organizes modules and sections, plans and explains the steps of each activity, prepares on line assessments and quizzes;
- students use or download materials shared in the on-line course by the teacher, read, write and collaborate in forums and on wiki pages, use collaborative tools and perform exercises and assessments;
- everyone should write in the forums of the course, ask a question or answer other questions;
- there is a peer collaboration under the supervision of the teacher who is also an on-line tutor.

There is a fluid interaction: classroom and home activities do not follow fixed schemas. The learning process alternates presence and physical distance; interaction, through media and technologies, can be synchronous or asynchronous in a "liquid" succession. **Technology is integrated into lessons, do but does not substitute them:** textbooks, personal devices, files, exercise books and notes should be used together.

The lesson moves from the classroom to the web, students come in the class having already read the theoretical contents prepared by the teacher and shared in the LMS of the school. **The lesson is flipped**: the students have seen the lessons, so the class time could be more active and becomes a workshop.

There is an overcoming of the hierarchical rule from teacher versus student:

- **creation of materials**: not only the teacher creates and publishes materials, students perform a co-creation and post files, maps or links;
- **interaction**: the course improves peer education and discussion in forums or wiki but also during classroom activities.

3.2 Collaborative and Group Activities

The activity in the classroom becomes more focused to problem solving: a discussion of possible solutions rather than an explanation from one channel, the teacher, to students like passive spectators. It is more dynamic: not only one channel of communication but two or more. The collaborative activity in the classroom and on the on-line course is made possible by using useful tools:

- **TeamUp** on-line Tool [4]: allows the teacher to randomly create groups with a dedicated vocal diary for each group; the diaries are managed and maintained by the students. The single registration keeps one minute of voice, but groups could record a lot of entries. The teacher can listen to the recordings before the lesson which may make the teacher aware of any problem areas;
- **Padlet** on-line Tool [8]: it is a virtual board to tap personal notes or to exchange notes for group work, it is often used in a recent MOOC [9];
- **Cmap Tool** [10], famous concept map Sw, is used collaboratively to revise theoretical concepts; it is useful as a homework task then enriched in the classroom, for example first of an assessment. Students create a map at home and deliver it on the Moodle platform, before the lesson the teacher displays maps and identifies the most complete. Those identified are discussed in the classroom. The maps are integrated and published during the course; this allows all students to perform a better preparation for the final assessment. This action also allows one to improve results on the checks.

3.3 Flipped Teaching

Before lessons:

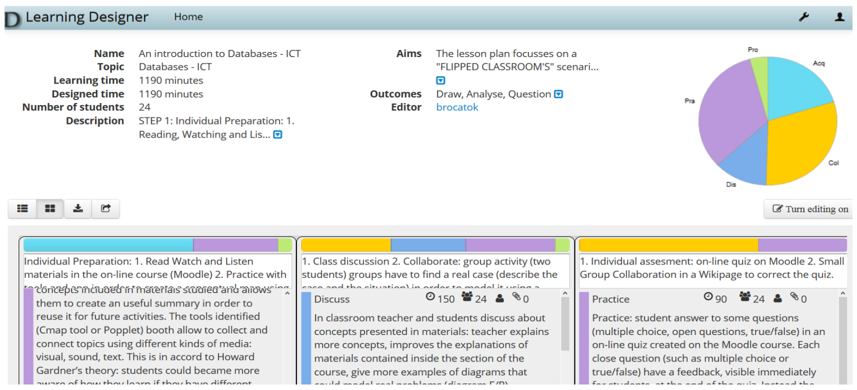
- **the teacher** uploads selected contents and files (choosing between ready materials and/or self-produced) in an on-line environment (LMS or on-line Tools);
- **students** perform activities like reading contents in the on-line environment, create personal maps and study; students can communicate with each other (peer) and with the teacher through forums and messages in the on-line environment.

Concepts take shape in the virtual classroom with the support of the teacher who can guide, trace the path, check, facilitate and give sense to the home activity. Each student prepares themselves at home for classroom activities.

During classroom lessons there are: real time discussions, negotiations of meaning, depth study, collaborative learning, synchronous learning, group activities. LIM or whiteboard are used for the negotiation of meaning.

3.4 Other useful Sw and Tools

- **"Prezi"** an on-line Tool [11]: to design collaborative and multimedia presentations;
- **"Google Apps"** Tool [5]: to store files in cloud, to share and write files, to assess;
- **"Socrative"** on-line Tool [12]: to create quizzes;
- **"Learning Designer"** Sw [13]: is a specific on-line graphic Tool used by teachers to plan class activities with simple drag and drop actions, small text descriptions and check lists. An example of a lesson, planned with the Sw, is contained in the following screenshot:



Name	An introduction to Databases - ICT
Topic	Databases - ICT
Learning time	1190 minutes
Designed time	1190 minutes
Number of students	24
Description	STEP 1: Individual Preparation: 1. Reading, Watching and Lis...
Aims	The lesson plan focusses on a "FLIPPED CLASSROOM'S" scenari...
Outcomes	Draw, Analyse, Question
Editor	brocatok

Individual Preparation: 1. Read Watch and Listen materials in the on-line course (Moodle) 2. Practice with

concepts included in materials studied and allows them to create an useful summary in order to reuse it for future activities. The tools identified (Cmap tool or Popplet) booth allow to collect and connect topics using different kinds of media: visual, sound, text. This is in accord to Howard Gardner's theory: students could became more

1. Class discussion 2. Collaborate: group activity (two students) groups have to find a real case (describe the

Discuss 150 24 0

In classroom teacher and students discuss about concepts presented in materials: teacher explains more concepts, improves the explanations of materials contained inside the section of the course, give more examples of diagrams that

1. Individual assesment: on-line quiz on Moodle 2. Small Group Collaboration in a Wikipage to correct the quiz.

Practice 90 24 0

Practice: student answer to some questions (multiple choice, open questions, true/false) in an on-line quiz created on the Moodle course. Each close question (such as multiple choice or true/false) have a feedback, visible immediately

In following text (in italics) there is the beginning of the resulting plan (is possible to print a pdf file of a plan directly from the site) containing, only the general part and the first step of the lesson planned.

Context

Topic: Databases – ICT *Total learning time: 1190* *Number of students: 24*
Description:

- ***STEP 1: Individual Preparation:*** *1. Read Watch and Listen materials in the on-line course (Moodle) 2. Practice with tools and app in order to create a conceptual map using tablets 3. Produce a file or a link, with the map, and upload it in the Moodle course of the class.*
- ***STEP 2: Collaboration:*** *1.Class discussion 2. Collaborate: group activity (two students) groups have to find a real case (describe the case and the situation) in order to model it using a database 3. Produce: group activity (two students) groups have to project the conceptual schema of the database (diagram E/R) and the model of the database connected to the case of*

the previous step 4. Share the collaborative description and the project of the Database schema connected with (diagram E/R) in a FORUM in Moodle.

- ***STEP 3: Assessment:*** *1. Individual assessment: on-line quiz on Moodle 2. Small Group Collaboration in a Wiki page to correct the quiz.*

Aims

The lesson plan focusses on a "FLIPPED CLASSROOM'S" scenario; it describes some activities to introduce an important concept for ICT students in a secondary school: "Introduction to projecting a Databases with E/R schema". Flipped activities are shared in the on-line course of the class in the Moodle environment of the school. Students and teacher, during their school activities, can use: an interactive whiteboard and a PC connected to the net in the classroom, a personal tablet for each (BYOD) and a fast wifi connections provided, instead, by school. During home activity students use tablets to access to materials shared in the on-line course by teacher, to collaborate in forums and in wiki pages, to use tools.

Outcomes

- *Draw (Knowledge): draw a map*
- *Analyse (Analysis): analyse a real case*
- *Question (Comprehension): Answer questions*

<u>Teaching-Learning activities (TLA): TLA 1 - Individual Preparation:</u>

- ***Read Watch and Listen materials in the on-line course (Moodle)***
- ***Practice with tools and app in order to create a conceptual map using tablets***
- ***Produce a file or a link, with the map, and upload it in the Moodle course of the class.***

Read Watch Listen	***240 minutes***	***24 students***	***Tutor is not available***

Each student, individually at home, logs into the Moodle on-line course and read, watch and listen the materials shared by teacher.

Practice	***120 minutes***	***24 students***	***Tutor is not available***

Students, at home, using their own tablets write a conceptual map using Cmap tool or Popplet on-line Sw or Popplet App for tablet (the choose is free). This activity helps students to identify the main concepts included in materials studied and allows them to create a summary in order to reuse it for future activities. The tools identified (Cmap or Popplet) booth allow to collect and connect topics using different kinds of media: visual, sound, text. This is in accord to Howard Gardner's theory: students could became more aware of how they learn if they have different kinds of learning activity.

Produce	***20 minutes***	***24 students***	***Tutor is not available***

When the conceptual map was ended, each student logs into the Moodle on-line course of the classroom and upload the map (a link, a file) in the area dedicated to the delivery.

3.5 Reflections

Through technology it is created a learning **environment for**:

- **student support**: students retrieve videos and different types of multimedia files, perform self-assessment tests, write questions in a moderated forum. Students cannot accumulate deficiencies and have different types of interactions to solve problems. Moreover, studying with contexts and environments that offer diversified media, students can achieve the **customization of a learning process**; this helps those who are in greater difficulty.

- **teacher support**: teachers can understand and investigate the student's level before the lesson in classroom. The on-line activity allows to highline them, first of the lesson, not in classroom. This allows to prepare more effective lessons: discussing, first, arguments non clear for students. Moreover teachers could use forums as an on-line tutor to support students also outside the class.

Technology will improve, in the future, their importance in our life. **We don't prepare our student for today but we prepare our students for tomorrow. So technology has to be integrated to** what students do in school with the guide of teachers: this process helps them in their future work and in their future life. If, this integrated process, starts at school, it could have better positive effects: could produce better long life learners that consciously use technologies, not only for access to information and consumption of it; but also to produce personalized contents, to share them and collaborate with others.

Technology could improve a **personalized learning**: students could use and reflect to materials shared or created from teachers, using different media, and collected in virtual class environments; they could review them in different and asynchronous time, they could easily personalize materials adding notes, links, voice. They could learn with more flexibility and they could reflect, in differentiated ways, about what they produce.

The use of technology is also a **solution to include**: teacher can positively use peer collaboration in classrooms, could have feedback also for this activity that sometimes isn't so visible during normal class activities. Moreover students could easily work in small groups and collaborate inside and outside school in fluent ways.

References

[1] "Scratch" Tool MIT Media Lab Cambridge: http://wiki.scratch.mit.edu/wiki/Scratch_2.0;

[2] "Code Org" Website: https://code.org/;

[3] "Moodle" Website and Community: https://moodle.org/?lang=en;

[4] "TeamUp" on-line Tool: http://teamup.aalto.fi. The manual for the use is visible on the link: http://teamup.aalto.fi/TeamUp-Manual.pdf;

[5] "Google Apps" Tool: http://www.google.it/intl/it/about/products/

[6] "Diagram Designer" on-line Tool: http://meesoft.logicnet.dk/DiagramDesigner/

[7] "Rise" Sw http://www.risetobloome.com/Page_1_S_NodeListing.aspx?ITEM=1404;

[8] "Padlet" on-line Tool: https://it.padlet.com

[9] MOOC course: http://www.europeanschoolnetacademy.eu/en/web//tablets-in-schools;

[10] "CMap" Sw: http://cmap.ihmc.us/

[11] "Prezi" on-line Tool: https://prezi.com/

[12] "Socrative" on-line Tool: http://www.socrative.com;

[13] "Learning Designer" on-line Tool: http://learningdesigner.org/

Active learning in a “Introduction to networks” course

Sophia Danesino
Istituto di Istruzione Superiore “Giuseppe Peano”
Torino, Italy
sophia.danesino@peano.it

Abstract. There is significant evidence that profound learning occurs when students are involved in creating, sharing, interacting and explaining. Traditional lessons tend to be particularly ineffective in some types of teaching. Telling students how devices work or describing the process of networking transmission can be boring, difficult to remember, even hard to understand. It is necessary to find new approaches in order to involve students in the learning process so to increase their engagement and make them more motivated and autonomous. Starting from this assumption, two innovating teaching methods have been experimented at Peano Institute in Torino, Italy, with encouraging results. In this work, we describe the way we used to increase the amount of active learning in our “Introduction to networks” course of the third-year curriculum. After introducing these techniques in 2013 we observed an increase of interest and participation in our students. This result suggests that the adoption of active learning pedagogies can contribute to increase personal work and improve the learning process.

1. Background

I.I.S. “G.Peano” (*http://iispeano.gov.it*) is a technical high school that offers a five-year course both in “Computer Science” and in “Electronics” (age of pupils: 14-19). It is located in the peripheral area of Torino, Italy, a highly industrialized city. The qualification combines academic study with work-based learning. The school is designed to equip students for a particular area of work, as well as giving them the general skills that are useful in any type of job. Upon completion of Higher National Diploma students can progress to university to complete a degree programme.
Enrollments have reached during the last ten years an average of 800 students per year. Most of them come from poor families and the percentage of immigrants is quite high. Such an unfavorable social background causes a high rate of dropouts. Consequently it becomes imperative to motivate students, to increase their interest in Computer Science and to improve their knowledge of the English language so to avoid dropouts and facilitate access to employment.
As active learning methodologies have demonstrated to be effective in terms of learning outcomes [1], we decided to experiment teaching strategies and technological resources that engage students in their learning process. Two apps have been chosen as particularly fit to encourage students to analyze and explain concepts: Popcorn Maker and Augmented Reality.

The first experiment involved two classes in their third year of high school (3AI 25 students and 3BI 29 students) in April-May 2013. The second one involved one class of 24 students in their fourth year in November-December 2014.

2. Popcorn Maker

Popcorn Maker was selected as a basic tool in the first experiment because of its specific features. Traditional video editors produce what can be described as *flat* video [2]: each frame is predetermined by the creator and the viewer has no other way to experience it. Popcorn Maker transforms the passive act of watching a video online into an active experience where the video makes connections across a variety of platforms [3].   In other words Popcorn Maker produces multimedia that can be influenced by data on the web.

Using a simple drag-and-drop interface (*https://popcorn.webmaker.org/*) users can grab live content like Twitter feeds, Google maps, photos, news feeds and links directly into an existing video and audio file producing multimedia that can be influenced by the actions of the viewer.

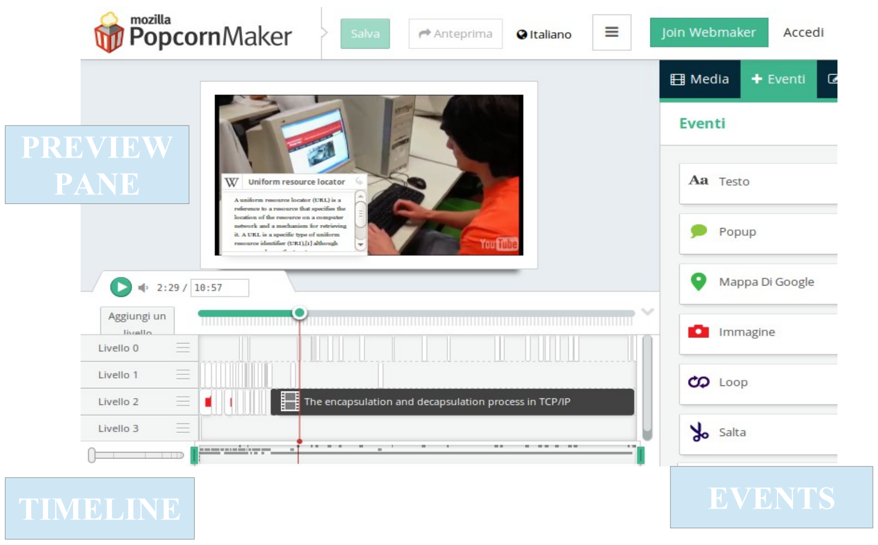


Figure 1 – Wikipedia information added to a video.

Structurally speaking, *Popcorn Maker* is divided into three sections (*Figure 1*): the preview pane, the timeline, and the feature list. In the center of the screen is the preview pane, where users may view the videos they have modified. Below that screen is the timeline, which displays the video as a linear timeline, punctuated by "events" (e.g. text or pop-up bubble) layered on top of the original video. To modify the videos, users drag and drop the events listed on the right of the screen into layers in the timeline [4].

The events users can add to videos include simple text overlay that can be modified with different fonts, sizes, colors, shadow, background color and link, as well as text overlay within a bubble and featuring an icon (e.g. an exclamation point). Both the shape and icon may be changed as shown in *Figure 2*.

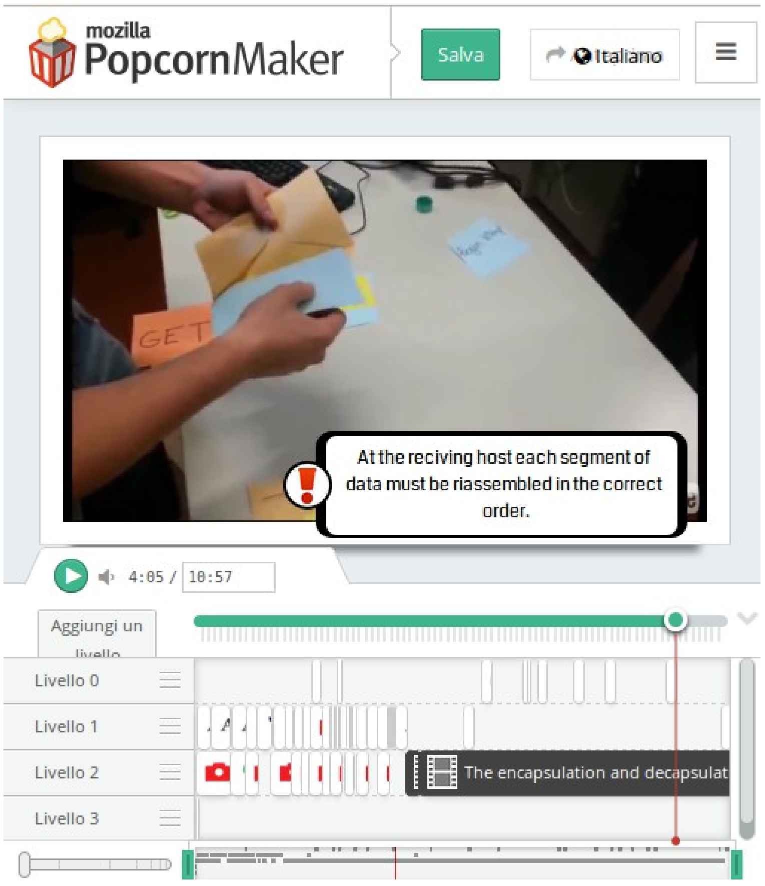
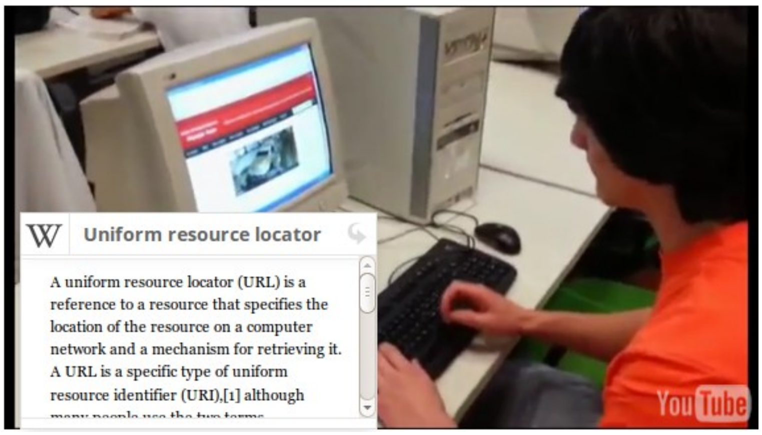

mozilla PopcornMaker
Salva
Italiano
At the reciving host each segment of data must be riassembled in the correct order.
4:05 / 10:57
Aggiungi un livello
Livello 0
Livello 1
Livello 2
Livello 3
The encapsulation and decapsulat

Figure 2 – Popup text added to the video.

Adding overlays means analyzing the content given: identifying parts, analysing relationships between parts, recognizing the organizational principles involved, classifying, ordering. Students are required to interpret information, not to simply recall information: learning is achieved through students selecting relevant information and interpreting it through their existing knowledge.

Once a project is finished and saved, it can be shared as a *Popcorn Maker* URL, exported as an HTML source file, or embedded in another webpage using provided embed code. *Figure 3* is an example of an augmented video with clickable interactive widgets: if you click the small window you can access the related information on Wikipedia website.



A uniform resource locator (URL) is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it. A URL is a specific type of uniform resource identifier (URI),[1] although

Figure 3 – augmented video.

Popcorn Maker is free. It gives schools, like other users/students, the freedom to copy and redistribute the software, with no obligation to pay for doing so (moreover Popcorn Maker allows the user to create his own version of Popcorn Maker by forking the code on Github) . Our students can keep working at home for free and this is important because we do not want our students to illegally install proprietary software on their PC. "*The most fundamental task of schools is to teach good citizenship, including the habit of helping others. In the area of computing, this means teaching people to share software"* [5].

This online tool was used in two classes of 25 and 29 students attending the third year. They worked at first on the TCP/IP protocol suite: working in groups of four/fine students they broke down the problems involved in moving data from one computer to another one and categorized these problems to four groups (the four layers in TCP/IP), then they identified the different addresses needed at each layer (MAC, IP, port number, DNS name). Afterwards, in greater detail, on the encapsulation/decapsulation process.

Stage 1 – Introduction

At the beginning the teacher tried to explain the process of encapsulation using the analogy of a packaging process but it didn't work. The story "The content is put inside bags, the bags are packed into boxes and the boxes are put inside cars. The bags, boxes and cars normally will contain information that are needed for the delivery" was boring and, taken out of context, had little meaning. So students were encouraged to simulate the process of getting a web page from the web.

Stage 2 – Simulation

Each student played a different role (i.e. layer) from the data-link layer to the application layer. They physically added control data to each data unit and encapsulated them into a new one (*Figure 4*).

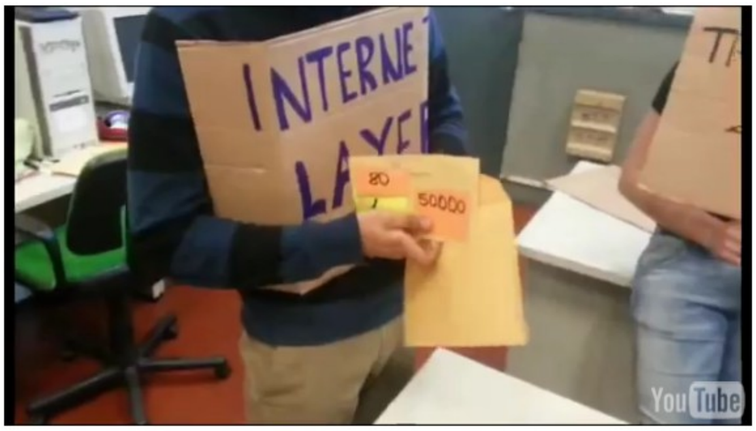


Figure 4 – Simulating the encapsulation process.

In the preparation phase of this activity students were divided into small groups (of four to five students) and each group was assigned a single step in the sending/receiving process to explore. Each group discussed the plan for its virtual play and prepared the envelopes and the information needed (ARP table, Routing table, DNS mappings...). Each group had to interact with the others in order to get a consistent information of the whole network and on the whole process being organised.

Stage 3 – Video recording

The video has been recorded in English as it was part of a Content and Language Integrated Learning (CLIL) lesson: students were learning networking (content) through the medium of English (a foreign language) and English by studying a content-based subject.

At the end of this step a final video was recorded (*https://youtu.be/_dU4bonmyeE)*. The results were quite impressive. Not only our students had been able to use a foreign language when speaking, but they also showed a remarkable creativity. Creativity is a critical component in enabling us to use innovative powers and a key resource in a knowledge-driven economy.

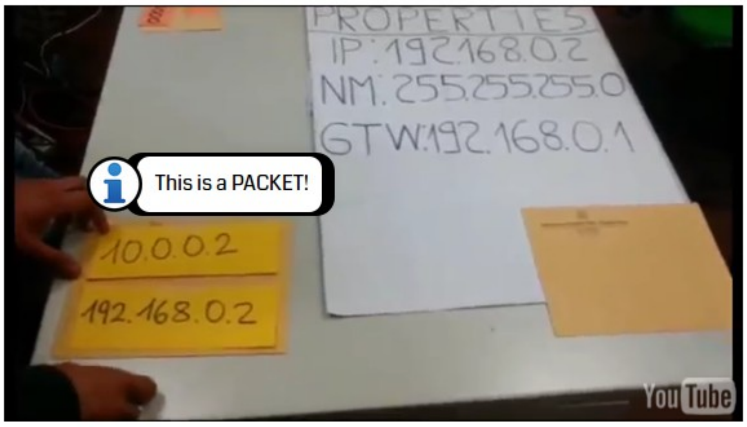


Figure 5 – A network layer data unit.

Stage 4 – Review

The last phase of the experiment included a full review of the entire process. The students were asked to explain the process of encapsulation/decapsulation improving the video previously made in the classroom using Popcorn Maker. Using its drag-and-drop interface they were asked to:

i. add hyperlinks for the viewer to click and get information on the web (i.e. related articles from Wikipedia, such as detailed information about a specific protocol layer as shown in *Figure 3*)

ii. add popups to the video describing the data unit shown (such as packets, segments, etc.) as in the following figure:

iii. insert images showing detailed schema of a video section, for example the IPv4 address structure as in the following figure:

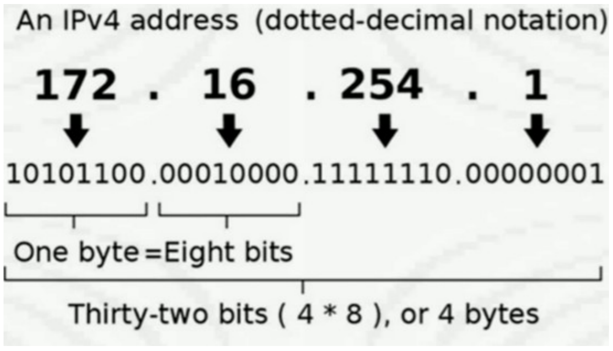

An IPv4 address (dotted-decimal notation)
172 . 16 . 254 . 1
10101100.00010000.11111110.00000001
One byte=Eight bits
Thirty-two bits (4 * 8), or 4 bytes

Figure 6 – IPv4 address schema.

This activity improved the overall quality of learning as students had ample opportunities to clarify, question, apply, compare and consolidate their knowledge.

The final work (*https://danesino.makes.org/popcorn/35th*) has been embedded into the school e-learning website (a Moodle learning environment, login as Guest): *http://informatica.peano.it/mod/page/view.php?id=1724*

A full analysis of the learning results is not possibile yet. Appreciable results have certainly been obtained in increasing interest and participation, All students, without any exception, have given a personal contribution. Improvements of proficiency levels and reduction of dropout rate will be assessed in the future. The school is planning to repeat the test next year.

3. Augmented Reality

The second active learning experience conserns Augmented Reality, i.e. the ability to see or hear contextually relevant information superimposed on what you see through the camera lens of an Android or iOS device such as a phone or mobile device. A classic example of AR is the fighter pilot's heads up display: data such as altitude, speed and fuel level appear on a transparent visor worn by the pilot, so that he does not need to look down at a control panel.

Augmented Reality (AR) allows educators and students to create layers of digital information on top of the physical world. It can be successfully used to make videos and apply them on top of trigger images such as book covers, photos, lab equipment. The information delivered increases engagement and enjoyment: it makes the difference between an ordinary lecture and a learning experience.

Augmented Reality not only can change the environment around students, it also encourages them to construct their own learning worlds. Students can place AR markers throughout the classroom. When markers are scanned, they trigger student-made videos, slideshows or images.

To implement a project in the field of Augmented Reality, Aurasma (*http://www.aurasma.com/*) was chosen as the most appropriate digital tool. Aurasma is a simple and immediate app that allows users to engage in and create Augmented Reality experiences.

At Peano Institute Aurasma has been used in several different ways in a classroom of 24 students attending the forth year. During an initial lesson on networking devices, such as switches and router, students were asked to think of a brief way to describe each device function.

i. Each student worked with a partner and each group was given the image of a specific device (such as the one in *Figure 7*). That image became their trigger image (a trigger is an image that activates media when scanned by an AR-enabled device).

ii. In order to define and describe it, some students drew an image containing text information (for example a description of a switch/router interfaces and physical ports were added to the device image), others recorded an audio description (for example a brief description of each layer were added to an image of the TCP/IP protocol suite).

iii. After that, the students opened the Aurasma app, uploaded the image they were given and added the text/audio as an overlay.

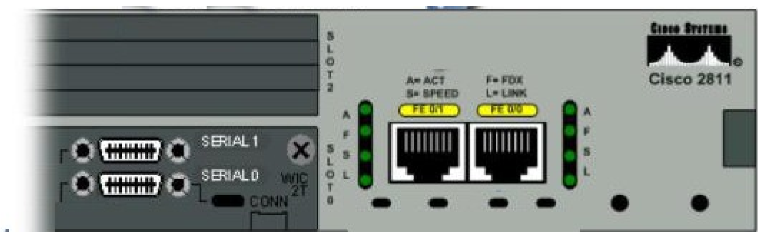


Figure 7 – Trigger image, example.

All images were stuck to the classroom walls (our public auras can be downloaded searching "peano" on Aurasma). As a result, any time a student wants to know more about a device, he can hover his tablet/smartphone above the image on the wall. The text/audio overlay, created by the students, will automatically begin playing (*Figure 8*).

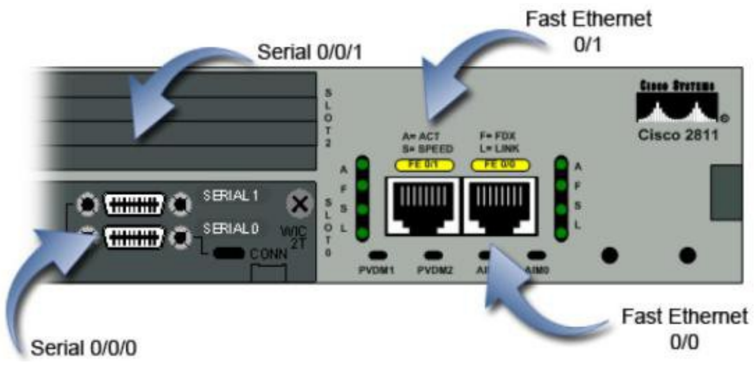


Figure 8 – Text overlay, example.

The experiment offered students a chance of engaging in personal work, made them less dependent on textbooks and predetermined ways of thinking and working.

The experiment will be repeated next year so to allow a better assessment of results, based on quantitative data. At present the outcome of the experiment is considered quite satisfactory in terms of student participation and interest.

4. Conclusion

The benefits of the activities described above are relevant. They include improved critical thinking skills, increased retention and transfer of new information, increased motivation and improved interpersonal skills. Nevertheless, there are a few observations related to the learning process that must be underlined.

Adopting active learning methodologies is a time-consuming process:

i. it requires a significant effort by the instructor to develop a plan for an active learning program

ii. students need a lot of class time to complete the activities (we doubled the class time in the second experiment and even more in the first one).

Although active learning requires a large initial time investment, we realized that our students have a better understanding of what they learnt, so that learning new information will become an easier and quicker process [6].

Another element must be taken into account: active learning can be challenging because of class size or room limitations such as fixed seating. So we suggest using a laboratory or a flexible environment as a classroom. A fast Internet connectivity is also mandatory.

Using active learning does not mean abandoning the lecture format, but incorporate active learning into our lessons can transform our classroom into an exciting, dynamic learning environment.

The described activities allow a twofold approach to digital competences. Not only students discover and practice digital tools while carrying out a task, but they also explore and explain the related contents. To properly learn something nothing better than having to teach it and this works also for young students.

References

[1] Michel Prince, *Does Active Learning Work? A Review of the Research*, Journal of Engineering Education, July 2004

[2] Mozilla support, *How is Popcorn Maker different from other video editors?*

https://support.mozilla.org/en-US/kb/how-is-popcorn-different

[3] Remake learning, *Popcorn Maker - Video beyond the box*

http://remakelearning.org/resource/popcorn-maker/

[4] Educade, *3 Lesson Plans Using MOZILLA POPCORN MAKER*

http://educade.org/teaching_tools/mozilla-popcorn-maker

[5] R.Stallman, *Why Schools Should Exclusively Use Free Software*, Free Software Fundation

[6] Mick Healey, Eric Pawson, Michael Solem, *Active Learning and Student Engagement: International Perspectives and practices in Geography in higher education*, Routledge 2009

Introducing recursion with LOGO in upper primary school

Ágnes Erdősné Németh

Batthyány Lajos Gimnázium, Nagykanizsa, Hungary
ELTE Faculty of Informatics, PhD School, Budapest, Hungary

erdosne@blg.hu

Abstract. Recursion is a powerful concept, but most of the students and teachers agree that it is difficult to learn, understand, and teach it. On the other hand, Logo is a powerful language that allows for explorations of recursion via visualization. This article demonstrates a new way of teaching and understanding recursion for the upper primary students with the help of Logo. The structure we used fits to the computational thinking approach, it helps to understand time and memory limits of computers, too. Understanding the key concepts of recursion prepares students for making sense of other types of programming concepts like memoization and dynamic programming and other more complex concepts.

Keywords: recursion, LOGO, primary school education, computer science, trees, computational thinking, fractals, L-systems

1 Introduction

Recursion is a key concept in the Computer Science field. It is a powerful tool for solving specific programming tasks and has features that sometimes make it a superior choice over other approaches. [2] Students learn recursion during their first programming course and learn its concept in math too. It is deep, rich, and interdisciplinary, but at the same time abstract, vague and difficult to explain or define [3]. Many agree that, while it is powerful and significant, it is difficult to learn and understand [4]. There seems to be an overall consensus on the difficulty of teaching it at all school levels [8], too.

On the other hand, Logo is a powerful language that allows students to explore recursion via visualization. This article demonstrates a new way of teaching recursion to the 11 and 12-year-old children with LOGO, on the basis of computational thinking. Visualization, associations with everyday life, and the variety of different samples might help understanding, encourage motivation, aid conceptualization, and give a strong basis for subsequent formal studies.

Recursion in programming textbooks often exists as the quintessential triumvirate: Towers of Hanoi, Factorial and Fibonacci. [10] These tasks are powerful, but they are not expressive enough for primaries, because children in primary schools need visualization to understand any idea. Most of the textbooks begin with iteration examples instead of real recursive tasks.

There is a noteable textbook teaching recursion with Logo by Hromkowitz [1], which begins with a simple definition: "Recursion is the process of repeating items in a self-similar way." His first example is an unlimited recursion without parameters which does nothing because the recursive call is the first step. The second example is drawing something before the recursive call. The next ones are variations of spiral with different angles and lengths with incrementing parameters infinitely. Just after this he shows the STOP command with a condition. After that he analyzes the depth of calls. These theoretical examples are followed by the traditional examples: nesting brackets, Koch-curve, Sierpinski-triangle, trees and snowflakes. This structure is suitable for adults because of its theoretical characteristic and not for children without taking advantage of the visualization power of Logo.

There is a remarkable Hungarian textbook teaching recursion with Logo [11], which introduces recursion with spirals - instead of using iteration for drawing spirals, which is more suitable for children with their previous experiences. Next step is implementing fractals. Their introduction of recursion procedure comes without explanation; the children have to learn it from samples: Cantor-dust, Cantor-set, Koch-curve, Koch-snowflakes, Sierpinski-triangle, Sierpinski-carpet and Sierpinski-curve. After learning all of them from samples, the Peano-curve and optional creative fractals are given as homework. There is no awareness of the foundation of recursion but it is a good choice for learning it alone.

In our paper we suggest a more conscious setup of teaching recursion while keeping the computational thinking approach in mind. Based on our experience it is comprehensible for all students at the age of 11-12 and develops their thinking methods. Using Logo for teaching recursion is a natural concept. In the drawings it is very natural to explore the drawing itself, at a very early age.

2 Approaches on teaching recursion with Logo

For introducing recursion we have used a number of principles and approaches from the field of informatics and mathematics. Some prerequisite knowledge was mastered before introducing recursion in Logo, such as: basic commands for drawing a picture in Logo (forward, back, left, right, penup, pendown, pencolor, fill), flow of control in a procedure, iteration (for and while), conditional, interaction between procedures; local variables (only the knowledge of parameters is needed) and scope of variables, triangles, polygons, the basics of geometry. The children had to write procedures and to use the development environment.

3 Discovering the recursive step

The first task is to familiarize with recursion via binary trees (Fig. 1.). This example comes from everyday life; the children imagine real trees growing from year to year.

We think that the best way to explain the growing tree problem is with a ready-made program.

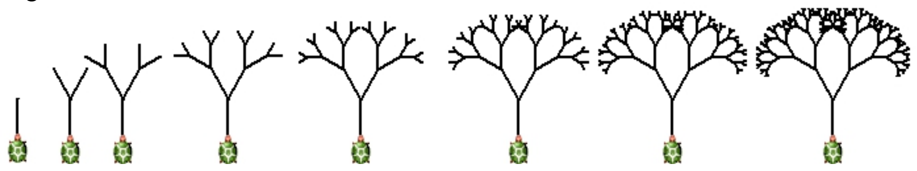


Fig. 1. Growing binary tree (tree 1..8)

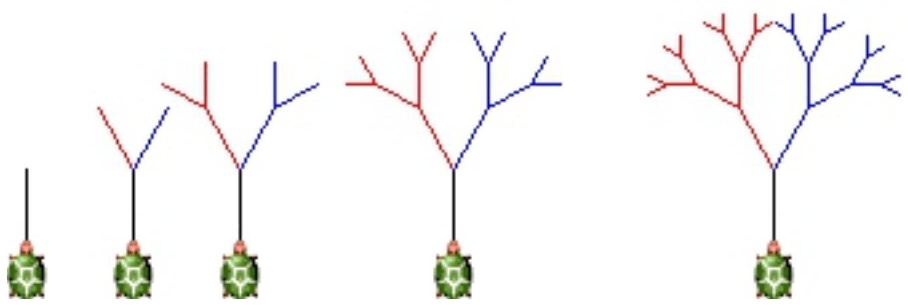


Fig. 2. Binary tree with colors (tree 1..5)

The above figure (Fig. 2.) is just for explaining the method. The goal of the presentation of above problem is explained by children: “The red part of the tree is the same as the one year younger whole tree. The blue part is a one-year younger tree too, just it erupts in another direction but it is in the same position as the red one. We could draw a line, turn left, draw a one-year-younger tree, turn right, draw a one-year-younger tree, turn left and go back to the initial position of the turtle.”

The children must articulate the strictly decreasing condition and the transparent status of the turtle with their own simple sentences. They have to decompose a recursive pattern into two parts, one of which is a smaller version of the original pattern. This means:

```
draw a pattern
  draw the base element
  move/turn
  draw a smaller pattern
```

After pupils have recognized, that the whole recursive procedure can be formulated by words, they could implement it.

From a computational thinking approach, visualization helps to express children’s thoughts about recursive pattern and helps them be familiar with this complicated idea by seeing this simple explanation.

4 Initial state

In the first example the initial state of recursion was very simple, a single line. It could be drawn simply in a recursive step. The next task is about the first state of the turtle in recursion, the initial step.

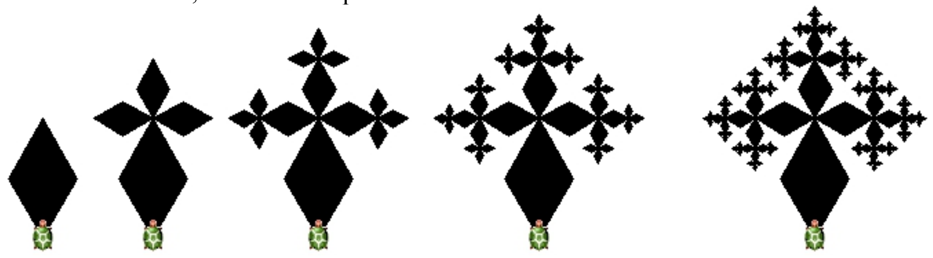


Fig. 3. Growing diamond (diamond 1..5)

In this example a diamond must be drawn as the first, initial stage, and the teacher must elicit that there must be a quitting condition: "when the recursion level turns to the first stage, we must draw an initial pattern, a diamond."

From the syntactic approach they have to recognize and formulate a template for recursive procedures:

```
procedure_name
  initial state
  recursive case
```

From a computational thinking approach abstraction helps to express pupil's thoughts about initial state.

5 Levels

The stage of the recursion call is always known throughout the process. In the next two examples we can use this information and we can make changes depending on the stage. In the next picture (Fig. 4.) there is an example where the width of a bough depends on the age of a bough.

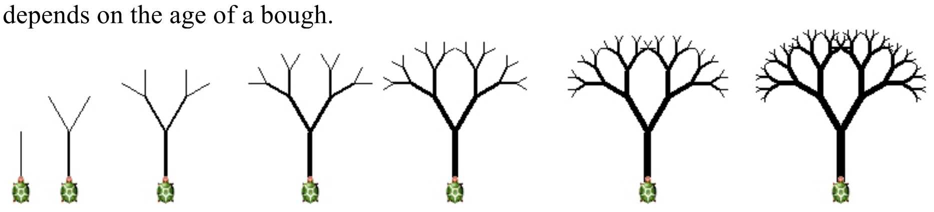


Fig. 4. Narrowing binary tree (narrow 1..7)

In every step we could write out the level information and observe the status of the stack. In the next picture (Fig. 5.) the color of a bough depends on the age, the stage of recursion, too.

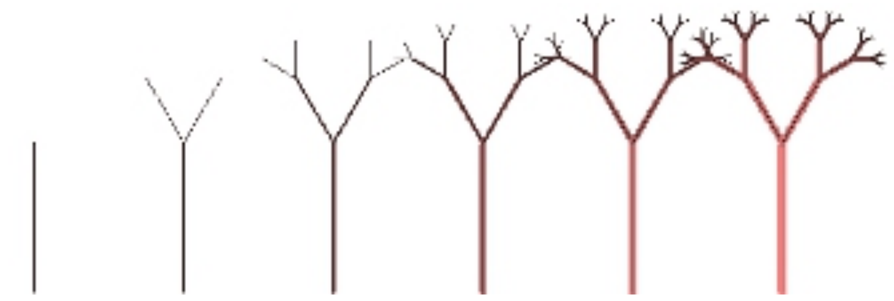


Fig. 5. Coloring binary tree (color 1..6)

In further studies - when they already have learned to use lists - there are a lot of creative opportunities to use this nature of recursion, that is, that we know the stage of the stack. There are tasks collections [11] to choose tasks for practicing this powerful and challenging topic. At advanced level it can be mixed with iterations and lists.

6 State transparency

Next step is exploring state transparency in recursive procedures. In every procedure there is a hard question: What is the turtle's position and direction before and after the next occurrence of recursive step? In the previous examples we made a starting position clear with drawing a turtle. In the next examples moving between recursive steps is not so easy, but exploring the drawing itself in a natural way makes it spectacular for children. Seeing a lot of examples children understand the importance of state transparency.

In the following examples (Fig. 6.; Fig. 7.) the children have to find first the position and direction of a self-similar drawing and the initial state of the picture, after that they can implement the whole procedure.

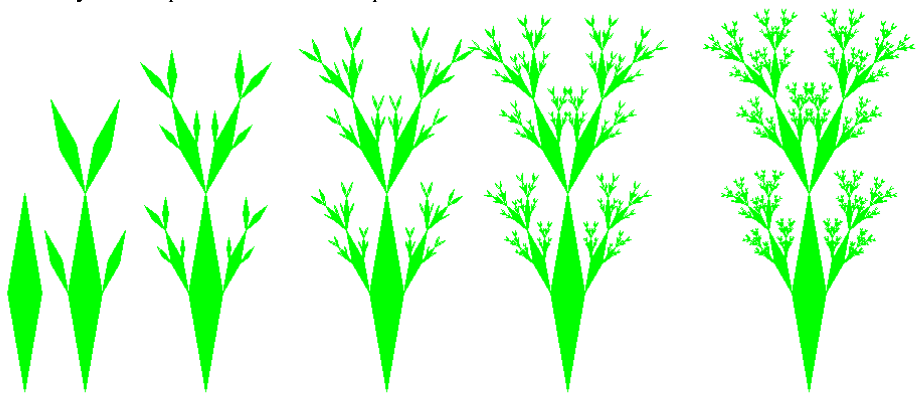


Fig. 6. Growing cactus (cactus 1..6)

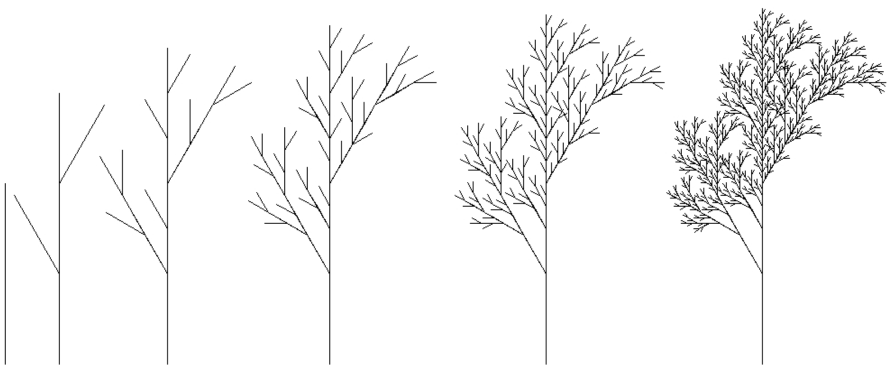


Fig. 7. Branching structure (branch 1..6)

7 Limit of data structures

There are two classical recursion problems to illustrate the limits of the recursion calls. The first one is the Sierpinski-triangle (Fig. 8.).

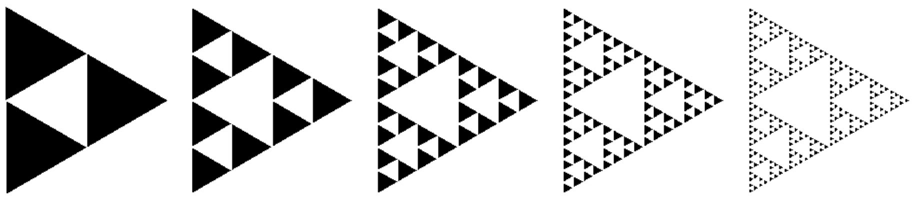


Fig. 8. Sierpinski-triangle (sier 1..5)

The second one is the Sierpinski-carpet (Fig. 9.).

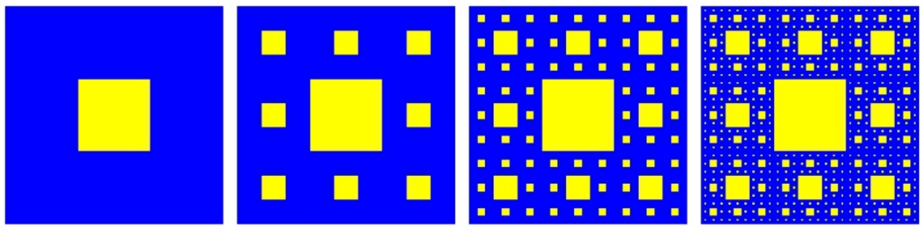


Fig. 9. Sierpinski-carpet (siercarpet 1..4)

These examples (Fig. 8. and Fig. 9.) visualize the problem of the finite property of data structures. We could not see any changes on the figure after the step (5-6), because the figure is too small. In the step (i) the area and side's length of triangle and square are lower than a pixel, so the Logo cannot display it. The computation goes on, but we can see nothing more on the picture, just only that the turtle turns very quickly. The number of the stage, where the drawing disappears can be counted easily ($1/i^n$).

From the computational thinking approach there is a impressive opportunity for visualizing the limitation of the computer data structures. In further studies the students will learn about the limits of data structures, such as integer, long integer, real, string, and they will be able to recall these examples.

8 Time costs

The next classical problem is the Koch-curve (Fig. 10.). This is a very good example for illustrating the exponential increasing running time. The students are very impatient and they think that computers are fast enough to compute everything in a flash. The first iterations are very fast, but from the 6th level it will be slower and slower. They can see the turtle turn and turn very fast and draw nothing, just compute the decreasing lengths which are lower than one pixel.

It is a good experiment to measure the time of making these pictures. Surprisingly, they will experience an exponential running time of the number of recursive steps.

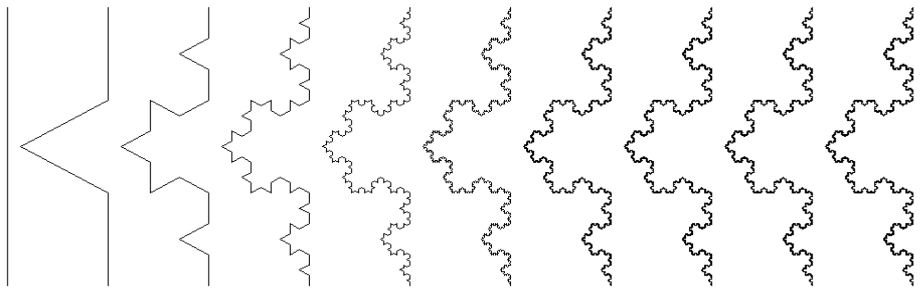


Fig. 10. Koch-curve (koch 1..10)

This trapeze task (Fig. 11.) is good for experiencing again both the data structure limitation and running time limitation.

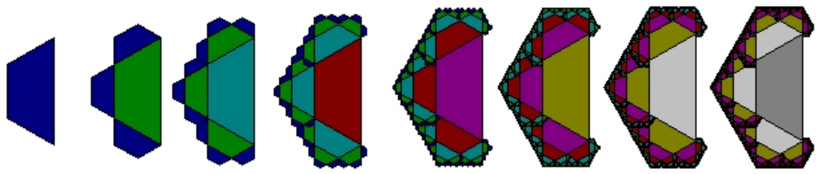


Fig. 11. Trapeze (trapeze 1..8)

From the computational thinking approach, the experience, namely, that the computation takes time, is a surprising discovery for students.

9 One recursive procedure calls another recursive procedure, which calls back the first one

The next cognitive level of recursion is when the procedure calls another one and it calls back to the first one. We think this concept is easy to understand with the visualization of Logo.

Children must carefully examine these pictures (Fig. 12.) to discover that once the triangle is right-shifted and in another time it is left-shifted. In a right-shifted triangle are left-shifted ones and vice-versa. We have to write two procedures which call each other.

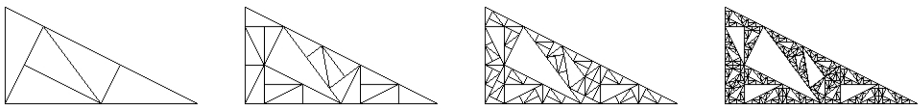


Fig. 12. Triangle (triangle 1..4)

In the next example (Fig. 13.) we have to decide on what the starting stage of the recursion is. This is one more parameter to think of at the initial stage. Using two procedures calling each other is a very hard type of recursion.

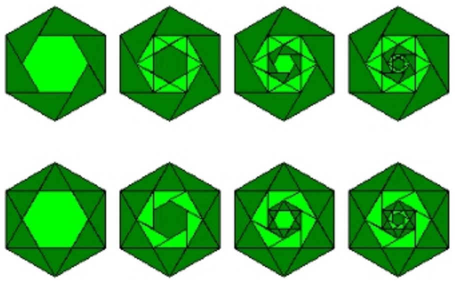


Fig. 13. Hexagons (hexagon 1..4 1, hexagon 1..4 2)

10 Conclusions

In our structure the concepts of recursion are drawing up consciously, simply, step-by-step. These examples are visual, easy to understand and raise awareness of the foundation of recursion. Children need a lot of tasks to learn and to practice the whole concept [13, 14, 15, 16, and 17].

Children have to predicate the basics with their own simple words. The visualization, with LOGO, helps them to do this. They learn about decomposing a recursive pattern into two parts, one of which is an original pattern, the other is a smaller version of the original pattern. They describe a template for recursive procedures and they understand how recursion works. The necessity of terminating the work of a recursive procedure is revealed, and correspondingly a termination rule has to be included in the procedure.

They learn about important concepts from computational thinking too, like the time of computation, the memory limit, the limit of data structures and the stack.

If they familiarize recursion with the visualization of LOGO, then later they will understand easier the top-down approach, the memoization and the bottom-up approach of dynamic programming, too.

The source codes of the discussed examples are available at www.microprof.hu/recursion.zip.

References

1. Juraj Hromkovic: Einführung in die Programmierung mit Logo: Lehrbuch Informatik, In: Vieweg+Taubner | GWv Fachverlage GmbH, Wiesbaden, 2010
2. Dr. Ivan Kalas & Dr. Andrej Blaho: I Beg Your Pardon Turtles: Don't Forget About Data Structures, In: Eurologo'97 Proceedings, Budapest, Hungary
3. Dalit Levy: Classification, Discussion, Recursion: Helping the Development of Computer-Science Concepts, In: Eurologo'97 Proceedings, Budapest, Hungary
4. Leron U.: What makes recursion hard, In: Proceedings of the Sixth International Congress on Mathematics Education (ICME6), 1988, Budapest, Hungary
5. John Murnane: To iterate or to recurse? In: Computers & Education, Volume 19, Issue 4, November 1992, Pages 387–394
6. M.C. Er: On the complexity of recursion in problem-solving, In: International Journal of Man-Machine Studies, Volume 20, Issue 6, June 1984, Pages 537–544
7. Derek Wilcocks, Ian Sanders: Animating recursion as an aid to instruction, In: Computers & Education, Volume 23, Issue 3, November 1994, Pages 221–226
8. Dalit Levy: Collaborative Conceptual Change: The Case of Recursion, In: Journal of Intelligent Systems 01/2002; 12(2)
9. Dr. John Close, Dr. Darina Dicheva: Misconceptions in Recursion: Diagnostic Teaching, In: Eurologo'97 Proceedings, Budapest, Hungary
10. Michael A. Wirth: The far side of recursion, In: Teaching mathematics and computer science, 2015/1 , pp 57-71
11. Viktória Heizlerné Bakonyi, László Zsakó: Strategy of guessing exercises – Variations of drawing trees, In: Proceedings of the 9th International Conference on Applied Informatics Eger, Hungary, 2014. Vol. 1. pp. 285–294
12. J. M. Wing: Computational thinking, In: Communications of the ACM, 49(3), p. 33-35, 2006. https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf
13. Rónai Orsolya Renáta, Vöröss Veronika: Lógós ecsetvonások, NJSZT 2008, http://matchsz.inf.elte.hu/logosecsetvonasok/ > lecke6.html & lecke7.html
14. LOGO versenyfeladatok tára 1998-2002, Ed.: Mészáros Tamásné, Zsakó László, NJSZT, 2002, http://logo.inf.elte.hu/peldatar/LogoPeldatar1998_2002.pdf
15. LOGO versenyfeladatok tára 2003-2007, Ed.: Heizlerné Bakonyi Viktória, Zsakó László, NJSZT, 2008, http://logo.inf.elte.hu/peldatar/Logo2003_2008.pdf
16. LOGO versenyfeladatok tára 2008-2012, Ed.: Heizlerné Bakonyi Viktória, Zsakó László, NJSZT, 2013, http://logo.inf.elte.hu/peldatar/LogoPeldatar2008_2012.pdf
17. NJSZT Logo Országos Számítástechnikai Tanulmányi Verseny archívum, http://logo.inf.elte.hu/logo_archivum.html

Experiences of the T4T group in primary schools

Fabrizio Ferrari[1], Alessandro Rabbone[2] and Sandro Ruggiero[3]

[1] Primary school De Amicis - IC Regio Parco
Torino, Italia
f.ferrari@to7120.com

[2] Primary school R. D'Azeglio – DD R. D'Azeglio
Torino, Italia
alessandro@rabbone.it

[3] Primary school N. Tommaseo – IC Tommaseo
Torino, Italia
sandro.ruggiero@gmail.com

Abstract. In this paper we describe the experiences carried out in three different primary schools during several years with pupils in their third, fourth and fifth grades (i.e. with pupils from 7 to 10 or 11 years old). The focus is on programming because we consider it a new tool for pupils to express their creativity while they are learning fundamental elements of computer science. Obviously suitable program development environments must be used, for example Scratch that is our choice for introducing programming. Our teacher experiences are focused on finding contributions to defining an interesting, affordable and sustainable school curriculum for CS in primary and lower secondary school. Such curriculum should introduce computing respecting the pedagogical achievements that have been identified by educators in the decades for the different school grades, allowing pupils to perform new kinds of activities also to the benefit of those longtime recognized achievements.

Keywords: Primary school education, computer science, Scratch

1 Introduction

We would like to draw the readers' attention to some pupils experiences about coding and computational thinking carried out in some primary schools in Turin, Italy.

We are a group of teachers interested in working on Computational Thinking and Computer Science (CS) who met during the T4T (Teachers For Teachers) seminars, organized yearly by the Informatics Department of the University of Turin. One of the aims of T4T is to introduce programming environments as new expressive tools to pupils and students of all ages and meanwhile teach them fundamental concepts of Computer Science [b].

Our teacher experiences were, and continue to be focused on, finding some interesting, affordable and sustainable suggestions for a school curriculum on CS in primary and lower secondary school. For us a proposal (and in the end a curriculum) is sustainable when it introduces new concepts but also it is respectful of the pedagogical achievements that have been identified by educators in the decades for the different school grades, allowing pupils to perform new kinds of activities also to the benefit of those longtime recognized achievements. Obviously it does not reduce to cross-disciplinary activities. Also, suggestions must be affordable for schools: this often means they should propose (almost) free activities.

In section 2 unplugged activities, yet introducing to computing, are described with their connections to the pedagogical achievements considered typical of the school-children age. In section 3 the computer based activities are described that have been experimented: they normally begin with scaffolded activities of easy-programming like using the Lightbot or Code.org and progress to use the Scratch development environment to implement stories, quiz or games. In section 4 the transition to computer programming in Scratch is presented also for its possible contributions to the linguistic abilities of the school-children particularly for their writing abilities. The conclusive section gathers some very preliminary reflections on the experiences.

2 Computer Science unplugged

Computer science unplugged activities are an important component of the experiences introducing to computing because of the obvious reason that they do not require the use of an intermediate device unknown to the school-children. Especially in the unplugged activities that require identification of their body and a android - robot, children seem to benefit greatly in terms of operational thinking. Another reason, more tangible, is that they normally are inexpensive activities and this is almost a mandatory requirement in many italian schools.

Searching on the internet and in schoolbooks for programming activities in primary schools we find three basic types of unplugged activities:

- Paper and pencil activities to move an android/robot. In the beginning commands are <u>independent</u> on both the person who gives the command and on the robot that has to perform the commands (typically the cardinal symbols N, E, S, W are used). Then commands are provided that require robot dependent movements such as Forward, Left, Backward, Right
- Boardgames using the same movement instruction as above, but in competition among players (i.e. Cody & Roby - http://codeweek.it/cody-roby/)
- Games on giant boards where pupils act as robot/android following school-mates commands.

We will focus on unplugged activities of the last kind. Unplugged activities are missing of an important element, present in every computer-based task: error checking. During unplugged activities you can’t have an immediate feedback telling you if you are right or wrong. We remember Cédric Villani recent words: “Coding, maybe, is the only activity where pupil can correct errors by themselves” (http://www.atelier.net/trends/articles/cedric-villani-programmation-seule-discipline-enfant-realise-auto-correction_436613).

In our experience we notice that the more pupils issuing commands identify themselves as the programmers, the more the robot/pupils follow to the letter what they have to do thus the activity turns out to be nearer to the computer based commands execution with its immediate errors visualization.

During our activities in the primary school D’Azeglio in Turin, pupils use the “natural” board shown in figure 1 that is a part of the school playground. It is made of 84 squares (about 80 cm for each side - 7 x 12) five of them occupied by a slide.

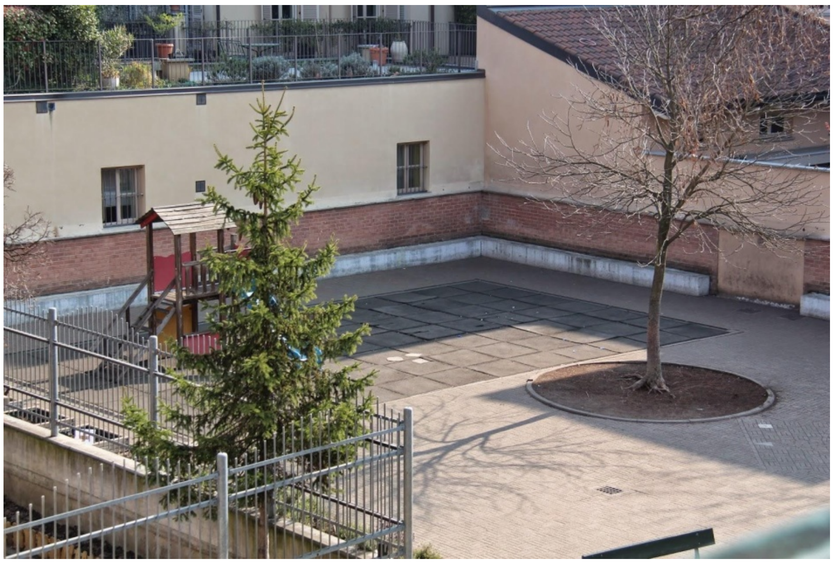
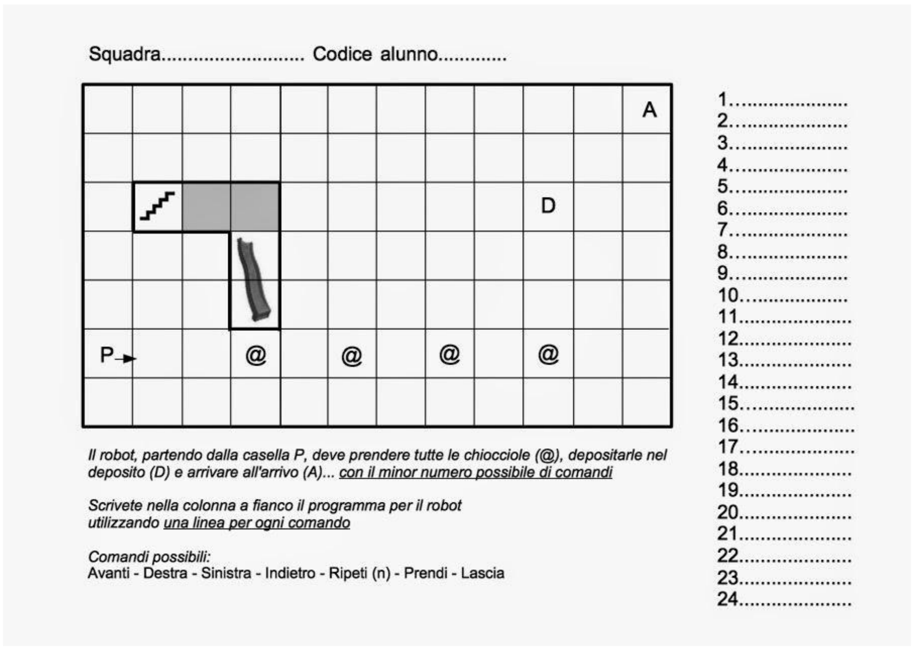


Fig. 1. The school playground

For the pupils of the 4th grade, the teacher prepared the following activities.

1. First a competitive role play as suggested by “Cody & Roby” (http://codeweek.it/cody-roby/duello/). One pupil acted as an Android/Robot and his/her mates gave her/him all the instructions in order to “catch” a second pupil acting as an enemy Android/Robot trying to enter in first pupil’s home-square. Pupils giving commands used only user dependent movement instructions F, R, L, B. This first exercise was useful for pupils to acquire confidence with the board and to learn to use a finished number of unambiguous instructions to communicate with the “Android/Robot”.
2. To movement instructions as in activity in point 1. other commands were added for: Repeat (n), grab, leave. The teacher prepared some sheets where the playground was drawn in every detail, see figure 2. Each sheet had growing difficulties tasks (similarly to what can be found in the Farmer activity in Code.org - https://studio.code.org/s/20-hour/stage/9/puzzle/4). On the right of

the sheet pupils could write their own code on numbered lines to reach the given target: one line, one command. For the “Repeat (n)” command pupils could draw on the left of the code a “(” sign from a line to another to precisely define which were the commands to repeat. Later on pupils shared their written code for the same task to discuss (and find) the best solution in order to solve the tasks with the less number of instructions. At the end pupils tried their code on the outdoor playground verifying their solutions.



Squadra.......................... Codice alunno............

Il robot, partendo dalla casella P, deve prendere tutte le chiocciole (@), depositarle nel deposito (D) e arrivare all'arrivo (A)... con il minor numero possibile di comandi

Scrivete nella colonna a fianco il programma per il robot utilizzando una linea per ogni comando

Comandi possibili:
Avanti - Destra - Sinistra - Indietro - Ripeti (n) - Prendi - Lascia

1....................
2....................
3....................
4....................
5....................
6....................
7....................
8....................
9....................
10..................
11..................
12..................
13..................
14..................
15..................
16..................
17..................
18..................
19..................
20..................
21..................
22..................
23..................
24..................

Fig. 2. The squared playground drawn on a paper

In figure 2. below the drawing of the squared playground there is the description of one of the activities suggested to the pupils: starting from P the robot must catch all the @ and bring them to point D, then the robot must go to A, with the lowest number of commands.

During unplugged activities schoolchildren are engaged in acting like they are robots moving according to a given sequence of instructions or like programmers deciding the sequence of instructions to be asked to a school-mate/robot to do a task, i.e. deciding the programs making the schoolmate/robot to do something. Soon pupils were ready to write down similar sequences of instructions making the computer to do something, that is ready to write programs for the computer with a suitable environment.

3 Computer-based programming

During the school years 2013-14 and 2014-15 the activity of computer programming (using both computer or other devices like iPad, tablet…) has been proposed in various ways and in different solutions, but all activities can briefly be classified as follows:

3.1 Puzzle solving (exercises on various websites)

We mean above all the courses offered by Code.org or by "Programma il futuro" (Italian version) in the framework of "One Hour of Code".

Still in our school labs the pupils have also experienced some puzzles with Light-Bot (http://lightbot.com/) and with Blockly-games' maze (https://blockly-games.appspot.com/maze?lang=it)

Some videos were initially screened on the interactive whiteboard and some examples puzzles were discussed and solved collectively. Later the children were invited to continue the path individually at home during the weekends or at school during the hours of the labs with the mates' help and the teacher's advice.

3.2 Using Scratch for the free development of personal projects

Usually the presentation of the Scratch environment has been very short and simple.

At the beginning of some sessions using the interactive whiteboard the teacher showed the students the development of a simple project that the children had to repeat step by step.

This "introduction" has never exceeded the length of 10 - 15 minutes.

Later the students were asked to develop their own original design or modify the example according to their taste or even to explore the Scratch website looking for projects to remix.

This 'free' activity has occupied most of the hours dedicated to labs. Cooperation among children, exchange and mutual support have been strongly encouraged. Even the use of a personal account in the context of the social environment of the site has been encouraged.

Lastly the printed and laminated Scratch Cards (https://scratch.mit.edu/help/cards/) were made available to students. Similarly some copies of the Getting Starter Guide (https://cdn.scratch.mit.edu/scratchr2/static/__90f8d4d8afbc51e3d823ca5efcc0ea53__/pdfs/help/Getting-Started-Guide-Scratch2.pdf)

3.3 Using Scratch to build a collective project

In our experience, Scratch has often been used as a tool for the implementation of projects related to curricular subjects or even when it was necessary to build something for a major event of the school life.

Children made animated backdrops for their theater performances, an interactive installation for Sciences exhibition (this one using a Makey Makey board - http://www.makeymakey.com/) and especially created animated stories with storytelling. Cappuccetto Rosso 2.0, which you can see in the appendix, is an excellent example of what we mean by "collaborative project"

Works on group projects has, however, involved the totality of pupils in classes.

The teacher, in this type of activity, has obviously taken on a more managerial role. He had to deal with the organization and coordination of teamwork, assigning modules or specific activities of the project (search for information, graphic design, implementation of sound or music, final assembly of the parts).

All above activities have been performed by schoolchildren 7-10 years old.

4 Coding strictness and written language teaching

Coding activity is very useful to introduce to the written language formalism, especially when pupils are foreign students and not native speakers. In a context of predisciplinary teaching, typical in primary school, our aim was to use the absolute stringency of the code (always verifiable through feedback of the computer) so that students would understand better the need of grammatical rules in written language.

During the school-year 2013/2014 we faced many pupils that aren't italian mother tongue even if they are italian native. In the primary school, teachers usually talk and read a lot to improve communication skills and vocabulary, but at the end the need of learning the written language rules arrives. Transition from oral language to written language is not so easy as it could seem at a first sight.

In the 5th grade of De Amicis school in Turin, we planned to use coding to make pupils easier to understand the need of rules for the written language. For the activities here described Scratch was the software for our pupils!

A similar intuition was already present in an educational robotics project described in [c] where pupils were using an Italian Logo-like language to program their Lego robots. In both cases teacher's target were:

- introduce pupils to a formal environment, i.e. an environment with strict rules pleasant and near to their way of thinking
- using of computers to improve motivation and to avoid the direct teacher-pupil interaction
- start using a new language, formally strict, but with immediate feedback
- developing comparative discussion about Scratch written language and italian written language.

Activities lasted an entire school year even if they were not so regular: sometime we worked more on Scratch; other times we worked more on written text, studying the syntax of the italian language.

Scratch tasks were not only focused on coding and developing problem solving skills, but the target was to learn about how strict can be a written language, whatever it is.

According to a choice of a non-directive teaching practice, children were invited to translate the code's commands in natural-language sentences (Italian).

The absence of any (traditional) interaction pupils-teacher but only pupils-pupils and pupils-computer strengthened this target.

The path was positive and the results were very good: pupils were motivated and the return on written language was encouraging. Every pupils understood that each written language needs fixed rules.

Differently from previous activities identified in paragraphs 2 and 3, this activity could not be included in a curriculum of CS, but it provides a good example of interdisciplinary operation.

5 Conclusions (very temporary)

The activities summarized here led our group to reflect on the differences in educational approach to the issues of coding, programming and Computational Thinking, in an attempt to start to define a possible curriculum for primary school.

It is immediately evident the large difference between an activity with the 'puzzles' and a 'free' activity using Scratch. In the first case, the learning goals are gradually fixed from the outside (more from the website, not so much by the teacher). In the second case it is the student who sets himself its objectives, if appropriately encouraged. This situation puts the child in more cognitive challenge: 'imagine' and plan a possible solution to his problem.

On the other hand the activity for preset sequences of concepts (sequence, iteration, conditions, functions, ...) offers the teacher the guarantee to offer, in a gradual way to difficulties, all the key concepts that a good curriculum for the primary school should provide.

All this means, therefore, that, at least from our point of view, it is not possible, anyway, decide once and for all, what is the best path. There are conditions, in everyday school life, quite variable from one situation to another. What may be advantageous in one, may not be so on another.

Certainly the experience of both types of activity at different times seems appropriate and advisable. For instance, we noticed that the children who had already experienced the path of Intro course Code.org, showed greater confidence and mastery in dealing with the planning of their Scratch projects.

Even unplugged activities can be integrated in a complementary way with other kinds of experience. The unplugged activities, especially in the initial stages of learning a concept - key, especially with younger children, can be an indispensable cognitive aid when, in live situations, you replicate the issues raised by digital puzzles.

[a] H. Abelson, A. Di Sessa, Turtle geometry, 1986

[b] T. Filippini and V.Vecchi (eds.) The one hundred languages of the children , Reggio Children Publisher, 1996/2005, 216 pagg., ISBN 978-88-87960-08-2, Reggio Emilia.

[c] G. Barbara Demo, T4T: a peer training model for in-service teachers, WiP-SCE 2015, Berlino

[d] G. Barbara Demo, G. Marcianò, Contributing to the Development of Linguistic and Logical Abilities through Robotics, Conference EuroLogo 2007, August 2007, Comenius University Editor, Bratislava

[e] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas, 1980

[f] A. Rabbone, Bambini che imparano a programmare, Blog (http://bambinicheimparanoaprogrammare.blogspot.it/)

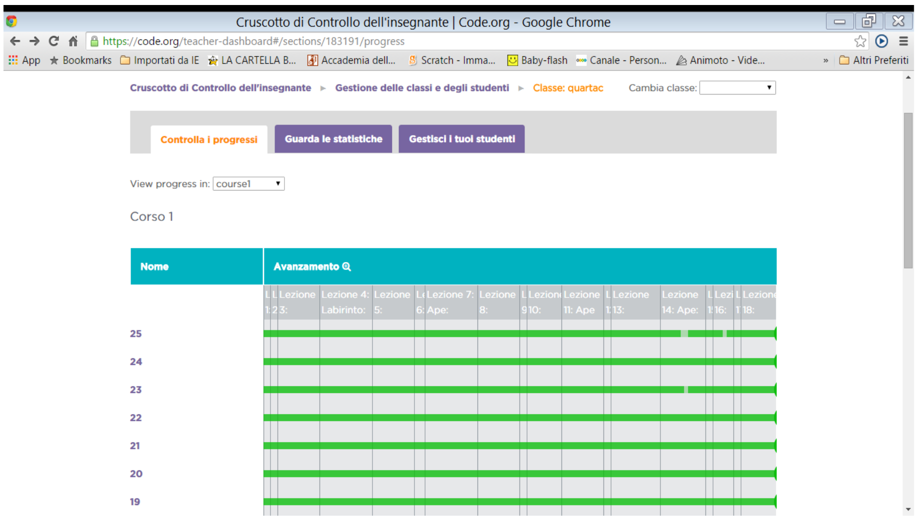
[g] B. M.Varisco, Nuove tecnologie per l'apprendimento, Garamond, Roma, 1998

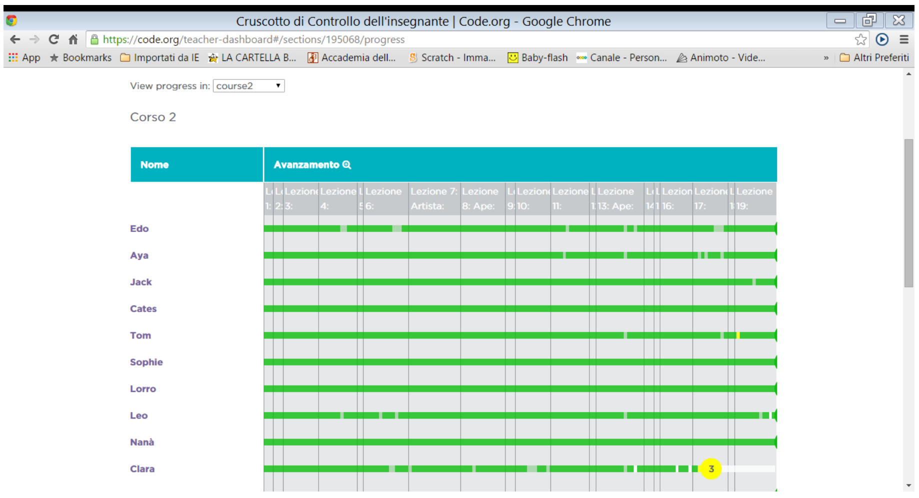
APPENDIX

RED RIDING HOOD 2.0

The 4th grade class of I.C. Tommaseo primary school began work on this project to participate in the Samsung Smart Code challenge 2015. This challenge focused on promoting coding and computational thinking through digital or analogical (unplugged) work.

During the school year the class experimented with the Scratch platform and many of the pupils enrolled with the community as individuals. Their work did not go further than attempts at sprite management or cartoon style scenes Apart from the lack of individual know how, after the general introduction to the class, the tool was used in spontaneous play. In December 2014 the teacher enrolled the class in the "Programma il futuro" initiative marking the start of a new way of working. All the pupils were enrolled on some "code.org" course and were invited to complete the work both at school and at home with the provision of free and collaborative periods. In a truly constructive way, the more competent pupils helped those with more difficulties to find solutions and finish the work whilst the role of the teacher was that of stimulus provider and moderator. Familiarity with block programming, the opportunity of guided work and the fact that there were objectives to be achieved meant courses 1 and 2 were soon completed and fairly accurately.




Cruscotto di Controllo dell'insegnante | Code.org - Google Chrome
https://code.org/teacher-dashboard#/sections/195068/progress
View progress in: course2
Corso 2
Nome
Avanzamento
Edo
Aya
Jack
Cates
Tom
Sophie
Lorro
Leo
Nanà
Clara

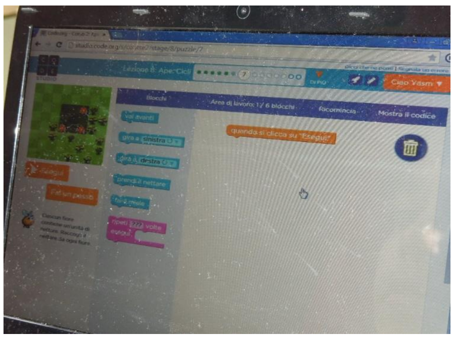
We then progressed to the 20 hour course which is under way at present. The whole class divided into two parallel groups is now working as a tutor for other classes that want to follow the “code” experience sharing both the equipment and the skills acquired.

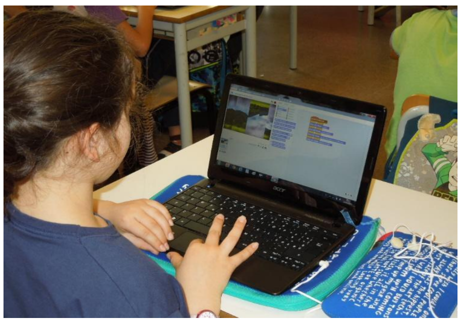
The pupils have thus learnt how to block programme as well as ensure safety and automatism on simple programming operations, for example the awareness of different points of view.

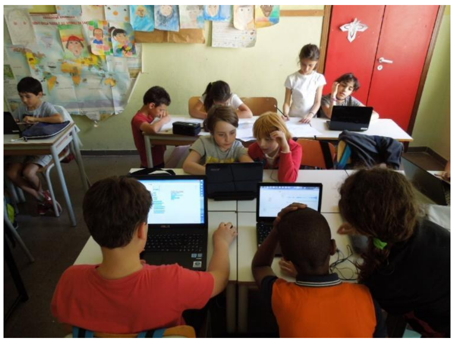
Taking part in the Samsung competition has provided the opportunity to test technological and non technological class skills in the construction programming and creation of an imaginary product. At the early stages the class went through a brainstorming phase on the “Padlet” (virtual wall); they abandoned the idea of a videogame in favour of videonarrating a fairytale. The class had previously been involved in a cartoon creation workshop (Calimero which is currently showing on RAI TV) and thus knowing about Propp functions, storyboard and storytelling they favoured animation. In view of limited time they chose to interpret the well known fairytale of Little Red Riding Hood rather than their own invention. Using a calendar we calculated that we had just four weeks to dedicate exclusively to the project (interrupting the teaching programme) which worked out to be about 60 hours school time. At this stage we identified the necessary roles to carry out the animation and then planned each step. Taking into account the children’s aptitudes and preferences we identified the following roles: Documentarists (those recording each phase and updating the scrum board), Screenwriters (those drawing up the paper storyboard and the dialogues), Scenographers (those drawing up or reelaborating the backgrounds), Phonics (those choosing and working on the soundtrack and the voices), Costume designers (those working on the costumes), Photographers (those documenting and photographing the sprites), Programmers (those mounting the scenes and the movement), the Special Effects Group (those elaborating the photographic and animation effects).

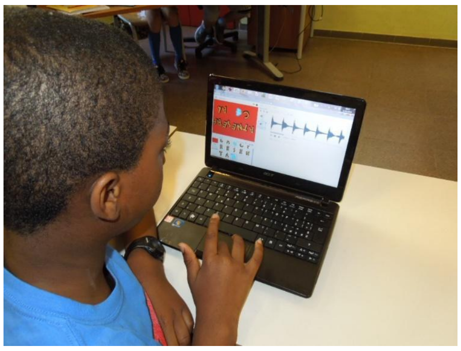
The groups worked autonomously and all the teacher had to do was coordinate the various phases. Decisions were taken collectively and the mounting done in real time on the whiteboard meaning the whole class was involved and often useful comments were made to find the best solutions. The project was beneficial in learning how to cooperate. It provided motivation to reach a common objective within a deadline. There was a true atmosphere of cooperative learning, relationships were improved and the pupils became individually and collectively responsible, peer education was enhanced as well as emotional skills. The class explored traditional and technological tools in order to adopt the best strategies to reach their objectives of quality, usefulness and aesthetics without neglecting the audience's emotional involvement in the final product. The use of internet, different types of software, personal devices together with the school premises and tools provided a modern and effective learning environment as well as food for thought over new initiatives. Experience in programming made it clear that the application of computational skills and thinking within school is decidedly useful both in daily problem solving and for the planning of projects and new initiatives.

Photos of the activities









Preparing programming exercises with efficient automated validation tests

Gregor Jerše, Sonja Jerše, Matija Lokar and Matija Pretnar

Faculty of Mathematics and Physics, University of Ljubljana
Ljubljana, Slovenia
{gregor.jerse, sonja.jerse, matija.lokar,
matija.pretnar}@fmf.uni-lj.si

Abstract. This paper presents Projekt Tomo web service, a service intended for automatic assessment of programming tasks in various programming languages and gives instructions for teachers on how to prepare problems with efficient automated validation tests.

Keywords: programming, education, automatic assessment, web service

1 Introduction

In order to achieve good programming skills, a learner has to solve many programming problems. Good and swift feedback about the submissions is vital for quick progress.
In an effort to save the teachers' time, a number of automated grading systems were developed. These systems use different approaches, as discussed below.
Setting up a special server provides a viable solution. Examples of such servers are Kattis Problem Archive[1], CodeChef[2] and Putka[3]. Students send their programs to the server, which then executes them, checks their behaviour and reports on the possible mistakes; all without any teacher intervention. A possible problem with this approach is the fact that the server must be powerful, as it has to be able to run and check the programs of all the students in the class at the same time during the lesson itself.
The students' tasks normally have simple solutions; however, the submissions that the students hand in often contain mistakes, which cause the program to loop indefinitely, and thus take up a large amount of the memory and processor time. Authors can recall numerous teaching occasions when the servers became overloaded, forcing the teaching assistant to switch back to ordinary teaching practice.
Apart from that, running third party code on the servers raises possible safety issues.
Those servers have a significant drawback from the teacher's perspective as well, as they do not usually provide a teacher's insight into the students' submissions. Students

1 http://open.kattis.com
2 http://www.codechef.com
3 http://www.putka.si

are engaged in a significantly high level of individual practice and experimentation in order to acquire basic competencies.
However, practice behaviours can be undermined during the early stages of instruction. This is often the result of seemingly trivial misconceptions that, when left unchecked, create cognitive-affective barriers [3].
Another alternative that has become rather widespread in the last few years is using the server for administration only, while the code itself runs on the student's computer. The CodeAcademy[4] page and Khan Academy[5] are currently probably the best-known examples of such a service. This alternative eliminates security and technical issues, as well as provides even faster feedback to the students as they do not need to upload the files on the server. Despite all its advantages, the alternative has a drawback. It is most easily executed using JavaScript programs, which run via a web browser. Therefore, most services offer assistance with the learning of either JavaScript or some basic form of another programming language that can be simulated in JavaScript. Another possible issue is that the students cannot use and thus get used to the common development tools.
All the known solutions have so far failed to provide proper support to our education process.
It should also be emphasized that a solution to support the learning process where the primary form of work are lab exercises with a group of students was needed, not an automated assessment tool, which requires relatively different design decisions (for a survey of such approaches see [1]).
Therefore, a web service called Projekt Tomo was developed. The first version of the system was created in 2010 by Matija Pretnar and Andrej Bauer [2]. In 2015, a new and enhanced version of the system was developed by Matija Pretnar, Gregor Jerše, Sonja Jerše and Matija Lokar.

2 Main objectives of Project Tomo service

A system, where students' submissions are tested locally, without the need for uploading or copying programs to a different coding environment was envisioned.
The students should be able to use existing coding environments for solving problems. This enables them to get comfortable with using environments that are used when working on real projects. This objective also has its practical roots. Namely, there are several courses where the usage of our service has already been foreseen. Due to various reasons in those courses, different environments (and languages) are used. Therefore, the students should not be forced to learn to use yet another environment.
Another factor behind the service is that students often solve their tasks outside the Faculty computer labs (at home, in dormitories …) where the quality of internet connection can be problematic from time to time. Consequently, it is important for the

[4] http:// www.codeacademy.com
[5] http:// www.khanacademy.org/computing/computer-programming

students to have the possibility to solve exercises (downloaded beforehand) without being online.
As explained before, such a solution is “server friendly” and not prone to misbehavior of the students’ code.
As much flexibility as possible in administering tests was another desired function. For instance, there should be the possibility to administer tests that check if a specific method was (or was not) used in a student’s submission. For example, if the students’ ability to write recursive programs is to be tested, non-recursive methods should not be accepted, even if they give expected results.
Subsequently, the main objectives were:

- local execution,
- using existing programming environment,
- being flexible enough to be functional with any programming language and
- providing as much flexibility as possible in administering tests.

3 Using the web service

Tomo web service works as follows: the students first downloads the files containing problems to their computer and start filling in the solutions, executing them locally in their favourite coding environment, while the server automatically stores and verifies the submissions.
At this moment the service supports Python and R programming languages. However, it is easily adaptable and the development towards the inclusion of C# and other languages is already on the way. Logging in is required for easier tracking of who made what changes and who solved which problems. Logging in is possible through use of LDAP system (for students of Faculty of Mathematics and Physics), Google account or Facebook account. Slovenian teachers can also login through their ArnesAAI[6]account.
Problems are organized in problem sets and courses according to topic they cover. Courses are managed by course teacher or teachers.

3.1 Using the web service as a student

After logging in, the students see all problem sets that are available in their courses (some problem sets might be hidden by teachers). The quota of the solutions accepted is shown beside each problem set. This way each student gets a quick overview of the progress made.
A very important design decisions was to call suitable submissions accepted, not correct as in the first version. Namely, after several discussions with students an incorrect interpretation about the correctness of the algorithm was observed. Now the fact that a student’s submission is evaluated only on a given set of test data is

6 https://aai.arnes.si/

emphasized. It could happen that a solution to be delivered by a student provides the expected output for the test taken, but would fail to do so in a different test.
At the top of a problem set there is usually an introduction that explains the topic covered in that section. The problems are listed underneath. They are independent from one another and each consists of one or more parts. The parts usually depend on one another and gradually become more difficult.
When the students select a problem, they must download the problem file from the server. Inside the file, there are the problem and the parts' descriptions with blanks for entering solutions.
All the functions necessary for communicating with the server and the tests for validating submissions are at the bottom of the file.
Students can open the file in any editor or programming environment they wish. Then they write the solutions in designated space between part descriptions. The students' submissions are checked and evaluated when the file is executed. Feedback is written on standard output and tells the students which solutions are accepted and which are not. For non-accepted solutions, the feedback usually includes information about which tests have failed.
If a student downloads the same problem file again, all the saved changes are included in the file. This way the student can continue solving the problem on a different computer, for example, at home.
After their submissions are accepted, the students can compare their solutions with the provided ones, written by the teacher. The teacher can choose to display the provided solutions even before a student's submission is accepted. Another option is not to show the provided solutions even after a student's solution is accepted. This is useful for exams.

3.2 Using the web service as a teacher

Users with teacher status add new problems and construct tests to test student solutions, show/hide official solutions to the problems and show/hide problem sets. As teachers can change the whole course, teacher status must be assigned with caution. Therefore, only the admin user can promote other users to teacher status.
After logging in, the teachers see a list of all their courses. When they enter a course, they see all the problem sets in the course.
The teachers can view a list of all the submissions for a problem. On this list, the teacher can see for each student which parts of the problems a student tried to solve and whether the submission was accepted. If the teachers want to examine the submission in more detail, they can download the student's attempt file to their computer. The teachers can also check the history of all the student's submissions for a problem. This option proved to be very useful in determining whether the students with similar submissions had cheated or not.
Teachers add problems by first clicking the Add new problem button. A form opens and the teacher fills the form with the problem title and description. When the teacher clicks the Add button at the bottom of the form, a new problem with a matching file is

generated. The problem title, description and all the functions for communicating with the server are already written in this file.
The teacher must then add a description of problem parts, provide solutions and a test for validating the students' attempts. To do this, the teacher must download a problem file, open it in a text editor and write directly into the file. When the file is executed, all the changes are saved on the server.
The problem description should be written in the comments at the top of the file. Text formatting is done with Markdown language[7] . Markdown language provides simple formatting options from which HTML can be rendered. Mathematical formulas can be written using LaTeX expressions are rendered by the browser using MathJax library[8].
Part's description should be written in comments between #=====.... as shown in example below.

```
# =====================================================
# Write a function `rectangle_area(width, height)`, which
# returns the area of a rectangle
# with width equals 'width` and height equals 'height`.
# =====================================================

def rectangle_area(width, height):
    return width * height

Check.part()
Check.equal('rectangle_area(3, 4)', 12)
Check.equal('rectangle_area(10, 10)', 100)
Check.equal('rectangle_area(1, 1)', 1)
```

The part's description is followed by the provided solution and tests. The tests and the provided solution are separated by `Check.part()` command.
Perhaps one of the most important features of the system is its richness in providing the teacher with suitable tools for constructing efficient tests. In the next section some of the most important features are discussed.

4 Most common validation tests

4.1 Check.equal

The simplest test is `Check.equal(expression, expectedResult)` which checks the result of the given expression and compares it to the expected result. The evaluated `expression` usually consists of a function application of the form `fun(arg1, arg2, ...)`, but it can contain any expression. If the results differ,

[7] http://daringfireball.net/projects/markdown
[8] http:// www.mathjax.org

the test emits an error from which the difference between the actual and the expected result can be seen.

```
# =================================================
# Write a function `distance(x1, y1, x2, y2)`, which
# returns the distance between points (`x1`, `y1`)
# and (`x2`, `y2`).
# =================================================
def distance(x1, y1, x2, y2):
    return ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** (1/2)

Check.part()
Check.equal('distance(0, 0, 3, 4)', 5)
Check.equal('distance(1, 2, 3, 4)**2', 8)
```

As is evident in the last example, the fact that an expression can be used, and not merely a function call, can come quite handy.
Following is an example of a feedback that a student would get for an incorrect submission that forgets to take the square root in the end:

```
  - Expression distance(0, 0, 3, 4) returns 25 instead of 5.
  - Expression distance(1, 2, 3, 4)**2 returns 64 instead of 8.
```

Note that the expression must be a string so that the feedback can show where the error occurred, while the `expectedResult` can be any Python expression, which will be displayed in its evaluated form.

4.2 Check.secret

Since the validation is performed on the students' computers, all tests are contained in the problem file. Students that are more cunning could thus look at the validation code and write a solution that passes all the listed tests, for example:

```
def distance(x1, y1, x2, y2):
    if (x1, y1, x2, y2) == (0, 0, 3, 4):
        return 5
    if (x1, y1, x2, y2) == (1, 2, 3, 4):
        return math.sqrt(8)
```

The students could even delete all the tests from the file, making the system mark such a submission as accepted. To prevent such cheating, the test `Check.secret(expression)` is used. It evaluates the given `expression` and sends the computed results to the server, where they are compared with the results submitted by the teacher (note that these are computed automatically when validating the teacher's solution).

A most common case is when `expression` consists of a function application, though this time there is no need to write it as a string because the location of the error will not be shown to the student (the system will, for example, just say "`Part 3 does not have a valid solution.`").
The students can still see the validation in the problem file, but it offers them little information because (1) they will not see the expected results and (2) the number of such tests can be quite large, as the teacher can easily test hundreds of seemingly random cases by simply writing:

```
for i in range(100):
    Check.secret(distance(i - 50, i * 17, i + 13, i/2)
```

which will compute the distance between pairs of points (-50, 0) & (13, 0), (-49, 17) & (14, 0), (-48, 34) & (15, 1) and so on… The `Check.secret` is thus especially useful when preparing exams or when there is a necessity to ensure that the students will solve their problems. In most cases, though, the students do not show a tendency to take shortcuts. Therefore, `Check.equal` tests are usually more appropriate, for they provide a better feedback.

4.3 Check.run

When programs that change local variables (for example when modifying lists or dictionaries) are tested, not only the result of an expression is to be compared, but also how it changed the state. In this case, Check.run(statements, expected_state) are used where statements is a list of Python statements (written as strings for the same reasons as in Check.run), and expected_state is a dictionary consisting of variable values expected after the statements are executed. For example:

```
# ======================================================
# Write a function `negate(numbers)`, which negates all
# the elements in a list `numbers`.
# ======================================================

def negate(numbers):
    for i in range(len(numbers)):
        numbers[i] = -numbers[i]

Check.part()
Check.run([
    'a = [1, 3, -5, 2, -6]',
    'negate(a)',
    'b = [1, -5, 2, 5]',
    'negate(b)',
    'negate(b)'
], {'a': [-1, -3, 5, -2, 6], 'b': [1, -5, 2, 5]})
```

Just like `Check.equal`, `Check.run` will not issue any warnings if the actual state matches the expected one. But if any of the values differ, a warning with an explanation will be issued. For example, if one wrote negate to leave the given list unchanged, the feedback would be:

```
- The statements
  >>> a = [1, 3, -5, 2, -6]
  >>> negate(a)
  >>> b = [1, -5, 2, 5]
  >>> negate(b)
  >>> negate(b)
set a to [1, 3, -5, 2, -6] instead of [-1, -3, 5, -2, 6].
```

4.4 Check.out_file

To check the submissions that are expected to generate files as output, `Check.out_file(filename, expected_contents)` is used. This function checks if the contents of the file with the given `filename` match the `expected_contents`, given with a list of lines. If any of the lines do not match, the student gets a comparison of both files, with * marking the lines that are different. For example, if the students are to write a function `write_alphabet(n, filename)` that prints the first n letters of the alphabet to the given file, the appropriate test is:

```
write_alphabet(3, 'test_output.txt')
Check.out_file('test_output.txt', [
    'a', 'b', 'c',
])
```

and the output in case the submission is incorrect would be:

```
- The output file test_output.txt
   equals     instead of:
    e        * a
    d        * b
    c        | c
```

4.5 Check.in_file

In order to test the programs that read from a given file, `Check.in_file(filename, contents)`is used. It generates a file with a given name and contents (supplied in the same way as in Check.out_file). The difference is that `Check.in_file` is used as a context manager – a function that changes the context in which the following functions are ran. This is best shown with an example.

Say that the students need to write a function `count_lines(filename)`, which returns the number of lines in a given file. Then, the tests would be:

```
with Check.in_file('test_input1.txt', ['a', 'b', 'c',]):
    Check.equal('count_lines("test_input1.txt")', 3)
with Check.in_file('test_input2.txt', []):
    Check.equal('count_lines("test_input2.txt")', 0)
```

As you can see, the usual testing commands are used inside a context, provided by `Check.in_file` (in the example, there is only one such test for each file, but there could be more). As seen in the example, `Check.in_file` can be used more than once to test solutions on different files. `Check.in_file` also changes the feedback so it can be seen which tests failed at which files. For example, if the student wrote a solution that had an off-by-one error, the feedback would be:

```
- For the input file test_input1.txt with contents
    a
    b
    c
  the following errors occurred:
  - The expression count_lines("test_input1.txt") returns
4 instead of 3.
```

5 Advanced validation

As the validation code is part of a Python file, more advanced tests can be constructed using not only the functions described above, but also everything that Python supports: conditionals, loops, functions, libraries, and more.

The basic command to use in such custom tests is `Check.error(message)`, which issues an error with a given message.

```
if f(100) > 1000:
    Check.error("The value f(100) is too large!")
```

The error can also be raised using an extended form `Check.error(message, arg1, arg2, ...)`, where message is given by a Python format string, where placeholders of the form `{}` which get filled with arguments can be included.

```
for x in [100, 200, 300, 400]:
    if f(i) > 10 * i:
        Check.error("The value f({0}) is too large!", i)
```

The tests also have access to the source of the submitted solution `under Check.current_part['solution']`, so they can make tests that ensure that a solution did not use `for` or `while` loops (e.g. if the students are to write solutions in recursive style). For example `ast` library[9] can be used for advanced analysis of Python source code.
`Commands Check.equal`, `Check.run` and `Check.out_file` return `True` or `False`. Therefore, they can be used to determine if additional tests should be run or not. For instance, if the first test fails, the submission is clearly not valid and additional tests are not necessary. However, if the goal is to provide the students with detailed information on which test data their programs fail, as many tests can be run as desired.

6 Conclusions

The Projekt Tomo service has been warmly welcomed by students and teachers alike. In the 2013/14 school year this service was used by 10 teaching assistants and about 500 students. Altogether, they solved 40.000 problems in 600.000 attempts.
The goal for the future is for Tomo to be used by schools and by individuals who want to learn to program by themselves. We want to accumulate a large base of well-prepared programming exercises that everybody can use and contribute to, therefore helping people to become better programmers.
Of course, there are numerous possibilities of improvement. One of them is the connection with a Moodle LMS. Tailoring appropriate feedback dependent on the history of solving a particular problem or a problem set is another area where more work is required.

7 Acknowledgements

This work was funded in part by the European Union, European Regional Development Fund and the Ministry of Education, Science and Sport. The operation is implemented in the framework of the Operational Program for Strengthening Regional Development Potentials for the Period 2007 – 2013, Development priority: Economic development infrastructure, Priority theme: Information society.

9 https://docs.python.org/3.5/library/ast.html

8 References

1. Ala-Mutka, K.M (2007). Survey of Automated Assessment Approaches for Programming Assignments, in: Computer Science Education, 15:2, 83-102, DOI: 10.1080/08993400500150747
2. Pretnar, M.: Spletna storitev za poučevanje programiranja. In: Vzgoja in izobraževanje v informacijski družbi, Lj, SLO, (2014)
3. Scott, M. J., Ghinea, G. (2013) Educating programmers: A reflection on barriers to deliberate practice. In: Proceedings of the 2nd HEA Conference on Learning and Teaching in STEM Disciplines, Birmingham, UK, 2013

Selected Spotlights on Informatics Education in Austrian Schools

Peter Micheuz[1] and Barbara Sabitzer[2]

[1] Alpen-Adria University Klagenfurt
Klagenfurt, Austria
peter.micheuz@aau.at

[2] Alpen-Adria University Klagenfurt
Klagenfurt, Austria
Barbara.sabitzer@aau.at

Abstract. In this paper we take a look on Informatics education in Austrian primary and secondary schools. The development of two reference models for digital competence and Informatics education should be seen as a big conceptual step forward, but regarding its nationwide implementation there is still a long way to go. Further, we report on a promising local initiative in Informatics for primary education as an outreach program. Then the current status of the development of a "curriculum reform light" for the obligatory subject Informatics in the 9th grade is pointed out. And finally, a major reform of the school leaving exam (Matura) at academic secondary schools including Informatics has been implemented in 2015 for the first time. In the last chapter we reflect on its general conditions, first experiences and results.

Keywords: School education, Informatics, Competence Orientation, Reference Models, Digital Competence, Curriculum, Final Exam, Matura

1 Introduction

Since the late 1980s, Informatics, ICT and Digital Media education at all levels of Austrian schools for general education have shown an inconsistent picture. Although there are many ambitious local and regional initiatives, Informatics education is not adequately represented in school, given the digital nature of our era. There is still a long way to go in Austria, especially in primary and lower secondary education.

This paper outlines the big picture of two comprehensive, coherent and comparable frameworks, including Informatics, ICT and digital literacy for all Austrian pupils and students in general education. It does not go into detail regarding vocational education at upper secondary level where the situation is much clearer, better structured and more binding.

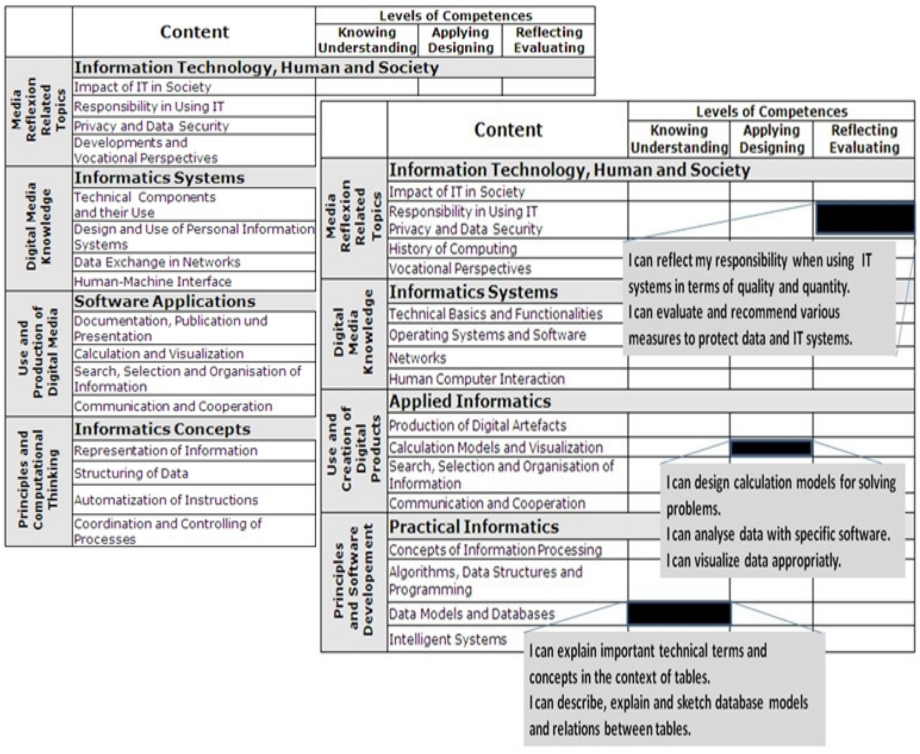
The Austrian school system encompasses elementary (grades 1 to 4, from the age of 6-7 years on), lower secondary (grades 5 to 8), and upper secondary level (grades 9 to 12) [1]. The secondary level is divided into two types of obligatory schools, namely

New Middle School (NMS), and academic secondary schools (AHS). This type of school comprises a four year lower level and a four year upper level, and concludes with the upper secondary diploma or school leaving exam (Matura) which entitles to study at universities. Currently about two thirds of the pupils attend the NMS and about one third the lower level of the AHS for four years.

Due to the lack of binding national IT-frameworks and central Informatics curricula at primary and lower secondary level so far, schools and teachers act independently, teaching, if at all, Informatics and ICT according to school specific curricula. As an undesired consequence, schools and students proceed and perform at extremely different paces.

2 Two Similar Frameworks for Digital Competence and Informatics Education for all Levels

The two competence models presented here refer to all pupils and students, from primary education on to the upper secondary level and Informatics Matura [2].



	Content	Levels of Competences		
		Knowing Understanding	Applying Designing	Reflecting Evaluating
Media Reflexion Related Topics	**Information Technology, Human and Society**			
	Impact of IT in Society			
	Responsibility in Using IT			
	Privacy and Data Security			
	Developments and Vocational Perspectives			
Digital Media Knowledge	**Informatics Systems**			
	Technical Components and their Use			
	Design and Use of Personal Information Systems			
	Data Exchange in Networks			
	Human-Machine Interface			
Use and Production of Digital Media	**Software Applications**			
	Documentation, Publication und Presentation			
	Calculation and Visualization			
	Search, Selection and Organisation of Information			
	Communication and Cooperation			
Principles and Computational Thinking	**Informatics Concepts**			
	Representation of Information			
	Structuring of Data			
	Automatization of Instructions			
	Coordination and Controlling of Processes			

	Content	Levels of Competences		
		Knowing Understanding	Applying Designing	Reflecting Evaluating
Media Reflexion Related Topics	**Information Technology, Human and Society**			
	Impact of IT in Society			
	Responsibility in Using IT			■
	Privacy and Data Security			
	History of Computing			
	Vocational Perspectives			
Digital Media Knowledge	**Informatics Systems**			
	Technical Basics and Functionalities			
	Operating Systems and Software			
	Networks			
	Human Computer Interaction			
Use and Creation of Digital Products	**Applied Informatics**			
	Production of Digital Artefacts			
	Calculation Models and Visualization		■	
	Search, Selection and Organisation of Information			
	Communication and Cooperation			
Principles and Software Developement	**Practical Informatics**			
	Concepts of Information Processing			
	Algorithms, Data Structures and Programming			
	Data Models and Databases	■		
	Intelligent Systems			

I can reflect my responsibility when using IT systems in terms of quality and quantity.
I can evaluate and recommend various measures to protect data and IT systems.

I can design calculation models for solving problems.
I can analyse data with specific software.
I can visualize data appropriatly.

I can explain important technical terms and concepts in the context of tables.
I can describe, explain and sketch database models and relations between tables.

Fig. 1. Models for Digital Competence (primary and lower secondary level, to the left) and Informatics Education (upper secondary level, to the right)

Lower secondary level (10-14 year olds) should be regarded as both a window of opportunity for and important phase of basic Informatics education. Standard learning objectives with clear expectations for teachers and students, based on a consistent, coherent and outcome-oriented reference framework, were overdue, and their development has been least triggered by the Digital Agenda [3], a framework for digital competence (left table in Fig. 1.) developed.

This model for "Digital Competence and Basic Informatics Education" incorporates many aspects. It is integrative, consistent, interdisciplinary and multidisciplinary.

A detailed discussion of the structure and example descriptors can be found in [4]. In general, curricula can be regarded as results of cultural traditions and findings from science and empirical research, including framework conditions given by educational policy. The competence framework for lower secondary education has been developed without referring to a national (core) curriculum because currently there is none.

One function of this model is to provide schools with guidance for implementing educational objectives. These can serve as a road map for policy makers, teachers, pupils and parents as well. A second is to form a basis for assessing educational outcomes in terms of accepted objectives. The competence matrix can also provide an orientation for individual diagnosis and supplementary support measures.

The framework and classification scheme (Fig. 1.) with four main categories and four content areas for each, together with about 70 "I can …" descriptors, has been disseminated among Austrian teachers. Many prototype subject tasks have been developed to illustrate and concretize the expected objectives and competencies within the Austrian project "DIGIKOMP" and the campaign "No child without digital competence." [5],[6].

Wir planen: Kein Kind ohne digitale Kompetenzen!
digikomp 8
INFORMATIONSTECHNOLOGIE, MENSCH UND GESELLSCHAFT
INFORMATIKSYSTEME
ANWENDUNGEN
KONZEPTE
Der digi.komp8-Kompetenzkatalog
KOMPLETT

Fig. 2. Planning grid for digital literacy and competence in lower secondary education

Subject to there being broad agreement on this project in all NMS and AHS, the acceptance of the standardized learning objectives and the development of tasks under a CC-license, schools should then be able to transfer theory into practice through effective implementation processes.

One current approach, especially in the NMS, is a planning grid (Fig. 2.) where teachers in many disciplines are invited to carry out selected tasks and to cover, if possible, all learning objectives of the competence model. This project, currently in progress, will be evaluated by the Austrian educational institute BIFIE.

Integrating these tasks in other subjects and/or implementing a new (interdisciplinary) subject are key issues for the future. Ongoing challenges include the supply of competent teachers, the development of competence-oriented curricula and corresponding teaching and learning material for the grades 5 to 8.

Currently, it does not seem to be realistic to implement compulsory Informatics lessons for all pupils at lower secondary level in a short time. However, there is hope that within a future major curricular reform, a new integrative and innovative subject covering all aspects of digital education including Informatics could be established.

3 Interventions and Initiatives at Primary Level

It speaks for the robustness of this model that there is an Austrian initiative to harness it as a framework for primary level within the project DIGIKOMP4 [7]. This model has the same main and subcategories as the framework for lower secondary level. The objectives are tailored appropriately to the particular age-group, but are still seen as very demanding. More than 50 tasks have been developed and published online. This can be seen as an ambitious endeavor to build IT-competency from a very early stage on. However, the general conditions in Austrian primary schools for a large-scale promotion and implementation of IT and Informatics are not beneficial. The reasons for this situation are manifold and need further investigation.

Whilst the situation in Austria is rather different from the top down / bottom up overhaul of computing education experienced by teachers in England, there are some interesting Austrian interventions and initiatives, typically focusing on outreach or of an informal, regional and project-driven character:

- Informatik erleben [9]
- Technik basteln [10]
- Wiener Zauberschule der Informatik [11]

Currently the funded project "Informatics – A child's play" and the idea of an "Informatics-Lab" [12] try to attract primary school pupils to Informatics. Many pupils in Austria associate the term informatics with the mere use of computers, tablets or smartphones. The main goal of the concept Informatics-Lab is to provide them with a better understanding of what Informatics really is. Other goals are to attract more pupils for the Informatics at an early age; the provision of co-operative learning environments where students teach one another through peer tutoring; development and use of neurodidactical lesson concepts; and the teaching of basic principles of computer science in a playful way [13].

Sample Workshops [http://informatikwerkstatt.aau.at]

- **All is logic** (Boolean Algebra)
- **1 + 1 = 10** (Binary Numbers)
- **Top secret!** (Codes and Encryption)
- **Fully networked** (The Internet and Networks)
- **Touchable computer** (Computer Systems, Hardware)
- **The data bus is on its way!** (Information Processing)
- **Well planned is half done** (Modeling and Diagrams)

The pilot project took place in July 2014 and recently 2015 at Klagenfurt University. Three research questions were the basis of an ongoing study:

- Is the concept Informatics-Lab able to increase the interest in Informatics?
- How are the individual learning workshops rated?
- Which learning methods helped the most in understanding the topics?

Feedback and replies of about 90 participants have been evaluated. The results were very promising and mainly positive. The answers about the learning methods are remarkable: The visitors rated the tutors most effective in explaining the topics. 75% said that individualized learning was very or rather helpful. More than 60% rated cooperative learning with peers and the booklet as very or rather helpful.

This regional project and initiative already seems to be having an impact on the positive perception of Informatics from an early age.

4 Special Case 9th Grade – A Compulsory Curriculum for All

Secondary academic schools (AHS, Gymnasium) provide a broad general secondary education at pre-university level for grades 9-12. Since the late 80s, Informatics has been compulsory in grade 9 and elective in the grades 10-12.

Due to a major reform of the school leaving certification process (Matura) in 2015, there has been a need for an educational guide providing recommendations for the structure and implementation of competence oriented curricula, tasks and final exams.

The similarity with the competence model for lower secondary level is obvious and deliberate. There are only a few changes in denotations which indicate the shift from digital competence (literacy) and ICT at lower secondary level to Informatics at secondary level. This model consists of four categories, each further divided into four independent areas. 80 descriptors in form of “I can …” statements describe the competences, providing more detailed information about the objectives and the corresponding topics and serving as the basis for the new competence oriented Informatics curriculum in the 9th grade. Taken as a whole, this provides teachers and students with a clear picture of Informatics.

The competence oriented curriculum, which is an amalgam of the old curriculum of 2003 [14] and the competence model in Fig. 1, is currently in review for approval.

Informatics, Human and Society
- Students describe the importance of computer science in society, evaluate its impact on individuals and society and examine exemplarily the advantages and disadvantages of digitalization.
- They take measures and apply legal principles related to data security, privacy and copyright issues.
- They describe and evaluate the development of computer science.
- They know professions related to Informatics and applications of Informatics in various occupational areas.

Informatics Systems
- Students describe and explain the structure of digital devices.
- They explain the functionality of informatics systems.
- They explain the basics of operating systems and handle graphical user interface and utilities.
- They describe the basics of networked computers.

Applied Informatics
- Students use standard software for communication and documentation as well as for the creation, publication and multimedia presentations of their own works.
- They apply standard software for calculating and visualizing.
- They know the basics of information management and use suitable software for the organization of their learning.
- They can explore sources of information, systemize, structure, evaluate, process digital content and apply different representations of information.

Practical Informatics
- Students explain terms and basic concepts of Informatics and put them into context.
- They understand, design and represent algorithms and implement them in a programming language.
- They explain basic principles of automata, data structures and programs.
- They use data bases and design simple data models.

Each curriculum is primarily only a (theoretical) paper. However, the implementation of these requirements and the consequent impact on students' competences is the other (practical) side of the coin. The issue of the intended, implemented and achieved curriculum is an interesting field of research. There are already some empirical results regarding the implementation of students achievement under the old curriculum [15]. Recent pilot research yielded insights into the actual contents which have been taught in some schools in the school year 2013/2014. The data were collected after informed consent of the schools from the central database of an Austrian wide digital class register. Most academic secondary schools have outsourced the lesson planning process where teachers have to record the subject matter they teach each week, providing a rich source of data on the planned, if not always enacted, curriculum.

The word cloud (Fig. 3.) gives an impression of the subject matter covered (or at least recorded as covered) by Informatics teachers. It is striking that there is very little programming or databases. Standard software widely dominates the content. Although the sample of data is very small and needs to be extended to yield valid results, the assumption that Informatics in the 9th grade is very application driven is justified.

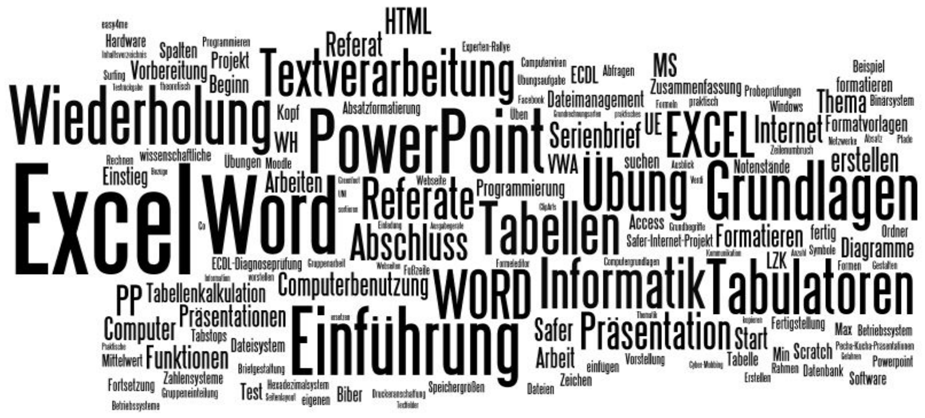


Fig. 3. Word cloud of Informatics lessons in the 9th grade of four Austrian schools.

5 Competence Oriented Informatics Matura – First Impressions and Results

A major reform of the final exam (Matura) at academic secondary schools has recently been implemented, with effect from the 2014-15 school year. All subjects, Informatics included, have been affected. A strong competence orientation, especially in the task construction provided the underlying philosophy for the reform. Whereas in languages and mathematics central regulations provided all students with the same tasks at the same time, in other disciplines, including Informatics, it has been left to the schools to select topics and develop individual tasks, based on particular curricula and competence models.

Oddly, the first exams under the new system (the Matura of summer 2015) took place before the new competency based curricula were in place, which will be adapted in the subsequent school year! Despite this flaw and criticism regarding the rather short preparation time for this major Matura reform, a nationwide evaluation actually yielded rather positive feedback, including for the oral Informatics Matura.

The new Matura includes the elective subject Informatics in grades 10-12, based on an open curriculum since 2003. Currently it is not competence oriented and consists merely of 11 different topics, without any indication of the level of knowledge expected in these. These broad topics are:

Basic principles of information processing, concepts of operating systems, construction and operation of networks, databases, learning and work organization, concepts of programming, artificial intelligence, expansion of theoretical and technical foundations of computer science, basic algorithms and data structures, computer science, society and the world of work, legal issues.

Unlike the main subjects (German, foreign languages and mathematics), there are no central regulations and centrally developed tasks for Informatics. Insted the teachers are responsible for the selection and sequencing of these topics in combination with appropriate software tools. Together with the competence model, these topics serve as the basis for the oral Matura.

Since 2012, a competence model for Informatics in upper secondary level (Fig 2.) now extends the curriculum with a better structure and orientation. It is not yet known to what extent this competence model is used by the teachers in their Informatics lessons. However, this model is one building block of official ministerial recommendations outlining some sample tasks [16].

The particular innovation of the Matura reform and the oral Matura is the mandatory competence oriented alignment of the tasks and the assigned number of topics. In the case of the elective Informatics, teachers at any particular schools have to agree on 12 topics which have to be publicly announced about six months before the exam.

School A	School B
- 01 History of Computer Science	- 01 Data Structuring and Modeling
- 02 Spreadsheet	- 02 Algorithms
- 03 Databases	- 03 Programming
- 04 Internet and Web 2.0	- 04 Office Programs
- 05 Operating Systems	- 05 Mathematics in Informatics
- 06 Data security and privacy	- 06 Computer Architecture
- 07 Software development and binary system	- 07 Fractals in Computer Science
- 08 Web Publishing/Markup Languages	- 08 Networks
- 09 Programming Languages and Concepts	- 09 Operating Systems
- 10 Digital Imaging	- 10 Web design
- 11 Hardware Basics	- 11 Artificial Intelligence
- 12 Network Technology	- 12 Data security and privacy

This table illustrates two examples with different approaches to Informatics content. There are about 340 secondary academic schools in Austria, although not all schools offer elective Informatics courses. That does not necessarily mean that there in all courses there are candidates who chose Informatics as an oral Matura subject.

As it is rather difficult to collect data from Austrian schools we only can estimate the figures. We estimate that in two thirds of the schools there are currently elective Informatics courses, suggesting that nationwide between 500 and 1000 students out of approximately 17000 chose the oral Informatics Matura.

If a school has candidates for the oral exam, the teacher has to prepare two "competence oriented" tasks for each topic. A competence oriented task has, by law, to cover three aspects of proficiency and requirements areas: reproduction, transfer and problem solving and reflection.

With these regulations, the legislator intended to transform the Matura from a knowledge-based to a competence-based assessment. Until these reforms, the tasks and questions were heavily content-based, asking for knowledge rather than the application of higher-level thinking through problem-solving or critical thinking. As long as it was only facts that assessments required, these will be what teachers teach and students learn, regardless of any efforts in curriculum renewal: 'that which we measure becomes that which we value'. Educators at all levels need expertise in the evaluation of competence and skills, building on a solid basis of content knowledge.

It has been a challenge for teachers to develop the 24 competence-oriented tasks, two for each topic.

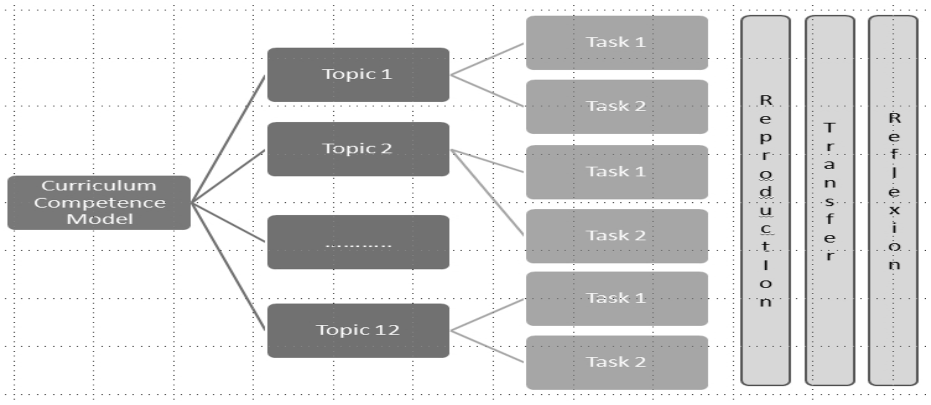


Fig. 4. Topics and tasks

All tasks have to be camera-ready a few weeks before the Matura, when the candidates have to draw two different random topics of which one has to be chosen. After this procedure the teacher chooses one of the two assigned tasks and hands it out to the student.

Part of an exemplary task, the reflective and problem-oriented element, is shown in Fig. 5. The whole task set of this particular school is shared as an open educational resource and can be downloaded from [17].

In Fig. 5. there is a fragment of a task is assigned to the topic "Simulation, Animation and Coincidence".

The operational part of the new Matura can be seen as a lottery.
You try to convince the school principal that it would make sense to carry out the draw of the topic numbers with the support of an informatics system.
The concept of random numbers, various developement tools and your competence seem to be good preconditions for an implemention.
Analyse the requirements for this digital soultion and develop a simple prototype.
Think of arguments which speak against a digital solution and put them to discussion.

Fig. 5. Example for the reflective and problem-solving part of a given task.

In the run-up to the Matura, a couple of month before the oral exams took place, the author conducted a little qualitative survey among a community of teachers involved about believes of strengths and weaknesses of the new Matura.

Strengths/Advantages	Weaknesses/Risks
All informatics teachers in one school must cooperate, there is no one man show any longer	No depth; interesting topics might be neglected. Tasks cannot be very demanding. For special topics (as robots, …) there will be less time.
Not only factual knowledge is assessed.	Fewer candidates for the Informatics Matura.
Standardization of the number of topics.	The Informatics Matura will be chosen only by very gifted and talented students. It is too difficult for the average.
The new Matura will bring more drive and motivation into the informatics lessons.	Many tasks have to be prepared for few students.
Practical tasks can be well covered through the second requirement area (transfer, application).	The practical part could be neglected.

This selection of 15 teachers' opinions is only a small sample of many interesting comments and insights. Assuming that most of the teachers are cooperative, the evaluation of experiences, the collection and categorization of topics and developed tasks could serve as a valuable resource for the improvement of the Informatics Matura and, moreover, also for the consolidation of the subject Informatics in general.

6 Conclusion

In this paper we have shed light on the current situation of Informatics education in Austria at different school levels in general education. We tried to sketch a picture of the spectrum from bottom up movements and initiatives to top down competence models, campaigns, curricula and central reforms, laying the focus on four active areas. Other projects such as the Beaver Contest and competitions in the field of IT, the influence of the European Computer Driving License on formal IT education, and the situation of teachers' pre- and in-service training have not been mentioned.

There are difficulties in comparing Austria's current status in and progress with Informatics education with that in other countries. However, comparable data from many countries that refer to national frameworks and formal implementation processes in existing school curricula suggest that Austria is trailing behind, in particular there is no coherent and nationwide education policy for IT and Informatics at primary and lower secondary level. By this standard, to start formal Informatics education for all at the 9th grade seems too late.

Nevertheless, as we indicate, there are non-formal activities and autonomous developments in individual schools which add life to the Informatics scene in Austrain education, and thus compensate to a certain extent for political omissions.

References

1. Bildungssystem in Österreich. http://www.bildungssystem.at
2. Micheuz, P. (2012). Towards a competence model for ICT and Informatics in general education at secondary level. In: Manchester Metropolitan University (Hrsg.): IFIP Working Conference. Addressing educational challenges: the role of ICT. Manchester: Manchester Metropolitan University, 12 pp.
3. DA (2008). Digital Agenda. http://ec.europa.eu/information_society/digitalagenda
4. Micheuz, P. (2011). A Competence-Oriented Approach to Basic Informatics Education in Austria, in I. Kalas, R.T. Mittermeir (Eds.) Informatics in Schools - Contributing to 21st Century Education, ISSEP 2011, Springer, 43-55.
5. Project DIGIKOMP8: http://www.digikomp.at
6. Narosy T.: Kein Kind ohne digitale Kompetenzen!. In: P. Micheuz et. al (Hrsg.): Digitale Schule Österreich. OCG. Wien (2013).
7. Mulley U. Zuliani B.: Ein Digitales Kompetenzmodell für die Volksschule. In: P. Micheuz et. al (Hrsg.): Digitale Schule Österreich. OCG. Wien 2013.
8. National Curriculum in England: Computing Programmes of Study, published 2013. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study
9. Informatik Erleben: http://informatik-erleben.aau.at
10. Technik basteln: http://www.technikbasteln.at
11. Wiener Zauberschule: http://www.ocg.at/de/wizik
12. Sabitzer, B.; Pasterk, S. (2014). Informatics – A Child's Play. 6th International Confernce on Education and New Learning Technologies (EDULEARN), July 2014, Barcelona, Spain.
13. Sabitzer, B. 2014. A Neurodidactical Approach to Cooperative and Cross-curricular Open Learning: COOL Informatics. Habilitation thesis. Alpen-Adria-Universität Klagenfurt.
14. Informatics Curriculum Upper Secondary Level Austria (2003) http://www.bmbwk.gv.at/medienpool/11866/lp_neu_ahs_14.pdf
15. Micheuz P.: Zahlen, Daten und Fakten zum Informatikunterricht an den Gymnasien Österreichs. In: B. Koerber (Hrsg.): Zukunft braucht Herkunft, INFOS 2009, LNI, Berlin. Online: http://subs.emis.de/LNI/Proceedings/Proceedings156/243.pdf (2009)
16. Official Guidelines for oral Informatics Matura https://www.bmbf.gv.at/schulen/unterricht/ba/reifepruefung_ahs_lfinf_24984.pdf?4k21fp
17. Portal for Informatics at academic secondary schools http://www.ahs-informatik.com/informatik-matura-neu/aufgabenpools/

Algorithms and well formatted texts: Introducing Computer Science Activities in Lower Secondary Schools

Martina Palazzolo

Istituto Comprensivo Ilaria Alpi, Milan, Italy
martina.palazzolo.5@gmail.com

Abstract. In order to raise awareness on computational thinking among teachers in our lower secondary school, we started a collaboration with university (Università degli Studi di Milano, Computer Science Department) for a training. We invited a team of computer scientists in all our 6th and 7th grade classes asking them to carry out an unplugged activity with a computer based final task (called 'algomotricity'). The aim of the project was to engage pupils in a computer science activity, but even more important was to show teachers the potential of informatics for a computational thinking approach in math and science. We got positive results and teachers expressed their interest. Over the next year we hope to continue this effort to gradually improve teacher self-competence, enable them to give the course themselves and gradually build up a new curriculum for mathematical and scientific area.

1 Introduction

During the last decade, Italian teachers have been urged to teach computer science, even if the subject was not (and is not yet) part of a general purpose curricula and there are no specialized computer science teachers in non-vocational schools (A recent survey of the Italian context appeared in [2]).

Thus, this urge was mainly confused with the use of office automation tools, Internet browsing and other communication facilities. Or, even worse, it was wrongly understood as the introduction of digital technologies in the teaching process, such as the adoption of the 'Interactive Multimedia Board' (IMB), [1] Maths and technology textbooks introduce computer science mainly explaining how to use MS Word and MS Excel and teacher training often focuses on the use of applications rather than on the educational aspect of computational thinking. In fact, this misinterpretation of computer science is so common [8]that the term *"applimatics"* was proposed in [3] to distinguish the use of applications and digital technologies from *"informatics"*, the actual scientific discipline. As pointed out by several authors in informatics (for example [6], [7])computer

[1] The Italian Minister of Education proclaims to have distributed 9'000 IMBs, on a total of 11'234 requests[5].

science has the potential to foster important scientific skills such as problem solving, creative and critical thinking.

In recent years, efforts aimed to expose young people to programming and computational thinking have been made and they are spreading in several countries, including Italy. An example is the Coderdojo initiative[2]where pupils are engaged in programming activities with a mentor who introduces them to the Scratch [3] visual programming language. Often these activities are proposed to pupils in a context where they can behave creatively, partially in contrast with the more formal classroom rules. Collaboration and creativity are encouraged and pupils are invited to exchange information and ideas. Coderdojos, however, are thought as extracurricular activities and they seem not to fit well in our school environment.

Last year a project called'Programma il futuro'[4], designed to be introduced in school, was launched. The project is the Italian version of the 'Code.org' initiative[5] and aims at introducing computer programming in school with the explicit goal that *"Every student in every school should have the opportunity to learn computer science"*. In fact, the main emphasis is on 'coding', on programming.

Thus, with the the dual objective of exposing our pupils to programming and computational thinking and to promote teachers' autonomy and improve our skills to deliver computer science activities in our classes, we started a collaboration with the team specialized in computer science education of the University of Milan (ALaDDin, [6]) who proposed entry-level labs. This paper reports this experience that involved two aspect of informatics: algomotricity (Maze workshop) and digitally formatted texts (Wikipasta workshop). Maze workshop was chosen for 6th grade pupils and Wikipasta workshop for 7th grade pupils.

The algomotricity strategy developed by the ALaDDin team was meant as *'a strategy that focuses on algorithmic concepts through motoric activities, which imply a mix of tangible and abstract object manipulations'*[4]. Maze workshop shows the core of computer programming with no particular interest in the syntactic issues aiming instead to develop problem solving and computational thinking related competences[4]. Starting with the development and discussion of pseudo code, pupils are led to understand and to formulate an algorithm defined as *"an effective procedure to reach, in finite time, a goal"*[4] . Algorithms, intended as detailed sequences of steps based on a number of given conditions are actually quite common in other discipline such maths, although during the teaching activities these are not explicitly declared as such, and no emphasis is put on the underlying computational thinking process.

An additional issue that we wanted to introduce in lower secondary school is the concept of 'digitally formatted texts'. One reason is that our pupils will be writers of digital texts in the future (if they are not already). Traditional and

[2] https://coderdojo.com/. In Italy: http://www.coderdojoitalia.org/.
[3] https://scratch.mit.edu
[4] http://www.programmailfuturo.it/, "Program the future".
[5] https://code.org/
[6] http://aladdin.di.unimi.it

digital writing are totally different processes with completely different rules (and tools, of course). A personal computer allows the processing of well formatted and accessible texts, provided that one can properly use it. We cannot continue to use a computer as a typewriter, completely ignoring the basics of the mark-up languages. In order to change pupils mental model of formatted texts and to introduce them to the idea of mark-up languages we asked the ALaDDin team to work in our 7th grade classes with their Wikipasta workshop, which they developed with this goal[1].

2 The Context

The project involves the lower secondary school in the context of a school that includes primary school (40 classes) and lower secondary school (27 classes). The lower secondary school is distributed in three different buildings. With an average of 20-25 pupils in each class, the lower secondary school has about 550 - 650 students. For 27 classes, nine maths and science teachers are required. Each teacher gives 2 hours of science lessons and 4 hours of maths lessons in a week per class. Usually, a maths and science teacher teaches in one 6th grade class, in one 7th grade class and in one 8th grade class for a total of three classes.

In the lower secondary school 9 classes have an IMBs (the already mentioned 'Interactive Multimedia Board'). Interactive boards are located in all the 8th grade classrooms (9 classes) to allow all the pupils to take advantage of this technology at least in the last year of lower secondary school. A survey carried out two years ago revealed that about 40 per cent of the teachers working in IMB equipped classrooms used them during their lessons. The IBM is appreciated by both teachers and pupils. Some teachers prepare lessons for IMB, but most of them use it for video or pictures. Over the last year, three of us adopted the Edmodo virtual class [7].The IMBs have been quite useful to connect to the virtual class and discuss with pupils their digital homework at school showing the outcome.

There are three computer labs, one for each building, but they are rarely used because of the nonfunctional infrastructure. Computers are too few, too old and too different to be used for teaching activities with a whole class.

Some of us show to be interested in informatics rather than *"applimatics"* and spontaneously followed specific workshops on computational thinking. Two of us spontaneously participated in a workshop that the ALaDDIn team organized about *'algomotricity'* and Maze in the Museo Nazionale della Scienza e della Tecnologia Leonardo Da Vinci in Milan in the year 2013 .

In 2011 two of us also participated in *"Progetto EST"* about Robotics, again organized by Museo della Scienza e della Tecnica Leonardo Da Vinci but could not integrate the project in their own curricula in the following years. Some interest for programming is evident but occasional.

For the project we are discussing here we focused on 6th and 7th grade pupils for a total of about 400 pupils and their 9 maths and science teachers.

[7] `www.edmodo.com`

3 The Project

The aim of the project is to improve our (teachers') skills and to help us to look at informatics with new eyes and reconsider the computer science as an interesting matter, whose concepts could fit in maths and problem solving rather than the use of applications such as MS Word, Excel or Power-point. In the future the goal will be to integrate computational concepts in our maths and science learning activities. The objective is also to teach teachers and pupils to write well formatted digital texts.

With the ALaDDin team we have chosen two of their proposals: Maze for 6th grade pupils (nine classes) and Wikipasta for 7th grade pupils (again nine classes).

3.1 Maze and programming

Maze activity introduces pupils to programming. The ALaDDIn team explain their Maze workshop and ensuing results in the article to be published in ISSEP in the current year[4]. Here it is shortly resumed: in Maze workshop pupils work in groups and are asked to write a pseudo-code program to drive one of them (the robot) in a maze built by chairs and tables. The workshop starts with an unplugged activity and ends with a computer activity where pupils program using a Scratch derived environment. The ALaDDin team showed us their *"algomotricity"* method based on kinesthetic learning activities. In their activities computers and software tools are of secondary importance but the link between unplugged activity and the use of technology is clear. As the ALaDDin team explained it in [4]:

> *"Maze workshop proposes a little real problem. Students need "to formalize" it, they must find their own way to a solution, not just code a predefine one. Thus the activity works on the interplay between algorithms and programs. When the pupils drive a blindfolded mate with a finite numbers of instructions, they reason on what is effective and feasible: and the power of the interpreter is in large part a choice they explicitly do. Moreover, after the pupils themselves have checked that their problem solve the task, when they compare it with others' programs, they discover that some of the assumptions they made are not valid in the slightly different context of the other teams' settings. [...] AlMa (Algomotricity and Maze workshop) aims at giving the students a meaningful problem to be explored in an open context. We provide just a few restrictions designed to support their own inquiry. [...] The open-ended activities proposed in AlMa, indeed, encourage the participants to formulate original ideas. Moreover, the team setting forces the pupils to convince the other mates that their proposals work correctly: they need to describe them properly and devise a way to show the correctness of their hypothesis. Thus pupils happen to put into practice and experience the "scientific method", even though they usually are unaware of this fact."*

3.2 Wikipasta

In 7th grade classes ALaDDin team presented their Wikipasta workshop. This activity aims to make pupils understand the meaning of formatted text pointing at mark-up languages but without talking about it. A methodology similar to Maze, based on unplugged and computer activities, was adopted. Also Wikipasta workshops have been organized within maths and science lessons. When we read or write a text on paper we can easily recognize functional part of the text such as the title, the paragraph, an ordered list, a figure, etc. thanks to graphical effects. The title usually is centered, colored and has bigger characters that make it distinguishable from the paragraphs. An ordered list is clearly identified by numbers before each sentence. When we asked ours pupils to identify the different part in a text and to explain their meaning they had no difficulties in explaining the function of the title, ordered list, figures and so on. They perfectly recognized the important informative role of a formatted text. But, while on paper formatted text and its content are merged, in digital texts these are two separate pieces of information. Using cheap materials (pasta, colored buttons) and a paper with a text without any formatting information, the ALaDDin team proposed an interesting game that enabled our pupils to understand how in digital texts content and formatting information are separately managed. They asked to format a text using pasta and colored buttons. Afterwards they understood that, adding a legend, they could effectively send formatting information separately from the text itself. Later, as explained by the ALaDDIn team in [1], they discovered the use of tags:

> *"Teams were again requested to reproduce a formatted text with objects and write down codifications rules precise enough to be followed by another team. However the game was made fun by the introduction of a "cost" for the objects. In fact, the more an object could be used to mimic a piece of formatting , the more it cost in order to promote its* symbolic *use, e.g., since spaghetti pasta could be easily associated to* underlying*, its cost was very high. The winner would be the team able to hand in an unambiguous codification with the lowest cost. The cost incentive was enough to let the pupils discover what is commonplace in mark-up languages: the use of tags at the beginning and at the end of (possibly overlapping) regions".*

In Figure 1 and In Figure 2 we can see two products from the Wikipasta workshop before and after telling them the cost of pasta and buttons.

The Wikipasta workshop allowed us to introduce the basics of HTML language in one of the 7th grade class during science lessons. At the end of the course most pupils succeeded in preparing a single web page using the main HTML tags and some pictures about their scientific experiences. The activity is described in the poster session (From Paper to Web Pages - Some Help from PirateBox) here at ISSEP of the current year [9].

In Figure 3 we can see the first web page that the teacher showed to pupils to identify the functional parts of the document. In Figure 4 we see the same

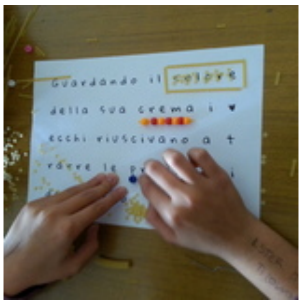


Fig. 1: wikipasta1

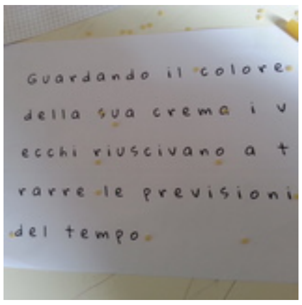
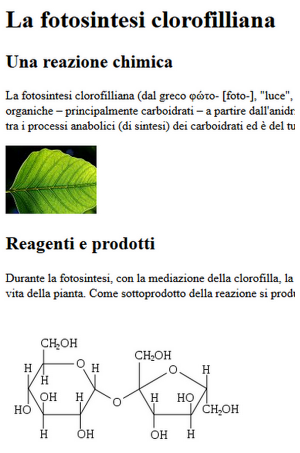


Fig. 2: wikipasta2



La fotosintesi clorofilliana

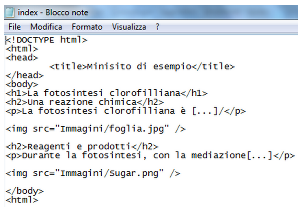
Una reazione chimica

La fotosintesi clorofilliana (dal greco φώτο- [foto-], "luce",
organiche – principalmente carboidrati – a partire dall'anidr
tra i processi anabolici (di sintesi) dei carboidrati ed è del tu

Reagenti e prodotti

Durante la fotosintesi, con la mediazione della clorofilla, la
vita della pianta. Come sottoprodotto della reazione si produ

Fig. 3: web-page



```
<!DOCTYPE html>
<html>
<head>
        <title>Minisito di esempio</title>
</head>
<body>
<h1>La fotosintesi clorofilliana</h1>
<h2>Una reazione chimica</h2>
<p>La fotosintesi clorofilliana è [...]/</p>

<img src="Immagini/foglia.jpg" />

<h2>Reagenti e prodotti</h2>
<p>Durante la fotosintesi, con la mediazione[...]</p>

<img src="Immagini/Sugar.png" />

</body>
<html>
```

Fig. 4: html

page written in HTML with a text editor where pupils started identifying the symbols used instead of pasta and colored buttons.

4 Feedback and Results

The ALaDDin team prepared questionnaires to evaluate the impact of their Maze activities. Each pupil answered the questionnaires at the presence of their maths and science teachers. The ALaDDin team also organized a focus group with a sample of students from each class. The AlaDDin team examined and analyzed them for their academic research and publications [4]. From informal talks with students we can report that most pupils were enthusiastic and enjoyed the activities, which have been described as interesting and fun. It looks like most of the pupils changed their previous idea that informatics is just word processing and web browsing. We (the teachers) observed that also pupils with special educational needs enjoyed Maze and Wikipasta workshops. Thanks to the setting and the methodology adopted by the ALaDDin team, they could actively participate in the working group with their own contribution. Some of us teachers asked pupils to summarize the experience with the ALaDDin team on paper and they all confirmed the positive feeling of their teachers, including those with special needs. Most of them, in fact, wrote that they enjoyed the activities, mainly because they succeeded in actively participating and interacting with their peers. In their reports on the Wikipasta workshop, pupils showed to have understood digital formatting rules using tags (pasta or special symbols) and expressed their interest both for the content and for the setting of the proposed activities. Our post-course talks showed that we appreciated these activities too. We are a bit worried about class management and felt not yet confident to autonomously face informatics concepts the ALaDDin team proposed. We appreciated the involvement we observed in our pupils and also understood what informatics could really mean when intended as computer science instead of simple use of applications or communication tools.

5 Discussion

The collaboration with the ALaDDin team has been fundamental to understanding programming and formatted text topics. We all, Maths and science teachers and pupils, could taste what informatics should be: a computer science related with problem solving as well as creativity without the need for any specific application. In spite our interest, some of us felt not to be yet ready to continue the activities by ourselves. A training thought just for us would be required before, during and after workshops in the classes. A stable collaboration with informatics specialists from universities interested in the educational aspects of computer science would be essential to help us integrate *"informatics "*(not *"applimatics"*) in our curricula. The Maze workshop would be a perfect starting point to introduce an interactive way to teach and fostr computational thinking, involvement and creativity. Some maths and science contents could be presented using the

ALaDDin "algomotricity" strategy of learning. As an example we tested the procedure to calculate the surface of different "every day" objects like shoe-boxes, a tube of tennis balls, etc. (this was the given problem). Working in groups, pupils were invited to identify a way to calculate the surface of the object they chose, define the strategy they adopted and describe it to mates. These activities have just begun, but a strong involvement was observed in pupils making their own hypothesis, testing it with their tools (scissors, ruler, pencils ...), trying to formalize a description to convince mates their strategy was good. As already mentioned "Thus pupils happen to put into practice and experience the *scientific method*, even though they usually are unaware of this fact". This involvement in the manipulative activities was important to stimulate pupils curiosity and interest when we proposed them the usual formula for geometry area calculus. In addition, the Maze workshop could introduce pupils to programming using, for example, the Scratch visual programming language. An important question to be asked is when this can be done. Taking time from the 6 hours dedicated to maths and science contents? Which contents could be considered to be equivalent to a programming activity? And how? Can this activity be integrated in the maths and science curricula?

Another important concept about informatics would regard digitally formatted texts. In a summary a pupil wrote:"During the Wikipasta workshop we understood that in a document we wrote two different information: meaning and formatting. Computers use different symbols to send this two pieces of information". We think this is a fundamental concept to teach about digitally formatted texts. This mental model is essential in order to write well formatted texts. This allows such text to be accessible in different ways to people, including those with some disabilities, like, for istance, blind people that use screen readers or other electronic devices to read a text. We chose to show them the basis of HTML language for different reasons: first, pupils told us that they mainly use PC to browse the internet and read web pages (other then for videos, musics and video games); second, it is a mark-up language that perfectly fits in the Wikipasta workshop; third, only a text editor and a browser are required to write and read a web pages without facing the problem of using a specific word processor.

6 Conclusion

The project described here focuses on two aspects of informatics: programming and formatting digital texts. To introduce them in our lower secondary school we started a collaboration with the ALaDDin team specialized in computer science education of the University of Milan. The aim of the project was twofold: to introduce informatics concepts to pupils and also to their maths and science teachers. Both pupils and teachers have been faced with a new vision of informatics and expressed their interest. The project has been summarized in the last teachers' board and was approved for the next year. This project has no focus on specific computer applications but on the science of computer itself. Informatics is not yet a discipline in the lower secondary school but we think that many informatics concepts regard also maths and science skills like problem solving and scientific method, which could be integrated in our curricula.

References

1. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M.: Exploring the processing of formatted texts by a kynesthetic approach. In: Proc. of the 7th WiPSCE. pp. 143–144. WiPSCE '12, ACM, New York, NY, USA (Nov 2012)
2. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M., Zecca, L.: Informatics education in italian secondary school. ACM Transactions on Computing Education 14(2), 15:1–15:6 (2014)
3. Lissoni, A., Lonati, V., Monga, M., Morpurgo, A., Torelli, M.: Working for a leap in the general perception of computing. In: Cortesi, A., Luccio, F. (eds.) Proceedings of informatics education europe III. pp. 134–139. ACM – IFIP (2008), http://www.dsi.unive.it/IEEIII/atti/PROCEEDINGS_IEEIII08.pdf
4. Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A.: Is coding the way to go? In: 8th international conference on informatics in schools: situation, evolution, and perspective. To appear
5. Ministero dell'Istruzione, dell'Università e della Ricerca: Scuola digitale — LIM. http://hubmiur.pubblica.istruzione.it/web/istruzione/piano_scuola_digitale/lim, last visit: July 2015
6. Papert, S.: Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc. (1980)
7. Taub, R., Ben-Ari, M., Armoni, M.: The effect of cs unplugged on middle-school students' views of cs. ACM SIGCSE Bulletin 41(3), 99–103 (2009)
8. Furber, S: Shut down or restart? The way forward for computing in UK school. The royal Society Education Section. DES2488 (2012)
9. Palazzolo, M., Mauri P.: From Paper to Web Page - Some Help from PirateBox. In: 8th international conference on informatics in schools: situation, evolution, and perspective. To appear

ZaznajSpoznaj - a modifiable platform for accessibility and inclusion of visually-impaired elementary school children

Matevž Pesek[1], **Daniel Kuhl**[1], **Matevž Baloh**[1], **Matija Marolt**[1]

[1]*University of Ljubljana, Faculty of Computer and Information Science*
E-mail: matevz.pesek@fri.uni-lj.si

Abstract. The educational and IT communities have produced a number of e-learning products, ranging from support-oriented platforms for online courses and learning to educational games. However, there is still a growing need for inclusive and accessible learning products. To meet the need, we developed an accessible online web and mobile platform for educational games which are highly modifiable and applicative to any learning domain. The paper describes the platform, its agile development process, and first results of the platform's evaluation for the blind and visually impaired elementary school children.

Ključne besede: e-learning, visually-impaired children, ZaznajSpoznaj platform, memory training, vision training, ICT inclusion

1 Introduction

E-learning draws more and more attention each year, by increased inclusion of e-materials in primary and secondary schools, new IT products focusing on this field, and by the rising affordability of technology.

E-learning introduces a number of challenges. Existing educational materials may soon become outdated and do not evolve with new technologies. Moreover, the materials, especially the educational games, usually include fixed content, which cannot be modified by the user. When the product gradually becomes outdated, the educational process and the game content drift apart. Thus, the teacher is faced with a decision to either modify the educational process to include the game or stop using the game. Additionally, the source code of many projects is unavailable — either the code is private (not open-source), or even worse, is lost.

For students with disabilities, e-materials introduce additional difficulties. Although the technology itself usually provides accessibility options to be used by people with disabilities (e.g. Android, Windows and other OS accessibility features), these are rarely usable for a specific product to the full extent. For example, the OS built-in accessibility features may not be fully compatible with an e-learning application and games are not adapted for use by these students.

Within the ZaznajSpoznaj project, we are developing a novel platform for inclusive and accessible educational games that would overcome some of these difficulties. The project focuses on the blind and visually-impaired children by supporting them in their elementary school learning process. The ZaznajSpoznaj platform offers a variety of repetitive games for memory and vision training, learning Braille and extending the typing skills. The games support accessibility for the blind and visually impaired and are developed to be extremely flexible and allow for modification of their content, thus offering teachers the possibility to adapt the games to several domains and modify them through time.

This paper presents the platform, which enables open-access to: developers with an open-source API for development; teachers who can modify the games in several aspects including content and visual appearance; and users who gain access to a growing community-based games database. The paper is structured as follows: the current state of related products is presented in Section 2. Further on, the ZaznajSpoznaj platform and its features are presented in Section 3, followed by a preliminary evaluation with visually impaired children in Section 4. We conclude the paper in Section 5.

2 Related work

Due to the increasing importance of e-learning there already exist approaches to make learning management systems (LMS) accessible for blind and visually impaired students [8]. In addition there are also various initiatives to make textbooks and other traditional learning materials accessible (e.g. [5], [2]). However, recent trends and developments introduce new challenges into learning processes in general and e-learning in particular. Two currently popular trends are mobile learning and game-based learning. The spread of mobile devices results in a greater demand to support mobile learning activities at every level of education. For example, research activities of Filho et al. address general aspects to make learning environments accessible on mobile devices, Ally et al.[1] are focused on the impact of mobile technology on learning processes, curriculum and education in general to make mobile learning beneficial. To foster mobile learning especially in developing countries the United Nations Educational, Scientific and Cultural Organization (UNESCO) published policy guidelines for mobile learning[10].

Despite all diverse learning concepts and modern technology, learning activities as such remain annoying to most students. To overcome motivational barriers, it proved beneficial to introduce game-based learning elements into the process. Depending on target groups, subject and learning objectives, different approaches can be applied. Especially elementary school children are likely to benefit from such approaches. For example, Leichtenstern et al.[4] use mobile devices to assign specific types of interaction within role plays. Görgü et al. [3] use mobile devices and augmented reality to motivate users for outdoor games. The approach of Schimanke et al. [9] focuses on the benefits of games to improve results of repetitive learning activities. The game-based approach also raises challenges related to accessibility and inclusion. Although some initiatives offer examples of web-based learning games [1][2] they lack the integration into a LMS and offer only limited opportunity for customization and personalization. In addition due to applied technology, they are not always available on standard mobile devices. Milne et al. [6, 7] present several examples for learning games for blind and visually-impaired children on standard mobile devices. But so far their approaches offer only limited opportunities for customization and lack integration into a LMS.

Existing e-learning platforms and approaches do not provide sufficient support for game-based learning in general and are not suitable for visually-impaired children. In addition they do not support the Slovenian language and offer limited support for customization by teachers that are not IT-experts.

3 The ZaznajSpoznaj platform

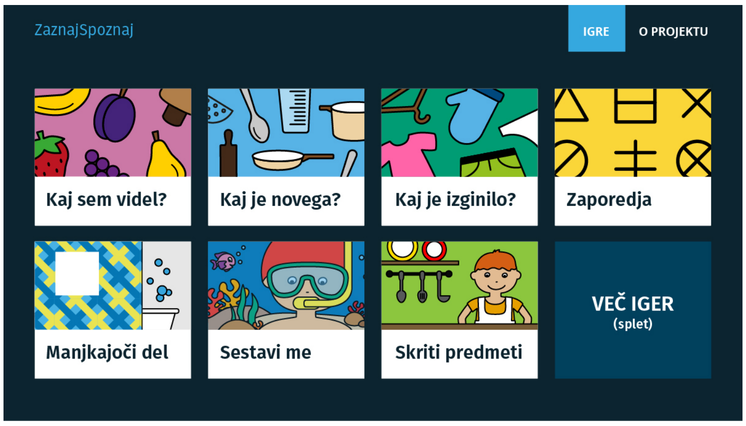


Figure 1: A layout of available games on a mobile platform. The default layout includes instances of seven games which can be played offline. The classes which the user is enrolled in, are accessible in online mode.

The focus of the ZaznajSpoznaj platform is to provide a series of modifiable educational games which include concepts that can be applied to a number of learning domains. Since the project focuses on the blind and visually impaired, these games are divided into three related categories: memory training, vision training and ICT inclusion. The memory training section contains games such as a standard memory game and finding the correct subset of the shown items. There are also several variants of these games, e.g. finding exact sequences, inverted subsets etc. The vision training category includes games for pattern matching and its variations. The ICT inclusion category consists of two training activities for learning Braille and touch-typing. All games offer the user and their teacher a supportive environment to achieve the learning goals (see Figure 1).

The ZaznajSpoznaj platform offers these games as modifiable templates by providing a special interface for the teachers. A teacher can take a template, define the

[1]http://allabilitiesplayground.net.au
[2]http://braillebug.afb.org

parameters of the game: the shown items, learning domain, and even the visual outlook of the game — which is important especially for the visually impaired. By modifying the template, teachers can create specific instances of the game for their students with specific needs.

Game templates can be further extended. All games communicate with the central ZaznajSpoznaj server hosting the ZaznajSpoznaj framework via an application programming interface (API). Calls to the API give the templates the information the teacher entered to define their game instances. The game is then customised according to the given parameters of the specific instance based on the received information from the API. Moreover, each game for each student can be personalized. If a user suffers from a specific visual impairment, their personal preferences are adjusted accordingly (e.g. colors, text font, text size etc.) and the entire contents of the game displayed in this personalized manner. We believe that this is an important aspect that increases accessibility of the games.

The framework supports several user roles: administrator, teacher and student. Framework administrators have the ability to add users, create classes, define teachers for classes and upload new templates (for games) and media files (images, sounds).

Each class in the framework has at least one assigned teacher. Teachers can edit their class' information and create new games or applications from the available templates.

A user can be a teacher in one class and a student in another as this role is tied to individual classes. Students are the framework's regular users. They enroll (or are enrolled by teachers) in classes and participate in the activities the classes offer. Each user is able to specify specific preferences, including the level of visual impairment. This setting then tailors the visual display and behavior of the site and games, suitable for the user. The current list of style preferences contains the following possible states: default, blind, inverted colors, high contrast, protanopia, tritanopia and achromatopsia. It is up to the game template developers to determine how each of these is processed in the game itself. ZaznajSpoznaj platform also supports multiple languages. Each template can provide multilingual language files and a user can define their preferred language.

These features of the platform allow for involvement of the community to further expand the array of functionalities developed within the ZaznajSpoznaj project. The API is open to any potential developer who can create new games and offer them for use through the platform. The high customizability of both, the games and user preferences, is important for inclusion of users with various impairments. We expect to increase the number of options in the field of visual impairment and add support others conditions (e.g. ADHD, games for the elderly and others).

4 Preliminary evaluation

The ZaznajSpoznaj project is still under development. The developed features are however continuously evaluated by the staff at Zavod za slepo in slabovidno mladino Ljubljana (ZSSM), who is the leading partner in the project. Due to agile development, we are able to adjust the developed features according to results of their evaluation. Additionally, we performed a preliminary evaluation of the ZaznajSpoznaj platform with three visually impaired children. Each child was evaluated by a teacher who inter-

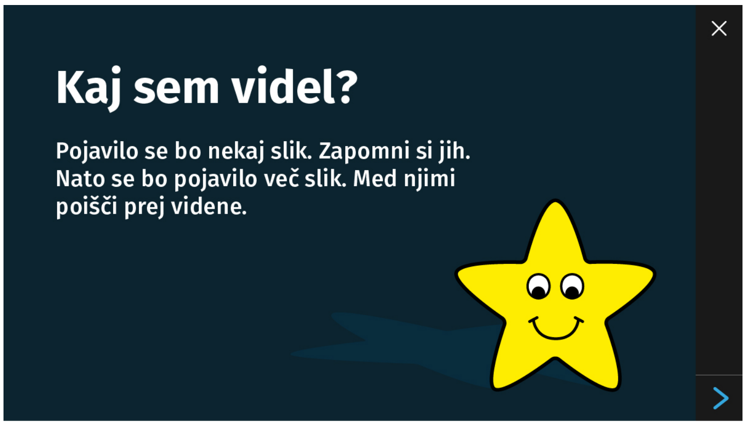

Kaj sem videl?
Pojavilo se bo nekaj slik. Zapomni si jih. Nato se bo pojavilo več slik. Med njimi poišči prej videne.

Figure 2: A layout with game instructions, accompanied by a star character which encourages the student throughout the playtime.

viewed the child and accompanied by an observer who wrote down her observation. A child was presented with a game of finding a subset of displayed images, shown in Figure 3. Children started playing the game on the basic level, where the task was to find two images and if they successfully played the game, the difficulty was automatically increased by increasing the number of images to find. We evaluated children's interactions, their response to success and failure and understanding of instructions displayed by the game.

Our findings show that all three children, aged between 8 and 10, were excited to play the game. The game character provided valuable visual encouragement (as shown in Figure 2) but the children also needed verbal support while playing the game. This finding suggests that we should implement vocal encouragement into the games. Two children needed help after failing several attempts on a higher difficulty level, the third needed help already at the basic difficulty level. One child needed additional explanation of the game, while another suggested modifying the appearance of the game to his favourite colors.

5 Conclusion

The ZaznajSpoznaj project provides an open-source platform and a template standard for accessible and inclusive learning games. Even teachers without specific IT-knowledge can freely modify content and customize games according to the needs and preferences of visually-impaired school children. This way they can enrich and diversify the learning process and include motivating and accessible elements of game-based learning into inclusive learning scenarios.

Although we only performed an initial evaluation on target users of the platform, it has already shown positive acceptance of the ZaznajSpoznaj concept by teachers and school children. Therefore ZaznajSpoznaj can be considered a significant advancement in using ICT for inclusion of blind and visually impaired children.

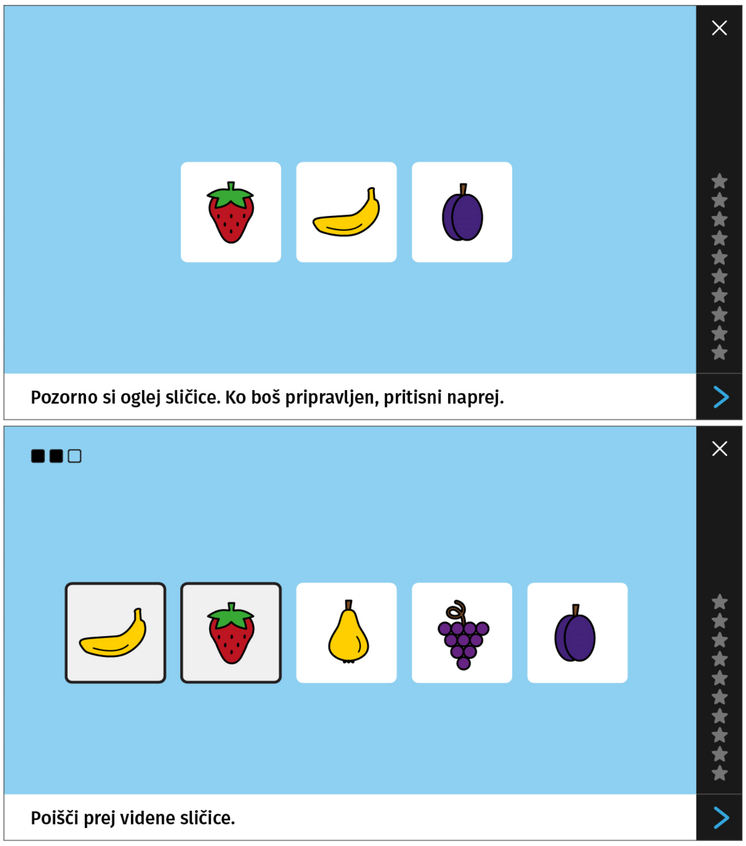

Pozorno si oglej sličice. Ko boš pripravljen, pritisni naprej.
Poišči prej videne sličice.

Figure 3: A screenshot of the game played during the evaluation. The user was asked to find the images (top), which are displayed as the subset of images (bottom). The number of images which need to be selected is indicated at the top left corner. The stars on the right side of the screen indicate the user's progress, which is tracked throughout the game.

We will base our future work on the results of our first evaluation stage. The next stage of evaluation will focus on the platform's interface and support for teachers to customize the games according to preferences and impairments of individual users. We further intend to evaluate the interface and to create and provide templates for new games with master students of the e-Learning course at the Faculty of computer and information science, University of Ljubljana. This will also help to extend the set of available games for our primary target groups and introduce new domains of impairments (such as ADHD).

References

[1] Mohamed Ally, Margarete Grimus, and Martin Ebner. Preparing teachers for a mobile world, to improve access to education. *PROSPECTS*, 44(1):43–59, February 2014.

[2] LarsBallieu Ballieu Christensen and Tanja Stevns. Biblus – A Digital Library to Support Integration of Visually Impaired in Mainstream Education. In Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs SE - 6*, volume 7382 of *Lecture Notes in Computer Science*, pages 36–42. Springer Berlin Heidelberg, 2012.

[3] Levent Görgü, Abraham G. Campbell, Kealan McCusker, Mauro Dragone, Michael J. O'Grady, Noel E. O'Connor, and Gregory M. P. O'Hare. FreeGaming: Mobile, Collaborative, Adaptive and Augmented ExerGaming. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia - MoMM '10*, volume 8, pages 287–301. ACM Press, 2010.

[4] Karin Leichtenstern, Elisabeth André, and Thurid Vogt. Role Assignment Via Physical Mobile Interaction Techniques in Mobile Multi-user Applications for Children. In Bernt Schiele, AnindK. Dey, Hans Gellersen, Boris Ruyter, Manfred Tscheligi, Reiner Wichert, Emile Aarts, and Alejandro Buchmann, editors, *Ambient Intelligence SE - 3*, volume 4794 of *Lecture Notes in Computer Science*, pages 38–54. Springer Berlin Heidelberg, 2007.

[5] Klaus Miesenberger and Reinhard Ruemer. Schulbuch Barrierefrei (Accessible School Books) – Co-operation Between Publishers and Service Providers in Austria. In Klaus Miesenberger, Joachim Klaus, WolfgangL. Zagler, and ArthurI. Karshmer, editors, *Computers Helping People with Special Needs*, volume 4061 of *Lecture Notes in Computer Science*, pages 32–39. Springer Berlin Heidelberg, 2006.

[6] Lauren R Milne, Cynthia L Bennett, and Richard E Ladner. VBGhost: A Braille-based Educational Smartphone Game for Children. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 75:1—-75:2, New York, NY, USA, 2013. ACM.

[7] Lauren R Milne, Cynthia L Bennett, Richard E Ladner, and Shiri Azenkot. BraillePlay: Educational Smartphone Games for Blind Children. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '14, pages 137–144, New York, NY, USA, 2014. ACM.

[8] Marko Periša, Dragan Peraković, and Vladimir Remenar. Guidelines for Developing e-Learning System for Visually Impaired. In *Universal Learning Design*, page 115, Berlin; Heidelberg, 2012. Springer-Verlag.

[9] Florian Schimanke, Robert Mertens, Oliver Vornberger, and Stephanie Vollmer. Multi Category Content Selection in Spaced Repetition Based Mobile Learning Games. In *2013 IEEE International Symposium on Multimedia*, pages 468–473. IEEE, December 2013.

[10] Mark West and Steven Vosloo. *UNESCO Policy guidelines for mobile learning*. United Nations Educational, Scientific and Cultural Organization, Paris, 2013.