ISSEP 2019

12th International conference on informatics in schools
Situation, evaluation and perspectives

Organizers:
University of Cyprus

18-20 November 2019
Larnaca, Cyprus

Edited by
Eglė Jasutė and Sergei Pozdniakov

ISBN 978-9925-553-27-3

The conference website: http://cyprusconferences.org/issep2019/

ISSEP 2019 COMMITTEES

General and Organising Chair

George A. Papadopoulos
University of Cyprus

Steering Committee

Valentina Dagienė
Vilnius University, Lithuania (Steering Committee Chair)
Andreas Bollin
University of Klagenfurt, Austria
Yasemin Gulbahar
Ankara University, Turkey
Juraj Hromkovič
Swiss Federal Institute of Technology Zurich, Switzerland
Ivan Kalas
Comenius University, Slovakia
George A. Papadopoulos
University of Cyprus, Cyprus
Sergei Pozdniakov
Saint Petersburg Electrotechnical University, Russia

Program Committee

Sergei Pozdniakov (Program Committee Chair)
Saint Petersburg Electrotechnical University, Russia
Erik Barendsen
Radboud University Nijmegen and Open Universiteit, Netherlands
Andrej Brodnik
University of Ljubljana, Slovenia
Christian Datzko
SVIA-SSIE-SSII, Basel, Switzerland
Ira Diethelm
Oldenburg University, Germany
Michalis Giannakos
Norwegian University of Science and Technology, Norway
Bruria Haberman
Holon Institute of Technology, Tel Aviv, Israel
Peter Hubwieser
Technical University Munich, Germany
Petri Ihantola,
Tampere University of Technology, Finland
Mile Jovanov
Sts. Cyril and Methodius University, Macedonia
Dennis Komm
Pedagogical University Chur, Switzerland
Mark Laanpere
Tallinn University, Estonia
Peter Micheuz
University Klagenfurt and Gymnasium Völkermarkt, Austria
Mattia Monga
Università degli Studi di Milano, Italy
Ralf Romeike
University Erlangen (FAU), Germany
Giovanni Serafini
Swiss Federal Institute of Technology Zurich, Switzerland
Maciej M. Syslo
Nicolaus Copernicus University, Toruń, Poland

Program at a glance

Monday, 18 November 2019

08:00 - 09:00	Registration
09:00 - 10:00	**Session 1** **Invited Speaker:** Charoula Aggeli
10:00 - 10:30	Coffee Break
10:30 - 12:30	**Session 2**
10:30 - 10:55	**Empowering the teachers with the NAPOJ - a grassroots movement towards computing teachers community of practice** *Andrej Brodnik and Matija Lokar*
10:55 -11:20	**An Exploration of Teachers' Perspective about the Learning of Iteration-Control Constructs** *Emanuele Scapin and Claudio Mirolo*
11:20 - 11:45	**Observing Abstraction in Young Children Solving Algorithmic Tasks** *Josina I. Koning, Hylke Faber, Menno Wierdsma, Henderien Steenbeek and Erik Barendsen*
11:45 -12:10	**Informatics Education in School: A Multi-Year Large-Scale Study on Female Participation and Teachers' Beliefs** *Enrico Nardelli and Isabella Corradin*
12:10 - 12:30	**Teachers' Perspectives on Artificial Intelligence** *Annabel Lindner and Ralf Romeike*
12:30 - 14:00	Lunch break
14:00 - 15:35	**Session 3**
14:00 - 14:25	**Machine Learning Unplugged - Development and Evaluation of a Workshop about Machine Learning** *Elisaweta Ossovski and Michael Brinkmeier*
14:25 - 14:50	**About Classes and Trees: Introducing Secondary School Students to Aspects of Data Mining** *Andreas Grillenberger and Ralf Romeike*
14:50 - 15:15	**Cybersecurity Within the Curricula of Informatics: The Estonian Perspective** *Birgy Lorenz, Kaido Kikkas, Tiia Sõmer and Edmund Laugasson*
15:15 - 15:35	**Comparing Approaches for Learning Abstraction and Automation by Object Orientation** *Arno Pasternak and Johannes Fischer*
15:35 - 16:00	**Coffee Break**
16:00 - 17:30	**Session 4** **Poster Session**

Tuesday, 19 November 2019

08:00 - 09:00	Registration
09:00 - 10:00	**Session 5** **Invited Speaker:** Alexei Semenov
10:00 - 10:30	Coffee Break
10:30 - 12:30	**Session 6**
10:30 - 10:55	**Holistic STEAM education: A perspective on training future teachers** *Arnold Pears, Erik Barendsen, Valentina Dagienė, Vladimiras Dolgopolovas and Eglė Jasutė*
10:55 -11:20	**Inquiry-based Learning in Computer Science Classroom** *Zuzana Tkáčová, Ľubomír Šnajder and Ján Guniš*
11:20 - 11:45	**Introducing Informatics in Primary Education: Curriculum and Teachers' Perspectives** *Valentina Dagienė, Tatjana Jevsikova and Gabrielė Stupurienė*
11:45 -12:10	**Unplugged Activities in the Context of AI** *Annabel Lindner, Stefan Seegerer and Ralf Romeike*
12:10 - 12:30	**International Comparative Study on Computational Thinking Education in K-12** *Yang Xing and Jin-Bao Zhang*
12:35 - 14:00	Lunch break
14:00 - 15:35	**Session 7**
14:00 - 14:25	**The Genesis of a Bebras Task** *Christian Datzko*
14:25 - 14:50	**From Bebras tasks to lesson plans - Graph data structures** *Lucia Budinská, Karolina Mayerova*
14:50 - 15:15	**Situated learning with Bebras tasklets** *Carlo Bellettini, Violetta Lonati, Mattia Monga, Anna Morpurgo and Martina Palazzolo*
15:15 - 15:35	**Facts, Figures, Remarks and Conclusions about the Austrian Beaver Contest with Regard to Computational Thinking and Computer Science Domains** *Peter Micheuz, Stefan Pasterk and Andreas Bollin*
15:35 - 16:00	**Coffee Break**
16:00 - 17:30	**Session 8**
16:00 -16:25	**PrivaCity. A Chatbot Game to Raise Privacy Awareness among Teenagers** *Erlend Bergen, Torjus H. Sæthre and Monica Divitini*
16:25 - 16:50	**Computer Science Problem Solving in the Escape Game "Room-X"** *Alexander Hacke*
16:50 -17:15	**CITY: A Game to Raise Girls' Awareness of Computing** *Evelyn Saxegaard and Monica Divitini*
17:15 - 17:35	**How beginner-friendly is a programming language? A short analysis based on Java and Python examples** *Jean-Philippe Pellet, Amaury Dame and Gabriel Parriaux*

Wednesday, 20 November 2019

08:00 - 09:00	Registration
09:00 - 10:00	**Session 9** **Invited Speaker:** Arnold Pears
10:00 - 10:30	Coffee Break
10:30 - 12:30	**Session 10**
10:30 - 10:55	**What Are Computer Science Educators Interested In? The Case of SIGCSE Conferences** *Noa Ragonis and Orit Hazzan*
10:55 -11:20	**Implementing a bidirectional Logo Debugger** *Renato Menta, Serena Pedrocchi, Jacqueline Staub and Dominic Weibel*
11:20 - 11:40	**Networked Embedded Systems in the Physical Computing Project "Smart City"** *Mareen Przybylla, Andreas Grillenberger and Andreas Schwill*
11:40 -12:05	**Wandering microbits in the public education of Hungary** *Andor Abonyi-Tóth and Zsuzsa Pluhár*
12:05 - 12:30	**Introduction to Computational Thinking for university students** *Zsuzsa Pluhár and Hajnalka Torma*
12:30 - 14:00	Lunch break
14:00 - 15:20	**Session 11**
14:00 - 14:25	**Person-Thing-Orientation and the Choice of Computer Science Courses in High School** *Michael Brinkmeier and Jascha Kemper*
14:25 - 14:50	**Enhancing Student Engagement in Multidisciplinary Groups in Higher Education** *Michael Opoku Agyeman*
14:50 - 15:20	**Understanding Artificial Intelligence - A Project for the Development of Comprehensive Teaching Material (work in progress)** *Michael Schlichtig, Simone Opel, Carsten Schulte and Lea Budde*
15:20 - 15:50	**Coffee Break**
15:50 - 16:50	**Session 12**
15:50 -16:10	**Two programming languages friendly to lower secondary blind students** *Ľudmila Jašková, Natália Kováčová and Mária Karasová*
16:10 - 16:30	**Accessibility of the Computer Science study for students with visual impairment** *Mária Stankovičová*
16:30 -16:50	**Music Computer Technologies in IT Training for Students with Deep Visual Impairment** *Irina Gorbunova, Anastasia Govorova and Alex Voronov*

Content

Papers

Posters

Music Computer Technologies in IT Training for Students with Deep Visual Impairment

[1] Herzen State Pedagogical University of Russia, 48, Emb. River Moika, 191186 St. Petersburg, RUSSIA
gorbunovaib@herzen.spb.ru

Abstract. The article analyzes the processes of information, transforming the environment of training of pupil with visual impairment. There is a necessity to change the content of the information education in connection with the use of specialized software, and hardware and digital educational resources. Special rehabilitation centers that teach computer science and contemporary information technology (IT) and allow student with visual impairment to adapt to the development and independent use of specialized IT, help in solving a number of important problems. The tiflotechnik class implements contemporary technological capabilities for training visually impaired information and music computer technologies. Analysis of the peculiarities of the use of music computer technologies (MCT) by rehabilitators with visual impairments shows that currently blind musicians have the opportunity to master a number of MCT-programs (sequencers, audio editors, etc.), which contributes to the most profound disclosure of their creative potential.

The development of IT and MCT offer unique opportunities for visually impaired students to provide and receive information almost in full. The authors of article describe the experience of working with students with deep visual impairment in the process of teaching them contemporary IT.

Keywords: Information Technology, Inclusive Education, Informatics, Music Computer Technologies Tiflotechnik Class, Visually Impaired

1 Introduction

It is believed that vision provides more than 80% of information about the outside world. The blind or visually impaired person is partially limited in the choice of the source of information. Most of these people do not have the opportunity to read books, newspapers, magazines and other flat-printed publications, navigate in space and move through streets without of an accompanying person’s help. The development of information technologies (IT) opens up unique prospects for pupil with visual impairment (we are talking about more severe cases, although they include [4], [5]) to provide and receive information almost in full. With the use of specially developed technologies, blind pupil are able to use such actions as creating, processing and editing electronic texts, reading flat-screen literature, printing materials from electronic formats in special Braille; maintaining various kinds of databases, communicating in social networks, as well as searching and placement of information on the Internet. All this gives the blind person the opportunity to get a profession corresponding to his/her interests [13], [15].

Special rehabilitation centers that teach music computer technologies (MCT) and contemporary IT and allow pupil with visual impairment to adapt to the development and independent use of specialized IT, help in solving a number of important problems.

2 Tiflotechnic Class: Teaching IT and MCT for Students with Visual Impairment

The workplace for the rehabilitant, who is trained according to the methodology developed by us, consists of the following components:

a. soft computer chair, which has the ability to pick up the height so that it is conven-ient for those who have a small balance of vision to look at the monitor screen;

b. two-tier computer desk where the student can place his/her hands for the convenience of free movement of the computer mouse, and, if necessary, the use of a Braille display for more accurate display and development of the information received;
c. the monitor screen diagonal of nineteen inches, set to the highest resolution and font magnification for people with residual vision;
d. on both sides of the monitor there are active speakers for playing audio files, as well as speech output using the system of voice assistants and the screen access programs; for sound playback, you can use headphones provided at each workplace;
e. high-quality vocal microphone, which allows rehabilitants to record and process their own voice;
f. on the left of the table there is a special double-sided high-speed Braille printer for printing on double-format sheets (newspaper format). This device allows you to print out electronic texts in Braille, so that the rehabilitant could not only hear the information reproduced by the speech synthesizer, but also read it;
g. scanning and reading machine, which provides the opportunity to make available to blind and visually impaired people a wide range of printed materials using the latest technology of optical character recognition of a scanned text which is read aloud in the chosen language and a user-defined voice, adjustable speed, volume, tone, listening material; it is also possible to conserve the information on digital media;
h. television, which is designed to read digital talking books, recorded on flash cards in a specially designed format. The player contains electronic memory and electronic bookmarks, which makes it easy to find the desired fragment, continue playback from the desired episode and move through the text, adjusting the speed of speech playback;
i. portable Braille display allows blind users to freely navigate the Windows operating system. All buttons can be individually customized to the needs of a particular user. The wheels allow you to scroll lines, sentences, paragraphs and entire documents. There is an additional way to navigate using the joystick or the rocker button. Using a combination of scroll buttons, rocker buttons, and curs or buttons, you can easily navigate through the body of a document of any format;
j. stationary video magnifier, which allows visually impaired people to read books, magazines, recipes, it allows you to consider the small details of any object;
k. GPS-device for blind and visually impaired users which is a compact device equipped with a navigator with voice control and designed to orient the blind and visually impaired student on the ground. Multi-transport navigation is carried out on foot or by car and you can find the road with the help of voice navigation; digital maps for exploring the area; getting real-time voice description of what surrounds you while walking; obtaining information about your location, etc.

In the class of tiflotechnic we provide a MIDI-keyboard for those who can realize their musical abilities; we have developed and implemented it in the educational process of the special program of teaching computer arrangements for people with impaired vision. The manuals have developed modules and classes aimed at providing opportunities for teaching computer musical creativity blind musicians using the "method of projects" [18] on the basis of music computer technologies (MCT) [10]. This has become possible thanks to the developments of scientists involved in solving the problems of computer modeling of the creative process, in particular, the modeling of the process of musical creativity and the use of its results in the system of contemporary music education [2], [7-9].

In addition to special equipment, there are also some speech programs (JAWS, NVDA, etc.) designed for voice access to the information displayed on the PC screen. This access is carried out by means of computer synthesis of a speech signal with its output on the sound-transmitting device. This allows the blind user to work with a wide range of programs for various purposes, using a text display mode, including text editors, table processors, programming systems, some educational games. (Read more about the possibilities of teplotehnich class analysis for teaching IT and MCT to students with visual impairment written in [11], [14].)

3 The Main Modules of IT Training for Visually Impaired Students

The main modules of IT training for visually impaired student consist of the following thematic blocks:

1. Familiarity with the special equipment installed in the class of tiflotechnic, the main features and possibilities of its practical application (more detailed information see: [14]).
2. The computer device and operating system. Computer capabilities for a person with visual impairment. It sets up the operating system for the correct operation of the voice assistant (JAWS, NVDA, etc.).

3. Work with a computer keyboard, the location of the keys and the main keyboard functions. Here we talk about the specific purpose of each key, what function it performs when working in the operating system, study the location of the keys on the number and letter rows. For those who want to learn how to type quickly, there are special keyboard simulators with voice prompts and special tasks for spell checking and error detection in fast printing.
4. Basic principles of work with programs of screen access (“voice assistants”), voicing the information displayed on the screen, which helps the blind or visually impaired person to freely navigate in the settings and commands of the operating system, to work with electronic, audio and video materials, to sound work on the Internet; system requirements and features that must be observed when using them (more detailed information see: [13]).
5. Principles of operation of operating systems of the ‘Windows’ line (desktop, start menu and context menu, basic system keyboard commands required by the user when working on the computer, etc. (more detailed information see: [14]).
6. What are “hot keys” and how they can help to make the work easier for the blind and visually impaired, not using a computer “mouse” in the work.
7. Working with files and folders using keyboard shortcuts, editing, filling and moving. Basic concepts about extensions and file types (audio, video, photos, etc.).
8. Work with data (information), transfer data to various removable media (flash drives, removable hard drives and CDs). Working with data without using a mouse, there are system commands for such operations. Archiving the information, to configure the archiver. Basic combinations for saving and deleting information from the hard drive or removable media. Types of media for high-quality and reliable storage of information.
9. Working with dialogs and pop-up system messages, what to do when the system messages appear, when you need to minimize or expand the system window, what key combinations you can use to switch between Windows, and what is convenient for working with multiple tasks at the same time.
10. Work in popular text editors (word processors, keyboard combinations used for text formatting, creation and formation of tables, spell checking, correction of spelling and syntax errors, the main differences between editors of different years of release, archiving of documents and text files to save space on the hard drive and removable media) (more detailed information see: [13]).
11. Scanning of books, documents and photos, translation of texts from paper to electronic versions, recognition of scanned flat-printed texts with the help of special programs to be able to reproduce them by speech synthesizers. Setting the speed of reading playback, choosing the voice tone, the best for the perception of electronic voice (more detailed information see: [12]).
12. Work with a “talking” electronic library, creation of electronic bookmarks for the convenience of listening, finding and keeping the fragments of the work, the translation of electronic texts in audio format for readability, “pocket” computers, mobile phones, tablets and iPods (more detailed information see: [11]).
13. Work with sound and music, listening to audio files, watching videos with the help of various programs of players. Practical work in popular audio editors, easy to use for the blind and visually impaired, digitization of rare unique recordings from magnetic tapes and other analog media, cleaning of noise and other extraneous sounds without loss of quality, conversion of audio and video materials, transfer from one format to another to save space on the hard disk and removable media (more detailed information see: [8]).
14. Work with e-mail, check the e-mail address, sending and receiving e-mail using a special e-mail clients come with features e-mail. Main shortcuts that are used when sending and receiving e-mail attachments (more detailed information see: [14]). 15) The principle of operation of the blind and visually impaired in the global Internet, with the help of voice assistants voice information is displayed on the screen, the main features of the search and download information from the Internet resources, “hot keys” to speed up the network and get quick access to the desired information, as well as the ability to quickly find the desired link (more detailed information see: [14]).
15. Work in popular search engines, search for the desired information using speech synthesizers (voice assistants), the presence of key combinations to speed up the work and sort out the data. Search and exchange of information in the global network and data exchange systems, the process and features of data acquisition, setting up special and additional programs, voicing information about the received data, speed, ratings, percentage of reception and transmission of information (more detailed information see: [7]).
16. Getting to know and work with social networks, posting personal information, searching for people by groups and interests, inviting them to participate and viewing the news feed. Work in networks that support not only e-mail, but also voice communication, such as ICQ, Skype, etc., features and principles of blind and visually impaired people in voice chat. Setting up a special program that alerts the actions taking place on the monitor

screen to carry out simultaneous communication in voice and electronic chats, which greatly increases the productivity of information (more detailed information see: [13]).

17. Methods of self-posting information to the global network, viewing information and, if necessary, save it to your computer, the main keyboard commands that help to navigate the work with the data (more detailed information see: [9]).
18. Familiarity with the thematic discussion of the Internet mailing for blind and visually impaired users, subscription, registration and direct participation in special electronic forums on interests (more detailed information see: [8]).
19. Possibility to work with modern touch displays, control icons with the help of specially provided gestures for the blind and visually impaired. Access to information through the voice assistant, which is provided on touch tablets (more detailed information see: [17]).

The method of teaching IT to visually impaired pupil involves doing homework after each lesson, so that students can analyze, consolidate and test the material in a reallife situation, and, in case of ambiguities, ask the appropriate questions.

4 Conclusion

The problem of teaching a blind student and a student with deep visual impairment contemporary IT lies also in the fact that the contingent is heterogeneous according to the type of existing visual pathologies, by type of visual impairment (total blindness or porcelina), at the time of the occurrence of the defect: blindness is congenital or acquired and so on. The number of hours of the rehabilitant's presence is determined depending on the need and level of his/her initial information training, as well as his/her health.

At the same time, each group of the blind, united on the basis of the severity of visual pathology, is characterized by the presence of certain psychological features that the teacher of information and communication technologies must bear in mind [10], [13]. Knowledge of these features will help to competently and accurately represent the training material necessary for the study of a group of rehabilitators.

Analysis of the peculiarities of the use of MCT by people with visual impairments shows that currently blind musicians have the opportunity to master a number of MCT programs (sequencers, audio editors, etc.), which contributes to the most profound disclosure of their creative potential. The MCT are an indispensable tool of educational process in propagating music masterpieces among the different social groups, as well as a unique technology for implementation of inclusive pedagogical process in training the people with disabilities [16].

Special rehabilitation centers that teach computer and contemporary IT and allow pupil with visual impairment to adapt to the development and independent use of specialized IT (see, for example, in the works [1], [3], [6], [18-21], [23-25]), help in solving a number of important problems.

High-tech information educational environment requires the search for new approaches and fundamentally new systems of education in the School of the Digital Age. “The ICT revolution brings along an increased potential for inclusion and participation, but also risks for exclusion and thus the responsibility for implementing eAccessibility. It provokes a growing number of challenging research questions. Old boundaries of concepts dissolve, new approaches and fresh thinking are formed: not only in technical terms, but also in social, economic, pedagogic and other terms” – G. Kouroupetroglou, General Chair of the16th International Conference on Computers Helping People with Special Needs ICCHP-2018 (July 11-13, 2018, Linz, Austria) said [22].

References.

1. Albouys-Perrois, J., Laviole, J., Briant, C., Brock, A.: Towards a multisensory augmented reality map for blind and low vision people: a participatory design approach. In: *International Conference CHI 2018* (2018)
2. Belov, G.G., Gorbunova, I.B. (2016).Music and Cybernetics. *Music and Time*, 11, pp. 25– 32.
3. Brock, A.M., Truillet, P., Oriola, B., Picard, D., Jouffrais, C.: Interactivity improves usability of geographic maps for visually impaired people. *Hum. Comput. Interact*. 30(2), 156–194 (2015)
4. Borgesa, J.A., Tomé, D. (2014). Teaching Music to Blind Children: New Strategies for Teaching through Interactive Use of Musibraille Software. *Procedia Computer Science*, 27, pp. 19– 27 https://doi.org/10.1016/j.procs.2014.02.004.
5. Brauner D. *Music Technology for the Visually Impaired* (2017). online at http://www.perkinselearning.org/technology/posts/music-technology-visually-impaired
6. Cheng R., Wang K., Lin S. (2018) Intersection Navigation for People with Visual Impairment. In: Miesenberger K., Kouroupetroglou G. (eds) *Computers Helping People with Special Needs. ICCHP 2018. Lecture Notes in Computer Science*, vol 10897. Springer, Cham

7. Chibirev, S.V., Gorbunova, I.B. (2013). Modeling of Musical Creativity Process with the Use of Music Computer Technologies. *Proceedings of Irkutsk State Technical University*, 4 (75), pp. 16–24.
8. Gorbunova, I.B. (2015). Computer Science and Music Computer Technologies in Education. *Theory and Practice of Social Development*, 12, pp. 428– 432.
9. Gorbunova, I.B. (2017). Music *Computer: Modeling the Process of Musical Creativity. The* World of Science, Culture, Education, 4 (65), pp. 145–148.
10. Gorbunova, I.B. (2012). Musical-Computer Technology: the Laboratory. *Mediamusic*. [Online], 1, pp. 5–7 http://mediamusic-journal.com/Issues/1_5.html.
11. Gorbunova, I.B., Govorova, A.A. (2015). Music Computer Technologies as a Means of Teaching People with Visual Impairment Musical Art. *Theory and Practice of Social Development*, 11, pp. 298–301.
12. Gorbunova, I.B., Romanenko, L.Yu., & Rodionov, P.D. (2013). Music Computer Technologies in Formation of Information Competence of the Contemporary Musician. *Scientific and Technical Bulletin of Saint-Petersburg State Polytechnic University. Humanities and Social Sciences*, 1 (167), pp. 39–48.
13. Gorbunova, I.B., Voronov, A.M. (2011). Music Computer Technologies in Teaching the Computer Science for Visually-Impairment Students at Higher Musical Schools. Contemporary Musical Education-2010: *Proceedings of the IX International Scientific and Practical Conference*, pp. 287–290.
14. Govorova, A. A., Voronov, A.M. (2017). Musical-Computer Technologies in Inclusive Education for People with Disabilities of Health Impaired: Analysis of the Existing Problems and Prospects of Use. *The World of Science, Culture, Education*, 2 (63), pp. 210–212
15. Voronov, A.M., Gorbunova, I.B., Kameris, A. & Romanenko, L.Yu. (2013). Music Computer Technologies in the Digital Age School. *Proceedings of Irkutsk State Technical University*, 5 (76), pp. 256–261.
16. Gorbunova, I., Govorova. A. (2018) Music Computer Technologies in Informatics and Music Studies at Schools for Children with Deep Visual Impairments: From the Experience. In: Pozdniakov S., Dagienė V. (eds) *Informatics in Schools. Fundamentals of Computer Science and Software Engineering. ISSEP 2018. Lecture Notes in Computer Science*, vol 11169. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-02750-6_29
17. Goncharova, M.S. , Gorbunova, I.B. (2016). Tablet (mobile) technology for professional music education. *Mediamusic*. [Online], 6. URL: http://mediamusicjournal.com/archive/6.html
18. Halloran, W.F. *The William Sharp "Fiona Macleod" Archive*. School of Advanced Study, University of London, London (2015).
19. Hargreaves, D.J., MacDonald, R. & Miell, D. *How do People Communicate Using Music. Musical Communication*, 1–26. Oxford University Press, Oxford (2005).
20. Hoshino, Y., Motoki, A.: A study on the displaying method to assist the intuitive understanding about braille mirror image. *Educ. Inf. Res*. 33(2), 31–36 (2017)
21. Katz, B., Kammoon, S., Parseihian, G., Gutierrez, O., Brilhault, A., Auvray, M., Truilliet, P., Denis, M., Thorpe, S., Jouffrais, C.: NAVIG: augmented reality guidance system for the visually impaired. *Virtual Reality* 16(4), 253–269 (2012)
22. Kouroupetroglou, G. (2018). *Welcome to ICCHP 2018!* [Online]. Available: http://www.icchp.org/welcome-chair-18
23. Kunz A., Miesenberger K., Zeng L., Weber G. (2018) Virtual Navigation Environment for Blind and Low Vision People. In: Miesenberger K., Kouroupetroglou G. (eds) *Computers Helping People with Special Needs. ICCHP 2018. Lecture Notes in Computer Science*, vol 10897. Springer, Cham
24. Thevin L., Brock A.M. (2018) Augmented Reality for People with Visual Impairments: Designing and Creating Audio-Tactile Content from Existing Objects. In: Miesenberger K., Kouroupetroglou G. (eds) *Computers Helping People with Special Needs. ICCHP 2018. Lecture Notes in Computer Science*, vol 10897. Springer, Cham
25. Weber H. (2018) Increasing Inclusive Capacity of Vocational Education and Training (VET) Organizations with Digital Media and ICT. In: Miesenberger K., Kouroupetroglou G. (eds) *Computers Helping People with Special Needs. ICCHP 2018. Lecture Notes in Computer Science*, vol 10896. Springer, Cham

Two programming languages friendly to lower secondary blind students

[1] Dpt. of Education, Faculty of Mathematics, Physics and Informatics, Comenius University,
84248 Bratislava, Slovakia
[2] Narnia Church Elementary School, 85106 Bratislava, Slovakia
[3] Elementary School for Visually Impaired, 84211 Bratislava, Slovakia
jaskova@fmph.uniba.sk, natalia.kovacova@narniaba.sk, maria.karasova@bee.sk

Abstract. In this paper, we describe our research in the field of teaching programming to lower secondary blind students. We were observing the misconceptions that these students had when acquiring basic programming concepts, such as commands, sequence of commands, and loops with a fixed number of iterations. We used two different programming languages to program simple melodies, stories or rhymes. Our intention was to compare the audio text-based programming language and the physical programming language in terms of their suitability for teaching blind learners of this age. We have found that both languages are suitable and extremely engaging for these students.

Keywords: Blind students, programming, audio programming language, physical programming language.

1 Introduction

Research in the field of teaching computing to lower secondary blind students has been taking place at our department since 2011 [1]. This research has produced pilot versions of programming environments designed for blind students and educational scenarios for teaching programming in these environments [2, 3].

We consider developing algorithmic thinking to be very important for blind students. People with vision loss have much less of a chance in finding a suitable job because of their impairment. However, if they have computer skills, they have a better chance of being employed. It is a great advantage if besides using applications, they have programming skills. A significant number of blind programmers are working successfully in software companies. We are convinced that the sooner the interest of blind learners in this profession is appropriately captured, the easier it will be for them to obtain the necessary qualifications.

The development of algorithmic thinking is also important for blind students who choose to pursue a career outside the IT field. The ability to follow the instructions, or to break down the problem into sub-problems and to describe the procedure for solving them, are important skills necessary for learning, working and living in society.

Key competencies related to programming are also set in the curriculum of computer science for students with visual impairment. In short, students should

- understand **concepts** command, sequence of commands, loop command,
- be able to use a formal notation **for writing** sequence of commands and **for interpreting** sequence of commands.

However, teaching programming requires an appropriate programming tool. A brief analysis of some programming environments in terms of their suitability for blind students is presented in the next chapter. We will describe the text-based audio programming language Alan (see Fig. 1) [4] and the physical programming language Torino [5]. We verified these two environments with blind students (of ages 12 to 14). We describe this verification in Chapter 3 and we summarize the results in Chapter 4. In the last chapter we compare our findings with the findings of other authors and present the conclusions.

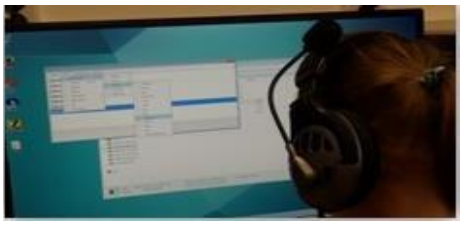


Fig. 1. A blind student is programming in text-based Alan environment

2 PROGRAMMING LANGUAGES FOR BLIND STUDENTS

2.1 Related work

Teaching algorithms and problem solving in a standard class with sighted students is usually done via software that heavily leans on visual representations – Logo [6], Scratch [7] and others. These environments allow the manipulation of virtual objects with a wide variety of commands. But these educational software products are not usable if we teach blind students – who work with computers using a screen reader and the only information they can work with is text and sound.

Block-based languages, like Scratch, are very popular, because these environments remove the syntax complexities and they can be a good choice for an introduction to programming. Milne [8] documented the accessibility challenges of block-based languages and developed Blocks4All prototype environment as a more inclusive alternative for Apple iPad environment. Unfortunately, the iPad is not widely used in Slovak schools yet.

We considered the use of audio programming language (APL) for the blind developed in Chile [9]. The pilot testing (conducted by the Sánchez and his colleagues [10]) showed that programming in this language was too abstract even for older learners (aged 17 to 20) and therefore unsuitable for the younger learners in our group. Nevertheless, it inspired us to create a simpler audio programming language Alan (see Chapter 2.3) for younger students.

Stefik with his team [11] developed an auditory programming environment called Sod-beans with a programming language called Hop. They also developed and verified multi-sensory (sound and touch) curriculum to make text-based programming easier and more accessible for people with visual impairments. But it was designed for older students as well and it is not usable for young children who may still be developing literacy and typing skills.

Horn and his colleagues [12] describe two tangible programming languages, Tern and Quetzal. The Tern language allows children to create a program to control a virtual robot Karel on a computer screen and the language Quetzal to control a LEGO MindstormsTM robot. The authors created these tangible languages for sighted children to facilitate their collaborative development of the programs in the group. For blind children it is problematic to program the robot Karel because they cannot follow his movement on the screen. The Quetzal language we found to be suitable for our blind and partially sighted students. They could create a program by hand and check the robot executing the program by touch. Unfortunately, both languages are not commercially available yet, because their use has been restricted almost entirely to laboratory and research settings. A more suitable solution for blind children is the physical programming language Torino, which we will describe later (see Chapter 2.4).

2.2 Development of own programming environments for blind students

We decided to develop our own programming tool. Our university students developed two programming environments Ladybug [13] and World of Sounds [14]. The Ladybug environment allows a user to control a virtual object moving on a virtual grid (on the computer screen). While the object was moving according to the instructions, coordinates of the passed fields were pronounced verbally by a screen reader to the blind users so that they received feedback. The application World of Sounds used a simple audio programming language to program a sequence of sounds – either selected audio files or read the entered text. However, these environments were not of such high quality to be used in the learning process. But during their verification, we defined what features a programming environment for blind novice programming students must have [15].

- The environment should work with a screen reader. Each response must have an appropriate sound form. It must be fully controllable using a keyboard.

- The environment should be controlled with the mouse and should also be of interest to partially sighted and able-bodied students - classmates of blind students.
- Basic commands should either be used to play different sounds, or to pronounce text. In addition, language should include more complicated commands that lead students to understand the concepts of loop, conditional statement, variable, and procedure.
- Block-based features – the user does not have to edit the language commands but selects them from the list.
- The result of the program should be a sequence of sounds (audio file or spoken text) that are played in the specified order.

2.3 Programming environment ALAN

We proposed a language that meets the criteria mentioned above. According to this proposal, a student of applied informatics has created the programming environment ALAN (see Table 1) as part of his bachelor thesis [4].

Table 1. Program in Alan environment

```
Say(What animal makes this sound?)
Repeat 3 times
   Play(donkey)
End of Repeat
Question(Is that a donkey?) Answer: Yes
   Say(Excellent!)
Else
   Say(No, it is a donkey) End of Question
```

The Alan environment allows a user to program audio stories or simple melodies. The environment uses a simple programming language with two basic commands: Play and Say. The Play command plays a sound that a student can choose from among several categories - Music, Tones, Story, Animals. However, the teacher can add any additional sounds and categories to the menu. The Say command says the specified text. In addition, the program offers the possibility to use a loop command with a fixed number of iterations. Furthermore, variables, branching commands, and subroutines can be used in the program as well. The environment was created primarily for blind students, so it can be fully controlled using the keyboard. From an educational point of view, the environment enables blind students to learn all the knowledge and skills required by the curriculum in a fun way.

2.4 Physical programming language Torino

While we were developing the Torino environment, the staff in the Microsoft Research Lab - Cambridge were developing the Torino (see Fig. 2) physical programming language [5]. This language enables to teach programming concepts and computational thinking to children aged 7-11 regardless of level of vision. To create code, the children connect command pods and adjust their settings to create music, stories, or poetry.

Individual commands are represented by tangible parts – modules. The sequence of modules creates a thread. Each thread can be connected to one of the four inputs of the main module (hub), whereas at each input we can choose a different set of sounds.

Torino is based on Sonic Pi language[1] developed by professor Aaron [16]. Currently, Torino has four types of modules (instruction pods) that students can use to create programs [17]:

- **Play pod** instructs the program to play a sound that can be altered using a dial that rotates through the number of available sounds.
- The **pause pod** adds a temporal suspension to the program.
- The **loop pod** enables a sequence of commands to be repeated.
- The **selection pod** enables different paths to be taken through a program depending on the values set on the two dials. The **merge pod** is used to merge the two paths back together so the program can continue after the commands in the chosen path have been carried out.

[1] http://sonic-pi.net/

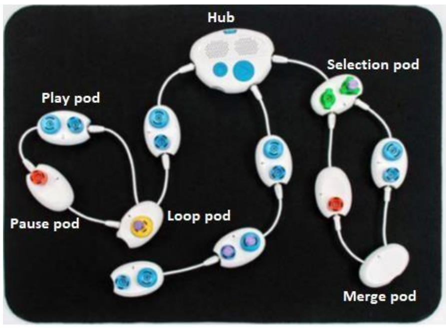
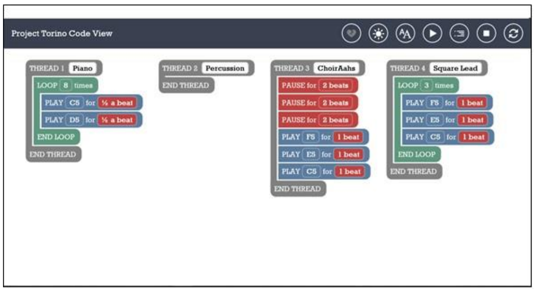


Fig. 2. Physical programming language Torino (source of image [5])

The main module (hub) communicates via Bluetooth with the service software (see Fig. 3) running on a computer or tablet. This program allows the teacher to set up individual categories of sounds that students use when creating programs.



```
THREAD 1 Piano
  LOOP 8 times
    PLAY C5 for ½ a beat
    PLAY D5 for ½ a beat
  END LOOP
END THREAD

THREAD 2 Percussion
END THREAD

THREAD 3 ChoirAahs
  PAUSE for 2 beats
  PAUSE for 2 beats
  PAUSE for 2 beats
  PLAY F5 for 1 beat
  PLAY E5 for 1 beat
  PLAY C5 for 1 beat
END THREAD

THREAD 4 Square Lead
  LOOP 3 times
    PLAY F5 for 1 beat
    PLAY E5 for 1 beat
    PLAY C5 for 1 beat
  END LOOP
END THREAD
```

Fig. 3. A software running on the computer with a text (block-based) version of program

Except of command pods there are plugs that can be inserted into the dials on the pods to allow further expansion of concepts to include constants and variables.

3 DESCRIPTION OF THE RESEARCH PROJECT

3.1 Aims and methods

In our research, we explored the acquisition of the following concepts by lower secondary blind students (without previous programming experience) - **a command**, **a sequence of commands**, and **a loop command**. We focused on the following two goals.

- To observe the misconceptions of blind students in solving tasks requiring the use of a sequence of commands and a loop with a fixed number of iterations.
- To explore the advantages and disadvantages of programming in Alan and Torino environments (in solving tasks aimed at using a sequence of commands and a loop with a fixed number of iterations).

There are not many blind students aged 14-16 in Slovakia, so our research sample was small. Therefore, we decided to use a case study qualitative research strategy. We had chosen the following research methods to collect and analyse data [18, 19].

- Observation of students (at computer science lessons).
- Semi-structured interview (with students and teachers).
- Analysis of educational outcomes of students.
- Analysis of video and audio recordings (done during computer lessons). Questionnaires (for students and teachers).

3.2 Participants

Our research involved five visually impaired lower secondary students. The observed students used a screen reader to work with the computer. Some students with severe vision impairment also used magnification software.

We tried to find out the most relevant information about the students studied. We conducted interviews with classroom teachers, special teachers and informatics teachers.

Table 2 shows the most important characteristics of these students.

Table 2. Characteristics of students

Student	Grade	Gender	Impairment	Computer skills
A	6	Girl	Severe low vision	Above average
B	6	Girl	Blind, Muscular dystrophy	Slightly below average
C	7	Boy	Blind	Slightly above average
D	7	Boy	Blind	Slightly above average
E	7	Girl	Severe low vision	Slightly above average

3.3 Educational activities with Alan

We developed an educational scenario for teaching programming in the Alan environment over the course of five lessons [20]. The aim of the first lesson was to introduce the Alan programming environment and two basic language commands - Say and Play. During the second lesson, students practiced using the basic commands to create longer sequences. The task was to create audio stories by combining spoken words with the set of sounds. During the third lesson, the students learned about the loop concept. The aim of the fourth lesson was to teach students how to debug the program. The fifth lesson was focused on analysing codes of programs with a loop command in terms of accuracy and efficiency. We also included a task to determine whether the result of two different programs is the same.

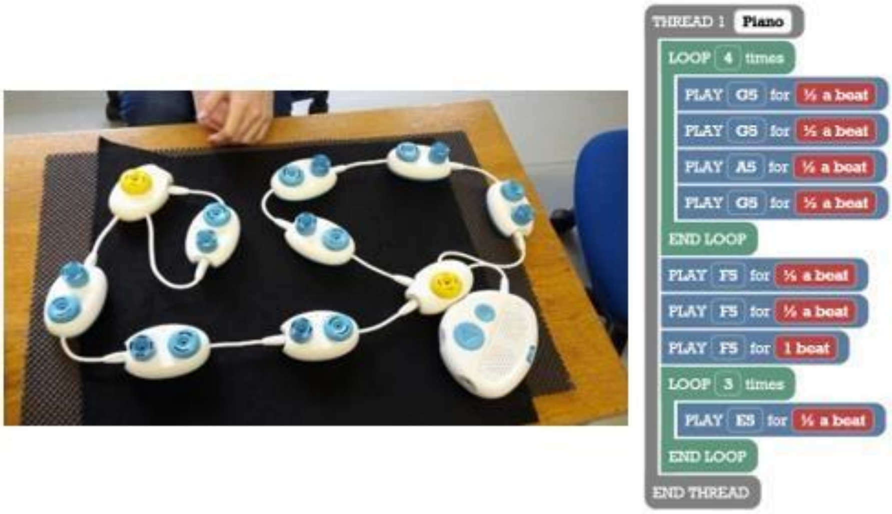
We implemented this educational scenario with two groups of students, in one group there were students from sixth grade (A and B), in the second group there were students from seventh grade (C, D, E). Each student had a computer (with headphones, screen reader and Alan environment). We discussed the tasks with students either in the group or individually. The aim of the discussions was not to disclose the solution to the students, but to bring them to the solution by asking questions.

3.4 Educational activities with Torino

While creating the educational scenario for the Torino environment, we were inspired by the teacher's manual created by the Torino developers. However, we decided to adjust most of the tasks so that students could create stories and songs in their native language and not in English. The scenario was planned for five lessons. During the first lesson, we focused on gaining basic skills in working with the physical programming language Torino. Students became familiar with the main module, as well as with the procedure for connecting play pods to the sequence and running the program.

During the second lesson, the students experimented with setting the sound parameters on the play pods. During the first two lessons students programmed simple stories or short sections of known melodies and rhymes. The theme of the third lesson was debugging. The tasks were to find and correct the following types of errors: incorrect command in sequence, incorrect order of commands in sequence, missing or redundant command in sequence. The fourth lesson was focused on the use of the loop command, with no commands placed before or after the loop. Tasks that required to place command sequences before or after the loop command were planned for the fifth lesson. We also attempted to assign tasks to use nested loops and sequence of loops (see Fig. 4).

This educational scenario was implemented with the same two groups of students from the educational scenario for Alan. So, students had already programmed in the Alan environment a few months before. We had two programming kits, so some students could use one for themselves and others worked in pairs. Students solved the tasks and when they did not know how to proceed, we helped them by asking questions.



```
THREAD 1 Piano
  LOOP 4 times
    PLAY G5 for ½ a beat
    PLAY G5 for ½ a beat
    PLAY A5 for ½ a beat
    PLAY G5 for ½ a beat
  END LOOP
  PLAY F5 for ½ a beat
  PLAY F5 for ½ a beat
  PLAY F5 for 1 beat
  LOOP 3 times
    PLAY E5 for ½ a beat
  END LOOP
END THREAD
```

Fig. 4. Students create a program with two loops (physical and text version of program)

4 Results

In our research, we have focused on misconceptions related to concepts − command, sequence of commands, loop command with a fixed number of iterations. We also wanted to verify the suitability of Alan and Torino environments to implement these concepts. The observed students worked with enthusiasm in both environments. In the Alan environment, students actively used various keyboard shortcuts. Editing the program did not cause them any major problems.

We have noticed that the physical programming language is more appropriate if the students work in pairs. They could cooperate better. However, a larger group is not appropriate because each student wants to feel by touch the sequence of pods. Ideally, each student should feel by touch which command is being executed. Thieme and the team of authors [17] also verified the Torino language with five pairs of students with different degrees of visual impairment, and they also found it to be a suitable programming tool supporting collaborative learning in pairs.

Regarding misconceptions, the students had no problems in solving tasks where it was necessary to create sequences of commands without loops. Problems were observed only when solving tasks that required the use of the loop command. We will now mention the observed problems for each programming environment.

In the ALAN environment, we noticed the following misconceptions.

- **Incorrectly placed command at the end of loop body instead of placing it outside of the loop.** Table 3 contains a comparison of the student's solution (the second column) and the correct solution (the third column). The task was to correct the program in the first column so that the gong sound was played four times instead of three times. The added command is formatted in bold and italics.

Table 3. Addition of missing command *Play(Gong)*

Original program	Student's solution	Correct solution

Repeat 3 times
    Play(Gong)
End of Repeat
	Repeat 3 times
	      Play(Gong)
	      Play(Gong)
	End of Repeat
		Repeat 3 times
		  Play(Gong)
		End of Repeat
		Play(Gong)

- **Misunderstanding the execution of commands placed in the body of the loop command.** We often encountered that students understand the repetition of a sequence of commands by first repeating the first command, then the next command, and so on, and the loop ends when the last loop command is executed several times. A similar experience has also been described by Grover and Basu [21], who focused on misconceptions in understanding loop in block-based programming language. On a sample of 100 high school students, they found that some of them understood the repetition of the sequence of commands in the loop as we had seen in our students.

- A similar misconception occurred in solving the task in which students should decide **whether the result of two different programs is the same**. The following two programs listed in Table 4 appeared to be the same for several students.

Table 4. The task was to compare the following two programs

Program 1	Program 2

`Repeat 5 times`
`  Play(pig)`
`End of Repeat`
`Repeat 5 times`
`  Play(hen)`
`End of Repeat`
	`Repeat 5 times`
	`  Play(pig)`
	`  Play(hen)`
	`End of Repeat`

In the Torino environment, we have observed the following phenomena.

- Some students were struggling to realize when to put a sequence of **commands before the loop and when inside the loop**.
- The **limited number of command modules** in the language had a motivating effect on the use of loops. Students considered when to use the loop command and when not. For example, they realized that if two or less commands were repeated, it would not be worthwhile to use a loop module.
- Using **nested loops** was challenging. Students preferred solutions without them.

Regarding motivation, we noticed that for the students who were not interested in music, programming of the music was rather difficult and less popular. These students preferred programming poems and short stories.

On the other hand, students with a musical talent preferred the Torino language, which, unlike Alan, allowed them to set the duration of the tones.

Student A preferred to work with the Torino physical programming language. As she is talented in music, programming the melodies was a good motivation for developing her programming skills. However, this student did not have any problems when working with ALAN environment as well.

For student B, we considered it appropriate to alternate several forms of learning programming, as we could not burden her with monotonous activity. Working with the Torino environment was quite difficult for her. She had problems connecting the command modules because of her physical impairment and the teacher had to help her.

Student C preferred to program in ALAN environment. He considered it difficult to connect physical programming modules. However, he liked to work in a group, so using a physical programming language was also useful for him.

Student D was as versatile as student A, so he had no problems with any of languages mentioned above.

Student E enjoyed working with a physical programming language the most. She is used to working in a group and she is also very communicative. Since she is interested in music, programming with a physical programming language has provided her with several opportunities to program music sequences.

5 Conclusion

During our long-term research on the development of algorithmic thinking of lower secondary blind students, we verified various programming environments.

We perceive the programming environments Alan and Torino presented in this paper as follows.

- We consider the Torino physical programming language to be an appropriate tool to teach students programming at an early age. Understanding the meaning of language modules is relatively intuitive and using a loop command is relatively simple. The limiting number of modules can motivate students to actively use the loop command in programs. Clarissa Correa de Oliveira [22] has mapped several different programming environments and considers Torino physical programming language to be the most promising learning tool for teaching blind learners. too.

- We also consider Alan to be suitable for students who have not yet encountered programming. However, they need to be able to use the computer keyboard and to work with files and folders.

We consider the Alan and Torino programming environments to be particularly suitable for lower secondary blind students because these languages are related to blockbased programming languages. Students are thus protected from unnecessary syntactic errors and they can concentrate on algorithms. Mladenović and the team of authors [23] compared the programming languages Scratch (block-based language), Logo and Python (text-based languages) in terms of misconceptions concerning the loop concept. They found that the least mistakes were made by students in a block-based programming language (Scratch).

Finally, the programming environments Alan and Torino allow students to learn programming in a constructivist way in the sense of Papert [24]. According to him, "students learn more effectively if they can solve their own tasks, building their knowledge themselves and expressing ideas through a medium that allows direct experience".

In the future we would like to find out how the observed students will use their knowledge to learn other programming concepts and skills. Students did not need to use variables in the loop command yet. A newer version of Alan was created in the last academic year [25]. In this version, students can define a variable and use it in a loop command. Also, the Torino physical programming language supports this concept. The branch command is implemented in Torino and in the new version of Alan. Thus, we see great scope for further exploration of misconceptions held by blind students when they combine different structured commands in programs.

Acknowledgment
This paper was created as part of a project funded by KEGA grant (Kega 018UK4/2019). Special thanks to the teachers and students of the elementary school for students with visual impairment where we conducted our research. Thanks to Microsoft Research Cambridge for offering us the Torino physical programming language for verification and for their supervision.

References

1. Jašková, Ľ.: How to start with learning informatics at elementary school for blind pupils (in Slovak) In: Trajteľ, Ľ. (eds.) Didinfo 2013, UMB, Banská Bystrica (2013).
2. Jašková, Ľ., Kaliaková, M.: Programming microworlds for visually impaired pupils In: Futschek, G., Kynigos, Ch. (eds.): Constructionism and Creativity 2014, Proceedings of the 3rd international constructionism conference, Östereichische Computer, Gesellschaft, Vienna (2014).
3. Jašková, Ľ.: Can We Use Origami Folding As An Effective Instrument For Developing Algorithmic Thinking Of Blind Pupils?. In: JTIE Journal of Technology and Information Education. Vol. 6, Issue 2, pp. 31-39, Palacký University in Olomouc, (2014), http://www.jtie.upol.cz/clanky_2_2014/JTIE_2_2014.pdf, last accessed 2019/08/29.
4. Kováč, M.: Programming environment for visually impaired pupils (in Slovak), bachelor thesis, FMFI UK, Bratislava (2016).
5. Morrison, C. et al.: Torino: A Tangible Programming Language Inclusive of Children with Visual Disabilities, Human–Computer Interaction (2018).
6. Blaho, A., Kalaš, I.: Imagine Logo Primary workbook. Logotron, Cambridge (2004).
7. Resnick, M. et al.: Scratch: Programming for All. Communications of the ACM, vol. 52, no. 11, pp. 60-67 (2009).
8. Milne, L. et al.: Blocks4All: overcoming accessibility barriers to blocks programming for children with visual impairments. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, p. 69 (2018).
9. Sánchez, J. et. al.: APL: audio programming language for blind learners. In International Conference on Computers for Handicapped Persons, pp. 1334-1341, Springer, Berlin, Heidelberg (2006).
10. Sánchez, J. et. al.: Blind learners programming through audio. In *CHI EA '05*, extended abstracts on Human factors in computing systems, pp. 1769—1772. ACM, New York (2005).
11. Stefik, A. M. et al.: On the design of an educational infrastructure for the blind and visually impaired in computer science. In Proceedings of the 42nd ACM technical symposium on Computer science education, pp. 571-576, ACM (2011).
12. Horn, M. et al.: Designing Tangible Programming Languages for Classroom Use. In: 1st international conference on tangible and embedded interaction, pp. 159—162. ACM, New York (2007), http://www.cs.tufts.edu/~jacob/papers/tei07.horn.pdf, last accessed 2019/08/29.
13. Bečvarová, Ľ.: Programming Environment for Children with Visual Impairment – Programming the Robot (in Slovak), bachelor thesis, FMFI UK, Bratislava (2013).
14. Sucháneková, J.: Programming Environment for Children with Visual Impairment – World of Sounds (in Slovak), bachelor thesis, FMFI UK, Bratislava (2013).

15. Jašková, Ľ.: Blind Pupils Begin to Solve Algorithmic Problems. In: Diethelm, I., Mittermeir, R.T. (eds.): ISSEP 2013, LNCS 7780, pp. 68—79. Springer, Heidelberg (2013).
16. Aaron, S. Code music with Sonic Pi, https://www.raspberrypi.org/magpi-issues/Essentials_Sonic_Pi-v1.pdf, last accessed 2019/08/29.
17. Thieme, A. et al.: Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In: Conference on Designing Interactive Systems, pp. 739–752 (2017), https://doi.org/10.1145/3064663.3064689, last accessed 2019/08/29.
18. Hendl, J.: Qualitative Research. Basic Methods and Applications. (translated into Slovak) Portál, Praha (2005).
19. Creswell, J.: Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research. Pearson Education, New Jersey (2008).
20. Kováčová, N.: The concept of cycle in teaching the programming of blind pupils of lower secondary education (in Slovak), doctoral theses, FMFI UK, Bratislava (2018).
21. Grover, S., Basu, S.: Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In: Proceedings of the 48thACM Technical Symposium on Computer Science Education (SIGCSE '17). Seattle, WA. ACM (2017).
22. De Oliveira, C. C.: Designing educational programming tools for the blind: mitigating the inequality of coding in schools (2017).
23. Mladenović, M.; Boljat, I.; Žanko, Ž.: Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. In Education and Information Technologies, 1-18 (2017).
24. Papert, S.: What is logo? And who needs it?, Logo Philosophy and Implementation. Logo Computer Systems Inc. (1999).
25. Kováč, M. Programming environment for visually impaired lower secondary pupils (in Slovak), master theses, FMFI UK, Bratislava (2018).

Teachers' Perspectives on Artificial Intelligence

Annabel Lindner[1] and Ralf Romeike[2]

[1]Computing Education Research Group, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
[2]Computing Education Research Group, Freie Universität, Berlin, Germany
annabel.lindner@fau.de, ralf.romeike@fu-berlin.de

Abstract. The growing importance of artificial intelligence systems in daily life leads to increasing demands for artificial intelligence (AI) as a topic in schools and the necessity to consider what all students in the 21st century should know about the topic. However, the successful integration of topics related to AI into teaching in K-12 requires proper preparation of teachers for this task. Therefore, we have conducted a questionnaire survey to get a first estimation of computer science teachers' knowledge about the subject area. The teachers, who are already interested in AI, are furthermore asked which competences and types of knowledge they would consider as most important for their students in the field of AI. The questionnaire also investigates the challenges of teaching artificial intelligence. The survey shows that the teachers' knowledge about artificial intelligence is broadly influenced by current "hype" topics and media coverage of AI. Teachers place a slightly higher value on sociocultural and technical knowledge about AI than on pure application oriented competences. Moreover, teachers perceive a lack of adequate teaching materials as well as best-practice examples and tools in the field of artificial intelligence. Despite these challenges, the integration of the topic into computer science education curricula is welcomed.

Keywords: Teachers' Perspectives · CS Education · Artificial Intelligence.

1 Introduction

With recent advances in technology that become visible in, e.g., the impressive skills of DeepMind's game AI AlphaZero [17], the translator DeepL [4], the development of self-driving cars [3], or smart assistants like Siri and Alexa, the importance of artificial intelligence systems in the modern digital world is rising and starts to affect our daily lives significantly. On account of this development, demands for the integration of artificial intelligence into computer science curricula in K-12 and the fostering of competences in the field of AI have been made by various institutions. For instance, the German government has approved a strategy paper on AI that promotes the extension of artificial intelligence education in Germany [2] and the G7 have committed to a vision on the future of artificial intelligence that also includes the area of education [8]. In response, material and curricula for teaching AI in K-12 have been developed worldwide (e.g. [22], [23], [15]).

However, besides providing useful teaching material, a central factor for successfully integrating the topic artificial intelligence into teaching is, apart from an appropriate didactic reduction of the subject area's contents, the adequate preparation and qualification of teachers. Diethelm et al. [5] highlight that material for computer science education developed by CS educators is often not used in an actual class. This circumstance is attributed to the disregard of teachers' perspectives in the development of teaching concepts and the assumption that teachers will be able to garner the required knowledge for any topic on their own. Thus, the inclusion of teachers' perspectives, i.e. the teachers' knowledge of the subject area, their explanatory models and their aims and expectations for class, is considered an important aspect for the development of effective teaching concepts and material that corresponds to the teachers' needs. Therefore, it plays an important role in their Model of Educational Reconstruction for Computer Science Education [5]. To get first insights into the teachers' knowledge about AI and to derive measures to support computer science teachers, we have conducted an online questionnaire survey that examines the current situation. In this context, the following research questions are addressed:

- What general knowledge of artificial intelligence do computer science teachers who are interested in the subject area dispose of?
- Which educational goals are pursued by these teachers in the field of AI and which challenges are they confronted with?

2 Related Work

2.1 Artificial Intelligence

A simple and concise definition of the term artificial intelligence can hardly be found. This is due to the fact that artificial intelligence can have various forms and that the phrase serves as a comprehensive term for different technologies and procedures. Furthermore, the AI community does not even agree upon a concise definition of intelligence as such (c.f. [7]). Therefore, Russell and Norvig [16] assign various definitions of AI to four general approaches of describing artificial intelligence: systems that act humanly, systems that think humanly, systems that act rationally or systems that think rationally. Thus, the central aim of AI is to understand, model and reproduce humanoid or ideal intelligent behavior in artificial systems whereby different techniques are used.

Görz et al. [9] identify a series of disciplines of AI but highlight that there are different opinions on how to systematically order them. They establish search, (human) cognition, knowledge representation and processing as well as inference and logic as central concepts of AI. These aspects are of high relevance in the field of AI and serve as the basis for more specialized theories and methods. In this context, neural networks, machine learning and data mining, dealing with constraints, planning, dealing with uncertain knowledge (by using e.g. statistical methods, fuzzy logic and theory of probability), non-monotonic logic2, case based reasoning3, language processing, and multi-agent systems are identified as subfields or rather theories and methods of AI (cf. [9]). These approaches are employed in different applications, e.g. typical ones for machine learning are classification tasks and pattern recognition. Hence, artificial intelligence cannot be equated with machine learning as it is currently often done in public discourse. Machine learning is rather one of the sub-domains or paradigms of AI and despite it being highly relevant in AI research and applications at present, considerations of the topic artificial intelligence in educational contexts should not be confined to machine learning but also take the other subfields and, in particular, the central concepts and big ideas of AI into account. This is especially important as the discipline of artificial intelligence is still in constant evolution and significant paradigm shifts cannot be ruled out.

In general, there are two different paradigms of representation in AI. Symbolic and sub-symbolic AI differ in the way in which knowledge is represented in the AI system. Symbolic methods explicitly represent knowledge by describing it symbolically, e.g. by using logic to represent facts and rules in expert systems. The sub-symbolic representation refrains from this explicit representation. Instead of manipulating symbolic representations of knowledge to solve problems, knowledge is stored implicitly. For example, in a neural network it is represented as a pattern of different weights, i.e. indirectly and not as composed of individual parts (cf. [9]). This implicates that often it is not explicitly comprehensible, how a sub-symbolic system determines a solution.

2.2 Perspectives of Teachers

Looking at current literature, research on teachers' perspectives on the topic of artificial intelligence is quite rare. Although there is already plenty of material and curricula dealing with the topic of AI in K-12[4], to our knowledge, there are no studies explicitly regarding teachers' perspectives on AI in secondary education and only some for higher education.

Wollowski et al. [20] carry out a survey with both AI instructors and AI practitioners to evaluate the concordance of current practice and teaching of AI in tertiary education. However, this survey does not focus on the instructors' individual knowledge but rather examines their course concepts. They found that instructed topics and the needs of AI practitioners match largely, both instructors and practitioners emphasize the importance of topics like search, knowledge representation and reasoning as well as machine learning. Differences are found in the practitioners' strong focus on systems engineering while educators tend to rather use more playful approaches to introduce basic AI topics. Furthermore, educators take a broader perspective including ethical, philosophical and historical aspects of AI instead of only dealing with AI tools and techniques.

Sulmont et al. [18] explore the pedagogical content knowledge (PCK) necessary for teaching machine learning to university students without a background in CS. In interviews with instructors, they determine preconceptions and barriers the students come across in the field of machine learning and identify the tactics instructors use to face them. The students' preconceptions rather center upon the reputation of machine learning than upon how it

[2] Method of deducing new facts from a base of facts and rules in which new facts can contradict old ones. Contradictions lead to the elimination of one of the facts.

[3] The process of solving problems based on solutions of similar past problems.

[4] e.g. [23], [15], https://experiments.withgoogle.com/teachable-machine, https://machinelearningforkids.co.uk/

works. Consequently, students tend to misconceive AI's abilities or do not consider themselves able to implement machine learning. To overcome these challenges, instructors work through and simulate machine learning algorithms in class. Furthermore, the instructors use specific data sets, real world, open-ended and domain-specific problems as well as extensive visualization. They also find that math and programming are challenges for non-CS students and that, therefore, some instructors tend to omit these aspects when teaching machine learning. They summarize that instructors face a series of challenges when teaching non-CS university students in the field of machine learning, but are able to overcome them by using certain tactics and adequate PCK. Despite that, the results of Wollowski et al. [20] and Sulmont et al. [18] are not transferable to secondary education as the underlying educational goals are not comparable.

Concerning the methodology of gathering teachers' perspectives on CS topics and teaching, a series of different approaches can be found in the literature. Yadav et al. [21] or Gretter et al. [10] are using semi-structured interview protocols to elicit teachers' perspectives. The study of Griffin et al. [11] combines teacher interviews with classroom observations in the form of field notes to analyze teachers' practices and their perspectives on the CS Principles Framework. Another central approach to elicit teachers' perspectives, which is also used in our survey, is conducting questionnaire surveys with closed questions. This method is pursued e.g. by Thompson et al. [19] to assess the adoption of new CS standards by teachers in New Zealand and by Grillenberger and Romeike [12], who have designed a questionnaire to elicit teachers' perspectives on data management – a new field of CS evolving from databases – and find that most teachers possess only a little knowledge about this, like AI, rather new topic of CSE.

3 Methodology

In order to evaluate the teachers' perspectives on artificial intelligence, a questionnaire survey was developed that aims to get a first impression of the teachers' content knowledge about AI as well as their experiences, expected challenges and objectives when teaching the topic. With this structure, the questionnaire focuses on content knowledge, pedagogical content knowledge, technological content knowledge, and technological pedagogical content knowledge of teachers according to the TPACK Model [14], which is a conceptual framework to structure professional knowledge of teachers. These components of the teachers' professional knowledge influence the teachers' views on certain CS topics and their teaching and are, hence, important for the successful development of material and for teacher training. General pedagogical and technological knowledge, which are also part of TPACK, are not subject or topic specific and are, therefore, not explicitly considered in the questionnaire.

The survey was conducted with 37 German secondary school computer science teachers from Bavaria, Berlin, and Brandenburg before they participated in a workshop about unplugged activities on artificial intelligence[5]. Thus, all participants are holding some kind of CS degree, are already interested in AI and are looking for possibilities to deal with the topic in class. However, this does not mean that all teachers have already taught AI or possess content knowledge about AI, the group of participants is rather heterogeneous. Consequently, the teachers have probably dealt with the contents of the subject area in varying degrees of intensity and have different levels of knowledge. This circumstance can help to assess how training material for teachers as well as teaching materials about AI should be arranged to build upon the teachers' previous knowledge.

As this survey (cf. Table 1) has the clear objective to gain a first general overview of the teachers' perspectives on AI, the questionnaire was kept short and consists of only five questions. To assess the teachers' content knowledge on AI, teachers were asked to evaluate certain keywords on their relevance for the subject area artificial intelligence. The keywords were selected based on the contents of widely-accepted reference books for the field of AI ([16], [6], [13], [7], [9]) and their examination of the subject area. The topics identified in the literature by structuring and matching the books' contents were summarized and partly reworded for the benefit of unambiguous comprehensibility. The keywords focus on methods and applications of AI, specific technologies like neural networks are therefore subsumed under these categories (neural networks are considered as one aspect of machine learning). This approach led to the following list of keywords that can also be assigned to the principles of AI identified in the Related Work section ([9], [20]): *Knowledge Representation, Search, Statistical Methods and Theory of Probability, Ethics, Classification, Machine Learning, Reasoning, Robotics, Algorithms, Natural Language Processing, Pattern Recognition, Approximation, Data Analysis*. Furthermore, distractors were added to identify random answers: *Sorting Algorithms, Simulation, Cloud Computing, Turing Machines*. These

[5] See https://ddi.cs.fau.de/schule/ai-unplugged/ for details about these activities.

aspects are not associated with AI in any of the reference books that were taken into account. However, this approach can only give a first indication of the teachers' knowledge of AI and does not provide insights into details or explanatory models, further evaluation of this aspect is, therefore, necessary.

Furthermore, the questionnaire examines if the teachers have already taught artificial intelligence and evaluates educational goals and challenges of AI in class. Adapted to the three perspectives established in the Dagstuhl triangle [1], the educational goals included in the questionnaire can be attributed to either a structural, application-oriented or socio-cultural perspective on the topic of artificial intelligence. These perspectives contribute to a holistic view of the topic. In terms of challenges and difficulties of teaching AI, a series of central factors were integrated into the questionnaire. They include the teachers themselves, teaching materials available and the topic itself. The survey questions are designed as Likert items and have to be answered on six-level Likert scales (cf. Table 1).

Table 1. The specific questions of the survey. All questions except question 2 are designed as Likert items.

1. How strongly do you associate the following aspects with artificial intelligence?

    Knowledge Representation, Search, Statistical Methods and Theory of Probability, Ethics, Classification, Machine Learning, Sorting Algorithms, Simulation, Reasoning, Robotics, Algorithms, Cloud Computing, Natural Language Processing, Pattern Recognition, Turing Machines, Approximation, Data Analysis

    Scale: 1-Very strongly — 2 — 3 — 4 — 5 — 6-Not at all — I cannot assess this.

2. Have you already given lessons on artificial intelligence?

    No.
    Yes, in regular class.
    Yes, in a special seminar.
    Yes, as part of extracurricular activities. Other.

3. I would like to see an explicit curricular integration of the topic AI.

    Scale: 1-Very strongly agree— 2 — 3 — 4 — 5 — 6-Very strongly disagree

4. When it comes to artificial intelligence (AI), I consider it important that all students...

    ...are able to assess ethical implications of AI systems, especially chances and risks for society.
    ...are able to identify the technical limitations of AI systems.
    ...develop a reflected personal attitude towards the use of AI systems.
    ...can use AI libraries (e.g. Tensorflow, Scikit-Learn) in their own programs.
    ...effectively utilize AI systems.
    ...are able to explain the functioning of machine learning processes.
    ...are able to assess the intelligence of AI systems.
    ...compare different methods that are used in AI systems.
    ...are able to identify use cases for AI systems.
    Scale: 1-Very strongly agree— 2 — 3 — 4 — 5 — 6-Very strongly disagree

5. What challenges and difficulties do you perceive when it comes to teaching artificial intelligence?

    I do not have the required expertise in this field.
    There is a lack of suitable teaching materials.
    There are no good best practice examples.
    There is a lack of appropriate tools.
    The subject is too complex.
    There is little or no time to deal with extra-curricular content in class.
    Scale: 1-Very true of me— 2 — 3 — 4 — 5 — 6-Very untrue of me — I cannot assess this.

4 Results

The questionnaire does not indicate how much content knowledge the teachers possess in the individual subdomains of artificial intelligence, but only gathers if the teachers generally have an idea of the field. Consequently, it cannot be determined if teachers who have already taught AI possess a more detailed knowledge about the field. For this reason, the results for teachers with (7 teachers) and without teaching experience (29 teachers) in the field of AI are not considered separately. The following aspects were found:

1. *The teachers' content knowledge of AI is largely influenced by social discoursesabout AI.* In terms of content knowledge on artificial intelligence, the survey indicates that AI-buzzwords frequently used in the media influence the teachers' idea of the field. Both median and mode values for the keywords *Classification, Machine Learning, Natural Language Processing, Pattern Recognition, and Data Analysis* show that these keywords are associated with the subject area very strongly (cf. Table 2). The other keywords obtain lower rates with median values between 2 and 3.

Table 2. Association of keywords with the subject area artificial intelligence, 1 - Very strongly, 6 - Not at all. When the No. of Answers is less then 37, participants have either not answered at all or indicated that they cannot assess the item.

Keyword	No.ofAnswers	Median	Mode	$\bar{d}_{0.5}$
Knowledge Representation	34	3	3	0.94
Search	36	2	2	0.72
Statistical Methods and Theory of Probability	37	2	1	0.76
Ethics	36	2	1	1.22
Classification	37	1	1	0.78
Machine Learning	37	1	1	0.46
Sorting Algorithms	33	3	1	1.52
Simulation	36	2	1	1.06
Reasoning	35	2	1	1.06
Robotics	37	2	1	1.00
Algorithms	37	2	1	0.97
Cloud Computing	36	2	1	1.25
Natural Language Processing	37	1	1	0.65
Pattern Recognition	37	1	1	0.22
Turing Machines	31	4	3	1.19
Approximation	35	2	2	0.97
Data Analysis	37	1	1	0.38

2. *The teachers' ideas of artificial intelligence diverge significantly.* The absolute deviation from the median $d_{0.5}$ shows clearly that the association of the keywords with artificial intelligence differs widely among the participants. Seven out of seventeen keywords have a deviation of $d_{0.5} \geq 1$. Still, all buzzword-keywords have a deviation of $d_{0.5} < 0.8$. Such deviation values also indicate that the teachers' ideas of what artificial intelligence comprises diverge significantly. These results cannot be brought in to draw detailed conclusions on the teachers' knowledge about the subject area, but the prominent assessment of the buzzwords indicates that teachers have only a rough idea of the subject area and its sub-topics, which is shaped by current hype topics.

This conclusion is also supported by the fact that the distractors, i.e. *Simulation*, *Cloud Computing*, *Sorting Algorithms* and *Turing* Machines, are not clearly perceived as rather not belonging to the field of AI. Except for *Turing Machines*, the median and mode values for the distractors do not stand out from the other keywords (cf. Table 2) and they are just as much associated with AI as the other keywords used in the survey.

3. *Most teachers do not have experiences with teaching AI yet.* The diverging ideas of artificial intelligence are directly related to the fact that 80.56% of the teachers questioned have not taught artificial intelligence until now and thus had no need to take a closer look at the topic. Furthermore, AI is not a compulsory part of teacher training programs in Brandenburg, Berlin, and Bavaria where this survey was conducted. Nevertheless, the majority of the teachers is in favor of a curricular integration of the topic, only 8.33% of the participants have a rather negative opinion on the curricular integration of AI as Table 3 indicates.

However, due to this result, the outcome of the subsequent items must be reviewed critically: with little content knowledge about the topic and no teaching experience, it is rather difficult to assess educational goals and challenges of teaching it. Therefore, for comparison, the results of the teachers who have already taught AI and should thus be more confident in assessing the goals and challenges were reviewed again separately. These results show the same tendencies concerning the importance of certain educational goals (e.g. 85.71% and 89.66% of the teachers with and without experience agree that students should be able to assess the intelligence of AI systems; socio-cultural aspects are considered most important in both groups). Nevertheless, those with teaching experience in AI generally consider the challenges of teaching AI less high although they still perceive the same kind of difficulties (e.g. only 85.71% perceive a lack of suitable teaching materials, compared to 94.29% in the entire group).

4. *Teaching concepts for AI should address a variety of educational goals.* The educational goals of teachers in the field of AI are not confined to one of the Dagstuhl-perspectives [1] but comprise each of the three aspects mentioned in the methodology section. However, socio-cultural aspects like the development of a critical personal attitude towards AI systems (100% agreement[6]) or the evaluation of social chances and risks of AI technologies reach the highest approval rates (94.44% agreement). The ability to actually utilize AI systems (77.78% agreement) and libraries (63.89% agreement) is considered least relevant by the teachers, being able to identify technical limitations of artificial intelligence (88.89% agreement) or to explain the functioning of machine learning processes (86.11% agreement) is regarded as more important.

Table 3. I would like to see an explicit curricular integration of the topic AI.

Response Option	%
1 - Very strongly agree	36.11%
2	30.55%
3	25.00%
4	8.33%
5 & 6-Very strongly disagree	-

The challenges of teaching artificial intelligence perceived by the teachers are diverse. 91.67% of the teachers agree that they lack profound knowledge about artificial intelligence, 94.29% consider it as a problem to find suitable teaching materials or best-practice examples (87.88%). The high complexity of the topic is considered a problem by 64.52% of the teachers and can therefore also be seen as a central challenge for integrating the topic in K-12 education. Furthermore, 71.43% of those questioned indicate that they do not have enough time to deal with extracurricular contents like AI in class.

5 Discussion

Even though the sample size is too small to permit general conclusions and reasonable statements about statistical significance, the results of the questionnaire survey show that even most teachers interested in the topic only have a rough idea about the subject area of AI. Their knowledge seems to be shaped by the latest (media) topics and developments in the field of artificial intelligence, topics of AI which are currently less popular, e. g. search, are less strongly associated with AI. In the teachers' perception, the subject of artificial intelligence seems, above all, to be linked with aspects of machine learning and typical application areas of AI such as language processing and data analysis. This might indicate that teachers are often not aware of the full scope of the topic and the aspects forming the basic principles of AI. Yet, it is these basic principles that are highlighted by AI instructors and practitioners on the university level, as the findings of Wollowski et al. [20] show, and which, therefore, should also be familiar to teachers and play a role in class. Nevertheless, this thesis should be verified with a more detailed investigation of the teachers content knowledge on AI. This could be done by using open questions which were not part of this questionnaire.

The strong focus on AI buzzwords also indicates that, for class, AI is not only relevant as a mere subject content but also serves to educate and inform about current public discourses and, therefore, is of high value for general education. Consequently, teaching AI should not only consider either a technological or a socio-cultural or an application-oriented perspective, as it is done by many of the concepts that already exist[7], but follow a

[6] In this section, agreement refers to the response options *1-Very strongly agree, 2* and *3* on the Likert-scale.

[7] e.g. http://moralmachine.mit.edu/, https://machinelearningforkids.co.uk

holistic approach that integrates computer science contents, media-ethical perspectives and aspects of practical and socio-cultural relevance. As the educational goals of the teachers questioned are not confined to one of the Dagstuhl-perspectives [1], such a wide perspective is clearly aimed for by them. With this in mind, teacher training should cover this new topic in various facets to allow for a holistic teaching.

Thus, the aspects mentioned need to be taken into account in the development of workshops, teaching materials and concepts for teacher training. To meet the teachers' needs, it is important to adapt the material to the teachers' prior knowledge about AI [5]. As, according to the survey, the teachers have only basic content knowledge about AI, the buzzwords already familiar from social discourses can form adequate starting points to introduce and discuss topics of AI. This approach also makes it possible to explore the existing explanatory models of teachers for certain AI phenomena[8] and, if necessary, to modify them. By clarifying and deepening terms like machine learning, referring to AI applications familiar from everyday life, as well as by establishing connections to AI sub-topics or principles which are already part of CS curricula, e.g. search or expert systems, familiarizing with the topic is facilitated. For example, as problem-solving by search is a fundamental concept of AI, teachers can be made aware that search strategies can also be mediated in the context of AI. This is equally possible with data structures like graphs and (decision) trees.

Moreover, to respond to the challenges perceived by the teachers questioned, it is necessary to develop innovative and creative material that offers low initial hurdles but also appeals to teachers who are already experienced in the field of AI. This can be achieved by providing detailed material which serves both to extend the teachers' content knowledge and to show suitable and innovative teaching approaches for different and complex topics from the subject area, as the findings of Sulmont et al. [18] show that instructors face a series of topic specific challenges when teaching about AI. The curricular integration of the topic would have to be accompanied by such actions to reduce the hurdles of teaching artificial intelligence and to guarantee an effective, competence-oriented alignment of AI education.

Nonetheless, to achieve this, the teachers' perspectives need to be examined in more detail (e.g. by conducting qualitative interviews) and teaching materials for AI should be developed in close cooperation with teachers to effectively meet their needs. This survey, which gives first insights into the teachers' perspectives on AI and whose general findings correspond to those of Grillenberger and Romeike [12] for the field of data management – an equally new topic for CS class –, can serve as a basis for further research on these aspects.

6 Conclusion

In sum, we found that even computer science teachers who are already interested in AI predominantly draw upon content knowledge about the topic that is largely influenced by the current social discussion about AI. Nevertheless, most teachers are in favor of introducing artificial intelligence into computer science curricula and have diverse educational goals that focus on technical as well as socio-cultural and application-oriented aspects. Furthermore, the teachers questioned perceive a number of challenges ranging from content knowledge deficits in the subject area to high thematic complexity and a lack of adequate teaching materials and time in class. All of these aspects need to be taken into account when developing AI training materials for teachers and teaching concepts. Thus, the following recommendations can be made: to connect to the teachers' prior knowledge, which is influenced by the media and social discourse, materials about AI should start with familiar buzzwords and then go deeper. Furthermore, the central principles of AI should be highlighted with varied examples and be linked to existing curricular topics in training materials. Finally, to ensure sustainable teaching about AI, teachers should be encouraged to take a holistic perspective on the topic and be supported in mastering the complex technical aspects of AI and its subfields with extensive training and teaching materials. Such starting points can simultaneously provide teachers with ideas on how to introduce the topic to students without being overwhelmed by the complexity of the field.

However, the results of the questionnaire also lead to subsequent research questions that need further investigation: Can the present results be transferred to a larger sample size? Moreover, it needs to be determined how the teachers acquired their knowledge about AI and to which extent the media really is important in this process of knowledge acquisition.

[8] A further general investigation of these explanatory models in the field of AI is necessary as they were omitted in this rather general survey.

References

1. Brinda, T., Diethelm, I.: Education in the digital networked world. In: Tatnall,A., Webb, M. (eds.) Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing. pp. 653–657. Springer International (2017)
2. Bundesregierung: Strategie Ku¨nstliche Intelligenz der Bundesregierung [Artificial intelligence strategy] (2018), https://www.bmbf.de/files/Nationale KIStrategie.pdf. Last accessed June 15, 2019
3. Conner-Simons, A., Gordon, R.: Self-driving cars for country roads (2018),http://news.mit.edu/2018/self-driving-cars-for-country-roads-mit-csail-0507. Last accessed May 23, 2019
4. DeepL GmbH: Deepl translator, https://www.deepl.com/translator. Last accessedMay 22, 2019
5. Diethelm, I., Do¨rge, C., Mesaros, A.M., Du¨nnebier, M.: Die Didaktische Rekonstruktion fu¨r den Informatikunterricht [Educational reconstruction for computer science education]. In: Informatik in Bildung und Beruf–INFOS 2011–14. GIFachtagung Informatik und Schule. pp. 77–86. Lecture Notes in Informatics (LNI) - Proceedings, Gesellschaft fu¨r Informatik e.V. (2011)
6. Ertel, W.: Introduction to Artificial Intelligence. Springer International (2018)
7. Frankish, K., Ramsey, W.M.: The Cambridge Handbook of Artificial Intelligence.Cambridge University Press (2014)
8. G7: Charlevoix common vision for the future of artificial intelligence (2018), https://www.bundesregierung.de/resource/blob/975254/1142208/ 9fdd8014296e52270d56d70df98ce7e0/2018-06-09-g7-kanada-artificial-intelligenceen-data.pdf?download=1. Last accessed June 17, 2019
9. Görz, G., Schneeberger, J.: Handbuch der ku¨nstlichen Intelligenz [Handbook ofArtificial Intelligence]. Oldenbourg (2014)
10. Gretter, S., Yadav, A., Sands, P., Hambrusch, S.: Equitable learning environmentsin K-12 computing: Teachers' views on barriers to diversity. ACM Transactions on Computing Education **19**, 1–16 (01 2019). https://doi.org/10.1145/3282939
11. Griffin, J., Pirmann, T., Gray, B.: Two teachers, two perspectives on CS principles. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education. pp. 461–466. SIGCSE '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2839509.2844630
12. Grillenberger, A., Romeike, R.: What teachers and students know about data management. In: Tatnall, A., Webb, M. (eds.) Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing. pp. 557–566. Springer International (2017)
13. Lämmel, U., Cleve, J.: Künstliche Intelligenz [Artificial Intelligence]. Hanser (2012)
14. Mishra, P., Koehler, M.: Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers College Record **108**, 1017–1054 (06 2006). https://doi.org/10.1111/j.1467-9620.2006.00684.x
15. ReadyAI: ReadyAI Curriculum (2019), https://www.readyai.org/curriculum/.Last accessed August 20, 2019
16. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (2009)
17. Silver, D., Hubert, T., Schrittwieser, J., Hassabis, D.: Alphazero: Shedding new light on the grand games of chess, shogi and Go (2018), https://deepmind.com/blog/alphazero-shedding-new-light-grand-games-chessshogi-and-go/. Last accessed May 27, 2019
18. Sulmont, E., Patitsas, E., Cooperstock, J.R.: Can you teach me to machinelearn? In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education. pp. 948–954. SIGCSE '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3287324.3287392
19. Thompson, D., Bell, T., Andreae, P., Robins, A.: The role of teachers in implementing curriculum changes. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. pp. 245–250. SIGCSE '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2445196.2445272
20. Wollowski, M., Selkowitz, R., Brown, L.E., Goel, A., Luger, G., Marshall, J., Neel,A., Neller, T., Norvig, P.: A survey of current practice and teaching of AI. In: Proceedings of the Sixth Symposium on Educational Advances in Artificial Intelligence (EAAI-16). pp. 4119–4124 (2016)
21. Yadav, A., Gretter, S., Hambrusch, S.: Challenges of a computer science classroom: Initial perspectives from teachers. In: Proceedings of the Workshop in Primary and Secondary Computing Education. pp. 136–137. WiPSCE '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2818314.2818322
22. Yu, Y., Chen, Y.: Design and development of high school artificial intelligencetextbook based on computational thinking. Open Access Library Journal **5**(09), 1–14 (2018). https://doi.org/10.4236/oalib.1104898
23. Zimmerman, M.: Teaching AI. Exploring New Frontiers for Learning. ISTE (2018)

Facts, Figures, Remarks and Conclusions about the Austrian Bebras Challenge with Regard to Computational Thinking and Computer Science Domains

Peter Micheuz[1], Stefan Pasterk and Andreas Bollin[2]

[1] Alpen-Adria Gymnasium Völkermarkt, 9020 Völkermarkt, Austria
peter.micheuz@bildung.gv.at
[2] Alpen-Adria-University, Universitätsstraße 65-67, 9020 Klagenfurt, Austria
{stefan.pasterk, andreas.bollin}@aau.at

Abstract. The interest in the Bebras movement and the number of participants increases every year. Participants of this international contest have to solve Informatics related tasks, requiring no specific education in that field. Each of the tasks covers one or more categories of Computational Thinking (CT) and can be assigned to special domains of Computer Science (CS). Although the practical definition of CT is still a moving target, this challenge can be seen as a measurement tool for CT competencies.

In this paper, we present an analysis of different aspects of the Austrian Bebras competition over the last three years. Accumulated statistical datasets are compared and analyzed as well as detailed task solutions. Moreover, the data provide information about CT skills and CS domains mastered by the pupils and, conversely, about difficulties in certain task categories. Accordingly, the evaluated data provided by the automatic testing system can be used to draw conclusions about the development of tasks.

Keywords: Bebras Challenge, Austria, Findings, Computational Thinking.

1 Introduction

The Bebras Challenge is an international initiative, aiming at promoting Informatics among teachers and pupils of all ages. Established in 2004 in Lithuania [6], the Bebras movement developed very well since then. The increase from 29 participating countries and 0.7 million pupils in 2013 up to 54 countries and 2.7 million of pupils in 2018 impressively demonstrates this phenomenon.

In the wake of this encouraging development a considerable scientific accompanying research and documentation work is growing, including this contribution as a further mosaic stone among Bebras-related publications.

This paper must be seen as a compromise between exemplary cumulated statistical data, collected and provided by the Bebras testing system and exemplary task specific considerations and results with regard to a well-accepted classification scheme.

The paper starts with hard facts about Austrian participation numbers in various contexts, taking into account some idiosyncrasies due to the diverse Austrian school system. The underlying raw data have been collected by the international Bebras website [6] and by the (Dutch) Bebras testing system. These findings are complemented by empirical results of a survey among the local Bebras organizers in 2016.

The next chapter begins with some considerations on the limits of Beaver tasks in view of a wider perspective on Computational Thinking in general, and represents a widely accepted two-dimensional approach to classify Beaver tasks with regard to Computational Thinking aspects and Computer Science domains.

Chapter 4 is dedicated to the Austrian selection of tasks and its alignment to an existing classification scheme, followed by a discussion about exemplary results of pupils' proficiency in various domains.

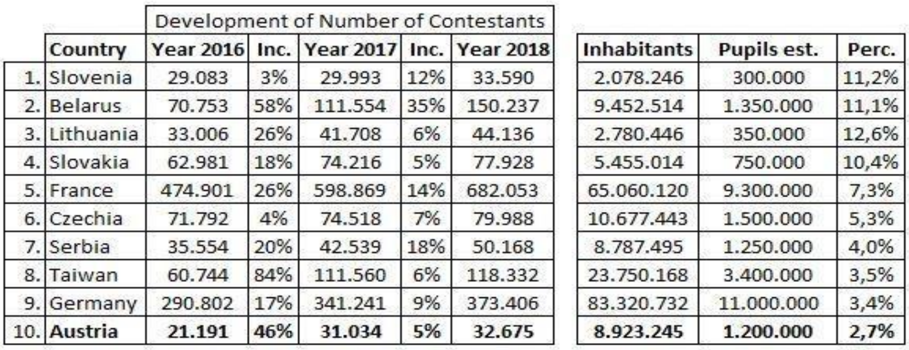
2 Facts, Figures and Remarks about the Austrian Participation

2.1 The Last Three Years

In the year 2018, the Bebras Challenge in Austria took place for the 12th time in succession, with celebrating the 10th anniversary in 2016 and the first participation in 2007, together with Latvia and Slovakia as the 8th country, after Estonia, Germany, The Netherlands, and Poland joined the forerunner Lithuania in 2006 [6].

Within 14 years, the Bebras challenge developed rapidly, currently encompassing 54 member countries from all continents. The last Bebras challenge attracted about 2.7 millions of pupils all over the world. Table 1 shows the top ten list of these countries and its specific development of participation within the last three years, considering its population in total and the estimated number of pupils. In this list Austria ranks in 10th place. It is striking that there was a big increase from 2016 to 2017 whereas the growth from 2017 to 2018 is moderate. About 3 out of 100 Austrian pupils have participated in the past year 2018, which indicates that there is still much room for expansion. Even the leader Slovenia attracts not more than about 11% of all the potential pupils.

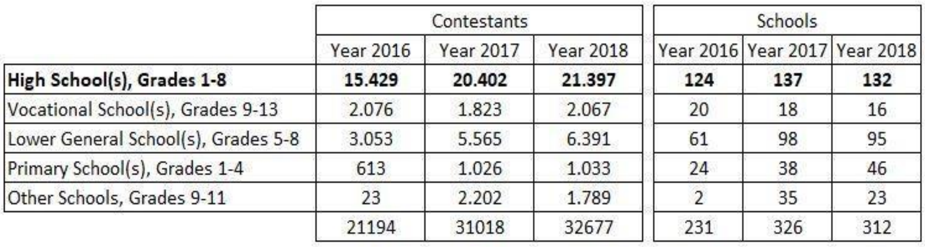
Table 1. Top Ten List with the Rank of Austria Among the Participating Countries



		Development of Number of Contestants							
	Country	**Year 2016**	**Inc.**	**Year 2017**	**Inc.**	**Year 2018**	**Inhabitants**	**Pupils est.**	**Perc.**
1.	Slovenia	29.083	3%	29.993	12%	33.590	2.078.246	300.000	11,2%
2.	Belarus	70.753	58%	111.554	35%	150.237	9.452.514	1.350.000	11,1%
3.	Lithuania	33.006	26%	41.708	6%	44.136	2.780.446	350.000	12,6%
4.	Slovakia	62.981	18%	74.216	5%	77.928	5.455.014	750.000	10,4%
5.	France	474.901	26%	598.869	14%	682.053	65.060.120	9.300.000	7,3%
6.	Czechia	71.792	4%	74.518	7%	79.988	10.677.443	1.500.000	5,3%
7.	Serbia	35.554	20%	42.539	18%	50.168	8.787.495	1.250.000	4,0%
8.	Taiwan	60.744	84%	111.560	6%	118.332	23.750.168	3.400.000	3,5%
9.	Germany	290.802	17%	341.241	9%	373.406	83.320.732	11.000.000	3,4%
10.	**Austria**	**21.191**	**46%**	**31.034**	**5%**	**32.675**	**8.923.245**	**1.200.000**	**2,7%**

Table 2 shows clearly the very inhomogeneous distribution among different types of schools, age-groups and grades. The Austrian High Schools (Academic Secondary Schools), hosting about 200.000 out of about 1.200.000 Austrian pupils, cover disproportionally many 2/3 of all participants. Almost 50% of them have taken part in the Bebras challenge within the last years, with the actual number of 132 out of about 330 schools of that type.

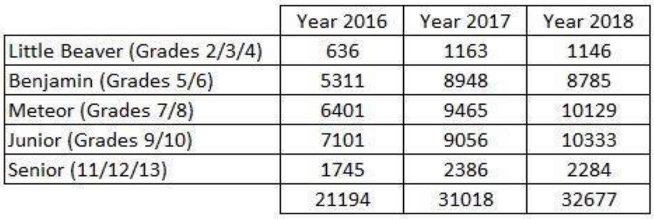
Table 2. Distribution of participation regarding school types.



	Contestants			Schools		
	Year 2016	Year 2017	Year 2018	Year 2016	Year 2017	Year 2018
High School(s), Grades 1-8	**15.429**	**20.402**	**21.397**	**124**	**137**	**132**
Vocational School(s), Grades 9-13	2.076	1.823	2.067	20	18	16
Lower General School(s), Grades 5-8	3.053	5.565	6.391	61	98	95
Primary School(s), Grades 1-4	613	1.026	1.033	24	38	46
Other Schools, Grades 9-11	23	2.202	1.789	2	35	23
	21194	31018	32677	231	326	312

About 2/3 of all participants are covered by pupils aged between 13 and 16 years. In Austria the categories are named Meteor and Junior. As table 3 shows, primary schools are still playing a marginal role (not even 1 of 300 pupils took part in 2018), and so do senior students in grade 11 to 13. The data showed that some primary schools offered the Bebras challenge even at grade 2 with 8-years old pupils.

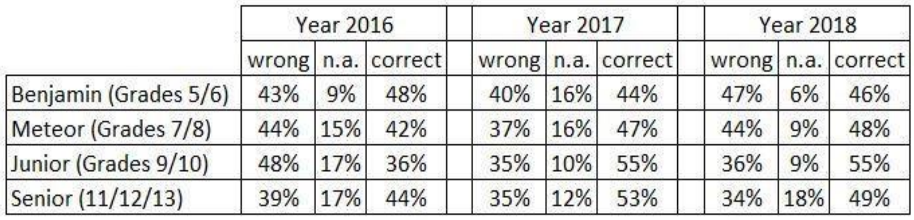
Table 3. Distribution of participation regarding Bebras levels.



	Year 2016	Year 2017	Year 2018
Little Beaver (Grades 2/3/4)	636	1163	1146
Benjamin (Grades 5/6)	5311	8948	8785
Meteor (Grades 7/8)	6401	9465	10129
Junior (Grades 9/10)	7101	9056	10333
Senior (11/12/13)	1745	2386	2284
	21194	31018	32677

Regarding the development of the overall performance over the last three years, there are comparatively little deviations without significant statistical outliers. The numbers in Table 4 indicate (but do not prove) a good annual task selection with a stable level of difficulties. The column labelled with “n.a.“ represents the not answered questions which has significantly decreased in the year 2018 except the category Senior.

Table 4. Distribution of participation regarding wrong and correct answers.



	Year 2016				Year 2017				Year 2018		
	wrong	n.a.	correct		wrong	n.a.	correct		wrong	n.a.	correct
Benjamin (Grades 5/6)	43%	9%	48%		40%	16%	44%		47%	6%	46%
Meteor (Grades 7/8)	44%	15%	42%		37%	16%	47%		44%	9%	48%
Junior (Grades 9/10)	48%	17%	36%		35%	10%	55%		36%	9%	55%
Senior (11/12/13)	39%	17%	44%		35%	12%	53%		34%	18%	49%

2.2 Exemplary Results of a Survey among School Coordinators and Pupils

In January 2017, the Austrian Bebras school coordinators of the challenge 2016, the key persons and drivers of Bebras at school level, were asked to take part in an online survey regarding organizational aspects, the competition platform and the tasks. 97 of 231 Bebras coordinators responded. A rough quantitative analysis shows that

- over 70% (of the coordinators) have organized the Bebras Challenge three times or more,
- more than 2/3 prefer the temporal extension of the competition to two weeks,
- during the challenge the pupils were strictly supervised in more than 80% of cases,
- the offer to work in teams of two was not recommended in 85% of the cases and less than half of the coordinators would be interested in this option in the future,
- in about 1/3 of all participating schools the Bebras challenge was also offered in classes without formal computer science lessons,
- the used (Dutch) competition platform was over 90% technically well-functioning and there were only little technical problems (less than 10%) at the schools,
- more than 90% of the Bebras tasks (2016) were found to be good,
- about 30% found the tasks on average too difficult and about half would want more easier tasks and less text,
- about 90% of the coordinators are in favor of interactive tasks,
- almost everyone stated that the competition was taken seriously by students,
- only 6% of the coordinators stated that they had ideas for Bebras tasks and less than 10% are willing to contribute to the Bebras task collection,
- finally, about 40% stated that the number of 21,000 Bebras participants (2016) in Austria is seen to be comparatively satisfactory.

3 The Bebras Challenge and Computational Thinking

3.1 Are Bebras Tasks Solvers Real Computational Thinkers?

Since the last year, Computational Thinking (CT) is an explicit, but not dominating part of the new Austrian curriculum for Basic Digital Education for lower secondary education (grades 5-8) within other domains, ranging from Computer Literacy to Media education. Within this CT related part of the curriculum pupils should be able to name and describe everyday processes, to use and build codes, to reproduce distinct instructions (algorithms) and carry them out, to formulate distinct instructions verbally and in written form and to program in one computer language [11]. This is one (Austrian) definition of CT which has become a buzzword with a multitude of definitions.

In a broader definition, CT shall be seen as „a cultural technique consisting of a set of skills needed to complete a task in a responsible, sustainable manner including problem solving, evolutionary and reflection steps. These steps encompass logical reasoning, algorithmic thinking, abstraction, generalization, decomposition, design/solution patterns, evaluation techniques, and as computers might be involved in the solution process, different representation forms. It also includes knowing about related disciplines like computer science and software engineering. As such, it should be thought to its fullest extent, but in an age-appropriate manner, at secondary level in Austria.“ [2]

The most comprehensive definition of CT is given by Denning/Tedre where „CT is the mental skills and practices a) for designing computations that get computers to do jobs for us and b) explaining and interpreting the world as a complex of information processes.“. Moreover, the authors distinguish clearly between „CT for beginners“ (K12 education) and „CT for professionals“ with a very rich skillset with six dimensions: machines, methods, computing education, software engineering, design, and computational science.

„If you try to understand what Computational Thinking is from media accounts (and many publications) you will hear a story of problem solving with algorithms, along with the ability to think at the many levels of abstraction needed to solve problems. But the K-12 education insights and debates barely scratch the surface of CT” [5].

In view of these multifaceted perspectives on CT, and the fact that our brains do not naturally think computationally and must be developed and trained, it is up to discussion if Bebras tasks should be so strongly and almost exclusively associated with CT as it is outlined in many publications. If we agree on CT as a set of trained and broad skills and look at the assertion that Bebras tasks can be solved at every level without prior Informatics education, we should see the Bebras challenge less as a role model for CT, but more as a valuable initiative for transmitting many Informatics concepts.

Since the Bebras’ start in Lithuania 2004, about - roughly estimated - one thousand Bebras tasks and their solutions, developed in a cooperative effort by enthusiastic Informatics educators worldwide, are publicly available. These tasks constitute their own reality with an unique character and increasing influence on Informatics teaching, and on the perception of what the core of „computing“ is about. The key idea behind these tasks is not to assess already learned factual and procedural computing knowledge, but to expose pupils to problems with a strong background of Informatics and CT concepts.

3.2 Categorizing Bebras Tasks

In different countries the Bebras tasks are published in form of booklets including the correct answers and descriptions of the computer science topics they deal with (e.g. United Kingdom [14], Australia [1] or Switzerland [13]). Some additionally categorize the tasks into given categories for computer science or Computational Thinking (e.g. United Kingdom, Australia). The booklet from United Kingdom [14] provides a categorization system based on the work of Dagienė et al. [8]. Dagienė et al. present a twodimensional system to categorize the Bebras tasks. As a first dimension they use five computer science concepts as domains and connect them with a set of keywords to clarify their meaning (in this paper we include only a snippet of the keywords).

1. **Algorithms and programming:** Algorithm; Binary search; Boolean algebra; Breadth-first search; Brute-force search; Bubble sort; Coding; …
2. **Data, data structures and representations:** Array; Attributes; Graph; Binary representations; Binary tree; Character encoding; Databases; …
3. **Computer processes and hardware**: Deadlock; Image processing; Memory; Peripherals; Priorities; Scheduling; Turing machine; …
4. **Communication and networking:** Client/server; Computer networks; Cryptography; Encryption; Parity bit; Protocols; Security; Topologies; …
5. **Interactions, systems and society:** Classification; Computer use; Ethics; Graphical user interface; Legal issues; Robotics; Social issues; … [8]

Computational Thinking skills build the second dimension. Based upon the work of Selby and Woollard [12] Dagienė et al. define five categories, adding notes how the use of the particular CT-skills can be assigned.

1. **Abstraction:** Removing unnecessary details; Spotting key elements in problem; Choosing a representation of a system
2. **Algorithmic thinking:** Thinking in terms of sequences and rules; Executing an algorithm; Creating an algorithm
3. **Decomposition:** Breaking down tasks; Thinking about problems in terms of component parts; Making decisions about dividing into sub-tasks with integration in mind, e.g. deduction
4. **Evaluation:** Finding best solution; Making decisions about good use of resources; Fitness for purpose
5. **Generalization:** Identifying patterns as well as similarities and connections; Solving new problems based on already-solved problems; Utilizing the general solution, e.g. induction [8]

This two-dimensional approach represents a possible categorization system for Bebras tasks describe them in more detail. In the Australian booklets comparable categories are used. With the help of the computer science concepts a mapping to the Australian Digital Technologies Curriculum is shown [1].

4 Results Regarding Computational Thinking and Computer Science Domains

4.1 Methodology

To categorize the Bebras tasks of the Austrian challenges the two-dimensional approach by Dagienė et al. [8] was used due to the fact that data of categorized tasks was available. So, in a first step the tasks of Austria and the United Kingdom were compared and mapped, considering language discrepancy. If a mapping could be found, the categorization was adopted. In the year 2016 18 out of 41, in 2017 22 out of 39, and in 2018 24 out of 39 tasks from the Austrian challenge could be mapped to tasks from UK. The tasks without a mapping were discussed and categorized by the authors following the instruction from the UK booklet:

- Each task is assigned to one or more Computational Thinking skills
- Each task is assigned to one Computer Science domain [14]

First, in the following section the results concerning category frequency are presented. Subsequently, the results of the Austrian challenge are discussed with regard to the distribution of correct answers within the categories.

4.2 Category Task Frequency

The results for the Computational Thinking categories show that the Bebras tasks contain a lot of ‘algorithmic thinking’ topics rising from 76% from 2016 up to 87% from 2018. In the year 2017 the number of tasks from the category ‘abstraction’ appeared more often compared to the other two years. The category ‘decomposition’ occurs most rarely in 2017 and 2018. Only in 2016 tasks including an aspect of ‘evaluation’ occurred more rarely. These results can be seen in Figure 1.

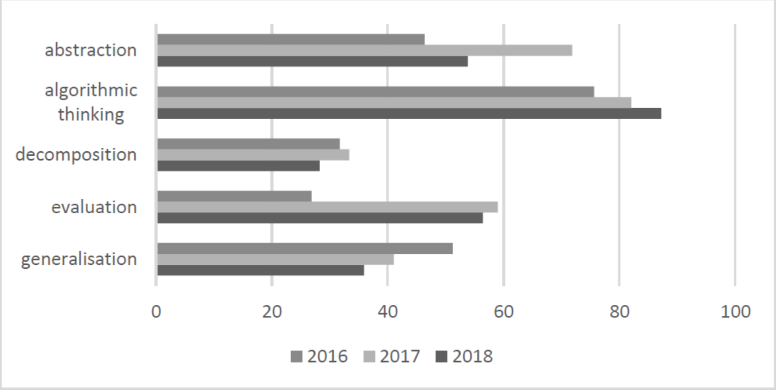


Fig. 1. Relative distribution of Computational Thinking tasks

Regarding the Computer Science domains, the two categories ‘algorithms and programming’ and ‘data, data structures and representations’ occur very frequently over all three years, as Figure 2 shows. The other three domains are not represented as often. This trend increased from 2016 to 2018: Domains which are represented frequently in 2016 and 2017, are represented in 2018 even more frequently, domains which occur rarely in 2016 and 2017, occur even more rarely in 2018. For instance, in the year 2018 there was no task in the Austrian Bebras challenge that belonged to the category ‘communication and networking’.

A comparison of all levels of the Bebras challenge shows the same picture. At all levels ‘algorithmic thinking’ for Computational Thinking and ‘algorithms and programming’ and ‘data, data structures and representations’ for Computer Science occur very frequently. However, the level two for grade 5 to 6 shows one exception. Figure 3 indicates that in this level the relative frequency of tasks from the category ‘decomposition’ is the highest. This is true for all three years, leading to the assumption that this was intended by the task selection.

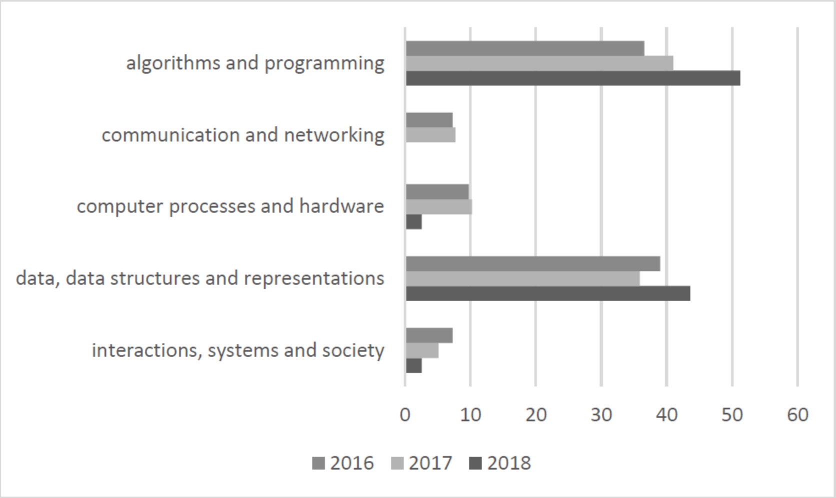


Fig.2. Relative distribution of computer science domain tasks

The results presented in this section shows that among the Computational Thinking categories tasks which demand algorithmic thinking skills are significantly prevalent. Among the Computer Science domains most tasks can be assigned to ‘algorithms and programming’ and ‘data, data structures and representations’. These distributions of tasks may have different reasons, as the tasks can be selected from an international pool of tasks but also can be invented by the (national) organizations. It may also be the case, that the particular task selection is intended. This would imply that the categorization system should be revised (at least) for the Austrian Bebras challenge.

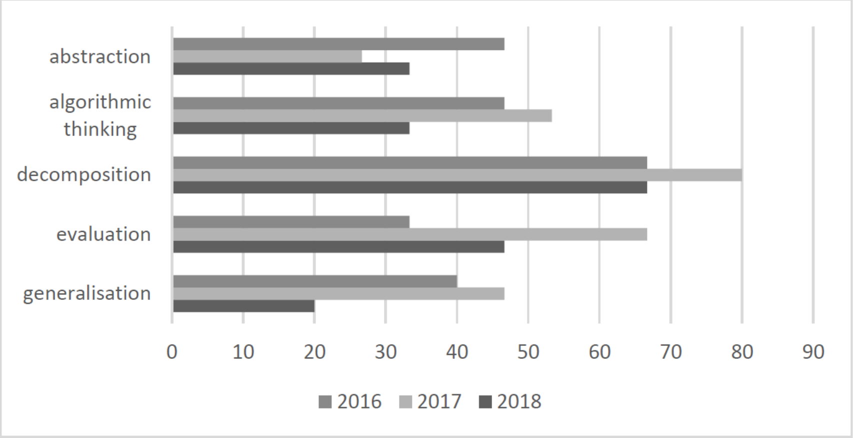


Fig. 3. Relative distribution of Computational Thinking tasks for grades 5 to 6

4.3 Answers within Categories

In this section the (correct) answers of the Austrian Bebras challenges over the years 2016, 2017 and 2018 are investigated. Obviously, the tasks are different from year to year which makes it difficult to compare the annual performance of Austrian pupils. Many other variables have to be considered, like the difficulty of selected tasks, the number of tasks within the categories or the fluctuation of participants. So, the comparison of the results over three years offers limited interpretation possibilities.

Figure 4 shows the relative student results regarding the years 2016-2018 for the Computational Thinking domain. Despite the aforementioned limitations, trends are visible over the three years. For example, in the category ‘abstraction’ the percentage of correct and wrong answers differs by 5% over all three years. In the category ‘decomposition’ the number of correct answers is at least 49% (2016), where the maximum of wrong answers is 42% (2016). 2017 nearly 57% of the answers within this category were correct, only 33% were wrong and 10% were not answered. Over all three years the number of correct answers is for at least 7% higher, compared to the number of wrong answers. For the category ‘evaluation’ also a trend over all three years is visible, as the ratios of the relative numbers of correct and wrong answers are very similar. In the two categories ‘algorithmic thinking’ and ‘generalization’ no trend can be recognized. However, the distribution follows the same pattern: in 2016 there were more wrong than correct answers, in 2017 there were far more correct than wrong answers, and in 2018 there were a bit more correct than wrong answers. This indicates that the national Bebras organizers aim at a task selection which yields a balance between correct and wrong answers.

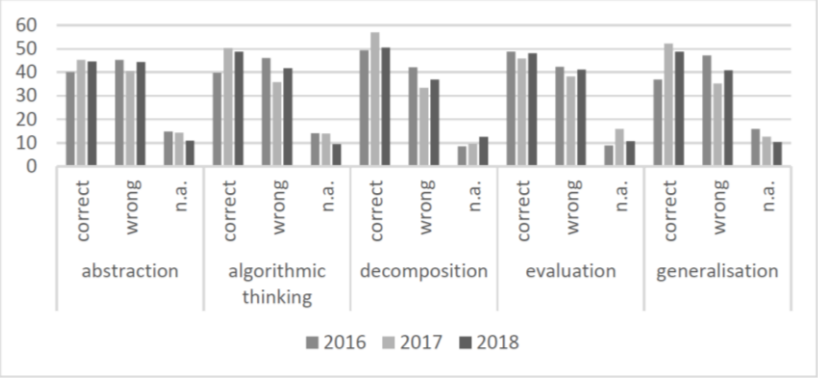


Fig. 4. Relative distribution of the answers for Computational Thinking tasks

Additionally, the results for the Computer Science domains show also interesting findings as it can be seen in Figure 5. Like in the Computational Thinking domain, again trends are visible in several categories. In the very frequently appearing category
‘algorithms and programming’ over all three years the ratio of correct and wrong answers is similar. All three years show a slightly higher percentage of correct answers.
The tasks from category ‘communication and networking’, which only occurred in 2016 and 2017, were answered correctly far more often (at least 20% more in 2017) than incorrectly. It has to be considered that each year only three tasks belonged to this category.

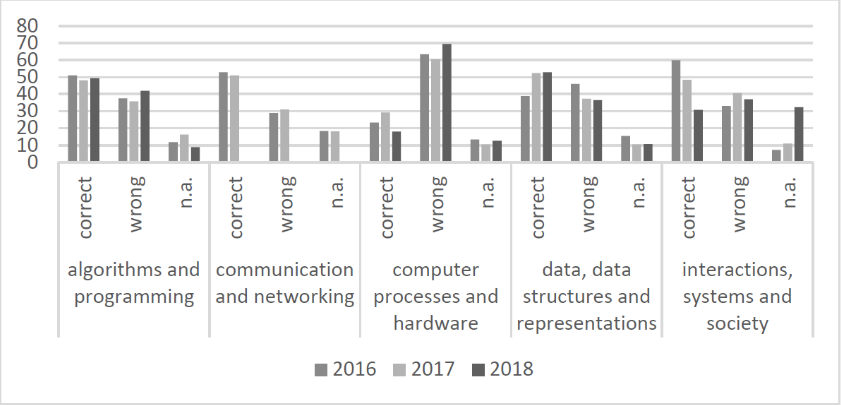


Fig. 5. Relative distribution of the answers for computer science domain tasks

The category ‘computer processes and hardware’ represents an unusual result, as there are over all three years significantly more wrong (at least 31% more in 2016) than correct answers. Here a very low number of tasks may bias the results as in the years 2016 and 2017 only four tasks and in 2018 only one task belonged to this category. In the category ‘data, data structures and representations’ which is the second most frequent category in the Computer Science domain, the number of correct answers was only in 2016 less, compared to the number of wrong answers. The years 2017 and 2018 were similar in the ratio of correct and wrong answers.

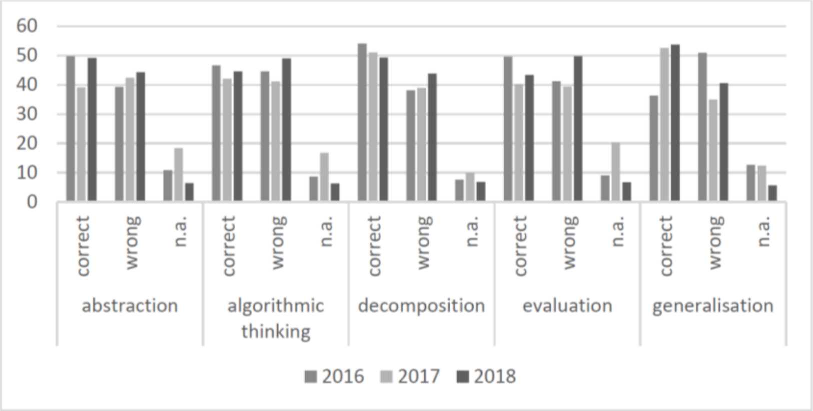


Fig. 6. Relative distribution of the answers for CT tasks for grades 5 to 6

Taking a look at grades 5-6, the Computational Thinking category is of interest. Figure 6 shows that over all three years a trend is visible in nearly all categories: The ratios between correct and wrong answers are more or less balanced.

Despite the limitations of measuring the development of the pupils’ performance and the difficulties of tasks over the period of three years, due to the lack of taking into account other influential preconditions and variables, it can be stated that the process of task selection works accurate and satisfactorily.

5 Concluding Remarks and Future Work

The Bebras challenge has a comparatively long tradition in Austrian schools and the numbers of participating pupils are still increasing from year to year. However, it seems that the growth from 2017 to 2018 has flattened out. So it will be interesting to observe the development of the Austrian Bebras challenge in the next years. It will take additional information and conviction effort from the Austrian Computer Society and the Ministry of Education among all schools and teachers to improve Austria’s 10th rank in the country table. The evaluation of test data with regard to different types of schools and the participating age-groups reveals clearly where to target the promotion for the next Bebras challenge in 2019.

Due to the page limit some other valuable statistical analyses have been omitted in this contribution (at the expense of a classification-related task study), such as the fluctuation of participating schools, the (average) percentage of participating pupils of schools including gender-specific evaluations, the correlation with formal Informatics education, and last but not least the pupils’ performances. It is planned to publish these facts and figures, together with new empirical data provided by a follow-up survey among the Bebras organizers at schools.

Regarding the categories provided by Dagienė et al. [8], the Austrian Bebras challenge includes a comparatively balanced number of tasks considering each of the categories from the Computational Thinking domain. However, some more tasks in the categories ‘decomposition’ and ‘generalization’ could even improve this balance. In contrast, the situation differs significantly in terms of the computer science domains. Only the two categories ‘algorithms and programming’ and ‘data, data structures and representations’ are very frequently represented in tasks. Tasks assigned to the categories ‘communication and networking’, ‘computer processes and hardware’ as well as ‘interactions, systems and society’ are exceptions for all examined three years. One conclusion might be that it is hard to find tasks fitting in the Austrian Bebras challenge for these three domains. Does the categorization system need a revision to be applied for the Austrian task selections?

The combination of the categories and the answers of the participants gives additional insights into the Austrian Bebras challenge. Although there are several limitations of the study due to different Bebras tasks over the years, the results indicate that in some categories the (performance) results are similar and relatively stable. Considering all variable conditions leading to these results, this is a remarkable finding about continuity and in task generation and selection by the Bebras community involved.

We propose further accompanying evaluation of Bebras data and even more scientific research in order to refine and expand the Bebras challenge in Austria and in other countries involved in the Bebras movement. And finally, we think of mutually agreed, comparable and desirable cross-national evaluations as well as of a revised classification scheme, or conversely, of an improved alignment of the Beaver tasks to the existing categorical framework referred to in this paper.

References

1. Bebras Australia: https://www.bebras.edu.au/bebras365/, last accessed 2019/30/06.
2. Bollin A., Micheuz P.: Computational Thinking on the Way to a Cultural Technique. In Empowering Learners for Life in the Digital Age, Springer International Publishing, pp 313, (2019).
3. Budinska L., Mayerova K., Siamndl V.: Differences Between 9-10 Years Old Pupils Results from Solvak and Czech Bebras Challenge. In: Pozdniakov S., Dagiene V. (eds.) Informatics in Schools 2018, LNCS, vol. 11169, pp 307-318. Springer, Heidelberg (2018).
4. Curzon P., McOwan P.: Computational Thinking. Springer, Berlin (2018).
5. Denning P., Tedre M.: Computational Thinking. MIT Press, Cambridge (2019).
6. International Bebras Homepage, https://www.bebras.org/?q=history, last accessed 2019/30/06.
7. Dagienė V., Sentance S.: It's Computational Thinking! Bebras Tasks in the Curriculum. In: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer International Publishing, S. 28-39, (2016).
8. Dagienė V., Sentence S., Stupuriene G.: Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. Informatica, Lith. Acad. Sci. 28, 23-44, (2017).
9. Dagienė V., Stupuriene G.: Bebras - A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking, Informatics in Education, Vol. 15 (1), pp 25-44, (2016).
10. Dagienė V., Stupuriene G.: Informatics Education based on Solving Attractive Tasks through a Contest. In: Comentarii informaticae didacticae. Torsten B. et al. (eds.). KEYCIT 2014, Potsdam, p. 97, (2014).
11. Digital Basic Education (in German): https://bildung.bmbwf.gv.at/schulen/schule40/dgb/index.html, last accessed 2019/30/06.
12. Selby C., and Woollard J.: Computational thinking: the developing definition University of Southampton (E-prints) 6pp, (2013).
13. Schweiz Informatik-Biber: https://informatik-biber.ch/de/aufgabensammlung/, last accessed 2019/22/08.
14. UK Bebras Computational Thinking Challenge: http://www.bebras.uk/answer-booklets.html, last accessed 2019/30/06.

Comparing Approaches for Learning Abstraction and Automation by Object Orientation

Arno Pasternak[1] and Johannes Fischer[2]

Fritz-Steinhoff-Schule Hagen
arno.pasternak@cs.tu-dortmund.de
Technische Universita¨t Dortmund
johannes.fischer@cs.tu-dortmund.de

Abstract. We investigate an important aspect of a ’CS for all’ curriculum in secondary education: how essential concepts for structuring data and actions are understood when using object oriented concepts for modeling and programming. Our primary research question is whether or not one should expose the students with large projects right from the beginning. To verify this claim, a group of 44 students was divided into 2 courses, where one course started programming with a (relatively) large project with several classes according to the ideas of K¨olling and Rosenberg from the year 2001. The other course started with a very small project with only one class corresponding to the concept of Sentence and Wait dating 2017.

Our evaluation shows that none of the approaches can be regarded as truly superior. Although it can be argued that the ’small project’ approach is slightly advantageous, our real findings are that the underlying curriculum seems to be unsuited for a world where CS as a school subject has grown beyond a specialist’s subject.

1 Introduction and Educational Setting

In Germany, computer science (CS) has been a common school subject in many federal states for over 30 years. Since this subject could not, for the most part, be credited as similar subjects as the natural sciences, it was has and remained a ’niche subject’ for many schools.

The school where one of the authors teaches has the following characteristics, which make it interesting as a research location for evaluating aspects of a ’CS for all’ curriculum:

- Almost half of the students at this school now participate regularly in computer science lessons at upper secondary level.
- The number of female students is almost the same as the number of male students.
- The proportion of students with an immigrant background is increasing and is now around 60%.
- Many students choose CS primarily as an alternative to natural sciences and not because they have a special connection to CS.
- students who deal more intensively with CS in their spare time, e.g. by programming, are becoming less and less frequent.
- Due to the school structure as a full-time school with afternoon lessons and sometimes more stressful requirements in the compulsory subjects, homework morale is very poor.

In short: CS has now become a completely normal school subject. However, the curriculum still assumes that due to a special interest in CS, the participants in these courses are also engaged in advancing their competences in implementation outside the classroom. As a result, the depth that is necessary to learn the basic terms (e.g. from OO) is not achieved to the degree as it used to be. Often, the syntactic elements of a language like *Java* alone pose great difficulties to the students. Therefore, in this study we compare two popular approaches for learning basic concepts [13,7] from abstraction and automation, which can be regarded as the most important aspects of computational thinking.

2 Basics of CS Education in Schools

In 2016 the CSTA formulated in their revised CS K–12 - Standards: ’It is intended to introduce the principles and methodologies of CS to all students, whether they are college bound or career bound after high school’ [3, p.7]. Correspondingly, the contribution of CS to general education is undisputed among didactics experts. This is further supported by focusing on concepts from *computational thinking* [15].

The CSTA standards describe the competencies and contents of CS as a school subject being part of a general education: The main focus is on basic structures and concepts, and *not* on the use of specific tools such as specific word processing or database software:

> 'The goals of [...] the curriculum are to engage students in using computational thinking as a problem-solving tool, teach them to use programming concepts and methods while creating digital artifacts [...]. [...] As students begin to master fundamental computer science concepts and practices, it is vitally important that they learn that these concepts and practices empower them to create innovations, tools, and applications' [12, p.15,16].

This focus on the teaching of concepts and principles also implies that when teaching programming, one should *not* focus on specific constructs and techniques of the chosen language that are necessary to express general ideas.

2.1 Concepts of Computer Science

What are the basic concepts of CS? First and foremost, Wing [15] names *abstraction and automation* as candidates, and somewhat later specifies: 'Computing is the automation of our abstractions'. The core idea of Wing's statements can be described as follows: By *abstraction*, a model will be created as an image of the real world. This image consists of a set of *abstract data structures and data objects* However, these are only a *static* description of the world. It is further necessary to abstract the *actions* such that one can (hopefully) solve a problem.

This connection of the abstraction of data *and* operations generates a *dynamic* model from the static one. This thinking in such basic structures is emphasized by *Denert*:

> 'I mean something else: Computational thinking. What's that? First, thinking in algorithms, that's fundamental. For information technology, I think we can agree on that. However, it is not sufficient. Thinking in structures are their necessary counterparts today more than ever, because the systems we build and administer are of an overwhelming complexity that can only be achieved through good structures[9]' [4].

The dynamic model is subsequently *implemented* on a (machine) concrete and then executed. Unfortunately, this machine does not 'understand' natural language. The artificial languages of the machines are an adaptation to *technical* conditions and represent a separate intellectual structure. These programming languages have to be learned and practised. It is an intellectually difficult activity, in particular for students, to 'code' a model in a concrete programming language and to test and correct this program.

Computational Thinking summarized as *abstraction and automation* is shaping society in a variety of ways today. CS in schools must, therefore, represent all aspects of it (with appropriate didactic reduction). Let us consider the specific aspects somewhat more precisely.

3 Abstraction

CS has become so important in practice because it helps to solve problems. To allow processing with a machine – the computer – the problem has to be analysed in a structured manner. This is done by abstraction of data and processes.

3.1 Abstraction of the Data

Modeling is a transformation of the problem into a reduced abstract world through the development of appropriate *modules*. The concept was originally described as follows:

> 'The preferred way to build a general-purpose system is *not* to build one computer program that will do all things for all people. Instead, what one should do is build a large number of small, single-purpose modules

[9] Original in German, translated by the authors

> that are flexible and that have extremely clean interfaces. The generality comes from the almost infinite number of combinations of such modules – combinations that very few designers ever would have been able to predict' [2, p. 374].

This definition makes clear how central the idea of modularisation is in CS. If such a module describes data, we speak of an *abstract data structure* or an *abstract data type* [10, p.95ff, p.172ff].

In education, object-oriented (OO) languages are often used today. The learners do not always know that modular modeling and programming does not require OO. As a side remark, we have observed that even our current students at university only know the OO representation and therefore believe that modular programming can only be accomplished with OO.

3.2 Abstraction of Actions

A systematic abstraction of actions leads to *structured programming*, which is summarised by *Constantine* and *Yourdon*:

> '[...] we have seen that we can generally minimize the cost of implementation, maintenance, and modification — three of the major technical objectives for current computer systems — by designing systems whose pieces are small, easily related to the application, and relatively independent of one another' [2, p.29].

Depending on the programming paradigm, *control structures* in imperative languages or *recursive functions and procedures* in functional languages are used. In general, *procedures* are used as a first mechanism for aggregating a sequence of statements into increasingly abstract statements, which then can be grouped further into libraries or modules.

3.3 Abstraction in School

Both forms of abstraction have to be taught in CS. Problem-solving is only possible when both are experienced and practised equally. An exclusive abstraction of the data leads to a description of the static world without function, and an exclusive abstraction of the actions leads to a variant of Mathematics. *Lewis* therefore makes this clear in his exemplary analysis of the OO approach:

> **Myth: Object-Orientation and Procedural Concepts Mutually Exclusive**
>
> 'An OO approach does not abandon the concepts that we admire in a procedural approach; it augments and strengthens them.
>
> We've known for years that solving a problem requires dissecting it and its solution into multiple, manageable pieces. We know that modularity and encapsulation are valuable design concepts. ...That idea underlies the definition of an abstract data type (ADT) .... These concepts are not only represented in an object-oriented approach, they are its cornerstone' [8].

Likewise, *B¨osz¨orm´enyi* correctly emphasizes that 'the concept of modularisation is much more fundamental than that of object-orientation' [1, p.142]. If we follow *Wirth*, it is not even a new, independent concept, but only a new technique. In an interview, he explains:

> 'Procedural programming is still the most common paradigm, and it will remain so, because the semantic gap between procedural languages and computers is smaller than for any other paradigm. Instruction sequences are represented by statements, and the state space by variables. I consider objectoriented languages also as procedural. ' [10].

4 Automation

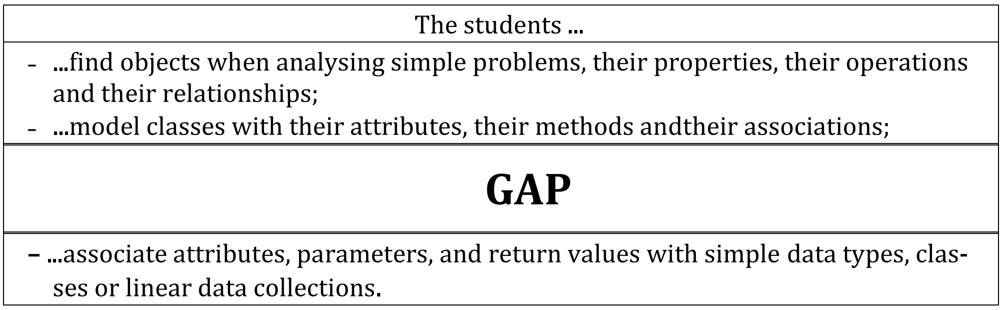
From the abstract description of data and sequences, a program has to be created, which is by no means simple or trivial. From the model, which consists of the abstraction of data and sequences, a program has to be developed in a concrete programming language. In accordance with the paradigm and the syntax of this language, the pro-

[10] www.simple-talk.com/opinion/geek-of-the-week/niklaus-wirth-geek-of-the-week/, last viewed: 2017/06/30

gramming language provides certain basic data types and control structures as building blocks. In order to proceed from the abstract model to the concrete program, a *gap* between the abstract ideas and the concrete structures of the programming language must be overcome.

4.1 The Modeling Gap

This *gap* is illustrated by the example of a recent curriculum for higher secondary education in a German state (see Fig. 1).



The students ...
- ...find objects when analysing simple problems, their properties, their operations and their relationships; - ...model classes with their attributes, their methods andtheir associations;
GAP
– ...associate attributes, parameters, and return values with simple data types, classes or linear data collections.

Fig.1. The Modeling Gap (highlighting of the ’gap’ by the authors).

For example, it is not at all evident for students to model the sorting of a stack of cards by an array or a list. The presentations in most schoolbooks suggest that this is self-evident, and many teachers teach it this way, mostly without further didactic reflection. However, if the teacher does not influence the students, they are more likely to model individual cards because they cannot or will not be aware of the fact that an array with indexing (or alternatively a list) is a suitable structure for this problem.

Likewise, there is a similar gap between the algorithmic abstraction and the specific programming language. Every teacher knows how long it takes for the students to recognize and apply the principle of iteration over a field as a suitable construct for solving certain problems. Only with this knowledge students can select the array as the appropriate data type with its functionalities in their chosen programming language.

A gap can only be overcome by jumping from one side to the other. But this has to be learned. A jump across a gap from one side is as complicated as it is from the other side. This means that if I succeed as a student to model reality, I have to learn how these models are mapped in my programming language. If I can move adequately in my programming language for small problems, I still have to learn ’to break down’ the reality into these smaller structures.

5 Teaching Approaches

Knowledge of data representation and control statements is therefore necessary. Both aspects must be learned and practised. The best way of sequencing and/or interweaving the two is not obvious.

5.1 OO-Guidelines: Start with Big Projects

The fundamental problem of OO systems and, in addition, the special syntactic structures of the *Java* language, which are difficult to understand for beginners, lead K¨olling and Rosenberg to make suggestions for a more successful initial introduction to Java [7]. In addition to other suggestions such as starting with *reading* code (instead of *writing*), one of their main claims is that the overhead of OO is relatively smaller the larger the system is. They conclude from this that teaching should start with a relatively *big* project.

5.2 PRIMM: Start with Small Projects

Sentance and *Waite* describe a related approach with their *PRIMM*-method [13].

Predict *summarise what given code will do on execution*
Run *execute code to test prediction*
Investigate *explain, trace, annotate, debug*

Modify *edit program to change its functionality* **Make** *design a new program*

But unlike K¨olling and Rosenberg, they say: ’Novice learners should be focused on very small tasks with single elements, with emphasis on reading and tracing code before they are expected to write code snippets.’ [13] Based on these considerations, it makes sense to use only *small* projects in class for beginners.

6 Description of the Investigation

6.1 Reason for our Investigation

Initially, we found the OO-Guidelines of Sect. 5.1 appealing and taught previous courses accordingly, but eventually found that the students found it particularly difficult to work with large projects right from the beginning. In the academic year of 2017/2018, we therefore conducted a comparative study with new groups (each taught 3 hrs/week for a full semester):

- One course was taught on the basis of a larger project according to the ideas of *K¨olling and Rosenberg* (Sect. 5.1).
- The other course was taught with one project to be developed on ideas of an approach of *PRIMM* (Sect. 5.2).

Despite this difference, the intended learning outcomes of the two groups were the same.

The first group started with a complete project with about 200 LOC that simulated a card game. Two players as instances of the players class play a simple card game. Each has a stack of stack_class cards, which consists of a certain number of cards from the playingcards class.

In the second group, the program (65 LOC) consisted at first only of a stack class. The individual cards had only a single number as their value. In the actual program, two stacks were instantiated and cards were alternately drawn from the stacks and compared.

The basic steps are similar in the *OO-guidelines* and in *PRIMM*: the programs are first read and analyzed. Subsequently, small changes are made and later additions are conducted. Both groups were taught such that they were at about the same step at any given point.

6.2 Composition of the Groups

The groups were assembled by the administration for structural reasons. Unfortunately, they were not the same size. One group had 27 students at the beginning, the other group only 17 students. The smaller group was taught according to the *OO-Guidelines*. Both groups were taught in a computerlab with 30 student computers, so that each student could work on their own computer. It was a challenge for the teacher to meet the demands of all students, especially in the large group, during the computer work phases. It goes without saying that the students in the smaller group had an advantage here.

6.3 Description of Empirical Investigation

We asked four times for a description of the terms *field, class, attribute, object, abstract class, abstract data type* and *OO programming* before the series of lessons and after about 2 months, at the end and after another 3 months. Additionally, it was asked how, for example, a car class for a car trade could look like. Reproductive requirements were therefore essentially needed.

The answers of the students were converted into a value of a Likert scale from 0 to 5 for the rating No answer, wrong, mostly wrong or incomplete, partly wrong/partially right, mostly right, right.

For a meaningful evaluation of the results, we calculated the *effect size d* according to *Hattie* [5] for the individual questions. This enables us to not only to not only compare the results of the two groups against each other, but also whether a positive learning effect can be observed at all. According to Hattie, the goal is to achieve an effect size of $d \geq 0.4$ in a one-year class of about four hours per week.

6.4 Selected Descriptive Statistical Results

The bar diagram in Fig. 2 shows that from the point of view of a teacher the results are rather disillusioning. The students’ statements on all questions were largely negative. This partially contradicted the feeling of the teacher during class conversation and while observing the active behaviour at the computer. This emphasizes the fact that it is extremely difficult, even for an experienced teacher, to decide whether more than superficial learning is actually taking place in the presence of active, interested pupils.

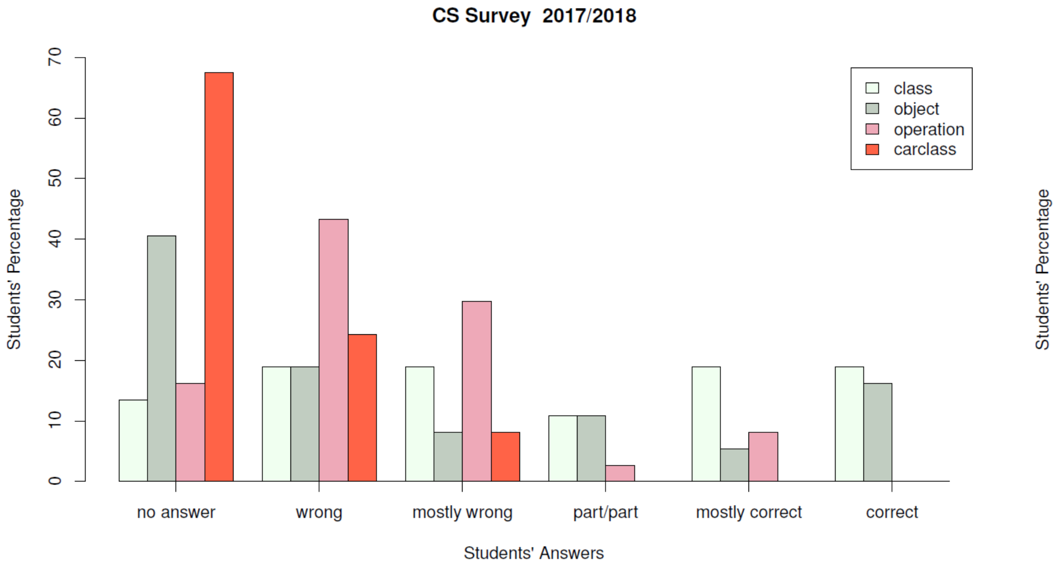


Fig.2. Post answers about the items *class*

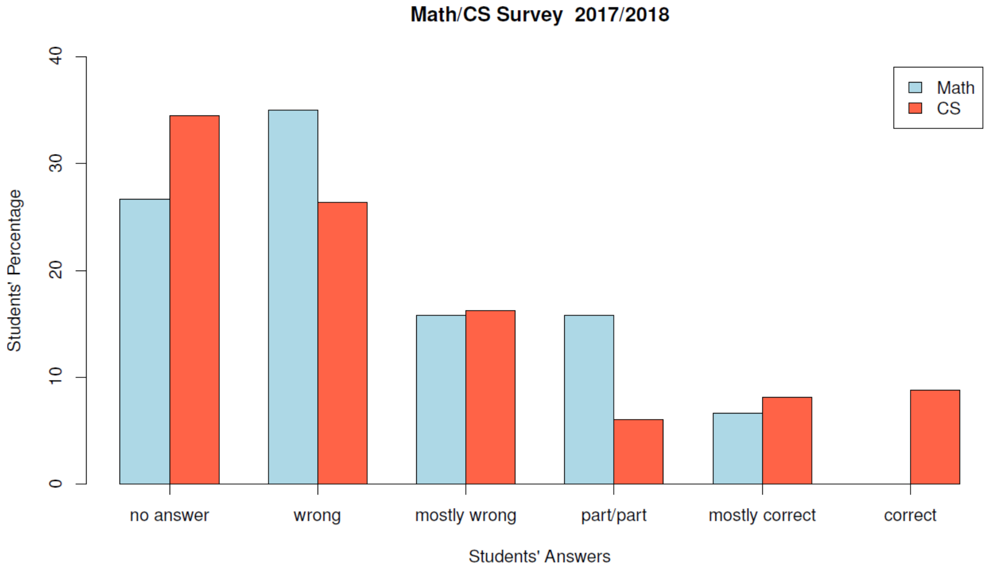


Fig.3. Math- and CS- answers *operation, object, example carclass*

However, this is not a peculiarity of CS lessons. As a sanity check, in the last lesson of the school year we asked the students about similar ’simple’ concepts of mathematics (like integral, derivation, etc.), which they have been exposed to much more intensively. The result is qualitatively not much different; see Fig. 3. The outcome in mathematics is all the more surprising since, in contrast to computer science, there is a lot of experience with mathematics teaching.

Still, Fig. 4 shows that an increase in learning can be observed in both groups regarding the understanding of new concepts such as *class, object, attribute*. It also becomes clear in Fig. 5 that such an increase does not come ’automatically’ by just using the appropriate concepts: although both groups worked with *arrays* (which had already been introduced before), no increase can be observed. This shows that a simple use of a structure without further reflection does not lead to further knowledge.

6.5 Evaluation of Selected Statistical Results

Initially, we were interested in the question whether teaching approaches with larger and smaller projects show significant differences. This, however, cannot be deduced from our data, since in both groups a significant number of students had great difficulties with these concepts.

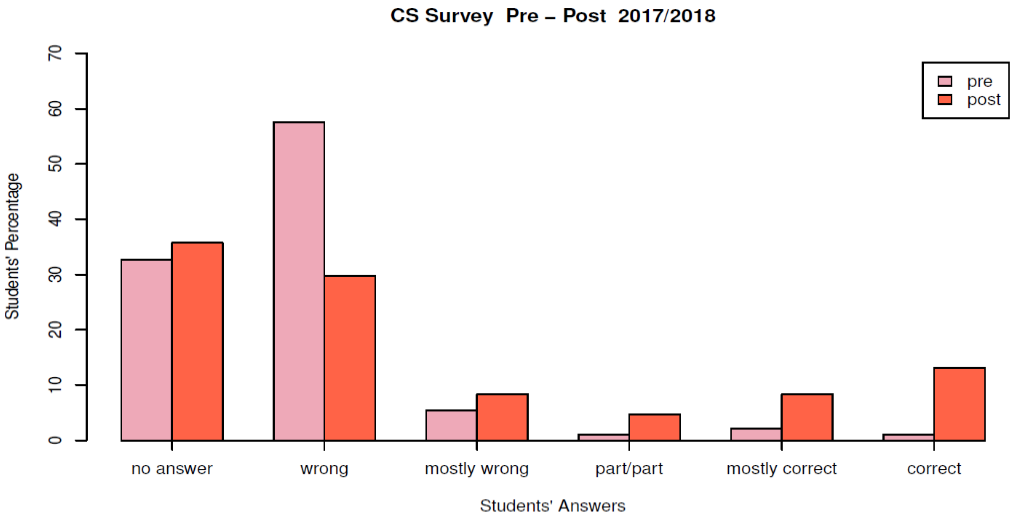


Fig. 4. Pre- and post answers about the new CS items

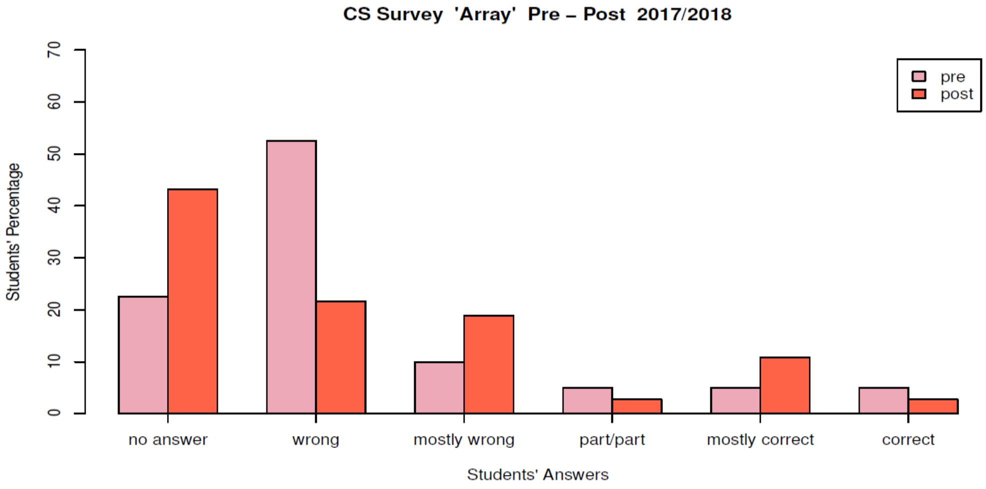


Fig. 5. Pre- and post answers about the data structure *array*

An assessment of the effectiveness of an educational measure can be made with the *effect size* according to *Hattie* [5, p.8]. In table 1 we can see the effect of our teaching on the concept of *arrays*:

Table 1. Effect size *d* pre-post * data structure *array*

Effect size	Array
Group OO-Guidelines	- 0.10
Group PRIMM	- 0.03

We can see that the students in both groups do not show an increase in the understanding of this data structure. This structure was used but not taught in this lesson.

By contrast tables 2 and 3, describing the learning progress in the new concepts from OO programming show that some of the students have made considerable progress. This is even unexpectedly large, since according to Hattie an effect strength of $d \geq 0.4$ is already satisfactory. With the individual terms this is almost always achieved, only with the explanation of the *OO programming* as 'meta term' this is clearly missed.

Table 2. Effect size *d* pre-post *OO – items*

Effect size	class	operation	object	carclass	oo
Group OO-Guidelines	0.98	1.39	0.55	0.15	0.18
Group PRIMM	1.21	0.36	0.56	0.07	0.18

.

Table 3. Effect size *d* OO-Guidelines - PRIMM *OO – items*

Effect size	class	operation	object	carclass	oo
pre	-0.19	0,70	0.17	0.31	-0.16
post	0,16	-0.11	0.19	0.08	-0.09

A comparison of the two approaches reveals an inconsistent picture. While in the group with the smaller projects a better description of the term class succeeds, the single terms like object, operation etc. are better described in the other group. Obviously, both approaches have made comparatively equal progress possible for the students.

6.6 Interpretation of the Results

Our intention was to examine which learning outcomes can be observed when learning the basic concepts of OO with the two approaches described in Sect. 5. Our groups did not consist of extremely motivated and/or interested students. The time invested by these students in addition to teaching is very small. Therefore, their programming skills are not very high.

The teaching approach in both groups was similar. In both groups there was a complete program for a game that was read, analyzed, slightly modified and then completed. In one group the project consisted of several classes and mostly several instances of this class as it is required by *K¨olling and Rosenberg* [7]. The other group started with a smaller project consisting of only one class according to the ideas of *PRIMM* [13].

The study showed that there is a significant learning process regarding the basic concepts of OO programming. However, these observations have to be made against the background that many students have a very weak overall understanding of these terms even at the end of the teaching unit, so that practically no competences are yet available with regard to the modeling of a new problem (definition of a car class).

Both groups have a similar increase in learning. However, it must be borne in mind that the group with the larger projects had a considerable advantage. It was significantly smaller (17 compared to 27 students) and in this group half of the students undertook written exams compared to only 3 students in the other group. Although this usually implies an increased motivation and interest in obtaining good grades, the small group did not achieve better results in our survey.

Even if the basic skills of students in CS do not differ qualitatively from those in Mathematics, the question arises whether a considerable number of students are overwhelmed with the entire subject of OO modeling and programming and whether a simpler approach should be chosen to meet the new challenges. Obviously, the OO modeling that many CS educators claim to be close to reality cannot be reconstructed easily by the students.

Thus the *Cognitive Load Theory (CLT)* [9] is to be considered, which was confirmed in a number of studies. It suggests to include only a few new elements in the learning process at a given time. The empirical investigations of *Hattie* [5] confirm this in an impressive manner. The connection of neuroscientific and pedagogical empirical knowledge is described in detail by *Hattie* and *Yates* [6].

Portnoff [11] demands a different approach to the teaching of novices on the basis of the *Siegmund study* [14], which has found that learning a programming language primarily addresses the linguistic rather than the logical area in the brain. These additional findings make it clear that novices must meet considerable requirements in addition to the aspects of process and data structuring when learning a programming language. This is still far from being didactically accepted and developed.

7 Summary and Outlook

In the last few years, the teaching of CS has changed at least in some schools. CS has become a 'normal' subject. This requires more than in the past a review of the intended competencies, in particular for basic modeling activites for data abstraction and automation.

In a comparative study we examined how working with projects that are first read, analyzed, slightly modified and then completed influences the learning process. In one course we worked with larger projects, in the other course with smaller projects.

By means of *effect size d* [5] we investigated and compared the two groups. In both groups an increase in learning can be observed; however, it is present in only half of the students. Hardly any growth can be detected in the other half. Even though this is similar in other subjects such as Mathematics, this situation is unsatisfying.

In further research we want to analyze the above-mentioned considerations and interpretations through further investigations. The next step is to investigate whether a teaching approach without a project of any kind is a better approach. In the forthcoming academic year, two teaching groups will be formed again. A group should be taught again according to the ideas of *PRIMM*. In the other group, very small examples optimized for the respective content will be presented. To reduce the influence from syntactic overload, these groups will be taught with the *Groovy* language, which uses Java-syntax, but enables the programmer to omit certain syntactic elements of a full Java program that can be distracting for beginners.

It should also be examined whether a completely different path to learning abstraction and automation can be more successfull than any OO-apprach. For this, the question of the used programming language will also have to be reconsidered.

References

1. Böszörmenyi, L.: Why Java is not my favorite first-course annote. Software – Concepts & Tools **19**, 141–145 (1998)
2. Constantine, L.L., Yourdon, E.: Structured Design - Fundamentals of a Discipline of Computer Program and Systems Design. Prentice Hall, London, 2 edn. (1979)
3. CSTA STANDARDS TASK FORCE: [Interim] CSTA K–12 Computer Science Standards: Revised 2016. Tech. rep., New York, NY, USA (2016)
4. Denert, E.: Lob der struktur. LOG IN (187/188), 33–37 (2017)
5. Hattie, J.: Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement. Taylor & Francis (2008)
6. Hattie, J., Yates, G.C.R.: Visible Learning and the Science of How We Learn -. Routledge, New York (2013)
7. Kölling, M., Rosenberg, J.: Guidelines for teaching object orientation with java. In: Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education. pp. 33–36. ITiCSE '01, ACM, New York, NY, USA (2001)
8. Lewis, J.: Myths about object-orientation and its pedagogy. In: SIGCSE 2000: Proceedings of the 31st ACM technical symposium on Computer science education. pp. 245–249. ACM, Austin, TX, USA (2000)
9. Paas, F., Renkl, A., Sweller, J.: Cognitive load theory and instructional design: Recent developments. Educational Psychologist **38**(1), 1–4 (2003)
10. Pomberger, G.: Software Engineering and Modula-2. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK (1986)
11. Portnoff, S.R.: The introductory computer programming course is first and foremost a *language* course. Inroads **9**(2), 34–52 (2018). https://doi.org/10.1145/3152433
12. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., Owens, B.B., Stephenson, C., Verno, A.: CSTA K–12 Computer Science Standards: Revised 2011. Tech. rep., New York, NY, USA (2011)
13. Sentance, S., Waite, J.: Primm: Exploring pedagogical approaches for teaching text-based programming in school. In: Proceedings of the 12th Workshop on Primary and Secondary Computing Education. pp. 113–114. WiPSCE '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3137065.3137084, http://doi.acm.org/10.1145/3137065.3137084
14. Siegmund, J., Ka¨stner, C., Apel, S., Parnin, C., Bethmann, A., Leich, T., Saake, G., Brechmann, A.: Understanding understanding source code with functional magnetic resonance imaging. In: Proceedings of the 36th International Conference on Software Engineering. pp. 378–389. ICSE 2014, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2568225.2568252
15. Wing, J.M.: Computational thinking. Communications of the ACM **49**(3), 33–35 (2006)

How beginner-friendly is a programming language? A short analysis based on Java and Python examples

Jean-Philippe Pellet[1,2], Amaury Dame[2], and Gabriel Parriaux[1]

[1]University of Teacher Education, Lausanne, Switzerland
{jean-philippe.pellet,gabriel.parriaux}@hepl.ch
[2]École polytechnique fédérale de Lausanne, Switzerland
amaury.dame@epfl.ch

Abstract. In this paper, we are interested in criteria to help us choose a programming language for a freshman programming course. The audience is future engineers who won't be computer scientists or IT professionals. We are therefore more interested in conveying elements of computational thinking and logic rather than full mastery of a given language. Following a rather exceptional situation where we had to give substantially the same course, in parallel, once in Java and once in Python, we relate here some syntactic and semantic aspects of the two languages which, in our experience, ease the teaching or learning of basic programming concepts. We argue that in quite a few cases, Python makes basic concepts easier to introduce because of less *syntactic noise* and less *conceptual noise*. We also propose a short list of syntax- and semantics-related desiderata for a beginner language—which neither Java nor Python completely answer.

Keywords: Choice of programming language · Computer science education · Syntax and semantics

1 Introduction

How to choose the first programming language keeps being one of the most debated questions in computer science education [11]. Lower-level languages are closer to the machine and may be able to convey a better conceptual model of how a computer works; higher-level languages let us express more concisely more complex computations, data manipulations, more general problem-solving strategies. Both can be desirable depending on the course objectives and the relevance of a language can never be assessed independently of its audience.

We teach introductory programming to college freshmen. They are future engineers, but they probably aren't future computer scientists, software engineers, or developers. Our course doesn't aim at making students proficient in the chosen programming language. It is jointly taught with another course presenting an introduction to the foundations of computing. For most students, these two courses will be the only ones fully dedicated to computing and programming. Our course duo is thus positioned as presenting the theoretical foundations of computing, discussing the bases of algorithmic approaches, reviewing open questions, and getting acquainted with the basics of programming. The focus is set on structured approaches to problem solving emphasizing higher-level, potentially transferrable skills like modeling, abstraction, decomposition, etc.—referred to by some as computational thinking [15].

Many articles have been written about what a first programming language should look like, some of which we mention in Section 2. Here, owing to our experience and to the context mentioned in Section 3, we decided to explicitly put focus on the comparison between Java and Python from a syntactic and semantic point of view. Language-design decisions, be them major or minor, have a large influence on how easy certain concepts can be taught with them, and how syntactically or semantically intermingled these concepts can be. Throughout Section 4, we show and discuss short code excerpts in Java and Python taken from the course and discuss the two approaches, introducing the concepts of syntactic noise and of conceptual noise. As we go through the discussion, we propose several desiderata for a programming language used in such a beginners' course. That list can be used to determine the extent to which, from the mentioned points of view, a language can be seen as appropriate for introductory teaching of programming. We finally conclude in Section 5.

2 Related Work

A quite comprehensive survey of literature on teaching introductory programming [11] covers 4 main topics, one of which is language choice. The authors mention many references and classify them into those which (a) discuss a given language, (b) compare two or more languages, and (c) present general pedagogical criteria for an easier language choice. Among those criteria, a distinction is made between internal criteria (related to the language itself) and external criteria (industry demands, trends, availability of teaching material, etc.). Our paper can be classified as presenting internal criteria for language choice while justifying them with a comparison between two languages.

An early paper written in a similar spirit [7] lists seven "deadly sins" of introductory programming language design. The authors present internal criteria and illustrate some of them with various popular languages at the time (which, however, do not include Java or Python). This is built upon by another paper that adds the consideration of external criteria [7].

Other authors point out that generally valid criteria are hard to define as different courses have different learning objectives [6] (hence our contextualization in the last section) and that language choice remains an ever-contentious issue.

Closer to our setting, we can read about a comparison of errors made in 60 beginner programs, 30 in Java and 30 in Python [5], and find that Python programs contained on average fewer syntax errors, logic errors, and better errorhandling code. The same authors further examine Python [4], concluding it is simple enough for introductory teaching in high school already.

So while the general topic of this paper has been discussed many times, the detailed syntactic and semantic elements we investigate below, to the best of our knowledge, have never been treated precisely in such a learning context.

3 Context & Initial Thoughts

In the next section, we'll present our observations and findings from the parallel
Java and Python courses we gave in the setting explained in the introduction. The courses consisted in 14 weeks of a lecture of 45 minutes followed by a 90-minute practical exercise sessions. The main teaching objectives are formulated as follows: at the end of the courses, students shall be able to...

- build a conceptual model of how a computer executes code;
- use the basic control structures of a chosen programming language (conditions, loops, functions);
- choose and use appropriate data structures (sets, sequences, associative maps) to solve a given problem;
- build simple data models and process data with them.

The code excerpts in the following section which support our discussion were directly taken from the lecture slides or exercise solutions. We want to point out that the excerpts were not specially crafted to demonstrate the superior adequacy of either language. They were historically written for the Java course, and were recently migrated over to Python for a new variant of the course given to another faculty in the same institution.

Before diving into the code excerpts, we want to address a short list of preliminary objections one could make to the analysis below and which we have heard during discussions with colleagues.

— *"You can't compare Java and Python. Java is typed, safe, compiled, efficient; Python is for fast prototyping and lacks all of the above. They have totally different use cases."*

Of course, we can compare them! We must compare them, since at some point a decision has to be made. This doesn't in any way negate their different relevance. They come with different tradeoffs—which just have to be discussed from a teaching and learning perspective, independently of their merits in the industry or in other fields.

— *"I don't want to choose a single language. I could very well teach Python* and *Java. Learning to transfer skills from one language to another is important."*

In the beginning, sticking to one language is important. We know that transference cannot be expected from novices and only comes with mastery [3,1]. Plus, the question of language choice would hardly be solved—one would need to agree on *two* languages instead of just one.

— *"You say needing to care about feature X of the language is a hindrance for beginners. But I want them to have to care about X: it teaches them how to be precise, how to care about formal details, and they'll end up knowing more about the lower-level foundations and how the machine works."*

Our semester is 14 weeks and there's only so much we can do in that time. We could of course have decided that such learnings matter and are relevant. But skipping them also gives us more time to address higher-level concepts that may more easily be applicable to other languages (and, more broadly, to general computational thinking), contrary to lower-level, language-specific constructs.

— *"One should start programming with a block-based visual language, which avoids the mentioned syntactic pitfalls."*

Yes, one could. Actually, we would strongly recommend such an approach for K–12 education, for instance. But we should stay aware that not all of the syntactic pitfalls are avoided simply by using a block-based language. The issue of conceptual noise remains, and visual languages have limitations of their own.

— *"I don't want to teach in any language that is not statically typed."*

This seems to be a common gripe. Many traditionally dynamically typed languages can now include type annotations in language extensions (or derived languages)[11]. *Gradual typing* thus becomes possible and allows to fine-tune the tradeoff between (roughly out) the extra typing necessary and the additional diagnostics a static type checker can provide. It sounds hardly possible, though, for a statically typed language to relax its need for types—although modern languages have type-inference capabilities that alleviate the feeling of being too "constrained" by the need of declaring types everywhere. Whether to type or not type is a huge theme on its own that we, for lack of space but with regret, don't discuss further in this paper.

4 Analysis & Desiderata

We now present a few code excerpts[12] and comment on the differences in ease of teaching and learning that we have observed. Note that, in such a short paper, we make no claim of exhaustivity or coverage whatsoever. Also note that the arguments and desiderata presented below are to be read with the target audience in mind—non-CS, beginner students, first semester of college—and we don't claim that they automatically suit a more general context.

Table 1. A slightly more complicated "hello world".

Java

```java
public class Demo {
  public static void main(String[] args) {
    int side = 4;
    int area = side * side;
    System.out.println(area);
  }
}
```

Python

```python
side = 4
area = side * side
print(area)
```

This could come right after a "hello world" example. Its intent is to show the concepts of (a) storing a value in a variable; (b) using an arithmetic operator; and (c) printing something to the console, and illustrate the way to determine the "entry point"/main function of the program. To newcomers, it is already pretty dense code, as there are indeed these three new concepts to figure out conceptually and to recognize syntactically[13]. The semantics here of Java and Python is really close, but the syntactic effort is quite different. Symbols like braces and brackets are worth pointing out because they need more explaining to understand since for beginners, they are not immediately linked to a concept they know—an exception may be the parentheses in the definition and application of a mathematical function[14]. But the opening and closing braces and brackets, the dot and the semicolon are all here "parasitic" in Java, because they are needed even for this very small example to run. Moreover, no fewer than five keywords are needed, under each of which a potentially complicated concept is hiding. Usually, we tell student not to worry about them now and just to accept them—which is problematic, because from one of the very first examples, this shows that there's a lot of unknown already in the basic lines they have to write.

[11] MyPy with type annotations since Python 3.5, TypeScript or Flow for JavaScript, type hints since PHP 5, Typed Racket, Typed Closure, etc.

[12] The width of one unit of indentation was chosen to be 2 columns for easier layout, but was always presented to students as 4 columns.

[13] There are other, less directly apparent concepts like scoping and memory management that can be discussed with this example and that we opted to leave out for now with beginners.

[14] For consistency with the syntax used for stack-allocated objets with custom constructors, C++ goes the other way and allows any variable to be initialized with a parenthesis-based syntax resembling function application.

We call this noise, defined as extraneous elements that have to be typed in (and, potentially, understood), while being unrelated, or very indirectly related, to the basic concept we wish to illustrate with some code except. This noise is strongly linked to the syntax of the language. We propose to call it syntactic noise when it is due to simple syntactic rules of the language without a strong underlying, independent concept (e.g., the need for semicolons in Java), and to call it conceptual noise when additional concepts need to be explained and understood to measure the implications of the syntax being used (e.g., the use of static in Java).

We see that the Python version, on the contrary, is terse and corresponds line by line to the three new concepts with no syntactic or conceptual noise. (We avoid the issue of typing, rapidly mentioned in Section 3.)

Desideratum 1. The first steps in a language (e.g., console print, variable assignment) should be painless and minimize both the syntactic noise and the conceptual noise.

Table 2. Creating an associative data structure containing 3 key–value pairs.

Java

```java
Map<Integer, String> numbers = new HashMap<>();
numbers.put(1, "one");
numbers.put(2, "two");
numbers.put(10, "ten");
```

Python

```python
numbers = {
  1: "one",
  2: "two",
  10: "ten"
}
```

This examples demonstrates the definition of a simple map (the preferred term in Python being "dictionary") containing three entries. If it is meant to solely illustrate the possibility of such a data structure, then the Python code does just that, without conceptual noise. The Java code as shown here needs the `new` keyword, the `Integer` object type (to beginners, subtly different from the basic type `int`), type inference with the diamond notation, and repeated method calls on the constructed objects. Syntactically viewed, Python here has a more declarative approach, whereas Java's approach is imperative and requires a more complex mental model of what is going on, which must then be explained. Conversely, Python has special syntax here: the braces, the colon and the comma all have a specific meaning in this context, which must be learned. We argue that the cognitive effort of it, however, is lesser than the one of learning the bits of the object-oriented concepts needed to grasp the essence of the Java code.

Note that, since Java 9, one could write: `Map.of(1, "one", 2, "two", 10, "ten")` (with up to 10 key–value pairs—yielding here an immutable map), wrapping lines as needed to achieve a style closer to the Python style. The syntax, however, denotes nothing else than a regular function application—looking in the middle of the argument list provides no immediate way for the reader to distinguish a key from a value without scanning further left or right, and even less if the keys and values have the same type.

This example here begs the question: as we introduce a new concept that is materialized in a language, when is new syntax desirable? On the one hand, reusing previously introduced syntax and concepts seems elegant: new ideas can be expressed in terms of simpler, basic constructs, and keeping the core of a language small and expressive enough to cover most needs sounds elegant. On the other hand, one can argue that a sufficiently different concept warrants new syntax. Having a specialized syntax dedicated to a concept makes it immediately recognizable in the code. Provided the use of that concept is sufficiently widespread, a dedicated syntax becomes actually an easier way to embody and convey the new concept, rather than showing how existing syntax can be "semantically overloaded" to support this new concept as well. Reusing existing syntax puts the emphasis on *how* a certain mechanism is implemented in the language, whereas special syntax better denotes the *intent* of the programmer[15].

Desideratum 2. Broadly used concepts sufficiently different from other concepts should have dedicated syntax (and not rely on existing syntactic constructs).

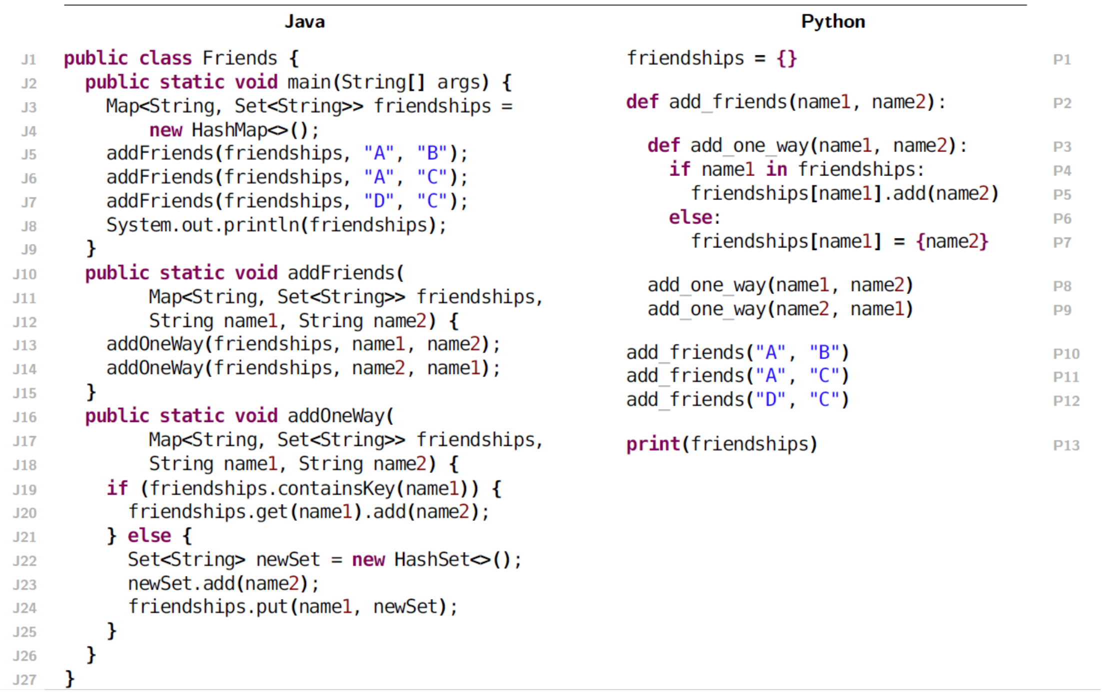
Basic data structures, we argue, fit these requirements. Data structures are often mentioned right next to algorithms as a constituents of programs, right up to the very name of classic textbooks [16]. To us, it seems reasonable to define such initial data with a syntax that looks declarative rather than imperative—just like defining the structure of graphical user interface is more conveniently tacked declaratively (e.g., React, Vue.js, SwiftUI, etc.).

[15] As a side note: this is one of the main arguments for having a dedicated syntax for implicit parameters in Scala 3, whereas they were written as regular parameter lists in previous versions. The mechanism was found embodying sufficiently different concepts (like typeclasses) and was found to be clearer with dedicated syntax [10].

A syntax denoting primarily a function call is maybe not best used to represent associative data. In that spirit, having specialized syntax to denote, for instance, a linear data structure (an array, list, or set), and an associative data structure (map) thus seems worthwhile. In this regard, Python has special syntax for list, set and (as shown above) dictionary literals; Java has special syntax for array literals only. Today, Java arrays are regarded as a low-level data structure, even being called "deprecated" in favor of generic dynamic lists [9]. The interest of special syntax for them seems clearly smaller than it would be for more dynamic and flexible data structures.

We also note that many operations mutating simple linear and associative data structures can be done via subscripting in Python. Subscripting exists in Java but only for arrays and only via `int` indices. In Python, lists for instance support subscripting via the indication of a single index like in Java, but also of slices of indices[16]. Slice-based subscripting supports assignment as well, thus enabling higher-level multiple modifications in a single statement. A syntactic advantage is that the assignment operator `=` used more widely in various circumstances where it denotes a mutation of the data structure on its left hand side. This is consistent with a simple variable assignment known from some of the very first code examples.

Table3. Extracting repeated code sections in methods/functions.



Java

```
J1   public class Friends {
J2     public static void main(String[] args) {
J3       Map<String, Set<String>> friendships =
J4           new HashMap<>();
J5       addFriends(friendships, "A", "B");
J6       addFriends(friendships, "A", "C");
J7       addFriends(friendships, "D", "C");
J8       System.out.println(friendships);
J9     }
J10    public static void addFriends(
J11          Map<String, Set<String>> friendships,
J12          String name1, String name2) {
J13      addOneWay(friendships, name1, name2);
J14      addOneWay(friendships, name2, name1);
J15    }
J16    public static void addOneWay(
J17          Map<String, Set<String>> friendships,
J18          String name1, String name2) {
J19      if (friendships.containsKey(name1)) {
J20        friendships.get(name1).add(name2);
J21      } else {
J22        Set<String> newSet = new HashSet<>();
J23        newSet.add(name2);
J24        friendships.put(name1, newSet);
J25      }
J26    }
J27  }
```

Python

```
friendships = {}                                  P1

def add_friends(name1, name2):                    P2

  def add_one_way(name1, name2):                  P3
    if name1 in friendships:                      P4
      friendships[name1].add(name2)               P5
    else:                                         P6
      friendships[name1] = {name2}                P7

  add_one_way(name1, name2)                       P8
  add_one_way(name2, name1)                       P9

add_friends("A", "B")                             P10
add_friends("A", "C")                             P11
add_friends("D", "C")                             P12

print(friendships)                                P13
```

This excerpt demonstrate the effort to factor out repeated code. Here, we are trying to populate a map linking names to sets of the names of their friends, symmetrically (i.e., if `"A"` is in the set of friends of `"B"`, then the converse must be true as well). We are trying to isolate two procedures (here, methods in Java and functions in Python): (a) a high-level procedure linking two new friends (lines J10–J15 and P2–P9) which, for its implementation, calls twice (b) a lower-level procedure adding an element to a set in a map (lines J16–J26 and P3–P7), creating the set as required[17].

One immediately evident difference is the block syntax: Java uses a bracebased syntax heavily based on C's whereas Python uses the colon character and significant indentation to determine block start and end. Much has been written about it already [13,4] regarding ease of reading, writing, editing, including caveats with copy-pasting, so we'll focus on other aspects. Another key difference is the ability, in Python, to define named functions much more flexibly. Java's method as always top-level members of a class whereas Python functions can be defined in any code block and have access to symbols of the enclosing lexical scopes (simplifying a bit). Python's `add_friends` and `add_one_way` can refer to `friendships` without it being passed along as an argument as in Java. There would be other ways to avoid the extra parameter in Java: declaring it as a field. But this comes

[16] For instance, `mylist[1:4]` yields a new Python list with elements 1 (incl.) to 4 (excl.). Custom classes can also support subscripting with arbitrary subscript types.

[17] Usage of a `defaultdict` was intentionally avoided here in Python as it requires passing a function as argument.

with its own conceptual noise: if it's a static field, one has to either explain its semantics or ask students to accept that it works as such. If it's a non-static field, the code in `main` needs to create a new instance. In both cases, the question arises of where to initialize the field (inline or in a method) and of initialization order, which may not follow the order of the lines in the file any more. Furthermore, in Java, both method definitions and field definitions may begin with as many as the same two keywords plus a type information (e.g., **`public static int`**), which can be confusing to the untrained eye. In Python, from the first keyword on, **`def`**, it is unambiguously clear that a function is being defined and no mix up with a variable (or attribute) is possible[18]. Also note that the possibility to define nested functions avoids polluting the outer namespace with symbols that are not used later anyway.

Finding repeated patterns in data and similar patterns in code is an essential part of computational thinking. Giving names to subprograms performing a given task is equally an essential means of abstraction, as it allows the creation of higher-level constructs based on lower-level ones. We believe it should be made as easy as possible in a language used for beginners.

> Desideratum 3. Functions should be syntactically easily definable and quickly recognizable so as to allow convenient code factorization.

Table4. Creating a simple compound data type with basic inheritance.

Java

```java
public class Rectangle extends BoardElement {
  public double width;
  public double height;
  public Rectangle(Point center,
         double width, double height) {
    super(center);
    this.width = width;
    this.height = height;
  }
  @Override public String toString() {
    return "Rectangle(c=" + center + ", w=" +
        width + ", h=" + height + ")";
  }
}

Rectangle r = new Rectangle(
    new Point(10.0, 10.0), 5.0, 2.0);
```

Python

```python
class Rectangle(BoardElement):
  def __init__(self, center,
               width, height):
    BoardElement.__init__(self, center)
    self.width = width
    self.height = height

  def __repr__(self):
    return (
      f"Rectangle(c={self.center}, "
      f"w={self.width}, h={self.height})"
    )


r = Rectangle(
        Point(10.0, 10.0), 5.0, 2.0)
```

This example serves the discussion of two concepts: the definition of a custom compound data type, and the basic introduction of class inheritance. They are treated separately in the text below. (Note that this Java code is not idiomatic: fields are typically private in Java with a public getter or setter as appropriate. In Python, attributes are technically all public and, by convention, are prefixed with one or two underscores to signal their non-publicness.)

The ability to define and use compound data types (i.e., a type constituted from several basic types of the language and possibly other compound types) is one of the first concrete steps of data modeling in a programming course. Students get taught that certain values that belong together in order to model a given entity should also "live" and "travel" together in code. In Java and Python, the most obvious way to achieve that is by declaring new classes[19].

Both the Java and the Python ways to declare a new class are relatively verbose and contain seemingly redundant elements. Both languages have a new keyword for it, **`class`**. In languages supporting class-based object orientation, classes are such a central concept that contesting the relevance of that keyword cannot be justified. The way to treat fields, constructors and constructor parameters, however, is different. Java requires the explicit declaration of all fields, whereas in Python, conventions dictate that the fields (referred to as attributes) be assigned in the `__init__` "constructor method". Owing to its dynamic nature, Python actually allows new attributes to be defined at later points of the lifetime of a created instance with no prior declaration, whereas Java forbids

[18] This says nothing of how, for instance, fields and methods should be *accessed*—Eiffel's uniform access principle [8] provides an nice abstraction over a class's implementation details and can be achieved in Python with *properties* (though not in Java).

[19] In Python, one could also used `namedtuples` for cases where methods are not needed. The data structures built this way, however, are immutable. Our take was not to skip discussions over mutable data structures, an essential side of imperative programming language.

it. Python is consistent here with its treatment of attributes and its treatment of, say, local variables: no declaration is needed. But in the perspective of basic data modeling, the essence of the definition of a compound type is the identification of what subparts it is made of. Java field definitions makes it clear, but Python makes it (at least partly) implicit. On the other hand, Java's syntax seems overly verbose for the very common use case where all fields obtain their initial values from the constructor: the identifier `width` appears no fewer than 4 times, compared to 3 in Python, owing to the nonexistent declarations.

It is notably difficult to explain to students the need for a simple constructor as shown in the example above. Especially the use of the same identifier as parameter name and as field, as is idiomatic in both languages, must be explained carefully. In Java, the use of the "`this.`" prefix also often is a source of confusion: it has to appear on lines J7 and J8, but is optional on lines J11 and J12 to refer to the fields. In general, referring to the current object is difficult to understand [12] and confuses beginners even more if it can be done in a non-systematic way. Of course, we as instructors could make it a rule to always use it even when optional. But the fact that students are bound to find code online that doesn't and, most probably, to end up writing code of their own that works without this prefix forces us to explain it anyway. Python here has another approach: the `self` parameter is always mentioned explicitly in the parameter list of methods, and only when prefixed by "`self.`" can attributes be referenced (also on lines P9 and P10). The always needed prefix improves on the syntactic consistency: no matter the context, class members (attributes and methods) are always prefixed by some expression evaluating to an instance, contrary to Java. It is thus always immediately apparent whether the code is accessing a local variable or a field, or calling a function or a method. But the mismatch in the number of parenthesized elements at definition site and at call site (where explicitly passing `self` mustn't be done) is confusing and requires additional explanations. Worse is the allowed coexistence of a way to call methods as functions, bypassing virtual dispatch, as shown on line P4, where `self` must be explicitly passed[20].

> Desideratum 4. Compound data types should be definable in a syntactically light way and should clearly allow the identification of their fields.

Both languages suffer from the fact that even after the above definition, there is no out-of-the-box mechanism for checking for the equivalence of two instances; i.e., if `r1` and `r2` are two `Rectangles` constructed with the same arguments, they will not be considered equivalent neither in Java (via `r1.equals(r2)`) nor in Python (via `r1 == r2`—more on that syntax below). Additional methods have to be implemented for that—even if, in most cases, their implementation is systematic and repetitive. The same goes for the generation of hash values for such instances: the default implementation is based on reference equality and not instance equivalence and will probably not act as intuitively expected when inserted into hashing containers such as sets and maps[21].

[20] Here, one could have written `super().__init__(center)` instead. But the fact remains: calling the constructor method of the superclass happens by using `__init__` name, in contradiction with the general guideline that "you shouldn't access it yourself if it has double underscores".

[21] At this stage, one cannot help but point to, e.g., Scala's way of defining such a compound type, where the definition of the constructor and the fields coincide

Table 5. Excerpt 5. Manipulating data structures representing numerical data.

Java

```java
J1   Vector3 v1 = new Vector3(1, 2, 3);
J2   Vector3 v2 = new Vector3(4, 5, 6);
J3   Vector3 v3 = v1.plus(v2).div(2);

J4   // Given a class similar to this (with a proper
J5   // toString() and equals() omitted here for brevity):
J6   public class Vector3 {
J7     public final double x, y, z;
J8     public Vector3(double x, double y, double z) {
J9       this.x = x; this.y = y; this.z = z;
J10    }
J11    public Vector3 plus(Vector3 v) {
J12      return new Vector3(x + v.x, y + v.y, z + v.z);
J13    }
J14    public Vector3 div(double d) {
J15      return new Vector3(x / d, y / d, z / d);
J16    }
J17  }
```

Python

```python
from numpy import array        P1

v1 = array([1, 2, 3])          P2
v2 = array([3, 2, 1])          P3
v3 = (v1 + v2) / 2             P4
```

Lines J3 and P3 both compute in a 3D space, the vector $v_3 = {}_{2}$. Java's impossibility to use arithmetic operators on custom types makes it impossible not to use method calls to accomplish this. But Python makes it more legible and writable by allowing operators (here in conjunction with a type defined in the `numpy` library). Allowing operators has the added benefits of reproducing the precedence rules we know from algebra, whereas, to the untrained eye,
distinguishing, e.g., `v1.plus(v2).div(2)` from `v1.plus(v2.div(2))` is difficult and may even be counterintuitive.

Note that the class definition in Java only serves to demonstrate how "arithmetic methods" are achieved, not to artificially lengthen the Java code. Conceptually, one must understand that, in order for chained method calls (as on line J3) to work, one needs to return an instance of the class, too. This in turns begs more questions that are avoided by Python's syntactic possibilities.

Another more difficult point of Java is equivalence checking: `==` denotes an identity check rather than an equivalence check, which must explicitly be done by means of the `equals()` method. It even gets trickier, because students rapidly get used to using `==` for all basic types and even for instances of `String`—and, more often than not, it actually works, since Java interns all strings literals compiled together. It only bites them later. Python's `==`, on the other hand, calls the method `__eq__` on its left operand, thus allowing custom behavior via standard syntax[22]. The same arguments can be made for the `<`, `<=`, `>`, and `>=` operators on instances of data types which are ordered.

Desideratum 5. Common arithmetic, equality, and comparison operators should work on compound data when relevant to ensure syntactic consistency with basic types.

Having stated these few desiderata, we regret not having evaluated them according to students' performance on tests or exams. This analysis emerged after the course was completed, at a point where all tests were already completed and turned out not to allow a proper isolation and evaluation of the aspects above. They are, however, based on our experience as teachers and are also stated in a way such that they are independent of this precise Java–Python comparison, being in principle applicable to any potential first language for beginners.

5 Conclusion & Outlook

Forty years ago already, researches wrote that "programmers tend to favor their first language to the extent of applying the style or structures of this language when programming in other languages" [14]. As teachers, we had better give serious thinking about the choice of the first language: especially when teaching to beginners clear embodiment of general concepts rather than of language-specific intricacies. The same author, however, writes that "the effects of poor teaching are an order of magnitude more significant than the effects of a poor first

[22] syntactically, and code for equality, hash value, and string representation is generated automatically (see https://docs.scala-lang.org/tour/case-classes.html).
Even if we all know that `==` is all but standard for beginners and that the confusion with `=` is extremely common.

language”— so, of course, we know that wisely choosing the first language is not the end of the story. Recently, ten quick tips for teaching programming were proposed [1], each of which is worth considering.

Based on our observations above, we have decided to move away from Java in favor of Python because many concepts were found to be more easily explainable without extraneous noise—while also noting that Python does not completely answers our desiderata. Another observation is that Python seems to be a choice more and more common at high school for students taking computing classes, such that the skillfulness at the start of the semester in Python may differ greatly among students. Choosing a radically different language (e.g., a functional language like Scheme or Haskell) was also mentioned as a way to start from scratch for everyone with high probability [2].

References

1. Brown, N.C.C., Wilson, G.: Ten quick tips for teaching programming. PLOS Computational Biology 14(4) (2018)
2. Felleisen, M., Findler, R.B., Flatt, M., Krishnamurthi, S.: The structure and interpretation of the computer science curriculum. Journal of Functional Programming 14(4), 365–378 (2004)
3. Gick, M.L., Holyoak, K.J.: The cognitive basis of knowledge transfer. In: Cormier, S.M., Hagman, J.D. (eds.) Transfer of learning: Contemporary research and applications. Academic Press (1987)
4. Grandell, L., Peltomäki, M., Back, R.J., Salakoski, T.: Why complicate things?: introducing programming in high school using python. In: Proceedings of the 8th Australasian Conference on Computing Education-Volume 52. pp. 71–80. Australian Computer Society, Inc. (2006)
5. Mannila, L., Peltomäki, M., Salakoski, T.: What about a simple language? analyzing the difficulties in learning to program. Computer Science Education 16(3), 211–227 (2006)
6. McIver, L.: Evaluating languages and environments for novice programmers. In: PPIG. p. 10 (2002)
7. McIver, L., Conway, D.: Seven deadly sins of introductory programming language design. In: Proceedings 1996 International Conference Software Engineering: Education and Practice. pp. 309–316. IEEE (1996)
8. Meyer, B.: Object-oriented Software Construction. Prentice-Hall (1988)
9. Naftalin, M., Walder, P.: Java Generics and Collections. O’Reilly (2007)
10. Odersky, M.: Some mistakes we made when designing implicits (and some things we got right). Talk given at Typelevel Summit 2019 in Lausanne (2019)
11. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J.: A survey of literature on the teaching of introductory programming. ACM SIGCSE Bulletin 39(4), 204–223 (2007)
12. Ragonis, N., Shmallo, R.: A diagnostic tool for assessing students’ perceptions and misconceptions regards the current object “this”. In: Pozdniakov, S.N., Dagiene˙, V. (eds.) Informatics in Schools. Fundamentals of Computer Science and Software Engineering. pp. 84–100. Springer International Publishing, Cham (2018)
13. Sanner, M.F.: Python: a programming language for software integration and development. J Mol Graph Model 17(1), 57–61 (1999)
14. Wexelblat, R.L.: The consequences of one’s first programming language. In: Proceedings of the 3rd ACM SIGSMALL symposium. pp. 52–55. ACM Press, New York, New York, USA (1980)
15. Wing, J.M.: Computational Thinking. Communications of the ACM, Viewpoint 49(3), 33–35 (2006)
16. Wirth, N.: Algorithms + Data Structures = Programs. Prentice Hall (1976)

Networked Embedded Systems in the Physical Computing Project “Smart City”

Mareen Przybylla[1], Andreas Grillenberger[2], and Andreas Schwill[1]

[1]University of Potsdam, August-Bebel-Str. 89, 14482 Potsdam, Germany
{mareen.przybylla,andreas.schwill}@uni-potsdam.de
[2]Freie Universita¨t Berlin, K¨onigin-Luise-Str. 24–26, 14195 Berlin, Germany
andreas.grillenberger@fu-berlin.de

Abstract. Physical computing is an appealing topic for CS education from primary school onward. With suitable tools, children and adolescents actively design and create their own interactive objects as tangible products of learning using methods and ideas of embedded systems. In this paper, we present a concept that uses a physical computing project as a framework to give students basic insight into connecting physical objects using embedded systems in time-limited contexts. The design and iterative development of a *Smart City* results in a product that contributes to the motivation of the students and can enthuse them for computer science.

Keywords: Physical Computing · Embedded Systems · CS Teaching · ProjectBased Learning · Smart City · Arduino · Snap!

1 Motivation and Background

Digital change is characterized, among other things, by the fact that many everyday objects perceive their environment with sensors and react to changes, exchange data via the Internet and communicate with each other, so that the physical and virtual worlds are blended. Accordingly, *embedded systems* have been a very important research and development area of CS for some time, resulting in many innovative products and applications. As a result, we are frequently confronted with CS phenomena in various contexts of our everyday lives including home automation, traffic and navigation or transportation and delivery. Consequently, everyone needs at least a basic understanding of and confidence in dealing with the complexity of these modern technologies. Computer science education therefore is faced with the challenge of adequately reflecting the relevant aspects in teaching and developing appropriate competencies in this area. Physical computing has proved its worth as an attractive approach to this topic: the design and realization of interactive, physical objects allows learners to develop concrete, tangible products of the real world, which arise from their imagination. This can be used in computer science education to provide learners with interesting and motivating access to various aspects of embedded systems design in constructionist and creative learning environments.

In this practical report, a concept is presented that gives heterogeneous groups of students from different backgrounds a motivating insight into a selection of central concepts of computer science in highly time-limited projects. In particular, the topic *networked embedded systems* is focused to teach learners how to network everyday objects with the help of embedded systems. In the process, various technical and organizational challenges are met, such as enabling wireless communication between different interactive objects or carrying out the project within limited time-frames. In the following, the relevance of the subject area for computer science education is briefly outlined and the technical background is discussed. Subsequently, the methodical approach is presented and reasoned based on existing requirements. Then content-related project goals, the pedagogy and structure of the projects as well as materials and tools are explained. Finally, impressions and experiences from different implementations are reflected and the respective challenges and solution approaches are discussed.

2 Embedded Systems in Computer Science Teaching

2.1 Representation in Curricula

Embedded systems increasingly gain importance in computer science teaching. The K–12 Computer Science Standards [5], for instance, recommend to let learners control robots, have them design, develop and discuss embedded systems, and to debate the effects of ubiquitous and pervasive computer-controlled devices of their everyday life. In the English national curriculum [6], already primary school students are supposed to design, analyze and correct programs according to specific objectives, including control or simulation of physical systems. The accompanying teacher's guide recommends the use of physical systems so that learners can work with sensors, lights and motors rather than pure simulations.

Also in Germany, computer science education at school level has dealt with this topic for more than 30 years: articles about integrating topics such as measurement, control and regulation, data processing and automation in school teaching have regularly been published [3]. In a more recent special issue on embedded systems of the German CS education journal LOG IN [7], various teaching examples, approaches and tools are presented that take up this trend. Different aspects of the broad topic embedded systems are also reflected in the recommendations for educational standards of CS by the German Informatics Society (GI) [1,2][23], which are used in many federal states as a basis when revising curricula. Typical application areas that are listed include robotics, process regulation and process control [2][24].

2.2 Central Content for CS Teaching

For the purpose of identifying content aspects in the subject area that are relevant for computer science teaching, the first author of this paper worked out common characteristics of embedded systems in various fields and application areas in a structured analysis of central technical literature. The identified commonalities are also relevant in physical computing, thus, the resulting extensive list was analyzed with regard to central aspects, i.e. such technologies, practices and principles, which are described as characteristic features of the relevant area in all disciplines. This means that they are recurrent and meet Schwill's horizontal criterion (see [16]) and are necessary requirements for developing competencies in the respective domains. These were then reduced to their essential properties by summarizing and abstracting subordinate terms and concepts[25].

The resulting technologies, practices and principles can be divided into three dimensions: Central topics, content areas and procedures of the subject area include *structure and properties* of embedded systems, which contribute to characterize and identify different systems and typical problems associated with their development, e.g. interfaces, peripheral devices and possibilities of data acquisition. Comprehensive *objectives, requirements and challenges* help to discuss typical questions and approaches in the design of embedded systems, e.g. system quality and real-time requirements. Proposed *practices* combine technical and educational considerations and offer strategies to prepare project work with embedded systems in a suitable way and in accordance with common procedures in the analyzed domains, e.g. tinkering and prototyping (see [12, p. 41ff.]).

3 Physical Computing in Project Lessons

In order to use a typical working method of computer science practice, a projectbased approach was chosen. Projects, according to Kilpatrick, "emphasize the factor of action, preferably wholehearted vigorous activity" and involve a "purposeful act carried on amid social surroundings" [11]. Project based learning has since become a popular method of teaching, especially in CS education [8,14]. Schubert and Schwill [15] describe the course of projects in computer science teaching as a structured sequence of steps closely related to the software life cycle with regular fixed points linking pedagogy and computer science. However, such a rather strict procedure is considered difficult to implement in class: On the one hand, project goals and procedures are often not compatible, since projects in school education are usually rather clear, so that there is neither the necessity of detailed planning nor the sense of elaborate documentation [9]. At the same time the learners are often overwhelmed with the complexity of larger projects, which frequently leads to unfinished or bad products [10]. As a remedy Kastl and Romeike [10] suggest to enrich project-based learning with agile methods of software development. Accordingly, agile projects start with a collection of ideas and initial planning, from which user stories emerge that form self-contained modules of the overall project. The actual project work is organized in cyclic iterations, within which all phases of the project work are run through (planning, design, implementation, discussion, testing, reflection) and which result in prototypes or at the end in the finished product. One promising aspect of integrating

[23] An English summary of the educational standards for lower secondary education can be found in [4].

[24] A more detailed description of the occurrence of embedded systems and robotics in CS education research and school teaching can be found in [12, p. 16ff.].

[25] The method and outcomes are described more detailed in [12, p. 40f.]

agile methods in project-based learning is the possibility to flexibly select suitable methods according to the respective project needs.

This procedure seems appropriate for physical computing teaching for several reasons: Through the iterative development of prototypes, the projects are given structure and the phases of planning and reflection, which are often neglected in teaching such projects (see [13]), are given a higher priority: Instead of only drafting a rough plan at the beginning of the project, which no longer receives attention over time, it is adapted to changing conditions. In addition, agile methods can be ideally integrated into physical computing activities: For project planning, the user view is usually taken first, which can be easily realized by creating user stories. During the creation of concrete tasks for programming and during implementation, the developer's perspective is then used to describe concrete program processes and identify suitable data sources and components.

4 Considerations for the Intended Project Implementations

In the planned project, the participants were to be involved in practical activities by networking objects and enriching them with sensors and actuators. This way, the basics of sensor-actuator control and networking objects with embedded systems should be addressed in a motivating manner. The following challenges had to be considered: A *strong heterogeneity of student groups* had to be assumed, since the project was to be implemented in contexts in which groups of different classes, grades and schools were formed. Therefore, a framework had to be found that appeals to all participants equally and allows them to take on adequate tasks depending on their performance level. The heterogeneity of the groups also meant that *no common prior knowledge of the content or methods* could be assumed. Required knowledge therefore had to be acquired within the frame of the projects. Furthermore, all implementations were subject to *different time restrictions*, so that the concept had to be flexibly adaptable. In order to meet these challenges, different elements of agile methods were used, e.g. the use of a project board with user stories, tasks and process representation, the iterative production of prototypes or pair programming. In order to maintain the character of physical computing and take up its basic contents and practices, particular emphasis was placed on the following design principles[26]: Integration of tinkering activities into dedicated learning phases (P1), creation of interactive objects (P2), development of working prototypes (P3), provision of an interesting theme to trigger imagination and creativity (P4), integration of methods of creative learning (P5), integration of technical aspects with art/crafting (P6), structuring work processes (P7), selecting tools (P8) and materials (P9) suitable for the target group and the intended projects, collaborative work on a joint exhibition (P10) and presentation of the final products (P11).

4.1 Project Theme

In order to provide an interesting theme (P4) for the project, the context "Smart City" was chosen, which offers clear links to the students' world of experience, triggers creative ideas and solutions through its openness (P4/P5) and brings with it phenomena that are typical for working with embedded systems. The aim of the project is to create an interactive model city in which embedded systems capture their environment in different areas (e.g. brightness, traffic) and control objects (e.g. activate lighting, control traffic lights). For this purpose, the participants should either enrich prefabricated objects with sensors and actuators or design their own interactive objects (P2) and network them with each other. Networking is associated with the challenge that different sub-projects have to communicate with each other, so that the planned project goes beyond typical physical computing projects. Furthermore, there is the difficulty, but also the chance, that several groups of students work on the same object and with the same microcontroller at the same time, but individual groups can also be involved in several sub-projects. A clear assignment between object and student group therefore no longer exists. Both challenges can be solved technically, as explained in section 4.3.

4.2 Project Structure

The project is divided into separate stages (P7) and can be flexibly adapted to different conditions. In the *introduction and motivation* (5 min) the participants are familiarized with the setting and get an overview of the available tools and planned processes. In a *learning phase* (P1), a detailed introduction to physical computing, the tools and components as well as the corresponding program elements is given in market place activities (variant A, approx. 1–2 blocks of 90 min, see also section 4.4). In shorter sessions, a tutorial provides insight into

[26] The derivation of these principles is described in [12, p. 135f.]

the necessary basics instead (variant B, approx. 10 min). All teams have manuals available, which they can use as reference books or to acquire knowledge as required. In a first *planning phase* (15 min) ideas are collected in brainstorming (P5), the rough layout of the city is planned and sketched on a model board and tasks are identified and prioritized. The "letters to the mayor" described below can be used to make suggestions. In the following phases of *project work and reflection* (90 min to several project days) groups of two work on the tasks. At regular intervals they present their working prototypes (P3) to the "mayor" of the city, reflect on their progress and define the next work steps. The mayor has the opportunity to bring in wishes and priorities. This role could be taken by a student or group of students as city council. In the projects described in this paper, however, it was assigned to the teachers or supervisors so that they could influence the project from a pedagogical point of view without the pupils perceiving this as a teacher's instruction. After several iterations the students present their "Smart City" in an *exhibition*, explain their inventions and discuss them with visitors (P10/P11, 15 min up to several hours, e.g. open-door day).

4.3 Tools: Hardware Decision and Programming Environment

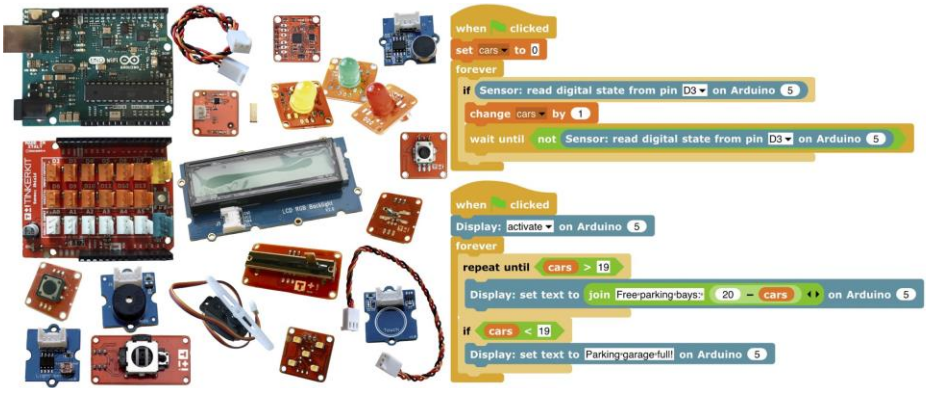
In order to meet the challenges outlined above, it was necessary to require as little prior knowledge as possible, both in working with embedded systems and in programming. For the selection of tools this meant that they should reduce entrance barriers as far as possible so that students can quickly get started intuitively. At the same time they should also be flexibly usable so that the projects were imposed with as few restrictions as possible and even complex ideas remained feasible (P8).

Hardware Concerning hardware, we used a combination of a microcontroller with a modular system: Instead of wiring sensors and actuators with breadboards in a complex and error-prone manner, such systems use conventional connectors, so that no knowledge of the electronics is required that goes beyond the distinction between components that are controlled digitally or analogously. In order to offer a large variety of sensors and actuators, a combination of the widely used and versatile platforms Seeed Grove and Arduino Tinkerkit is used (fig. 1). To increase compatibility, the chosen microcontroller boards were based on the widespread Arduino platform and required WLAN capability for networking. For this purpose, two possible boards were tested and used: Arduino Uno Wifi and the compatible Wemos D1. To control the microcontrollers wirelessly, a firmware had to be developed and the programming environment extended accordingly27. In order to make it as independent as possible from the programming environment used in the project, all communication among and with the systems is handled with an interface which is based on the REST principle. This can be used with all tools that implement the HTTP protocol. We decided against a persistent storage of the programs on the microcontrollers, since at present, available solutions permit only either live configuration or persistent programming28 and it was considered more important that the impact of program changes was immediately visible to the learners. To support the use of all sensors and actuators available for the projects, relevant options were made accessible via the REST-based interface and implemented for both microcontroller platforms.

Programming Environment Because of the expected heterogeneity of our learner groups, we aimed at a programming environment that allowed both experienced and inexperienced learners with regard to programming to work together and thus to enable both simple and technically demanding projects. We decided for the block-based programming environment Snap!, which is flexibly expandable and by default supports HTTP access and thus allows to use REST interfaces. For this purpose, we developed an extension for controlling the microcontrollers via WLAN. The blocks implemented for this purpose were designed in such a way that they build purely on blocks existing natively in Snap! and thus can be used in any Snap! derivatives such as Snap4Arduino or IoT-Snap (fig. 1). In the long run, this will make it possible to switch flexibly between WLAN and USB when controlling the microcontrollers and thus, it will allow to take advantage of both the persistent storage of programs and the live configuration of microcontrollers.

[27] To control microcontrollers via USB, often the standardized Firmata protocol is used, which is based on serial communication, but can not be used for WLAN communication. Thus a suitable firmware was developed, which can be downloaded from Github together with the programming environment and learning materials:
https://github.com/maprzybylla/LEGO-Smart-City.

[28] A corresponding approach is in development, but was not yet mature enough for a productive use in schools (see http://microblocks.fun) at the time of the implementation of our projects.



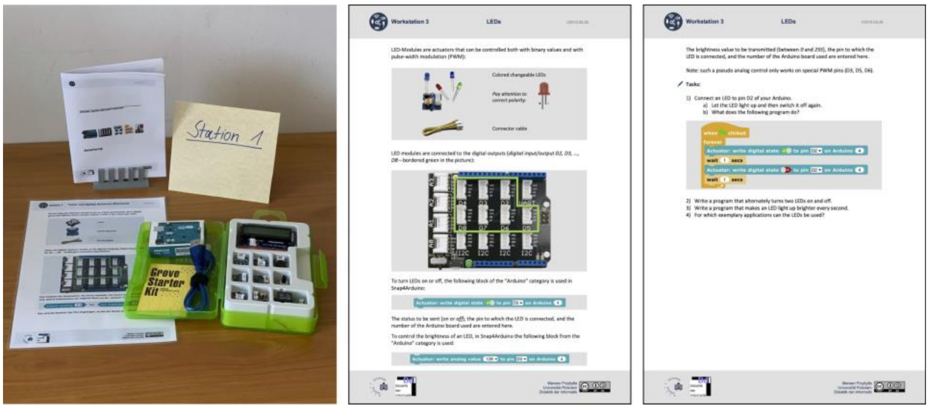
```
when ⚑ clicked
set cars ▾ to 0
forever
  if Sensor: read digital state from pin D3 ▾ on Arduino 5
    change cars ▾ by 1
    wait until not Sensor: read digital state from pin D3 ▾ on Arduino 5

when ⚑ clicked
Display: activate ▾ on Arduino 5
forever
  repeat until cars > 19
    Display: set text to join Free·parking·bays:· 20 − cars ◂▸ on Arduino 5
  if cars < 19
    Display: set text to Parking·garage·full! on Arduino 5
```

Fig. 1. Arduino Uno Wifi with Tinkerkit and Grove components and IoT-Snap scripts.

4.4 Work Materials: LEGO Bricks, Craft Materials, Manuals, Market Stalls, Letters

For the construction of the Smart City, *model building boards* are laid out with writable foil on which the students can plan and draw the layout of their city. *LEGO bricks* are used to construct the buildings, which is advantageous because the teenagers can handle them easily, are creative and need only a few further aids. A disadvantage is the limited mechanical compatibility of the LEGO bricks with the TinkerKit and Grove components. For the integration of sensors and actuators into the projects, therefore, e.g. *adhesive tape*, *hot glue*, *wire* and *cord* are used, which so far proved to be generally practicable (P6/P9). All project groups had access to comprehensive *manuals* and *cheat sheets* as references describing the interaction of hardware and programming environment. For some groups, additional *market stalls* were set up (fig. 2) for market place activities, in which they independently worked out the relevant contents on the basis of various tasks. The market stalls were designed in such a way that they can be solved by students with little prior knowledge, but at the same time are not trivial. For differentiation there are *extra tasks*, which deepen selected contents and skills. All market stalls have the same structure and the corresponding *worksheets* contain a list of the required components, assembly instructions and finally several tasks with increasing complexity. The learners receive *route cards* on which they document their progress and on the basis of which the advancement is also visible for the teacher. In order to stimulate ideas, a scenario is presented to the students in which the citizens of the city are called upon to participate with suggestions and project proposals in the further development of their city towards a "Smart City". Some citizens have already used this opportunity and sent *letters with suggestions* to the mayor, e.g. noise level measurement at the goods station to keep the night's rest, an alarm system to guard the city treasure or an automated greenhouse to increase the vegetable harvest. These letters are presented to the learners and they are asked to brainstorm additional ideas.



Workstation 3 LEDs

LED-Modules are actuators that can be controlled both with binary values and with pulse-width modulation (PWM):

Colored changeable LEDs

Pay attention to correct polarity

Connector cable

LED modules are connected to the digital outputs (*digital input/output D2, D3, … D8*—bordered green in the picture):

To turn LEDs on or off, the following block of the "Arduino" category is used in Snap4Arduino:

```
Actuator: write digital state ● to pin D2 ▾ on Arduino 4
```

The status to be sent (*on* or *off*), the pin to which the LED is connected, and the number of the Arduino board used are entered here.

To control the brightness of an LED, in Snap4Arduino the following block from the "Arduino" category is used:

```
Actuator: write analog value 128 ▾ to pin D3 ▾ on Arduino 4
```

Universität Potsdam
Didaktik der Informatik

Workstation 3 LEDs

The brightness value to be transmitted (between *0* and *255*), the pin to which the LED is connected, and the number of the Arduino board used are entered here.

Note: such a pseudo analog control only works on special PWM pins (D3, D5, D6).

Tasks:

1) Connect an LED to pin D2 of your Arduino.
   a) Let the LED light up and then switch it off again.
   b) What does the following program do?

```
when ⚑ clicked
forever
  Actuator: write digital state ● to pin D2 ▾ on Arduino 4
  wait 1 secs
  Actuator: write digital state ● to pin D2 ▾ on Arduino 4
  wait 1 secs
```

2) Write a program that alternately turns two LEDs on and off.
3) Write a program that makes an LED light up brighter every second.
4) For which exemplary applications can the LEDs be used?

Fig.2. Market stall with Grove Starter Kit, manual and work sheets.

5 Experience

The project has so far been carried out with 14 groups (approx. 215 students aged 10 to 18). It was tested in several different variants, each with distinct framework conditions, which are explained in the sections below and evaluated with regard to the following questions:

1. How do the methods, tools and materials help to achieve the project goals?
2. Which problems arise depending on the different conditions and how canthey be solved?

5.1 Group Compositions and Framework Conditions

Half-day projects with heterogeneous groups At the University of Erlangen-Nuremberg the concept was used to convey a positive image of computer science to school students and to enthuse them for this subject in four-hour projects. The participants came from different schools and classes and were very heterogeneous in their age structure. In addition to the short technical introduction to physical computing (variant B), the students also received a detailed introduction to agile practices in projects. A project board was used to visualize user stories and tasks in the process, stand-up meetings were called regularly and the status of the prototypes was presented to the mayors.

Project week in a school with a heterogeneous group A further implementation took place at a grammar school during a project week before the school festival. Also here the goal was to inspire students to take computer science courses. At the same time the elective subject should be present at the school festival with the developed “Smart City”. This project was designed to fill two school days with six lessons each, so that much more time was available for the development of the model city. Here, too, the project participants formed a very heterogeneous group (students of grades five to eleven). In this implementation, variant A (market place activities) was implemented and the project board as well as individual stand-up meetings were used.

Half-day projects with homogeneous groups As a third variant, the concept is regularly used at the University of Potsdam in three- to five-hour projects. The challenge is that parallel activities take place in which the students participate in smaller groups, so that not the whole project team is present continuously. For time reasons, variant B of the learning phase (tutorial) is preferred in this scenario. The use of agile methods is largely dispensed with, these are reflected above all in the structure of the implementation: User stories manifest in the aforementioned letters to the mayor and stand-up meetings are present in the form of discussions with the mayor.

5.2 Results and Observations

In all implementations similar positive experiences were gained: By using agile methods either explicitly or in the structure of the projects, a strong differentiation between the student teams working in pair programming took place automatically. Thus, in general little frustration arose, since the groups independently chose adequate tasks that were neither too difficult nor too simple. At the same time, most of the participants seemed very eager to learn, as the challenges were usually self-invented so that intrinsic motivation arose to achieve the project goals that had been set. This was most noticeable in the group who presented the results to the public at the school festival, probably because this was the group with the longest time-span available and thus with the most advanced and detailed results. Groups who received more detailed introductions and had longer learning phases beforehand, achieved more sophisticated results, however, it must be noted that they also had more time available for the project phase. In the other groups, it was possible for the students to start working immediately, but they mostly worked only on their sub-project and focused less on networking different objects. Impressions from the projects are shown in fig. 3.

The choice of hardware and software seemed to contribute to the success of the projects. Due to the use of power banks, the microcontrollers were not bound to one location, but could be placed anywhere in the “Smart City”. However, some problems occurred in groups where only little time was available to master technical hurdles. In projects that used Arduino Uno Wifi, large latencies occurred during WLAN communication, thus it is recommended to use the faster and more powerful Wemos D1 microcontrollers (or comparable devices) when possible. The extensive amount of sensors and actuators gave rise to a large variety of ideas. The students were able to implement their projects using the Snap! programming environment regardless of their age groups.

Difficulties sometimes occurred when students tried to install sensors and actuators in their LEGO brick objects, so we provided them with hot glue guns in cases where no less permanent attachment was possible.

Students had to be very careful not to damage electronic components by the heat. If used on smooth surfaces, the adhesive could later be removed without leaving any residue.

There was a great diversity in the overall projects, ranging from rather small cities that were planned in detail and lovingly designed to large and less detailoriented cities. However, the individual projects that were put together in the cities were surprisingly similar in nature, even in groups without the letters to the mayor: for example, there were sensor-controlled lighting elements, traffic lights and information displays in every city. Despite the strong self-regulation during the projects, it was therefore automatically necessary for all learners to deal with suitable data acquisition, the control of various actuators and conditional statements, among other things, so that the project proved to be well suited for the targeted teaching of basics of sensor-actuator control and networking objects in the subject area embedded systems. The project was also suitable to introduce the most inexperienced students to basic programming concepts without overwhelming them.

Although agile methods such as stand-up meetings and time estimation were introduced and a project board was made available in most groups, these were used only sparsely by the students. Stand-up meetings did not always take place regularly, but only when really needed, for example when it was necessary to clarify how the city should be organized because sub-projects ran the risk of blocking each other. As the size of the projects increased, however, such methods seemed to become more relevant and thus they were usually observable in the longer projects.

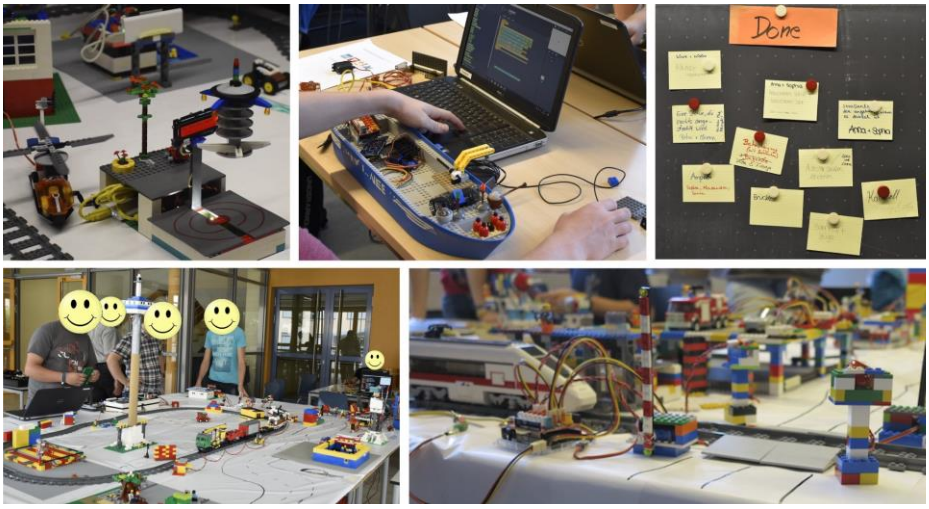


Fig. 3. Impressions from the projects (top left to bottom right): a spaceship landing platform, students programming a cargo ship with loading crane, “done” section of a project board, smart city with television tower with rotating platform and automated railway crossing.

6 Conclusion and Perspective

All in all, the “Smart City” project has proved successful in all its implementations. The chosen context motivated the students and aroused their interest in microcontrollers and their significance in reality, but also allowed a lot of room for creativity in the project implementations. The hardware and software were easy to use for the students, the only problem being latencies caused by the control of the microcontrollers via WLAN, which had to be taken into account in the projects and can be avoided with appropriate hardware decisions. The orientation towards agile methods was useful to structure the project work, especially through the iterative and prototype-oriented development of the sub-projects.

In order to solve some minor existing problems, initial ideas are available, such as the design of 3D-printed adapters, which make it easier for students to connect components with LEGO bricks. In general, the concept proved to be well suited to give learners with different abilities a motivating insight into microcontroller programming, the interaction of sensors and actuators and the networking of embedded systems. Depending on the concrete objectives, the idea can be adapted technically, methodically and in terms of content to the respective learning group. For this, only little effort is needed, which makes the concept very flexible. In addition, the project offers good starting points for subsequent lessons. For example, app development for smart home control with mobile devices could be just as well integrated as discussions on the potentials and dangers of increasing digitalization.

References

1. Arbeitskreis Bildungsstandards: Grundsa¨tze und Standards fu¨r die Informatik inder Schule. Bildungsstandards Informatik fu¨r die Sekundarstufe I. [Principles and Standards for Computer Science in Schools. Educational Standards for Computer Science in Lower Secondary Education]. Supplement to LOG IN 150/151 (2008)
2. Arbeitskreis Bildungsstandards SII: Bildungsstandards Informatik fu¨r die Sekundarstufe II [Educational Standards for Computer Science in Lower Secondary Education]. Supplement to LOG IN 183/184 (2016)
3. Baumann, R.: Eingebettete Systeme verstehen. Teil 1: Kreatives Experimentierenmit Arduino [Understanding Embedded Systems. Part 1: Creative Experimentation with Arduino] **32**(171), 33–45 (2012)
4. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS: Educational Standards for Computer Science in Lower Secondary Education. SIGCSE Bull. **41**(3), 288–292 (2009)
5. Computer Science Teachers Association: CSTA K-12 Computer Science Standards,Revised 2017. http://www.csteachers.org/standards (2017)
6. Department for Education: Computing programmes of study: key stages 1 and 2. National curriculum in England (2013) Fachbereich Erziehungswissenschaft und Psychologie der Freien Universita¨t Berlin(ed.): Eingebettete Systeme. No. 185/186 in LOG IN – Informatische Bildung und Computer in der Schule, LOG IN Verlag GmbH, Berlin (2016)
7. Fincher, S., Petre, M.: Project-based Learning Practices in Computer Science Education. In: FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. vol. 3, pp. 1185–1191. IEEE (1998)
8. Hartmann, W., Na¨f, M., Reichert, R.: Informatikunterricht planen und durchfu¨hren[Planning and conducting computer science lessons]. eXamen.press, SpringerVerlag, Berlin Heidelberg (2007)
9. Kastl, P., Romeike, R.: "Now They Just Start Working, and Organize Themselves"First Results of Introducing Agile Practices in Lessons. In: Proceedings of the 15th Workshop in Primary and Secondary Computing Education. pp. 25–28. ACM, New York, NY, USA (2015)
10. Kilpatrick, W.H.: The project method: the use of the purposeful act in the educative process. Teachers College, Columbia University (1929)
11. Przybylla, M.: From Embedded Systems to Physical Computing: Challenges ofthe "Digital World" in Secondary Computer Science Education. Doctoral thesis, Universita¨t Potsdam (2018)
12. Przybylla, M., Romeike, R.: The Nature of Physical Computing in Schools. In: Proceedings of the 17th Koli Calling International Conference on Computing Education Research. pp. 98–107. ACM (2017)
13. Romeike, R., Go¨ttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. pp. 48–57. ACM, New York, NY, USA (2012)
14. Schubert, S., Schwill, A.: Didaktik der Informatik [Didactics of Computer Science]. Spektrum Akademischer Verlag, Heidelberg, 2 edn. (2011)
15. Schwill, A.: Computer Science Education Based on Fundamental Ideas. In: Proceedings of the IFIP TC3 WG3.1/3.5 Joint Working Conference on Information Technology: Supporting Change Through Teacher Education. pp. 285–291. IFIP, Chapman & Hall, Ltd., London, UK (1997)

Understanding Artificial Intelligence – A Project for the Development of Comprehensive Teaching Material

Michael Schlichtig, Simone Opel, Lea Budde, and Carsten Schulte

Paderborn University, https://cs.uni-paderborn.de/ddi
{michael.schlichtig,simone.opel,lea.budde,carsten.schulte}@uni-paderborn.de

Abstract. Artificial intelligence (AI) has the potential for far-reaching – in our opinion – irreversible changes. They range from effects on the individual and society to new societal and social issues. The question arises as to how students can learn the basic functioning of AI systems, what areas of life and society are affected by these and – most important – how their own lives are affected by these changes. Therefore, we are developing and evaluating school materials for the German ”Science Year AI”. It can be used for students of all school types from the seventh grade upwards and will be distributed to about 2000 schools in autumn with the support of the Federal Ministry of Education and Research. The material deals with the following aspects of AI: Discussing everyday experiences with AI, how does machine learning work, historical development of AI concepts, difference between man and machine, future distribution of roles between man and machine, in which AI world do we want to live and how much AI would we like to have in our lives. Through an accompanying evaluation, high quality of the technical content and didactic preparation is achieved in order to guarantee the long-term applicability in the teaching context in the different age groups and school types. In this paper, we describe the current state of the material development, the challenges arising, and the results of tests with different classes to date. We also present first ideas for evaluating the results..

Keywords: Artificial Intelligence · Machine Learning · Teaching Material · Societal Aspects · Ethics · Social Aspects · Science Year · Simulation Game.

1 Introduction

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) have become part of our everyday lives without many of us noticing it. Consequent changes are extensive and may be irreversible, including strong societal effects. For instance, the advances in creating so-called deep fake videos using artificial neural networks can impact the credibility of videos as evidence and in the news by using them, e.g. for discrediting politicians or further misuse (cf. Chesney and Citron [6]). Hence, it is vital to be aware of such technological advances for responsibly taking part in society. Nevertheless, according to a recent Bitkom study [4], most Germans now know the term AI, but cannot properly explain it. However, neither computer science nor AI are anchored in school teaching yet and many teachers – also from computer science – only have rudimentary knowledge about AI, so these teachers are not prepared to discuss it competently with their students. Nonetheless, it seems essential to us that students acquire a comprehensive understanding of how AI systems work – topics like AI and ML should be part of gaining digital literacy for all students.

Albeit, computer science education is not always part of the curricula in Germany and is sometimes integrated into subjects. Even if we see the integration of computer science education otherwise rather negatively – it offers the opportunity to include teachers from ethics, social studies or politics in the introduction of AI, and thus the opportunity to bring the topic to schools in a very broad and differentiated way.

For this reason, we decided to develop teaching materials for teachers also from non-computer science to introduce a *basic understanding of Artificial Intelligence and Machine Learning* to secondary school students.

Teachers for ethics, philosophy or social science have a different perspective on the issues arising from the use of AI in our society, but non-computer science teachers lack a deeper understanding of computer science and the technology of AI. For this reason, all teaching materials about AI and ML for teachers also from non-computer science has to be developed in a way that they also gain competences about AI and ML without needing to understand how to program and implement ML systems.

To meet the challenges of also addressing teachers from other subjects, and at the same time reaching many teachers, we established cooperation with the nationwide campaign of the ”Science Year”, which is organised and funded by the Federal Ministry of Education and Research as well as by various foundations[29].

2 AI for all Students - Youth Campaign of the Science Year

As mentioned before, our materials will be distributed as part of the ”Science Year”. For this reason, we shortly present this annual campaign, our concept for the teaching material and our development process.

2.1 Conceptual Challenges

The Science Year deals with one topic each year – the current year is dedicated to the topic of artificial intelligence and includes very different initiatives, actions and materials around this topic. Former topics have been, e.g. energy research, sustainability, marine research or research on digital societies. We are currently implementing one of the annual initiatives, which is aimed at all students in the age from 12 up to 18 years (secondary schools) of all school types.

Therefore, the materials developed have to deal with considerable heterogeneity in terms of age and skills of the target group. Hence, we have to balance the levels of language, content – depth but also by considering prior knowledge – and learning tasks between preteens attending middle schools and almost adult students of grammar schools who also have computer science lessons.

A second challenge consists of the heterogeneity of the teachers. We expect that at least half of the teachers who use the resulting materials will not be teachers for computer science.

2.2 Development Approach

All these challenges caused the decision that we should analyse in particular our existing draft curriculum on data science, which has been developed in our ”Project Data Science and Big Data at School (ProDaBi)[30] [12]. Since we are already working on this curriculum for the use at different school levels and also take social aspects into account, we decided to use it as a basis. After that, we further reviewed existing curricula and frameworks [1] [15] [11] to match our issues with them.

However, we came to the conclusion that not all found topics could be used for the planned teaching material and removed all lectures which include programming or need in-depth technical knowledge – instead, we focus on social and ethical aspects.

Furthermore, we spoke informally with several non-computer science teachers to survey what they know about artificial intelligence and machine learning. Most of the teachers read some articles in newspapers and magazines or watched films about artificial intelligence, but all said that they were far from comprehensive technological knowledge.

For this reason, we explored a meaningful way to combine explaining technical knowledge to teachers with designing teaching materials. As a result, we attempt to create materials which integrate content knowledge with pedagogy. Therefore, teachers can also gain ”pedagogical content knowledge” (cf. Shulman [16]) in these topics.

In further response to all the challenges described above (see chapter 2.1), we decided to follow a ”design-based research” approach [8] and therefore developed the material in several cycles. Currently, we have completed the last evaluation cycles of the various worksheets and activities with several courses from different schools (7th up to 11th grade, age from 12 to 18 years, middle and grammar schools) and are in the process of layouting and producing the materials.

2.3 Implemented Teaching Materials

In contrast to other projects, this initiative requires close coordination with the organisers of the Science Year. Consequently, we discussed together, which types of material should be developed. We proposed the content areas which have been approved by the Science Year.

[29] https://www.wissenschaftsjahr.de/2019/english; last access: 2019-09-04

[30] Website: https://www.prodabi.de

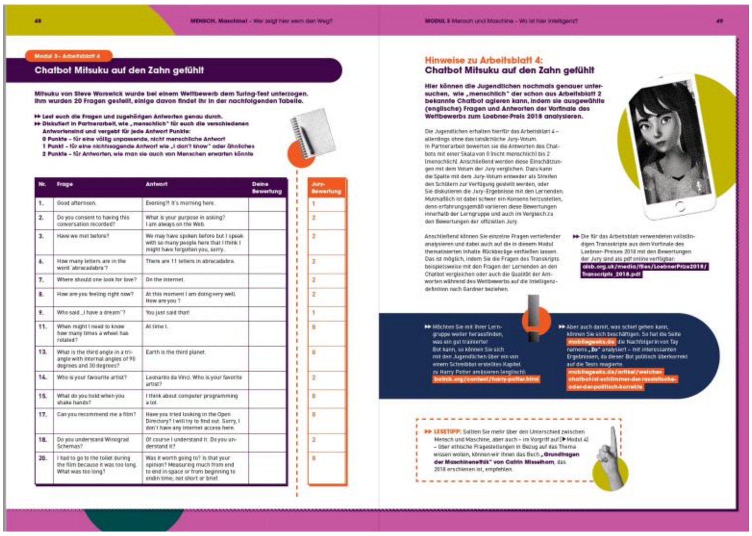
- *The core element is a simulation game to explain and to reflect how machine learning works.* We developed the simulation game based on ”hexapawn” by Martin Gardner [10], which has been enhanced several times, among others as ”Sweet Learning Computer” by the project *CS4FN* [9]. An extended description of this game has been presented on WiPSCE ’19 [14] as a poster.
- *Every Science Year a booklet for teachers is published.* This booklet (about 85 pages) contains explanatory texts and background information as well as pedagogical tips, which are always accompanied by worksheets and activities for the students. Concerning the explanatory texts, we focused on the explanations of the underlying concepts of AI as well as fundamental ideas of the further topics, for supporting the non-computer science teachers in particular. Additionally, the booklet contains suggestions to arrange lessons with the worksheets and outlooks on how to continue after each unit.
- *We also decided to publish a learning diary for students.* This diary can be used by the students themselves, but can also be part of the lessons. It consists of about 24 pages and should motivate the students to deal with the material and further information on their own to deepen their understanding of AI. They can do several quizzes, and we included links to additional videos, texts on how artificial neural networks work or a construction manual to build an original ”MENACE”[31].

Discussions regarding which content areas should be included, led to the joint decision to structure all materials in several *modules*, each dealing with a different set of issues. This approach simplifies, especially for teachers with a limited time budget, selecting the content that is important to them or to distribute topics between teachers of different subjects in interdisciplinary learning scenarios.

3 Developing the Materials and Modules

In the previous section, we tried to outline the different challenges and conditions that are contained in this project. In this section, we describe the contents and structure of the teaching materials, which has been designed to achieve the following general goals:

1. Explaining how Artificial Intelligence works
2. Opening of the social discourse on Artificial Intelligence
3. Reducing possible existing misconceptions



48 MENSCH. Maschine! – Wer zeigt hier wem den Weg?

Modul 3 – Arbeitsblatt 4

Chatbot Mitsuku auf den Zahn gefühlt

Mitsuku von Steve Worswick wurde bei einem Wettbewerb dem Turing-Test unterzogen. Ihm wurden 20 Fragen gestellt, einige davon findet ihr in der nachfolgenden Tabelle.

▸▸ Lest euch die Fragen und zugehörigen Antworten genau durch.
▸▸ Diskutiert in Partnerarbeit, wie „menschlich" für euch die verschiedenen Antwortensind und vergebt für jede Antwort Punkte:
0 Punkte – für eine völlig unpassende, nicht menschliche Antwort
1 Punkt – für eine nichtssagende Antwort wie „I don't know" oder ähnliches
2 Punkte – für Antworten, wie man sie auch von Menschen erwarten könnte

Nr.	Frage	Antwort	Deine Bewertung	Jury-Bewertung
1.	Good afternoon.	Evening?! It's morning here.		1
2.	Do you consent to having this conversation recorded?	What is your purpose in asking? I am always on the Web.		2
3.	Have we met before?	We may have spoken before but I speak with so many people here that I think I might have forgotten you, sorry.		2
6.	How many letters are in the word 'abracadabra'?	There are 11 letters in abracadabra.		2
7.	Where should one look for love?	On the internet.		2
8.	How are you feeling right now?	At this moment I am doing very well. How are you ?		2
9.	Who said „I have a dream"?	You just said that!		1
11.	When might I need to know how many times a wheel has rotated?	At time t.		0
13.	What is the third angle in a triangle with internal angles of 90 degrees and 30 degrees?	Earth is the third planet.		0
14.	Who is your favourite artist?	Leonardo da Vinci. Who is your favorite artist?		2
15.	What do you hold when you shake hands?	I think about computer programming a lot.		0
17.	Can you recommend me a film?	Have you tried looking in the Open Directory? I will try to find out. Sorry, I don't have any internet access here.		0
18.	Do you understand Winograd Schemas?	Of course I understand it. Do you understand it?		2
20.	I had to go to the toilet during the film because it was too long. What was too long?	Was it worth going to? Is that your opinion? Measuring much from end to end in space or from beginning to endin time; not short or brief.		0

MODUL 3 Mensch und Maschine – Wo ist hier Intelligenz? 49

Hinweise zu Arbeitsblatt 4:
Chatbot Mitsuku auf den Zahn gefühlt

Hier können die Jugendlichen nochmals genauer untersuchen, wie „menschlich" der schon aus Arbeitsblatt 2 bekannte Chatbot agieren kann, indem sie ausgewählte (englische) Fragen und Antworten der Vorfinale des Wettbewerbs zum Loebner-Preis 2018 analysieren.

Die Jugendlichen erhalten hierfür das Arbeitsblatt 4 – allerdings ohne das tatsächliche Jury-Votum. In Partnerarbeit bewerten sie die Antworten des Chatbots mit einer Skala von 0 (nicht menschlich) bis 2 (menschlich). Anschließend werden diese Einschätzungen mit dem Votum der Jury verglichen. Dazu kann die Spalte mit dem Jury-Votum entweder als Streifen den Schülern zur Verfügung gestellt werden, oder Sie diskutieren die Jury-Ergebnisse mit den Lernenden. Mutmaßlich ist dabei schwer ein Konsens herzustellen, denn erfahrungsgemäß variieren diese Bewertungen innerhalb der Lerngruppe und auch im Vergleich zu den Bewertungen der offiziellen Jury.

Anschließend können Sie einzelne Fragen vertiefender analysieren und dabei auch auf die in diesem Modul thematisierten Inhalte Rückbezüge einfließen lassen. Das ist möglich, indem Sie die Fragen des Transkripts beispielsweise mit den Fragen der Lernenden an den Chatbot vergleichen oder auch die Qualität der Antworten während des Wettbewerbs auf die Intelligenzdefinition nach Gardner beziehen.

▸▸ Die für das Arbeitsblatt verwendeten vollständigen Transkripte aus dem Vorfinale des Loebner-Preises 2018 mit den Bewertungen der Jury sind als pdf online verfügbar: aisb.org.uk/media/files/LoebnerPrize2018/Transcripts_2018.pdf

▸▸ Möchten Sie mit Ihrer Lerngruppe weiter herausfinden, was ein gut trainierter Bot kann, so können Sie sich mit den Jugendlichen über ein von einem Schreibbot erstelltes Kapitel zu Harry Potter amüsieren (englisch): botnik.org/content/harry-potter.html

▸▸ Aber auch damit, was schief gehen kann, können Sie sich beschäftigen. So hat die Seite mobilegeeks.de die Nachfolgerin von Tay namens „Zo" analysiert – mit interessanten Ergebnissen, da dieser Bot politisch überkorrekt auf die Tests reagierte. mobilegeeks.de/artikel/welcher-chatbot-ist-schlimmer-der-rassistische-oder-der-politisch-korrekte

▸▸ LESETIPP: Sollten Sie mehr über den Unterschied zwischen Mensch und Maschine, aber auch – im Vorgriff auf (▶ Modul 4) – über ethische Fragestellungen in Bezug auf das Thema wissen wollen, können wir Ihnen das Buch **„Grundfragen der Maschinenethik" von Catrin Misselhorn**, das 2018 erschienen ist, empfehlen.

Fig.1. Example of one of the final pages of the teacher booklet. The layout has been developed by the supporting designers of the Science Year.

As beforehand mentioned in chapter 1, in general, the leading questions for content development are: ”What areas of my life and society are affected by AI?” and in particular ”To what extent does all this affect **me**?”.

[31] The foundation for Gardner’s hexapawn: https://www.mscroggs.co.uk/blog/19

To meet these goals and answer these questions, we structured the topics covered to answer these questions step-by-step:

- Module 1: *Introduction - students' everyday experiences with AI* This module is intended to activate and systematise existing experience with AI. Also, it should help to raise students awareness of their existing attitudes.
- Module 2: *How does Machine Learning work? – The Simulation Game "Man, Machine!"* This module consists of the simulation game named "Man, Machine!" and helps the students to understand how ML and AI work. They simulate a learning machine and observe the learning process. The material also consists of worksheets in order to reflect the newly gained experiences. This game has been presented in detail before [14].
- Module 3: *What is the difference between man and machine?* Here the differences between man and machine are systematised, and the concept of intelligence, in particular, is examined more closely.
- Module 4: *Historical Overview of the Development of Artificial Intelligence* In this module, the various aspects from the other chapters will be linked by reflecting their historical context.
- Module 5: *The distribution of roles of man and machine - ethical and societal aspects* In this module, the distribution of roles between man and machine is examined more closely. This includes not only the role of the developers but also the role of the users – because users also contribute to the further development of AI systems through their data input or data traces.
- Module 6: *Workshop: In which AI world do we want to live?* The students should develop scenarios and ideas about what kind of AI world they want to live in and how they think the future handling of AI should be shaped.

Each module is designed for four up to eight school lessons. Since we know that these materials could cover an entire school year, we worked on several proposals on how to combine the modules. As shown in fig. 2, modules 1 and 2 (simulation game) can be used as a mini-course unit. We provide an additional teacher's booklet for the game online – it contains several variation possibilities and extensions to use the game more flexible. If a teacher has little time, these two modules are sufficient to get a first idea of how AI and ML work.

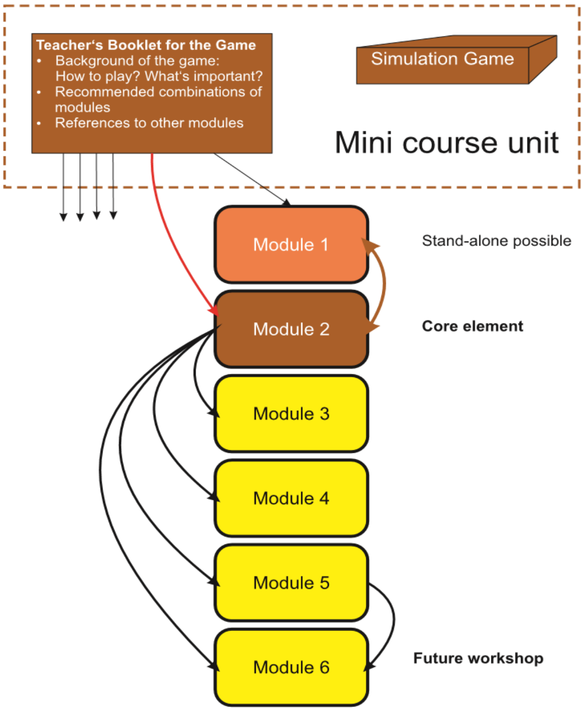

Teacher's Booklet for the Game
• Background of the game: How to play? What's important?
• Recommended combinations of modules
• References to other modules
Simulation Game
Mini course unit
Module 1
Stand-alone possible
Module 2
Core element
Module 3
Module 4
Module 5
Module 6
Future workshop

Fig. 2. Concept of module structure for all materials. The Core element is the simulation game in module 2, which can be used stand-alone with module 1.

The following modules are based on the game but are independent of each other, as we could test with classes in detail.

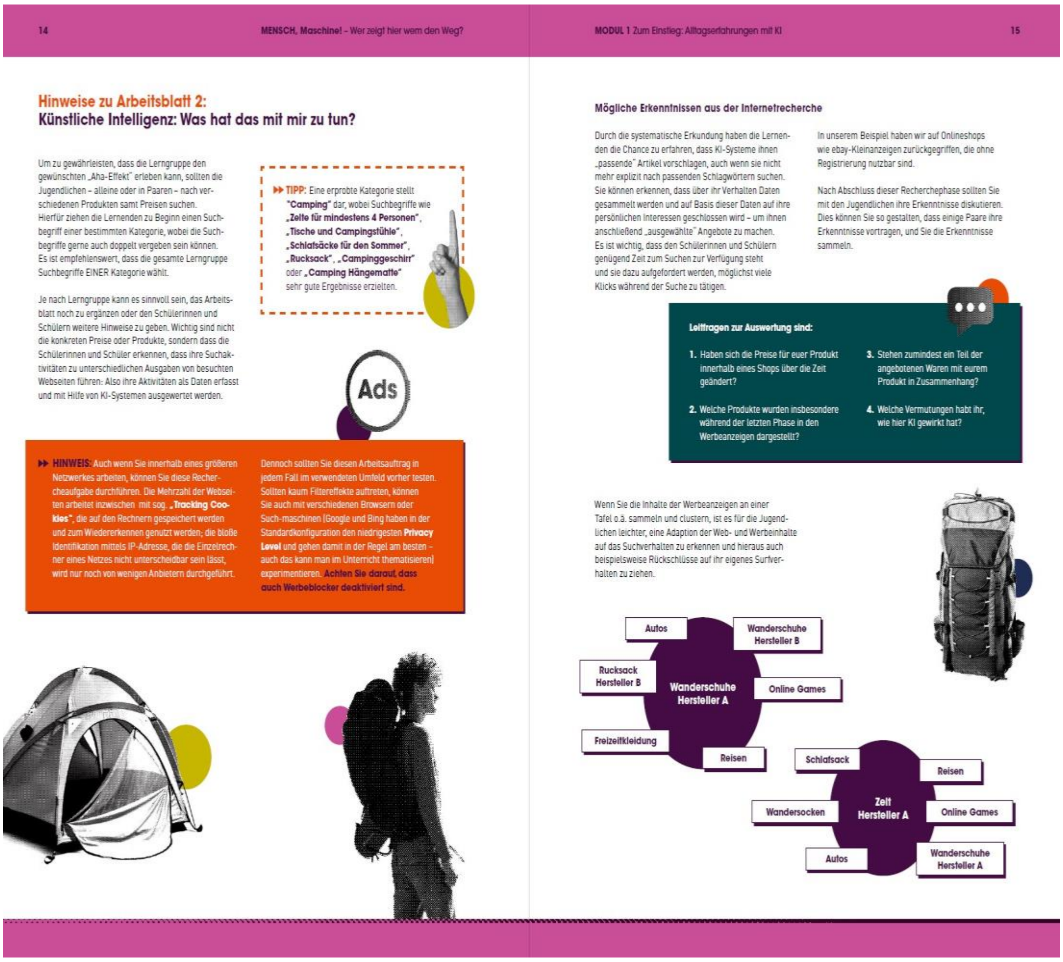
In order to present the goals and contents of each module even more apparent, we will now present the modules in more detail.

3.1 Module 1: Students' Everyday Experiences with AI

Artificial intelligence - what does it have to do with my life? And what is it actually? Questions that more and more people are asking themselves and want to answer for themselves.

To answer these questions, students first collect, reflect and systematise their prior experiences with artificial intelligence and their current ideas concerning AI without having to deal immediately with the technology behind it.

They then perform internet research and observe how the content displayed – in particular advertisements – changes and adapts to their behaviour (see fig.3). In this way, they can make initial assumptions about how AI could be included in these systems and what is essential for an AI: *Everything an AI can learn is based on the received amount of data, which must be structured appropriately*. This knowledge is vital and can also be used to formulate a first working definition of the terms AI and ML.



14 MENSCH, Maschine! – Wer zeigt hier wem den Weg?

Hinweise zu Arbeitsblatt 2:
Künstliche Intelligenz: Was hat das mit mir zu tun?

Um zu gewährleisten, dass die Lerngruppe den gewünschten „Aha-Effekt" erleben kann, sollten die Jugendlichen – alleine oder in Paaren – nach verschiedenen Produkten samt Preisen suchen. Hierfür ziehen die Lernenden zu Beginn einen Suchbegriff einer bestimmten Kategorie, wobei die Suchbegriffe gerne auch doppelt vergeben sein können. Es ist empfehlenswert, dass die gesamte Lerngruppe Suchbegriffe EINER Kategorie wählt.

Je nach Lerngruppe kann es sinnvoll sein, das Arbeitsblatt noch zu ergänzen oder den Schülerinnen und Schülern weitere Hinweise zu geben. Wichtig sind nicht die konkreten Preise oder Produkte, sondern dass die Schülerinnen und Schüler erkennen, dass ihre Suchaktivitäten zu unterschiedlichen Ausgaben von besuchten Webseiten führen: Also ihre Aktivitäten als Daten erfasst und mit Hilfe von KI-Systemen ausgewertet werden.

▸▸ TIPP: Eine erprobte Kategorie stellt **"Camping"** dar, wobei Suchbegriffe wie **„Zelte für mindestens 4 Personen"**, **„Tische und Campingstühle"**, **„Schlafsäcke für den Sommer"**, **„Rucksack"**, **„Campinggeschirr"** oder **„Camping Hängematte"** sehr gute Ergebnisse erzielten.

▸▸ **HINWEIS:** Auch wenn Sie innerhalb eines größeren Netzwerkes arbeiten, können Sie diese Rechercheaufgabe durchführen. Die Mehrzahl der Webseiten arbeitet inzwischen mit sog. **„Tracking Cookies"**, die auf den Rechnern gespeichert werden und zum Wiedererkennen genutzt werden; die bloße Identifikation mittels IP-Adresse, die die Einzelrechner eines Netzes nicht unterscheidbar sein lässt, wird nur noch von wenigen Anbietern durchgeführt. Dennoch sollten Sie diesen Arbeitsauftrag in jedem Fall im verwendeten Umfeld vorher testen. Sollten kaum Filtereffekte auftreten, können Sie auch mit verschiedenen Browsern oder Such-maschinen (Google und Bing haben in der Standardkonfiguration den niedrigesten **Privacy Level** und gehen damit in der Regel am besten – auch das kann man im Unterricht thematisieren) experimentieren. **Achten Sie darauf, dass auch Werbeblocker deaktiviert sind.**

MODUL 1 Zum Einstieg: Alltagserfahrungen mit KI 15

Mögliche Erkenntnissen aus der Internetrecherche

Durch die systematische Erkundung haben die Lernenden die Chance zu erfahren, dass KI-Systeme ihnen „passende" Artikel vorschlagen, auch wenn sie nicht mehr explizit nach passenden Schlagwörtern suchen. Sie können erkennen, dass über ihr Verhalten Daten gesammelt werden und auf Basis dieser Daten auf ihre persönlichen Interessen geschlossen wird – um ihnen anschließend „ausgewählte" Angebote zu machen. Es ist wichtig, dass den Schülerinnen und Schülern genügend Zeit zum Suchen zur Verfügung steht und sie dazu aufgefordert werden, möglichst viele Klicks während der Suche zu tätigen.

In unserem Beispiel haben wir auf Onlineshops wie ebay-Kleinanzeigen zurückgegriffen, die ohne Registrierung nutzbar sind.

Nach Abschluss dieser Recherchephase sollten Sie mit den Jugendlichen ihre Erkenntnisse diskutieren. Dies können Sie so gestalten, dass einige Paare ihre Erkenntnisse vortragen, und Sie die Erkenntnisse sammeln.

Leitfragen zur Auswertung sind:

1. Haben sich die Preise für euer Produkt innerhalb eines Shops über die Zeit geändert?
2. Welche Produkte wurden insbesondere während der letzten Phase in den Werbeanzeigen dargestellt?
3. Stehen zumindest ein Teil der angebotenen Waren mit eurem Produkt in Zusammenhang?
4. Welche Vermutungen habt ihr, wie hier KI gewirkt hat?

Wenn Sie die Inhalte der Werbeanzeigen an einer Tafel o.ä. sammeln und clustern, ist es für die Jugendlichen leichter, eine Adaption der Web- und Werbeinhalte auf das Suchverhalten zu erkennen und hieraus auch beispielsweise Rückschlüsse auf ihr eigenes Surfverhalten zu ziehen.

Fig. 3. Example page of module 1. These pages contain additional explanations for the teachers as well as supporting hints for the reflection of the previous worksheet. The layout has been developed by the supporting designers of the Science Year.

3.2 Module 2: How does Machine Learning work

How can a machine learn? What does it mean "to train the model"? As introduced in chapter 2.3, we intended to provide a low-threshold and vivid introduction into these topics, as it is well possible with a playful simulation.

In cooperation with our partners, we developed the current cardboard game named ”Man, Machine!” to explain how ML works (see fig.4 and fig.5).

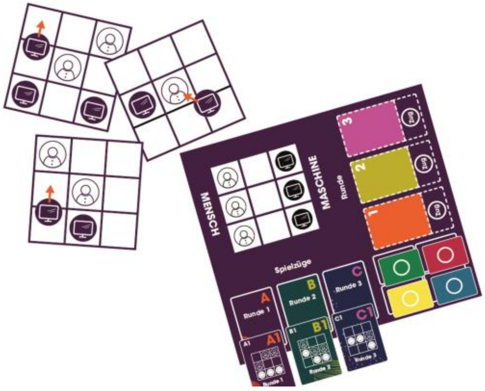


Fig. 4. Finished game design. The layout has been developed by the supporting designers of the Science Year.

Since this module can be used to explain many aspects of artificial intelligence and machine learning, much additional information about the technical aspects of these techniques has been included. An important learning objective is that the students do not over- or underestimate ML based on their experiences. On the one hand, the computer (the trained model) is perfect in this example because it is so simple. On the other hand, actual learning methods are more complex and trickier and can, therefore, be more complicated than the example suggests. Therefore we focused on a scientifically correct presentation of the technological basics and concepts.

3.3 Module 3: Difference between Man and Machine - What is Intelligence?

This module aims to investigate the differences between man and machine in detail. In particular, the question is whether artificial intelligence is really intelligent – and how to investigate this.

This module examines the concept of artificial intelligence in order to gain a better understanding of the subject area. The students should understand that AI systems are computer-driven applications and services in areas which mostly cannot be automated with conventional machines and thus have been human domains so far.

The question arises, what the differences between man and machine could be. Possible candidates to discuss are: People have a body, are socially integrated, cannot forget on command (machines do), people generalise strongly from a few examples, machines instead need many examples (cf. [5]).

Currently, human creativity is often mentioned as a difference (e.g. [2]), but creativity can also be ascribed to the computer. Therefore students need to explore different aspects of human intelligence to learn about the difference.

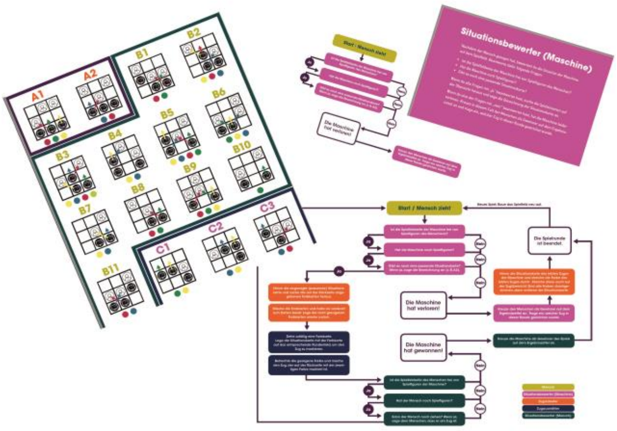

A1
A2
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
C1
C2
C3
Situationsbewerter (Maschine)
Start / Mensch zieht
Die Maschine hat verloren!
Die Maschine hat verloren!
Die Maschine hat gewonnen!
Die Spielrunde ist beendet.

Fig. 5. Impressions of the additional game material. The layout has been developed by the supporting designers of the Science Year.

For this purpose, we have developed several worksheets and activities. After a short introduction into the concept of ”intelligence”, students can test freely available chatbots like Mitsuku[32] for their behaviour and thus allow conclusions to be drawn about their ”intelligence”. A further perspective on the difference between man and machine is offered by the Turing test [17], a long-accepted test for distinguishing between man and machine, which is also introduced by the working material. Students can review and assess the answers given by Mitsuku during the contest for the Loebner Prize 2018 to recognise the differences between man and machine even better.

3.4 Module 4: A Historical View: The Development Path of the AI

In this module, various thematic aspects are linked and classified through a historical view. We focus on the changing role of computer science in the development of technical solutions.

Students can learn how the role and functioning of computer systems change over time.

Several stages of development will be discussed like the ”Dartmouth Conference”, the implementation of the ”Perceptron”, an early simplified replica of the human brain or the development of expert systems that can draw conclusions from previously entered knowledge and rule bases. Further, the so-called AI winter, which was caused by the limited computing power and memory capacity of the computer systems at that time [7], is a meaningful topic of discussion.

As these topics could be very boring for students, we developed an activity in which students should make a stop motion movie about the history of AI.

Students should work in groups, and each group creates one scene about one moment of AI history. To support the teachers, we created a comprehensive scenario, we included references to several tutorials for students and teachers and added worksheets to evaluate the resulting films.

An essential part of this module is the change of interaction and the roles of man and machine, furthermore the change in problem-solving – from algorithms to data-powered learning strategies. The students should understand that the solution for a problem is no longer given entirely by man (in the form of algorithms), but instead, examples are selected and compiled, using this, the machine learns to solve the problem.

3.5 Module 5: The distribution of roles of man and machine – ethical and societal aspects

In this module, the ”distribution of roles” between man and machine will be examined more closely. This includes not only the role of the developers but also the role of the users – because they also contribute to the further development of AI systems through their data input or data traces.

We talked to students and teachers before the development of the module, because this topic is very complex and includes many different aspects. Thus, we looked for examples which could be interesting for most of the students. In the end, we chose the topic of ”autonomous driving” for this module as the leading example. Furthermore, since many ethical aspects are addressed in this module, it contains fewer worksheets, but several role plays.

After an introduction into the six levels of autonomous driving students can decide at which level their own cars should operate without further discussion of risks and advantages – the results are recorded and can be used for later comparison. We added role plays to introduce the students, for instance, to the so-called dilemma of the ”trolley problem”, which can also be discussed comprehensively by younger students through their own experience. Depending on the age and previous knowledge, we then suggest discussing various other topics, such as face recognition in public or several rating systems. This module can be completed by a further discussion about the question, how much responsibility they are *now* willing to delegate to a machine, such as a vehicle AI.

3.6 Module 6: Workshop: In which ”AI world” do we want to live?

The concluding module can be connected to various modules and topics. We suggest doing this module in a project week or similar.

In this module, the students conduct a so-called ”future workshop” [13] about their own ideas on how AI should be integrated into their future lives. Leading questions in this module are ”Which AI world do we want to shape?”, ”What should interpersonal life in this AI world look like?” and last but not least, ”What does this mean for us now?”

We created a comprehensive scenario with several variations for the teachers. A typical future workshop consists of three phases, first the ”phase of criticism”, followed by the ”imaginative phase”, in the end, a ”realisation

[32] https://www.pandorabots.com/mitsuku/

phase”. The teacher booklet contains worksheets, ideas and tips for each phase, like how to prepare a debate, to build scenes or ideas on how to take different perspectives into account. With these ideas, the students can step by step visualise how they imagine their future with AI should look like – and how they themselves can actively participate in shaping their future.

4 Evaluation and Testing of Existing Materials

So far, we evaluated and revised all materials several times with different teachers, courses and age groups. We are always amazed by the level at which even younger students can discuss ethical and social topics if they only get the right stimulus. In this respect, role-plays and all activating forms, in particular, have been successful. In particular, the own replay of the trolley problem, when a student gets the opportunity to either have a group of classmates run over by a train formed by another classmate – or his best friend. Interestingly, almost all the students chose to use the switch and to sacrifice their best friend. The argumentation for this solution is mostly found in utilitarianism. Arguments in Kant’s sense are usually only found in the course of the discussion, especially when further questions are asked in the sense of the ”moral machine” [3].

We are currently developing an online questionnaire which will be sent to all teachers who ordered the teaching materials. We plan to evaluate how teachers use the material, how attitudes and self-concept for AI from teachers and students have been changed by using the teaching material, and how to improve the material further.

5 Summary and Conclusion

In this paper, we present our project in progress, which provides teaching materials to introduce artificial intelligence to students of different ages and school types.

Although the material has not yet been distributed, the experimental evaluation in the design-based research approach so far made us very optimistic that the materials can successfully be used ”out of the box” by teachers from different topics. We are also confident that we will be able to use and further develop the teaching material within our ”ProDaBi” project.

The project for the ”Science Year” is still very challenging, but we are confident that our materials can help to improve the competence of AI of many students and teachers. All these materials have been designed to use for lesson preparation as well as in class and will be sent to around 2000 interested schools all over the country in autumn 2019 for free.

We are very excited to receive feedback from the teachers!

References

1. Anderson, P., Bowring, J., McCauley, R., Pothering, G., Starr, C.: An undergraduate degree in data science: curriculum and a decade of implementation experience. In: Proceedings of the 45th ACM technical symposium on Computer science education. pp. 145–150. ACM, New York (2014)
2. Aoun, J.E.: Robot-Proof. Mit University Press Group Ltd, Cambridge, MA (2017)
3. Awad, E., Dsouza, S., Kim, R., Schulz, J., Henrich, J., Shariff, A., Bonnefon, J.F.,Rahwan, I.: The moral machine experiment. Nature **563**(7729), 59 (2018)
4. Bitkom: Ku¨nstliche Intelligenz: Bundesbu¨rger sehen vor allem Chancen. https://www.bitkom.org/Presse/Presseinformation/Kuenstliche-IntelligenzBundesbuerger-sehen-vor-allem-Chancen (2018)
5. Briggs, S.: What machine learning is teaching us about human learning.https://www.opencolleges.edu.au/informed/future-of-education/machinelearning-teaching-us-human-learning/ (2017)
6. Chesney, R., Citron, D.K.: Deep fakes: a looming challenge for privacy, democracy,and national security (2018)
7. Chollet, F.: Deep Learning with Python. Manning Publications Co., Greenwich,CT, USA, 1st edn. (2017)
8. Cobb, P., Confrey, J., diSessa, A., Lehrer, R., Schauble, L.: Design experiments ineducational research. Educational Researcher **32**(1), 9–13 (2003)
9. Curzon, P., McOwan, P.W.: Computer science for fun (cs4fn): The sweet learning computer: Machine learning. www.cs4fn.org/machinelearning/sweetlearningcomputer.php (2016), accessed: 2019-03-10
10. Gardner, M.: Mathematical games. Scientific American **206**(3), 138–144 (1962)
11. Grillenberger, A., Romeike, R.: Developing a theoretically founded data literacycompetency model. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education. pp. 9:1–9:10. WiPSCE ’18, ACM, New York, NY, USA (2018)

12. Heinemann, B., Opel, S., Budde, L., Schulte, C., Frischemeier, D., Biehler, R., Podworny, S., Wassong, T.: Drafting a data science curriculum for secondary schools. In: Proceedings of the 18th Koli Calling International Conference on Computing Education Research. pp. 17:1–17:5. Koli Calling ’18, ACM, New York, NY, USA (2018)
13. Mu¨llert, N.R.: Zukunftswerksta¨tten. In: Zukunftsforschung und Zukunftsgestaltung, pp. 269–276. Springer (2009)
14. Opel, S., Schlichtig, M., Schulte, C.: Developing teaching materials on artificial intelligence by using a simulation game (work in progress), accepted. In: Proceedings of the 14th Workshop in Primary and Secondary Computing Education. WiPSCE ’19, ACM, New York, NY, USA (2019)
15. Ridsdale, C., Rothwell, J., Smit, M., Ali-Hassan, H., Bliemel, M., Irvine, D., Kelley,D., Matwin, S., Wuetherick, B.: Strategies and best practices for data literacy education: knowledge synthesis report. Dalhousie University (2015)
16. Shulman, L.S.: Those who understand: Knowledge growth in teaching. EducationalResearcher **15**, 4–14 (1986)
17. Turing, A.: Computing machinery and intelligence. Mind **49**, 433–360 (1950)

Accessibility of the Computer Science study for students with visual impairment

Mária Stankovičová

Support Centre for Students with Special Needs, Comenius University in Bratislava, Slovakia
maria.stankovicova@rec.uniba.sk

Abstract. We present our research in the field of education of university students with visual impairment. The visually impaired students are interested in the Computer science study. Students with visual impairment require special attention not only in technical-material support, but also in learning strategies. We want to share our experience gained from the qualitative research aimed at case study of the Computer Science students with visual impairment. We focus on describing problems related to their study. We have identified three major areas, which we assumed essential for outlining possible solutions that would be helpful for teachers in the education process: 1. Programming environment accessibility for students with visual impairment. 2. Effectiveness of used teaching methods and forms for students with visual impairment. 3. Impact of visual impairment to students' understanding of programming concepts.

We also briefly introduce activities of The Support Centre for students with special needs, where we offer services for the students with disabilities to improve their integration in courses of the studies. The cooperation with teachers is also very important. We present the most commonly used assistive technologies for students with visual impairment. Our aim is to develop students' abilities to the benefit of their own as well as of the society.

Keywords: Accessibility, Visual impairment, Computer science study, Integration, Assistive technology.

1 Introduction

Increasing interest in Computer science study relates with development of technologies. Especially, for people with disabilities, assistive technologies represent an access to information and an opportunity to become more independent. Our research study aims to understand the special needs of visually impaired students in the study of computer science in tertiary education. We consider that the visual impairment affects the way the students obtain the information and create conceptions. An accessible form of the study material is essential for them.

The key subjects of Computer science are mathematics and programming and the significant requirement is to understand programming concepts. Most information has graphic representation and relevant description of context is needed. Understanding the specific needs of visually impaired students relevant to process of inclusive education is essential for the teachers that strive for more effective learning methods.

2 Background

The staffs of Support Centre for Students with Special Needs at Comenius University provide service and counselling for students with disabilities. Our effort is to prevent and to eliminate information and physical barriers which may occur in any situation. We regularly organize meetings and workshops for teachers and students. We discuss various methods and forms of education applicable for students with visual impairment. Our activities are divided into several areas described below.

2.1 Educational and technical activities

We organize the practical training courses for students and applicants to improve their skills to use assistive technologies. Students can borrow devices and use them in lectures during academic year. The following special devices are available especially for visually impaired students: screen reader[33], braille display[34], braille embosser[35], screen magnifying software, desktop and portable video magnifier, scanner, optical character recognition software.

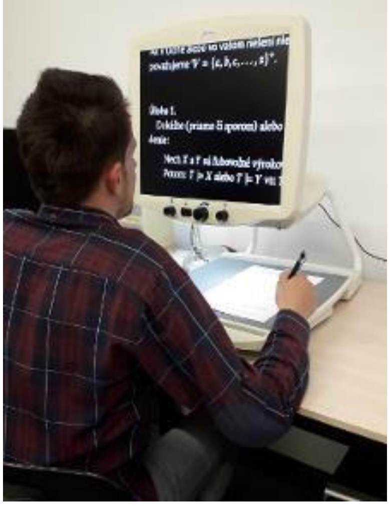


Fig. 1. Student uses video magnifier.

2.2 Counselling

We provide the information about special educational needs of students to academic staff to set appropriate educational conditions. The students with visual impairment use assistive technologies and require the preparation of accessible documents. To understand the possibilities and the limitations of assistive technology is essential for the teachers to set up appropriate educational conditions.

We give advice on prerequisites and necessary academic skills to prospective students to be acquainted in standards of chosen subject and to estimate extent of support. We organize preparation courses for applicants for university study. The participants receive important information about curricula and have opportunity to meet students with disabilities who study similar study subject.

2.3 Assistance

We offer individual mobility training (orientation in campus, becoming familiar with university information system). The staffs of The Support Centre regularly meet with students to give them advices on learning support strategies, study discipline, time management.

We assist teacher to prepare exams in accessible form for students with various disabilities. We coordinate the transforming of literature into a suitable form – electronic, Braille, large print, converting images to tactile graphics.

[33] Screen reader – a software using text-to-speech engine to translate text information on the screen into the speech

[34] Braille display – a peripheral device connected to a computer for displaying text in braille characters

[35] Braille embosser – a peripheral device to emboss braille writing system onto a paper

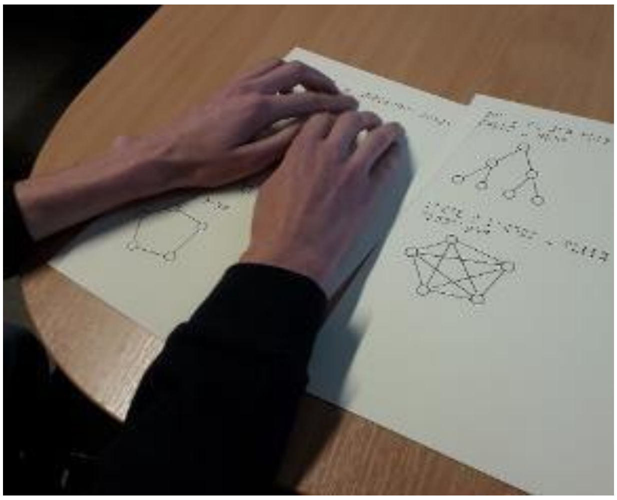


Fig. 2. Student reads documents with tactile graphics printed on braille embosser.

2.4 Cooperation with departments of the University

Following our experience students with visual impairment must be well prepared for studying at university. We aware of the importance of quality education at elementary and secondary schools, therefore we participate in projects aimed at supporting teaching informatics.

We cooperate with the Department of Informatics Education in several projects. In recent years we participated in project Computing Education for Blind and were concerned in developing of teaching materials in informatics for lower secondary school blind students [1]. Within this project we focused on the following areas of informatics: Text processing, Sound processing, Programming and problem solving, Working with mathematics formulas, ICT[36] for communication and information retrieval.

We have a very close cooperation with academic staff that is responsible for content of university webpage and academic information system. We help to identify problems of accessibility and regularly inform them.

3 Purpose of the research

One of the main themes of inclusive education is to improve the quality of the educational opportunities afforded students with disabilities [2]. The aim of our research study is to understand the special needs of the students with visual impairment and to observe how the visual impairment impacts their study of the computer science in tertiary education. Understanding how to satisfy special needs of students with visual impairment is the way towards inclusive education.

We assume high level of ICT literacy of applicant for studying computer science. The students must use various software programs and they also need to understand informatics concepts. Naturally, students could gain some basic programming skills in secondary education. However, misconceptions may occur. [3] Our research question is: How does visual impairment determine capability to solve tasks in computing?

Diversity of teaching materials unfortunately does not mean that every source of information is accessible for everybody. Accessibility of teaching materials means that the students obtain suitable form of the materials, which they can read without help of another person. The materials with mathematical notation indicate one aspect of the peculiar difficulty of the transforming into suitable form. The other aspect is searching for the suitable way of performing mathematical calculations. The accessibility of graphical information is also a serious issue.

Many sources are focused on visual perception, for example blackboard writing or printed books and therefore these sources are inaccessible for students with visual impairment. [4]

Many authors in their research study propose using assistive technology and give ideas how to make mathematics accessible applying LATEX [5, 6]. Our experience was similar, but in earlier period we have found useful AMS – ASCII Mathematical Script designed at the University of Karlsruhe and at Technische Universität Dresden [7]. Our students learned AMS notation quickly and they easily perform mathematical calculations. The disadvantage is rare usage. We suppose that AMS notation is used only at our university. We also have experience

[36] ICT – Information and Communication Technology

with the LAMBDA system [8], which was translated to Slovak language [9] and is used in elementary and high schools.

Blind students read 2-dimensional notation linearly. This way of reading is totally different as the sighted students read. The sighted students can find relations in formulas easier. For the students with visual impairment it could be hard to search relations because the length of the linear notation of the formula may take more than two rows.

We realized that the great attention is paid to exploring the accessibility of individual tools in programming environments for assistive technologies. [10, 11] The underlying idea is that visual impairment should not be a disadvantage when using any development environment. A survey among blind programmers has shown interesting ways to cope with the inaccessibility of utility tools. The inaccessibility of specific debugging tools forces the programmer with visual impairment to look for alternative solutions that may not be effective and may take a considerably longer time to implement them (for example using alert boxes). This process is not easy for a student who is just learning to program. The situation is also complicated by the awareness that his intact classmates use the tools of the development environment safely and progress while he stagnates or loses time by solving problems he might not have.

We consider useful for us the research resulting the design and implementation of accessible debugging tools for existing development environments. A very interesting project is the development of a plugin for Eclipse environment called StructJumper. [12] Students with visual impairment can quickly navigate and understand the structure of a code. Another interesting project is the implementation of a plugin for the Visual Studio called CodeTalk. Microsoft's development team has identified accessibility issues and suggested solutions. The plugin is part of the environment from 2015/2017. [13] For us, the results of these studies are important for removing barriers. Some might argue that all this is purely technical. But we think that wrongly chosen development environment will also have a significant impact on learning outcomes. The struggling with disproportionate conditions is a waste of student’s time and energy and he or she misses the essential – to study.

4 Research

4.1 Research methods

We decided to employ qualitative research strategy to gain an insight into the specific needs of visually impaired students in the study of informatics subjects in tertiary education. The case study research is the framework for the methods of this study. [14] We decided to use interviews, direct observations, and participant observations to collect relevant data. [15] The interview questions will focus on accessibility of teaching materials and software. The direct observations of lessons will be aimed at understanding the actual usage of the assistive technology. After the observations the participants fill structured questionnaires to increase reliability of observation.

We will also collect data from structured interviews with the teachers, who conduct obligatory courses. The questions will focus on methodology, usage of offered support, difference of behavior of visually impaired students.

Participant observation provides certain unusual opportunities for collecting case study data. The distinctive opportunity is the ability to perceive reality from the inside viewpoint. The Support Centre acts as mediator between the student and the teacher. We cooperate to find methodical approach and technical provision. Every found solution represents important source of research data.

4.2 Participants

We have asked actual students and their teachers, who conduct obligatory courses for participating on the research. This academic year one blind student and two students with low vision study computer science at our university. The inequality of visual impairments signifies different impact on their study.

The blind student “A” can read Braille, tactile graphics and uses computer with screen reader. The student does not have visualization, because the impairment occurred from early childhood. We consider this may affect the understanding of tasks with spatial relations.

Two students with low vision are slightly different. The first student “B” can read Braille and uses computer together with magnifier and screen reader. The student has good visualization but is not able to use handwriting. We assume, that for both students, “A” and “B”, the modification of tasks, which require drawing, is needed. The second student with low vision “C” is not able to read Braille and uses computer with magnifier. The student has good visualization and can use handwriting.

4.3 Findings

The objects of observation were teaching methods, used assistive technology and students' activities. The observation took place during two semesters by direct participation in exercises (programming and mathematics). We have identified three different teaching approaches during programming courses and one teaching approach during math lesson describing below.

Interactive cooperation. The teacher chose an interesting interactive teaching method – discussing with the students about the solution proposals. The selected student wrote the solution design in the web application environment, the other students read the code on their own computers and could comment. The interactive web app turned out to be inaccessible to the screen reader. The blind student was disoriented in the interactive application precisely because the screen reader could not interact interactively with the line in which the change was in progress – someone was writing. The code could only be read from start to finish. Although the student could listen to what the classmates talk about and deal with the assignment itself, he could not fully participate in the joint solution. In addition, the blind student needed consultations about the tasks he did not understand during the exercise.

The partially sighted student used a screen magnifier and had no major orientation problems and involvement in a common task solution. Only in jobs where code editing took place at multiple locations did it experience a slight time lag.

Individual work. The teacher chose the method of individual work. Students independently dealt with assignments. They passed the solutions through a web application that ran automatic code functionality testing. The teacher consulted students' questions individually. A web-based solution testing application was accessible for a screen reader. Students with visual impairment (both blind and partially sighted) dealt with tasks at their own pace (comparable to classmates).

Small groups. Students dealt with tasks in smaller groups (after 2 or 3), where the robot was programming. Robot handling – pressing the start button (too small to grip by blind person) was ensured by either the assistant (participant observation whereby the researcher was actively involved) or the sighted classmate. In developing the program, students with visual impairment could participate fully. Students used a computer with assistive technology (screen reader or magnifier).

Pre-set examples. For the students with visual impairment there were pre-set examples that were explained by the teacher during math lessons. The blind student used AMS notation. For students with low vision, the tasks were prepared in MathType editor. They solved tasks in MathType editor. In addition, all the students needed extra consultations about the tasks which they did not understand during the lesson.

5 Conclusion

We have described the activities of the Student Support Center in relation to students with visual impairment. We provided a brief overview of specific needs, focusing on how students work with assistive technologies. We presented the intention of our research project and outlined the closest plans. We assume that this research study provides the compact view on the special needs of visually impaired students in the study of computer science in tertiary education. We hope that the explaining of the impact of visual impairment on the computer science education will help to set up suitable educational condition.

References

1. Jašková Ľ., Stankovičová M.: Teaching materials in informatics for lower secondary school blind students. Informatics in Schools: Focus on Learning Programming Cham: Springer, 2017 S. 37-48 Lecture Notes in Computer Science; Vol. 10696 (2017)
2. European Agency for Special Needs and Inclusive Education: Five Key Messages for Inclusive Education. Putting Theory into Practice. Odense, Denmark: European Agency for Special Needs and Inclusive Education. ISBN: 978-87-7110-517-9 (2014)
3. Lister, R.: Toward a Developmental Epistemology of Computer Programming, WiPSCE'16 Proceedings of the 11th Workshop in Primary and Secondary Computing Education, ACM, Münster, Germany (2016)
4. Sokolov V., V.: Modern typhlo-information technologies in teaching students with significant visual impairment (from experience). In: Collection of articles of the 5th International scientific conference "Inclusive education: practice and perspectives", Baku, Republic of Azerbaijan, (2011)

5. Bexten E.M., Jung M.: LATEX at the University of Applied Sciences Giessen-Friedberg — Experiences at the Institute for Visually Impaired Students. In: Miesenberger K., Klaus J., Zagler W. (eds) Computers Helping People with Special Needs. ICCHP 2002. Lecture Notes in Computer Science, vol 2398. Springer, Berlin, Heidelberg (2002)
6. Gonzúrová W., Hrabák P.: Blind Friendly LaTeX. In: Miesenberger K., Karshmer A., Penaz P., Zagler W. (eds) Computers Helping People with Special Needs. ICCHP 2012. Lecture Notes in Computer Science, vol 7382. Springer, Berlin, Heidelberg (2012)
7. ASCII Mathematikschrift 2001, http://www.szs.kit.edu/download/ams.pdf, last accessed 2019/06/25
8. Schweikhardt W., Bernareggi C., Jessel N., Encelle B., Gut M.: LAMBDA: A European System to Access Mathematics with Braille and Audio Synthesis. In: Miesenberger K., Klaus J., Zagler W.L., Karshmer A.I. (eds) Computers Helping People with Special Needs. ICCHP 2006. Lecture Notes in Computer Science, vol 4061. Springer, Berlin, Heidelberg (2006)
9. Horňanský M.: Access to Mathematics for blind via Lambda editor in Slovakia, diploma thesis, Faculty of mathematics, physics and informatics Comenius university in Bratislava (2009)
10. Mealin, S., Murphy-Hill, E.: An exploratory study of blind software developers. Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on. IEEE, (September 2012), 71-74.
11. Albusays, K., Ludi, S., Huenerfauth M.: Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges. In Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17). ACM, New York, NY, USA (2017)
12. Baker, C., M., Milne, Ladner, R., E.: StructJumper: A Tool to Help Blind Programmers Navigate and Understand the Structure of Code. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). ACM, New York, NY, USA, 3043 (2015)
13. Polturi, V., Vaithilingam, P., Iyengar, S., Vidya, Y., Swaminathan, M., Srinivasa, G., CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 618, 11 pages (2018)
14. Creswell, John W.: Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research. Boston: Pearson, 4th edition, ISBN10 0131367390, ISBN13 9780131367395 (2012)
15. Yin, Robert K.: Case Study Research and applications: Design and Methods. London: Sage, 6th edition ISBN 978-1-5063-3616-9 (2018)

International Comparative Study on Computational Thinking Education in K-12

Yang Xing[1] and Jin-bao Zhang[*]

[1] College of Educational Technology, Beijing Normal University, China
xywzy468129@163.com
[*]College of Educational Technology, Beijing Normal University, China
zhangjb@bnu.edu.com

Abstract. As a new product of the computing age, computational thinking is considered to be a thinking method and strategy by using computational tools and methods to solve problems. It has the potential to play a vital role in the overall and lifelong development of everyone. In recent years, computational thinking education has gradually approached the horizons of educational researchers. However, the development of research and practice are still in the early stage. The questions such as unclear research objectives, not extensive research levels and not deep research issues are gradually revealed. What and how to teach become a huge problem for frontline teachers.

In order to find out strategic measures to solve practical problems, we selected USA, England, Lithuania, France and China as the research object. Each country has its own plan to develop CT at the K-12 stage. With the focus on CT education, the purpose of this paper is to compare and discuss the representative development path through the lens of international perspectives and cultural contexts to provide a general framework of CT education across the globe. We divide the whole framework into three parts: production, learning, and research. And we describe the framework as a tree, and the branches represent the dimensions, consisting of five parts: policy making, non-profit organization, industry promotion, teacher training, course implementation. Each dimension is related to each other and echoes each other to form a large whole system. The root of the tree represents the needs of students, indicates that all the dimensions are to serve for students. The aim is to develop computational thinking, logical thinking, digital competences of the students who need to prepare for the digital world.

Keywords: Computational thinking, Computational thinking education, Computer science, Digital economy, General framework.

1 Introduction

Nowadays, the word of "intelligence" has become a part of our life. The wonderful relationship between "artificial intelligence" and "humanity" has always been a hot topic of discussion, and there exist "replacement theory" and "control theory". In order not to be eliminated by the intelligent society, ideas and methods will become the basic survival skills residents of the intelligent era need. In recent years, we have witnessed an active discussion surrounding the important role of computational thinking (CT). Therefore, the owner of computational thinking should be the general public, and computational thinking education should be given high priority.

The main goal of CT education is to prepare the younger generations for the opportunities and challenges of the future economy where computing permeates virtually every aspect of society. Regardless of their ultimate field of study or occupation [1]. For example, in the process of learning computer science, students need to do more than just the application of new technologies, but complete the process of deconstruction and reconstruction based on understanding. The development of the future society also calls for the talent who owns new ability such as creativity, curiosity and critical thinking.

As a result, many countries favor the development of CT education: the three pillars of politics, economy and education support each other. Not only does it own national policy support, the promotion of the economic industry, but also it has theoretical and practical explorations in the field of education. The way in which CT education is promoted varies. Some national education departments issue relevant policies, carry out curriculum reforms or integrate CT as an independent discipline in the K-12 stage. Besides, strong economic industries such as Google, Microsoft have also joined the wave of promotion, using their own advantages to create a platform. For example, Google has a dedicated website for CT education(www.google.com/edu/computation- al-thinking) which publishes videos related to CT, providing teachers with different teaching methods and relevant resources for educators and managers.

2 Understanding Computational Thinking

2.1 Definition of CT

Computational thinking, a term mentioned by Papert in 1980 [2], generate attraction from researchers, practitioners and policy makers in the education field since 2006, through a seminal article by Wing [3], defined as a series of thinking activities that cover the breadth of computer science, such as problem solving, system design, and human behavior understanding, using the basic concepts of computer science. At the same time, other scholars also have their own interpretation of CT. The Computer Science Teachers Association and the International Society for Technology in Education [4] have developed an operational definition that provides a framework and vocabulary for computational thinking:

Computational Thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them;
- Logically organizing and analyzing data;
- Representing data through abstractions such as models and simulations;
- Automating solutions through algorithmic thinking (a series of ordered steps);
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources;
- Generalizing and transferring this problem-solving process to a wide variety of problems.

In August 2016, the CSTA released the K–12 Computer Science Standards. This stresses the problem-solving aspects, as well as abstraction, automation, and analysis as distinctive elements of CT: “We believe that computational thinking is a problemsolving methodology that expands the realm of computer science into all disciplines, providing a distinct means of analyzing and developing solutions to problems that can be solved computationally. With its focus on abstraction, automation, and analysis, CT is a core element of the broader discipline of computer science” [5].

2.2 CT’s relationship with programming

Computer Science is considered as the established academic term for the scientific discipline underlying the current digitalization and information technology [6]. And programming learning is the focus of computer science teaching. There are examples of countries around the world where programming or CS has been or will be introduced into early childhood education. For example, the European School net (October 2015) published a report surveying the current initiatives and plans in 20 European countries:16 countries integrate coding in the curriculum at national, regional or local level: Austria, Bulgaria, the Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia and the UK [7]. The analysis of many sources reveals that a variety of terms concerning CT are used including coding, programming, informatics, algorithmic thinking and so on. It can be seen that computational thinking and programming are closely related. But does programming education equal to CT education? What’s the deep relationship between programming and CT?

Based on the understanding of the running process of computer programs, programming is the method to achieve the control of the computer. It belongs to the tool level. However, CT covers not only the technical layer, but also the thinking of using tools and methods to solve problems. In general, it is agreed that CT and programming are not overlapping sets: “thinking as a computer scientist means more than being able to program a computer” [8]. Although programming plays an important role in the development of CT, it’s only part of it. In addition, CT includes other core ideas such as problem decomposition and abstraction, which are more based on the reality of life. The ultimate goal of CT education is to solve real problems. It is difficult to achieve this high requirement only through programming skills training.

Despite these distinctions, programming can make CT concepts more concrete and become a tool for learning. During the process of learning, the art of programming changes the children perspectives from application users to application creators so that they can express themselves freely. Last but not least, it also provides an opportunity to incorporate CT into other subjects, which is helpful to develop digital literacy and information literacy from an interdisciplinary perspective.

3 Objectives and Related Work

Scoping reviews of the CT education around the world and the country report have been conducted in the past but they only focus on the single aspect such as the policy, assessment, teacher development or the experience of teaching. However, these factors are not combined to form a large framework system. Among them, the European SchoolNet report description is more comprehensive, involving CT definition, integration with curriculum, programming skills assessment, teacher training and many other aspects [9]. As for the area of practice, Fredrik et al. reviewed how ten different countries have approached introducing computing into their K–12 education.The most common model was to make it compulsory in primary school and elective in secondary school [10]. Shuchi et al. summarized pertinent research on CT in K–12 involving the environment, tools and assessment [11]. Furthermore, in order to help teachers involve in teacher education and decision makers understand how and when CT can be included in their local institutions, Linda et al. discussed the current state of CT in K-9 education in multiple countries in Europe as well as the United States [12]. And Nicol R. interviewed five educational technology leaders with CS expertise from California, Florida, New Jersey, Missouri, and Maryland to collect the state’s support for CS program teachers [13]. As for policy, Yu-Chang et al. analyzed Reports, white papers, and policy documents across the global, summarizing four development trends: collaboration and partnerships across sectors and national boundaries, rationales taking a broad perspective and referring to common themes, a redefinition of digital competence, and an emphasis on broadening access and interest [14]. Stefania et al. discussed the most significant CT developments for compulsory education in Europe and provided a comprehensive synthesis of evidence, including implications for policy and practice [15].

The International Computational Thinking Bebras Challenge is dedicated to developing the thinking skills of 3-18 year olds, founded in Lithuania in 2004. It has been held for sixteen years. Analyzing data of 2018 Challenge(Fig.1), we found that the number of the French participants is the most, equaling to dozens of times of other countries’. And the USA, England, China also encourage students to participate in the challenge. In order to explore the idea of curriculum reform such as “British computing course”, “American computer science for all campaign”, we selected USA, England, Australia, Lithuania, France and China as the research object. Our goal is to contribute to the discussion around early education by 1) exploring the status of CT in K-12 education from different countries 2) exploring why the studied countries develop the CT perfectly 3) constructing the general development framework for CT education 4) preparing ideas and guidelines for collaboration and partnerships to introduce and enhance CT in the global education.

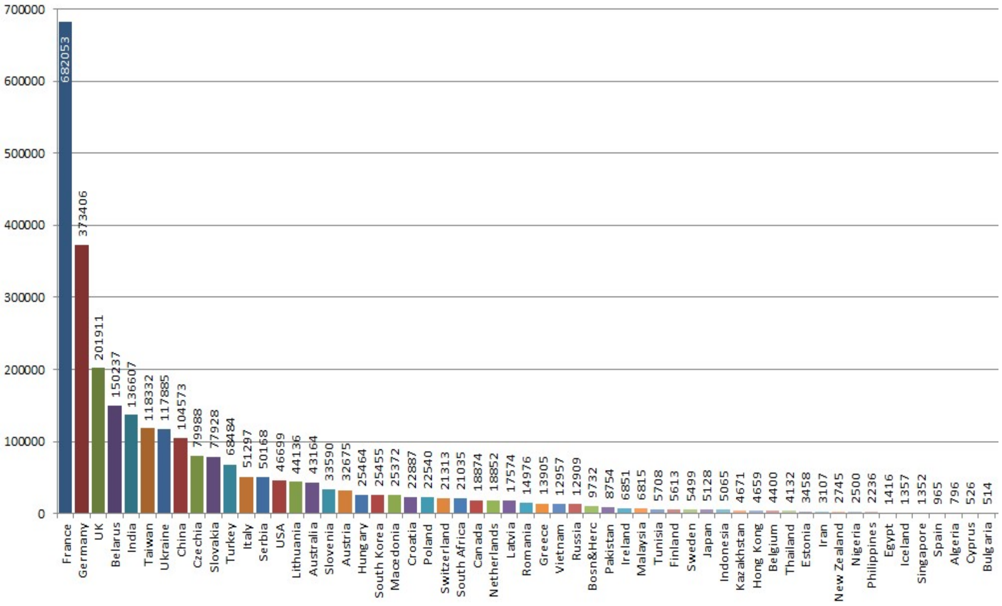


Fig. 1. The statistics of the participants in Bebras Challenge 2018 from countries.

4 Status of CT In Education

Despite this widespread interest, successful CT integration in K-12 education still faces unresolved issues. It has presented some challenges for educators, as computing has traditionally been primarily a university level discipline, there are no widely accepted general standards for what computing at K–12 level entails [16]. Through the analysis of domestic research literature, it is found that CT education research is still in its infancy. 80% of the research involves CS courses in higher education, and K-12 researches focus on the information technology curriculum, lacking interdisciplinary integration education. After interviewing with primary and secondary school teachers in Beijing, China, it is found that most of the teachers don't have a background in CS and they are not familiar with the basic concepts of CS. The frontline teachers are very appreciative of the importance of developing students' CT in the basic education stage, but there isn't a clear curriculum system. It is difficult to combine CT with the courses currently taught. What and how to teach becomes the biggest problem facing teachers. Not to mention the assessment.

Based on the domestic CT education issues, this study summarizes the development experiences of other countries (USA, England, Lithuania, France) and explores the possibility of mutual learning, cooperation and interaction between countries to promote common development. To study ways and models for how CS or CT can be introduced, we select the following criteria to focus on:

- Policy making
- Role in primary and secondary level curriculum (compulsory/elective/other)
- Integrated or a subject of its own
- Computing tools and assessment
- Teacher training (in-service and pre-service)
- Organizational support
- Economic industry promotion

Country reports

4.1 A. USA

In USA, the control of education primarily depends on the state and local government, which represents that the status of CS may vary significantly across the states. But there is a national program named "computer science for all" in 2016, initiated by President Obama's bold new initiative to empower a generation of American students with the computer science skills they need to thrive in a digital economy [17]. During the process, every state promotes CS education in the K-12 stage actively. In order to keep high quality, they provide the teachers with training for their own CT skills. Moreover, businesses and philanthropies like Apple, Facebook, Microsoft and Code.org have also played a major role in building a movement to expand CS nationally. For example, Microsoft announces its campaign to have all states adopt "Make CS Count" policies and invest in CS education as a complement to existing $75 million initiative.

There is also a great effort the main organizations including ACM, CSTA, Code.org, CIC, and NMSI make to develop a Framework for K–12 Computer Science Education (https://k12cs.org/). The framework consists of five percepts and seven practices. States, districts, and organizations can use the framework to inform the development of standards and curriculum, build capacity for teaching computer science and implement computer science pathways. It concentrates on the following issues: What should be taught and what the children can do after graduation? How to carry out CS education in early childhood? How to plan a course to ensure the effective connection of compulsory and secondary education? How to assess the development of CT?

However, there are few places in the USA where CS is required for students. At best, high school students have the option of taking electives in technology or, in some places, taking the Advanced Placement Computer Science course which is currently a Java programming class. The CS AP course allows students to take a standardized test, which if they pass, provides them with college credit [18]. This new focus on computational thinking has been further supported by an educational programming renaissance that has brought about the development of new computer science teaching tools (for example Alice, AgentSheet, Blockly, Kodu and Scratch).

Teachers in the US are required to receive their primary teacher certification in some discipline other than computer science and then meet additional requirements to receive a supplemental computer science endorsement [19]. The gap between the availability of teacher certifications or endorsements and the requirement to teach CS subjects is of great concern. Luckily, The generous funding from the National Science Foundation, from industry leaders such as Google and Oracle, and from CSTA makes Efforts to develop teacher training programs [20].

4.2 B. England

With the changes in the digital age, the “tools” curriculum information communication technology that cultivates students' software application capabilities has been questioned by British academics and industry. In October 2012, to ensure Britain competes and thrives in the global race, the government sets out plans to boost the teaching of computer science, supported by industry experts such as Microsoft, Facebook, BT and IBM [21]. In January 2013, because of the importance of computer science for both education and the economy, the government announced that it was included in the EBacc exam [22]. In February 2013, the Ministry of Education announced that the new curriculum, computing, was officially implemented, replacing the original ICT curriculum and taking computational thinking as the core goal of the course [23]. Therefore, the UK chose to set up an independent curriculum to develop student computing thinking, rather than using programming as an intermediary for development.

The content of the Computing course mainly involves three aspects: computer science, information technology and digital literacy. The course objectives are divided into four phases, covering both primary and secondary education. The main aims are to ensure that all students [24]:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyze problems in computational terms, and have repeated the practical experience of writing computer programs to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology

To ensure every child in every school has the right to world-class computing education, a community called CAS (Computing at school) established, supported financially by BCS, Microsoft, Google, Ensoft and the UK Committee of Heads and Professors of Computer Science. Unlike an organizer, CAS is more like a public platform where teachers and experts can share their understanding of computing courses, interact what difficulties they encounter during student performance, improve the structure of the curriculum, and improve the quality of teaching. At the same time, CAS provides instructional programs and instructional syllabus for each stage, from the most basic teaching essential resources to the high-level activity curriculum programs, including Barefoot Computing and Quick Start Computing and Teach Primary Computing for primary educators.

The teachers who are currently employed in a school and delivering computing lessons can take part in the teacher training of CAS. And the participants who satisfy the required credit can get a certification of BCS to recognize their computing knowledge and ability. In addition, CAS executes a project which trials some new initiatives aimed at improving girls’ participation in computing.

4.3 C. Lithuania

In Lithuania, CS is referred to as informatics. The first few after-school classes in programming were established in several Lithuanian secondary schools in 1970– 1975. There also appeared the first ideas to establish a school to teach secondary school students programming [25]. So teaching informatics started in the early 1980s with programming. As a part of the Education Reform in 1997, the Informatics core curriculum went through a major revision and it was expanded from teaching two years to four years (in total 136 hours) with more focus on application and the processing of information [26]. After the new reform of secondary schools in 2005，the subject "informatics" is rename "Information Technology".And the main attention of schools was paid to satisfy students’ need and foster computer literacy.

Methodology on teaching CS at secondary school level was strong and well designed by Lithuanian's researchers. The IT subject is taught by IT teachers who have various types of training. Some of them have a CS or mathematics degree combined with education, while others are teachers of other subjects with little training in IT [27]. The students of grades 5-10 are required to take part in the course 1 hour per week. IT includes five knowledge areas: information; digital technologies; algorithms and programming; virtual communication; security, ethics and legal principles. At the upper secondary level (grade 11-12), IT is an elective subject offered in basic and advanced modes. The advanced course includes electronic publishing, database design and management, and programming [28].

Apart from the course, there are also exams and contests concerning informatics. The curriculum of Informatics exam closely corresponds to the content of the programming module. Three main fields are emphasized: algorithms, data types and structures, and constructs of a programming language [29]. To introduce CS concepts, some interesting tasks related to real life with the knowledge of CS are set up for different age students. The international challenge on Informatics and Computational Thinking Bebras focus on concepts-based tasks. All students in secondary education under the age of 20 are invited to compete in Lithuanian Olympiads in Informatics [30]. These tasks can include a wide range of concepts within CS including algorithms and programs, both sequential and concurrent; data structures like heaps, stacks and queues; modelling of states, control flow and data flow; human-computer interaction; etc.

4.4 D. France

The history of informatics education in France has some similarities with other European countries. In the 1980's, informatics was referred as a new elective course in senior high schools [31]. But there exists statement that the goal of the course was not to teach programming languages, but to promote the algorithmic way of thinking as being useful for all other subjects [32]. Since then, there has been a constant debate about whether it should be a separate discipline or an aid to other disciplines. In 1999, informatics education disappeared from French high school [33].

Many years later, some scholars realized the importance of computer science to the future development of students, the Ministry of Education is also gradually concerned. As for elementary school, the curriculum covering programming, unplugged activities et.al is established by groups of experts-the Superior Curricula Council, appointed by the MoE. During the process, the teacher may advisory the researcher for specific points, usually more technical or unclear content [34]. As for secondary school, in 2012, algorithmic was introduced in mathematics curricula at grade 11 and an elective Computer Science course (called ISN) has been introduced at grade 12. The ISN course adopted project-based learning and teaching methods and it was structured into four themes: information comprehension, algorithmic, languages and programming, computer architecture [35].Students' learning in ISN must be assessed during an oral presentation of the outcomes of one of the projects they worked for during the year. In September 2015, the Superior Curricula Council proposed new curricula for kindergarten, elementary and middle school [36].

The educators for ISN could be teachers from scientific subject matter (mathematics, physics and chemistry) and technological subject matters (applied and industrial sciences) [37].All teachers were motivated and had inclinations towards digital literacy, but none had previous knowledge of programming. So most ES teachers are trained mainly by Superior Schools of Teaching and Education-SSTE (Ecoles sup´erieures du professor at et de l'´education) which deliver them a master [38]. Beside this, begun in France in 2016, Class'Code is a consortium of universities, companies and the government, offering educators who work with children ages 8–14 free online CT courses and resources as well as face-to-face meetings with facilitators throughout France [39].

4.5 E. China

In China, computer science is usually offered at a university, and it's called information technology in primary and secondary school. However, it is not an assessment subject for important exams (such as college entrance examinations). In the context of exam-oriented education, CS and computing education has long been decontextualized and considered irrelevant by many students. However, since 2007, China has progressively transformed and restructured computing education through CS0 (College Computers course) reform by integrating CT into the curriculum. The CS0 reform took four different approaches/contexts/content areas, including "CS0 course for the deaf", a MOOC on C Programming, a MOOC on College Computers and a CS0 course designed for health majors [40].

As computational Thinking and related concepts (e.g. coding, programming, algorithmic thinking) have received increasing attention in the educational field, introducing them into compulsory K-12 classes becomes necessary. Although Chinese CT education started late, it has also been moving forward. In 2017, the Ministry of Education issued the "Standards for Information Technology Courses in Ordinary High Schools" in which CT was regarded as one of the core literacy of information technology disciplines. The original information literacy goal was further developed into digital literacy, creative ability, and computational thinking. The course is set to compulsory, elective, and optional compulsory subjects, including artificial intelligence, 3D design, open source hardware, database, information system and other subjects [41]. Moreover, the new college entrance examination reform program piloted in Zhejiang Province include the information technology course in the seven-choice three subjects. It is the first time that information technology is recognized as the assessment subject in the key examinations [42]. The emergence of computational thinking has injected new vitality into the information technology curriculum.

In the practice of CT training, in addition to a small number of unplugged computer science gamification activities, programming is widely recognized as an effective way Graphical programming tools Scratch and App Inventor become the first choice.

5 Summary and Conclusions

Different countries have different institutional settings according to their own development, but education is global and extensive, regardless of country. To adapt to the future economic development, the education departments of various countries pay more attention to the thinking training of students in K-12. The visible hand of "policy" and the invisible hand of "industry" form a huge educational circle to promote the development of CT education jointly. With the focus on CT education, we select USA, England, Lithuania, France and China as the research object. Each country has its own plan to develop CT in K-12:

Table 1. Overviews of different countries.

Country	USA	England	Lithuania	France	China
Subject	Computer Science (own subject)	Computing (own subject)	Information technology (own subject)	Computer Science course(own subject), mathematics(integrate d)	Information technology (own subject)
Primary/secondary	compulsory/ compulsory	compulsory	compulsory/ elective	elective	compulsory/ compulsory elective
Focus	computational thinking, critical thinking	computational thinking, computer science	computational thinking, CS knowl edge	information comprehension, algorithmic, computer architecture	programming knowledge, digital literacy
Tools	programming language, graphical programming	Programing language, graphical programming	programming language	programming unplugged activities	programming graphical programming
Assessment/exam	standardized test	EBacc exam	exams and contests	national exam	exams and contests
organizational support	Apple,Microsoft,Facebook,code.org, ACM,CSTA ,CIC,NMSI	Microsoft, Google,Intel Facebook, BT,IBM,BCS,CAS	/	SSTE,SCC, Clss'Code	CCF
Characteristic	k-12 framework, five percepts, seven practices	CAS : a public platform	Bebras Challenge	Coteaching between teachers and researchers	/

As shown above, we can conclude that economic industry and non-profit organizations play an important role in the process of curriculum reform in some countries (the USA, England). On the one hand, non-profit organizations call together experts, researchers and teachers who are committed to the development of CS. On the other hand, financial support of the industry funds for the teacher training, curriculum development and other aspects of the project. In addition, the UK focuses on establishing an open communication platform, listening to the voices of each participant and solving problems together. There're also countries with institutional support, as well as France, but the most impressive point is the joint teaching of educators and researchers. It shows that the close integration of educational theory and practice, which can be known as a model of education. Lithuania, the founding member of the Bebras Challenge, is very different and has always attracted students' interest in CS with the Bebras Challenge and the Informatics Olympics.

6 Suggestion and Future Prospects

However, CT is referred to as a way of thinking. Its most evident characteristic is abstraction. So we think it is not appropriate to say "cultivation" but to say "development". The purpose of this paper is to compare and discuss the representative development path through the lens of international perspectives and cultural contexts to provide a general framework of CT education across the globe.

After summarizing, we divide the whole framework into three parts: production, learning, and research. And we describe the framework as a tree, the branches represent the frame dimensions, consisting of five parts: policy making, non-profit organization, industry promotion, teacher training, course implementation. Each dimension is related to each other and echoes each other to form a large whole. The root of the tree represents the needs of students, indicates that all the dimensions are to serve for students. The aim is to develop computational thinking, logical thinking, digital competences of the students who need to prepare for the digital world. The Ministry of Education usually releases guidance document which the schools and organizations must follow. The document usually brings together a variety of stakeholders, including students, parents, teachers, administrators, universities, industry leaders, nonprofits and government agencies. It also charges for the certificate qualification of the teachers and the requirement teachers should satisfy. The most important part is course implementation in that students can exercise their thinking and learn something new here. And it can give feedback to policy makers, practitioners and researchers. The teacher training is to form a big circle of teachers so that they can share their ideas and experiences and to provide course guidance and material for teachers. The organization about CS and industry like Apple, Microsoft usually build a relationship of cooperation. They build platform and provide financial support for the teacher training. In a word, the implementation of perfect education needs efforts from multiple departments.

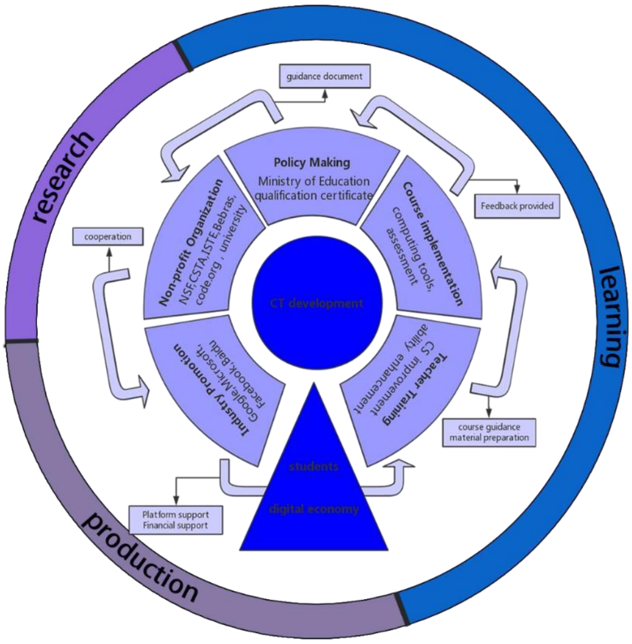


Fig. 2. The general development framework.

Computational thinking is becoming ever increasingly important to our society. However, CT or CS in primary and secondary education has not been well developed.

Some measures should be taken to change the current situation. One of the most effective ways is to open up the door of each country so that they can learn from each other. Though every country has different conditions, they can summarize the successful experience and apply it combined with the actual situation. There is also a research comparing and discussing the computational thinking educational policy initiatives emphasizes the trend of collaborative relationships and partnerships that cut across sectors and national boundaries [43].

As we all know, cooperation between countries may bring great progress in the economy. It is reasonable to believe that cooperation can also promote the development of education. Here are a few suggestions:

- With the development of transportation, the members of the Ministry of education can visit the CT related course in other countries by themselves. The course implementation can give feedback to the policy making. They can interact with the teachers and students to get first-hand information. As experts of education, their suggestion to the improvement of the course plan is also valuable for the teachers.
- With the developed Internet, the teachers from each country can take part in a public community where they can share the experience of their native curriculum and resolve the questions for each other. Otherwise, it is necessary to build a Competitive relationship. For example, the teachers can make groups of ten formulating a course plan during a semester and comparing the teaching achievement in the end.
- The industry promotion plays a great role in development. The industries like Google may provide financial support or teaching material for the teacher training. They can record the process of the training as videos so that many people can get access to the resource through MOOC. The role of the industries can also be an intermediary to combine the countries together.

References

[1] Computing in the Core, http://www.computinginthecore.org/.2014, Last accessed 2019/6/28.

[2] Seymour Papert: Mindstorms: Children, Computers, and Powerful Ideas[M].New York-Basic Books :285-286(1980).

[3] J. M. Wing.:Computational thinking. Communications of the ACM 49(3),33-35, (2006).

[4,8] CSTA & ISTE: Operational definition of Computational Thinking in K-12 education. (2009) https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

[5] CSTA:K-12 Computer Science Standards, Revised 2017.(2017) https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf

[6,10,16] Heintz, Fredrik., Mannila, Linda., Färnqvist, Tommy.: A Review of Models for In-troducing Computational Thinking, Computer Science and Computing in K-12 Education. In: Proceedings Of The 46th Frontiers In Education, pp. 1-9. (2016).

[7,9] European Schoolnet: Computing our future:Computer programming and coding Priori-ties,school curricula and initiatives across Europe.(2015).

[11] Grover, S., Pea, R.： Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher, 42(1), 38-43(2013).

[12,31] Mannila, Linda., Settle, Amber.,Dagiene Valentina. et al. : Computational Thinking in K-9 Education. ITiCSE-WGR 2014 - Working Group Reports of the 2014 Innovation and Technology In :Computer Science Education Conference. (2014).

[13] Howard, N: EdTech Leaders’ Beliefs: How are K-5 Teachers Supported with the Inte-gration of Computer Science in K-5 Classrooms? Technology, Knowledge and Learning, 24(2), 203-217(2018).

[14,39,43] Hsu, Y., Irie, N., Ching, R:Computational Thinking Educational Policy Initia-tives (CTEPI) Across the Globe. TechTrends, 63(3), 260-270(2019).

[15] Engelhardt, Katja.: Developing Computational Thinking in Compulsory Education - Implications for policy and practice. JRC104188(2016).

[17] White House: “Computer science for all,” https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.(2016) Last accessed 2019/6/28.

[18] Mannila, Linda. , Settle, Amber., Dagiene, Valentina. et al. : Computational Thinking in K-9 Education. ITiCSE-WGR 2014 - Working Group Reports of the 2014 Innovation and Technology In:Computer Science Education Conference. (2014)

[19,20] Gal-Ezer, Judith, and Chris Stephenson. "A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States." ACM Transactions on Computing Education (TOCE) 14.2: 1-18(2014).

[21] New Industry-Backed Plans to Boost Computer Science Teaching and Help Britain Compete in the World[DB/OL]. http://www.education.gov.uk/inthenews/inthenews/a00215981/new-industry-backed-plans-to-boost-computer-science-teaching(2013) Last accessed 2019/6/26.

[22] Computer Science to Be Included in the EBacc [DB/OL]. http://www.education.gov.uk/inthenews/inthenews/a00221085/ebacccompsci. (2013) Last accessed 2019/6/26.

[23] Computing Programs of Study for Key Stages 1-4 [DB/OL].) http://computingatschool.org.uk/data/uploads/computing-04-02-13_001.pdf(2013) Last ac-cessed 2019/6/28.

[24] Department for Education:"National curriculum in England: Computing programmes of-study,"https://www.gov.uk/government/publications/national-curriculumin-england-computing-programmes-of-study.(2013) Last accessed 2019/6/27.

[25] Dagiene, V: The Road of Informatics. Vilnius, TEV(2006).

[26,28] Dagiene, V., Stupuriene, G.: Informatics Concepts and Computational Thinking in K-12 Education: A Lithuanian Perspective. Journal of Information Processing24(4), 732–739(2016).

[29,38] Benaya, T., Zur, E., Dagiene, V., Stupuriene, G. : Computer Science High School Curriculum in Israel and Lithuania – Comparison and Teachers' Views. Baltic Journal of Modern Computing, 5(2), 164-182 (2017).

[30,33,35] Dagiene, Valentina, Jurate, Skupiene.: Olympiads in Informatics 1: 37-49. Web. (2007)

[31] Tort, Françoise & Drot-Delange, Béatrice. Informatics in the French Secondary Curric-ula: Recent Moves and Perspectives. 7780. 31-42 (2013).

[32] Baron, G.-L.: L'informatique, discipline scolaire?: le cas des lycées. PUF, Paris (1989)

[34] Vanea Chiprianov, Laurent Gallon. Introducing Computational Thinking to K-5 in a French Context. 21st Annual Conference on Innovation and Technology in Computer Sci-ence Education, Jul 2016, Arequipa, Peru(2016).

[36] Conseil Sup´erieur des Programmes. Projet deprogrammes pour les cycles 2,3,4 (French), (2015). 15

[37] BO, n. 36, 6 oct. 2011 - http://www.education.gouv.fr/ pid25535/bulletin_officiel.html? cid_bo=57998

[40] Pan, T. -Y. Reenergizing CS0 in China. In P. J. Rich & C. B. Hodges (Eds.), Emerging research, practice, and policy on computational thinking. pp. 351–362. Cham: Switzerland, Springer. (2017).

[41] Ministry of Education of the People's Republic of China. Information Standards for In-formation Technology in Ordinary High Schools [M]. Beijing: People's Education Press, (2018).(in Chinese)

[42] Ministry of Education of the People's Republic of China,Eight provinces released com-prehensive reform plan for college entrance examination, http://www.moe.gov.cn/jyb_xwfb/gzdt_gzdt/s5987/201904/t20190424_379274.html.(2019). Last accessed 2019/08/27(in Chinese)

Categorization of theoretical tasks in the middle school competition PRASK

Michal Anderle

Department of Computer Science
Comenius University, 842 48 Bratislava, Slovakia
anderle@dcs.fmph.uniba.sk

Abstract. Slovakia has a rich history of various algorithmic competitions. Middle school algorithmic competition PRASK is a new addition to them as it was only founded in 2015. Upon creation, there were several problems that needed to be addressed. Mainly, how to introduce middle schoolers to algorithm design throughout a series of tasks without relying on previous algorithmic knowledge or ability to write code in one of the programming languages. This led to the introduction of theoretical tasks that complemented standard programming ones. These theoretical tasks have been using a sequence of graduating problems that direct pupils towards the design of the final algorithm.

So far, more than 80 original tasks were created, 31 of them are theoretical. In this paper, we will present a closer look at these theoretical tasks. They will be categorized based on computational thinking skills required from pupils and also according to the levels of the cognitive domain of Bloom's taxonomy. In the end, emerging patterns will be presented and described.

Keywords: algorithmic competition · middle schools · task categorization · PRASK.

1 Introduction

Informatics competitions are an interesting tool for teaching students. They can present various concepts of computer science to pupils and while doing so, popularize it among them. Depending on the character of the competition, a task can be a small programming exercise or a theoretical problem focusing on some computer science concept, often delving deep into the given issue.

Slovakia in particular has a rich history of various algorithmic competitions that began in the early 1980s. They vary in formats, types of tasks and age categories of contestants. Bebras challenge focuses on the motivation of pupils, trying to introduce them to a variety of informatics topics using tasks that are solvable in 3 minutes [1]. National Olympiad in Informatics or Correspondence seminar in programing (KSP) [2] concentrate more on in-depth analysis using complex tasks that require advanced knowledge and can be solved during longer periods of time.

In 2015, an alternative to KSP for middle schoolers was founded – the PRASK competition [3]. It is intended for pupils in middle school (approx. 13-15 years old) that are interested in computer science, but even total beginners should be able to compete in it. One of the main goals is to teach pupils various informatics concepts, which is reflected in its tasks.

This competition is organized by volunteers, university and high school students without proper teacher training. Therefore, it would be interesting to look at the tasks of the PRASK competition, their structure and topics and evaluate whether they are suitable for middle schoolers and whether they helped to develop their computational thinking.

2 Research

PRASK is an algorithmic competition for middle schoolers (age approx. 13 to 15) that is organized in Slovakia. Four times a year, a set of five tasks is presented and contestants have at least one month to come up with solutions. These solutions are then graded by organizers and sent back with feedback.

There are three types of tasks in PRASK – theoretical, practical and programming. In our research, we decided to focus on theoretical tasks. In them, contestants are often confronted with some well-known algorithm or data structure. However, no prior knowledge is required from pupils, all rules and relations are described in the task statement. The goal of participants is to combine them and present some formal procedure.

In this preliminary research, we have tried to categorize the theoretical tasks of the PRASK competition according to their statements. We were interested whether there are some patterns that can be spotted and what is the structure of a typical task. This can be helpful in subsequent research when we try to analyze the pupils' solutions.

First, we have gathered statements from theoretical tasks published between January 2015 (start of the competition) and March 2019. It included 29 different tasks that were analyzed. After we first read through the statements it was apparent that statements were too long and extensive for proper categorization. However, each task is divided into several subtasks (number of subtasks differs from task to task, the median of them is 5), so separate subtasks were categorized. The total number of analyzed subtasks was 135.

For analysis of subtasks, two categorizations were used. The first comes from similar informatics competition – Bebras. It is two-dimensional categorization proposed by Dagiene et al. [5] in 2017. This categorization is based on the work of Selby and Woollard about computational thinking [4] and classifies tasks by both used informatics concepts and computational thinking skills. There are five main skills defined in it: abstraction, decomposition, algorithmic thinking, evaluation and generalization.

The second categorization was based on the cognitive domain defined by Bloom's taxonomy [6], which is a widely used tool for assessing learning objectives of tasks or activities. It defines six levels of cognitive domain: knowledge, comprehension, application, analysis, synthesis and evaluation.

3 Preliminary Findings

So far, one researcher read through all theoretical tasks from PRASK competition multiple times and categorized them based on the above-mentioned categorizations, refining and correcting this categorization during each round of reading. Even though the results are partially subjective, some emerging patterns can be spotted.

In two-dimensional categorization, we have focused on computational thinking skills. Occurrences of these skills were counted across all subtasks and also for subtasks according to their order in the statement. Even though not all tasks have an equal number of subtasks, we wanted to find out, if the beginning of the problem is different from the end. Only subtasks f), g) and h) were excluded as they had 7, 3 and 2 occurrences respectively. Results are displayed in Table 1.

Table 1. Number of occurrences of computational thinking skills across subtasks.

Subtask	Number of	Abstraction	Algorithmic thinking	Decomposition	Evaluation	Generalization
a	29	75.86%	72.41%	10.34%	55.17%	31.03%
b	28	67.86%	78.57%	14.29%	53.57%	60.71%
c	28	57.14%	75%	25%	46.43%	64.29%
d	21	47.62%	80.95%	28.57%	42.86%	76.19%
e	17	35.29%	82.35%	29.41%	35.29%	94.12%
total	135	59.26%	77.78%	20%	45.93%	64.44%

In the Table 1 several trends can be spotted. The most represented skill is algorithmic thinking, the least represented is decomposition. This can be explained by the structure of the tasks. The existence of the subtasks divide the problems into smaller parts, therefore the decomposition is often worked out in advance by task setters and contestants can focus on algorithmization.

We can also notice the decrease of the use of abstraction and increase of the use of a generalization throughout the problem. The structure of the subtasks is such, that the first subtasks introduce the problem and setting, letting contestants remove unnecessary details and spot key relations, and later subtasks are about putting all obtained information together.

The categorization according to Bloom's taxonomy was done in a similar fashion. However, we decided to exclude the Knowledge level of the cognitive domain as it was not that informative for our research.

According to Table 2, analysis level had the greatest number of occurrences. It is the most important step while discovering new relations and uses, which is often at the core of the tasks. The least prevalent was the evaluation level, which is the most difficult level of the cognitive domain. It was contained in subtasks, in which pupils were asked to prove some argument or needed to think about complexity and optimality of their solutions. We can also notice that subtasks a) and b) are mostly on comprehension and application level and synthesis level is present mostly in higher subtasks.

Table 2. Number of occurrences of levels in cognitive domain of Bloom's taxonomy.

Subtask	Number of	Comprehension	Application	Analysis	Synthesis	Evaluation
a	29	96.55%	89.66%	31.03%	0%	6.9%
b	28	42.86%	89.29%	96.43%	21.43%	7.14%
c	28	10.71%	60.71%	100%	57.14%	14.29%
d	21	0%	57.14%	100%	76.19%	19.05%
e	17	0%	58.82%	100%	100%	23.53%
total	135	32.59%	74.07%	83.7%	47.41%	14.81%

These findings paint a nice picture of a structure of the PRASK task. In the following research, we are going to let other researchers categorize these tasks, as we can then use data triangulation to strengthen our claims. Also, we are planning to look at which levels of Bloom's taxonomy appear in subtasks together and how often. This could lead to the description of an archetypal PRASK task, which could be a useful tool for the problem setters and other researchers.

Acknowledgment
This research was supported by the Comenius University in Bratislava Grant UK/235/2019.

References

1. Dagiene, V., Futschek, G.: Introducing informatics concepts through a contest. In: IFIP Working Conference: New Developments in ICT and Education. Universite de Picardie Jules Verne, Amiens (2010)
2. Correspondence seminar in programming, https://ksp.sk (2019)
3. Anderle, M.: PRASKan Algorithmic Competition for Middle Schoolers in Slovakia.In: Olympiads in Informatics, Vol. 12, pp. 147–157 (2018)
4. Selby, C., Woollard, J.: Computational thinking: the developing definition (2013)
5. Dagiene, V., Sentance, S., Stupuriene, G.: Developing a two-dimensional categorization system for educational tasks in informatics. Informatica, 28(1), pp. 23–44 (2017)
6. Bloom, B.: Taxonomy of Educational Objectives: The Classification of Educational
7. Goals. Handbook 1 Cognitive Domain McKay. New York (1956)

Learning Computer Science at a Fair with an Escape Game

Sébastien Combéfis[1,2] and Guillaume de Moffarts[1]

[1]ECAM Brussels Engineering School, Woluw´e-Saint-Lambert, Belgium
sebastien@combefis.be
[2]Computer Science and IT in Education ASBL, 1348 Louvain-la-Neuve, Belgium
guillaume.demoffarts@csited.be

Abstract. There exist several pedagogical devices to introduce computer science (CS) concepts to young pupils. No matter the resource used and the targeted age group, the learning can be done in an autonomous way with books or videos, at school with a teacher, during an event/fair with animators, etc. The challenge is always to keep the learner interested and involved with the activities. This paper presents how an *escape game* has been used to foster attendees of a fair to learn CS concepts. They were told a story in which they are locked in a strange bunker. To escape, they have to solve riddles needing basic understanding of CS concepts, such as binary numbers, text ciphering, or bubble sort algorithm, for example. Stands managed by animators allow the visitors to learn the CS concepts through interactive animations. Informal feedbacks from participants and stand animators were good.

Keywords: Escape Game · Computer Science Concepts · Fun learning

1 Introduction

Computer science (CS) concepts are not only taught to higher education students, but also to younger pupils [2,8]. Not all CS concepts can be taught to younger pupils, at least not in the same way than with higher education students. Specific pedagogical devices have been developed, focusing on computational thinking, digital literacy, etc. but they are not always easy to find, unless using a specialised digital library [4]. The *Computer Science and IT in Education ASBL* is a nonprofit organisation whose goal is to promote computer science at large. It regularly organises the *Computer Science Day* (CSDay), a fair where children and adults with no prior CS knowledge get the opportunity to be introduced to several CS concepts. The visitors of the fair can visit stands to learn CS concepts thanks to a team of animators, teaching them these concepts.

1.1 Related work

Particular pedagogical devices and resources must be selected to teach CS concepts to people without any CS background. For example, *CS Unplugged* activities [1] or the *Computer Science Field Guide* can be used to grow algorithmic thinking and encourage programming [5]. The place of learning is also important: it can be a classroom or other dedicated places, such as computer science fairs [7]. Such a place is perfect to allow its visitors to try different things at their own pace, following their own preferences. One challenge is to make sure that everyone gets the opportunity to take part to all the activities.

Using serious games to teach and learn CS concepts can be very effective, in particular for programming [9,3]. Learners enjoy playing game and are pushed to make progress and actively learn new things, if the design of the game is good. Also, the level of the game should be adapted to its audience, to not discourage its players. Finally, escape rooms and escape games provide challenging and motivating games. If well-designed, they can be used for educational purpose in the form of an escape classroom [6,10].

1.2 Motivation

This paper is about the *escape game* put in place during the CSDay 2019 fair to foster visitors to learn and be involved with the activities, in a fun and motivating way. Indeed, an observation made during the previous editions of the fair is that it was not easy to encourage its visitors to go through the different stands and animations. Just telling them to follow a route to learn several things was not motivating enough. Following the gamification process and taking advantage of the attractiveness of escape games resulted in a new way to organise a fair and to naturally attract visitors to the stands, to teach them CS concepts, without computers, in a fun and challenging way and allowing them directly practice.

2 Escape Game Design

The visitors of the fair receive a sheet of paper with a story at the entrance, telling them they got stuck in a strange cave from which they want to escape! No timer, no pressure, they can collect and solve the riddles during all the day, knowing that the only constraint is to escape before the end of the fair.

The game takes place in several rooms, between which visitors were free to move. To help them solve the riddles, they had a log book they are receiving incrementally and clues they are collecting. To understand and interpret the clues, visitors should use some CS concepts. They can learn them hopping by a stand to ask an animator explain them. The riddle to solve is a pretext to learn a new concept, visitors are led where we want them to go.

2.1 Riddle

What kind of riddles do the visitors have to solve? For example, at some point in the game, visitors were directed towards an old computer in a hallway, whose keyboard was altered, some keys being highlighted. After a visit to the stand about data ciphering, where visitors can learn about the Cæsar cipher, they manage to find a meaningful word from the highlighted letters of the altered keyboard. How are the visitors helped? Words written in bold in the log book are used to attract their attention to guide them towards the riddle they have to work on or the stand they should visit to make progress. Another riddle is a drawing of a strange circle cut in six parts, with digits from 1 to 6 placed on these parts. Visitors have to find a number from this drawing, and must visit the algorithm stand to understand how to solve the riddle.

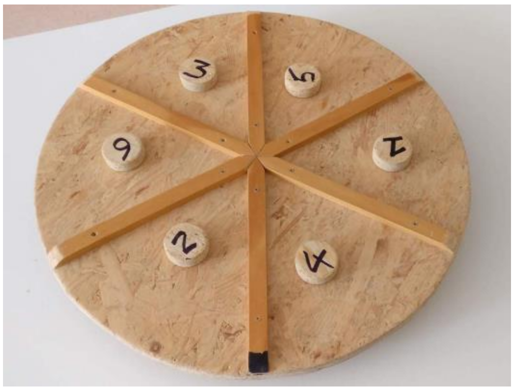


Fig. 1. A turntable is used during the fair to teach visitors the notion of algorithm and understand how a computer can sort numbers with the bubble sort algorithm.

2.2 Activity

What kind of activities are proposed? They are similar to CS Unplugged pen and paper activities to involve learners and to interactives from the Computer Science Field Guide to allow learners to experiment with the problem to solve. For example, visitors can understand the bubble sort algorithm thanks to a strange turntable with digits, shown on Figure 1. They have to follow instructions on a flowchart to manipulate it, and observe that the digits get sorted in ascending order. They can then make the link with the riddle with the drawing of a strange circle and understand that the number to guess is the number of loop iterations.

3 Conclusion

The escape game of the CSDay attracted more visitors to the event and to the activities organised during the fair. Visitors were happy and, in particular, the goal (a) to make them go through all the stands to learn CS concepts, (b) to be interested and involved in their learning by practicing and experimenting directly, (c) and to have a good time with a fun activity was clearly achieved. No formal survey about the game has been conducted yet, but informal feedbacks from the attendees show that the three objectives have been met. Some children refused to leave the fair until they solved all the riddles. Some participants to other events that took place during the fair told us that the escape game was an excellent idea to spend time between the other events.

Some drawbacks have also been observed. The level of difficulty of the escape game was not adapted to all the accepted age groups. A more guided path in the game should have been proposed for younger children. Also, the pedagogical device requires a lot of human resources the day of the fair. The team of the 2019 edition had 10 full-time people for the whole day, which was not enough.

To conclude, this first edition of the fair with an escape game, which managed to attract more than 34 persons and make them busy for a whole day learning CS concepts, was a big success. The combination of activities without computers, mixing the philosophies of the CS Unplugged and the Computer Science Field Guide activities with an escape game, fostered the learning of CS concepts thanks to the gamification process. Future improvements should include a multi-path story to better accommodate the different age groups and the design of new riddles and activities for uncovered CS concepts.

References

1. Bell, T., Alexander, J., Freeman, I., Grimley, M.: Computer science unplugged: School students doing real computing without computers. The New Zealand Journal of Applied Computing and Information Technology **13**(1), 20–29 (2009)
2. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS: Educational standards for computer science in lower secondary education. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education. pp. 288–292. ACM, New York, NY, USA (2009)
3. Comb´efis, S., Beresneviˇcius, G., Dagiene˙, V.: Learning programming through games and contests: Overview, characterisation and discussion. Olympiads in Informatics **10**(1), 39–60 (2016)
4. Comb´efis, S., de Moffarts, G.: TLCS: A digital library with resources to teach and learn computer science. Olympiads in Informatics **13**(1), 3–20 (2019)
5. Comb´efis, S., Van den Schrieck, V., Nootens, A.: Growing algorithmic thinking through interactive problems to encourage learning programming. Olympiads in Informatics **7**(1), 3–13 (2013)
6. Dietrich, N.: Escape classroom: The leblanc process–an educational "escape game". Journal of Chemical Education **95**(6), 996–999 (2018)
7. Fitzgerald, S., Hines, M.: The computer science fair: An alternative to the computer programming contest. In: Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education. pp. 368–372. ACM, New York, NY, USA (1996)
8. Hazzan, O., Gal-Ezer, J., Blum, L.: A model for high school computer science education: The four key elements that make it! In: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education. pp. 281–285. ACM, New York, NY, USA (2008)
9. Kazimoglu, C., Kiernan, M., Bacon, L., Mackinnon, L.: A serious game for developing computational thinking and learning introductory computer programming. Procedia - Social and Behavioral Sciences **47**(1), 1991–1999 (2012)
10. Nicholson, S.: Creating engaging escape rooms for the classroom. Childhood Education **94**(1), 44–49 (2018)

Music Computer Technologies and Musical Informatics Training Course for Students

Irina B. Gorbunova, Andreas Kameris, Elena N. Bazhukova

[1,2,3] Herzen State Pedagogical University of Russia, 48, Emb. River Moika, 191186 St. Petersburg, RUSSIA
gorbunovaib@herzen.spb.ru

Abstract. The beginning of the 21th century was marked by the introduction of computer and communication technologies in all spheres of human activity. Global changes have occurred in the way information is transmitted and presented. Digital technology has penetrated into music and music education. The achievements of sound recording, the technology of creating musical compositions combined with the new mass media possibilities have defined the previously non-existent areas of development and distribution of music, and require such knowledge that musicians who have received a classic music education do not have. The article describes the basic, which is devoted to development of actual theoretical and practical basis of raising knowledge of students at the music department with usage of music computer technologies.

Learning to play electronic musical instruments acquires a special relevance in the system of general music education at the school of the Digital Age, such an opportunity can also be provided in the system of additional education of students. Today in the class of electronic musical instruments it is necessary to meet the requirements of modern society in connection with the growing social demand for a higher level of professionalism, it is necessary to conduct a quality educational process and develop new, including digital technologies in the field of art. The discipline "Music Informatics", which is constantly evolving, since the mid-70s (IRCAM, France), is designed to solve a number of educational problems in this direction.

Keywords: Music Computer, Musical Informatics, Music Computer Technologies

1 Introduction

The development of a new approach to higher, vocational musical education has resulted from the need to solve the most acute problems in mass music pedagogy, including the current state of general music education. So, for example, at the annual International research and practical conference Contemporary Musical Education, held together by the Herzen State Pedagogical University of Russia and Saint Petersburg State Conservatory named after N. A. Rimsky-Korsakov from 2002 to 2019, musicologists, teachers of musical disciplines, researchers, teachers-practices noted that despite outstanding creative achievements, the possibilities of music education in mass pedagogy are not fully exploited and the teaching methods in the system of general music education do not change significantly.

One of the ways to solve this problem would be to find new educational technologies. They need to be improved by creating musical programs that would allow flexible and diversified use of the variety of rich pedagogical tools in teaching music and the huge possibilities of a music computer, contemporary music computer technologies (MCT).

Musical Iinformatics, in conjunction with other disciplines, helps integrate the modern musician in his or her professional activities. The purpose of the course of Musical Informatics is creating pre-conditions for expanding the professional capabilities of the musician with the use of modern digital technologies and facilitate the fuller use of the creative potential of the performer, composer or teacher of music.

The purpose of the subject is mastering music computer technologies, the software and hardware used in the professional activities of the musician and acquiring, by him or her, experience for working with digitized and synthesized sound and musical material in various formats as well as many other similar things.

2 The Discipline of *Musical Informatics* Includes the Following Sections:

The Introduction

The Subject of Musical Informatics

Music, Mathematics, Informatics: The Bounds of Their Interaction
Analysis as concerns harmony in works of music art and mathematical methods of their description.
Tentamen novae theoriae musicae ex certissimis harmoniae principiis delucide expositae (on the theory of music by L. Eller).
The descriptive-symbolic conception of music («Grammatica Speculativa» by Ch. S. Peirce and others).
On the characterization of various aspects of music creativity or "Musical Mathematics".
Musiques formelles by I. Xenakis.
Music programming.
Audio-visual synthesis.

Architectonics of Acoustic and Digital Musical Sound
Sound vibrations.
Musical sounds (fundamental tone, harmonics, notes).
The spectrum of sound.
The intensity and volume of sound.
Stereophonic parameters.
The modulation of high frequency vibrations.
The theory and practice of preserving sound.
Digital recording, the processing of sound and playback.

Musical Synthesizers
Extracts from the history of musical synthesizers.
Electronic musical instruments.
The musical instrument as synthesizer of musical sound.

Technologies of Sound Synthesis
The basic types of sound synthesis.
Sound filters: low-pass, hi-pass, band-pass, notch.
Equalizer.
Sound cards.
Audio mixing console.

Music Computer
Music computer: excerpts from history.
Computer modeling of music creative elements.
MIDI.
Music computer hardware.
Music computer software.
Music computer setting.
Music computer as the new multifunctional multi-timber musical instrument.
Music computer as the tool of teaching in the context of the basic methodological principles of the primary, secondary and higher additional professional and inclusive musical education.
Music computer as an instrument of the performer.

Digital Audio Workstation
The creation of a project, settings, the basics of working with an audio fragment.
The dynamic sound processing.
The frequency sound processing.
The spatial sound processing.
Reverberation.
Modulation effects.
Mixing and mastering of the musical project.

Digital Musical Synthesizer as a Modern Software-Hardware Complex for Teaching Musical Informatics
The modern digital musical synthesizer as part of the subject of music informatics.

Digital musical synthesizer: instrument controls.
Digital musical synthesizer as a means of performing music.

Professional Music Software
Audio editors.
Note editors.
Program music designers.
Automated musical arrangers.
Sequencers.
Teaching musical software.

On-line Music Teacher Assistance Services
A review of sheet music on-line software.
On-line audio editors.
Programs for microphone sound recording on-line.
Music studios.

Appendices
Appendix 1. A table of General MIDI instruments (GM)
Appendix 2. *Mathematical methods of research in musicology* course by M.S. Zalivadny
Appendix 3. Armenian universal analytical chart *АК-4* by V. Goshkovsky
Appendix 4. Examples of the use of matrix recording in analyses of music structures
Appendix 5. The examples of developing the original software by students participating in the Musical Informatics course: *MetronomKa* by A. Bungova and T. Bungov

Design for Curriculum on Programming Education in Primary Schools of Japan from 2020

Takeharu Ishizuka[1], Daisuke Hironaka[1] and Tatsuya Horita[2]

Fukuoka Institute of Technology, Junior College, Fukuoka, Japan
ishizuka@fit.ac.jp
Tohoku University, Sendai, Miyagi, Japan

Abstract. Programming education will be introduced in all primary schools in Japan from April 2020. In Guidelines for Standard Education in Japan called "Courses of Study" published by The Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT) in 2017, there appeared the word "programming" for the first time in Japanese primary education history. The legal basis of the school learning is the description of the "Courses of Study" and all schools in Japan have to follow it. Programming is positioned as a part of the ability to use information (including to use ICT) in the "Courses of Study". However, there are very few descriptions and no specific curriculum is indicated. In this paper, a pilot curriculum which is just started from September 2019 at one primary school in Japan is proposed. Then some leaning materials what to be presented at ISSEP 2019 are introduced.

Keywords: programming, primary school, Japan.

1 Introduction

1.1 New Guidelines for Standard Education in Japan

All kindergarten, primary schools, lower secondary schools and upper secondary schools in Japan are required to follow the guidelines for standard education named "Courses of Study" by MEXT [1] in order to secure the level of education nationwide. The courses of study are revised approximately every ten years. The latest version of them were published in 2018 for upper secondary schools and in 2017 for other kind of schools. Education under the new courses of study begin from April 2018 for kindergarten, from April 2020 for primary schools, from April 2021 for lower secondary schools and from April 2022 for upper secondary schools. Keywords in the latest revision of the courses of study for primary schools are "active learning", "programming", "subject for foreign language", and so on. It is the first time that word "programming" was appeared in Japanese primary education history. In this section the parts involved in programming will be introduced.

In Chapter 1 (General Provisions) of the new guideline, it is indicated that the ability to use information is the qualities and abilities that form the basis of learning.

Then, it is stated that the following learning activities should be carried out systematically in order to foster the ability to use information as follows:

1. Learning activities for mastering basic operation of the digital instruments such as typing characters that is required as a basis for learning.
2. Learning activities with programming experiences to acquire the logical thinking to make computer doing the processing intended by children.
3. Although there are no independent subject for programming itself, there are three descriptions of programming in the new guideline as followings;
4. To draw regular polygons in Arithmetic for Year 5, programming should be used for repeating the same procedure and drawing various regular polygons in the same way of drawing with changing some parts.
5. The nature and function of electricity in Science for Year 6, programming can be used to learn that there are tools whose motion changes according to the given conditions.
6. In the process of exploratory learning, programming can be used in the Period for Integrated Studies from Year 3 to Year 6.

1.2 Guide for Programming Education at Primary School

Programming is positioned as a part of the ability to use information in the "Courses of Study", but there are very few descriptions and no specific curriculum is indicated. Then, "Guide to programming education at primary school (1st Ed.)" was published by MEXT in March 2018 and 2nd editions in November 2018 [2]. In this guide, learning activities related to programming at the elementary school level were classified into six categories (see in Table 1.)

Table 1. Table captions should be placed above the tables.

Category	Learning activities to be carried out
A	in the unit indicated in the course of study
B	in the unit in each subject in the course of study except Category A
C	separately from each subject but in the class under the curriculum
D	as club activities, etc for specific children under the curriculum
E	outside the curriculum but the venue is a school
F	outside the school

2 Curriculum for Primary School

2.1 Designing a Curriculum for Primary School in Japan from 2020

Designing a curriculum, three main viewpoints as follows were reflected.

From the Viewpoint of Curriculum Management. All schools must do the activities in Category A of Table 1 for Year 5 and 6 because they are stated in Courses of Study. However, it is impossible to carry out the programming activities just learning the corresponding unit. Therefore, children need to have acquired basic programming skills before then as Category C.

From the Viewpoint of Connection with Secondary School Education. To implement real programming education at secondary schools, it is important for all primary school students should experience to make easy programs in the class.

From the Viewpoint of Developmental Stage. In consideration of the developmental stage of the child, learning the skills for coding itself should be set in Year 3 or 4 children. However, even lower grade children can acquire the skills on programming thinking with activities of unplugged and applications as Category C.

2.2 Pilot Curriculum at Primary School

Pilot Curriculum for one primary school is shown below. This school is a municipal and has 15 normal classes with about 400 students. This pilot curriculum is just started from September 2019.

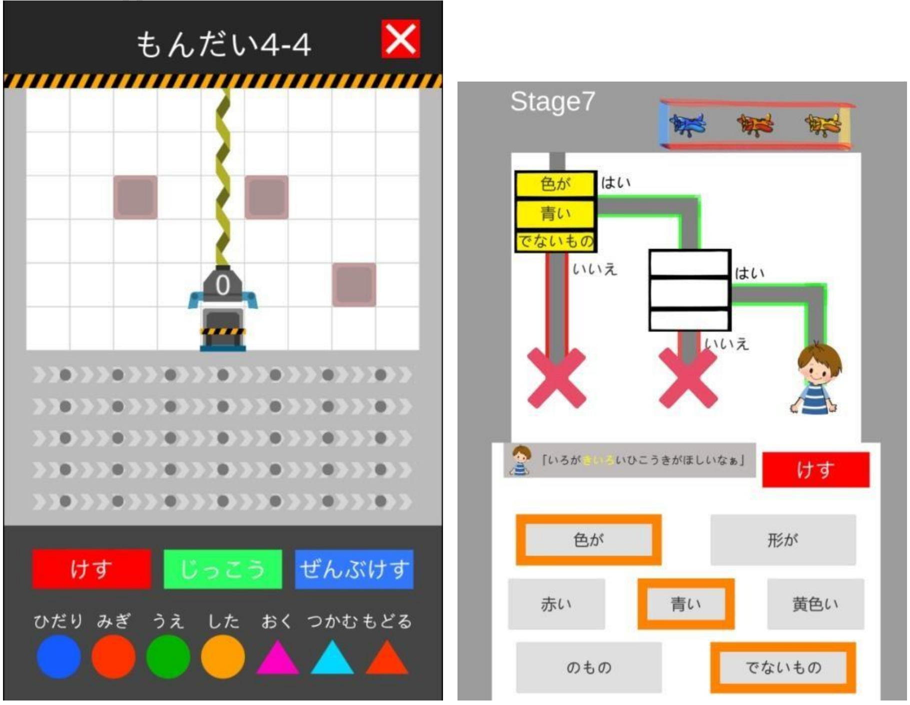
7. To draw motion objects with Viscuit[3] for Year 2.
8. To practice for thinking things in order with learning programming thinking purposed application for children with special needs assistant.
9. To learn programming skills for Year 4, 5, 6.
10. To draw regular polygons in arithmetic class for Year 5.
11. To use electromagnetic switch module with micro:bit in science class for Year 6.
12. To use a drone for integrated studies for Year 5.
13. To use micro:bit as a sensor in the period for integrated studies for Year 6.
14. Club activities in the curriculum and out of the curriculum for Year 4, 5, 6.

3 What to be Presented at ISSEP 2019

Following items are going to be presented at ISSEP 2019.

3.1 Applications for Acquiring Programming Thinking

These applications (see Fig.1) were developed for acquiring programming thinking for even lower grade children. These applications are used for 2) and 8) in Section 2.2.



「いろがきいろいひこうきがほしいなぁ」

Fig. 1. Sample screens of applications for acquiring programming thinking. In left figure is for training for ordering and right one for conditional branch.

3.2 Electromagnetic Switch Module with micro:bit

This learning program is developed for understanding to control sensor switch for 5) in Section 2.2.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 17K00993 and MEXT KAKENHI Grant Number 18H01045.

References

1. The Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT) : Courses of Study for primary school., Tokyo (2017).
2. The Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT) : Guide to programming education in primary school., Tokyo (2017)
3. Viscuit, https://www.viscuit.com/ Last accessed 30 Jun 2019

Integrated Digital Education – Computational Thinking For Everyone

Corinna Kröhn[1] and Barbara Sabitzer[2]

[1]Johannes Kepler University, Linz, Austria corinna.kroehn@jku.at
[2]Johannes Kepler University, Linz, Austria
barbara.sabitzer@jku.at

Abstract. When Austria introduced the new mandatory subject "basic digital education" in September 2018 for all pupils in grade 6 (twelveyear-old), computational thinking finally found its way into the school system. Schools can decide if they offer specific subjects or if they implement the curriculum in an integrative way in several subjects. However, schools still fight the problem who is teaching and how, because most schools just cannot install a stand-alone subject. This results in the fact that non IT teachers have to teach the contents of this curriculum, too. Hence, creative ideas for the implementation of the new curriculum are required because the schools need support for a cross-curricular introduction to computational thinking.

Keywords: computational thinking · digital education · teacher education · cross-curricular · STEM.

1 Introduction

Before 2002, digital education policies in the European Union aimed to improve infrastructures in schools [2]. The EU ICT Cluster study from 2010 shows that the ratio between digital devices and the number of learners decreased significantly. Additionally, the reliability and the speed of internet connections increased throughout the European Union [3]. There is also strong evidence that these improvements in infrastructure provision have not systematically been translated into the integration and routine use of digital technologies [2].

In 2011, all European countries had digital education policies in place, either as standalone policies or as part of a national ICT strategy. The strategic weight of these policies remained on nurturing students' digital competences, justified by future economic benefits [4].

2 Basic Digital Education In Austria

When the Austrian government introduced the new subject "basic digital education" in 2018, it was the first major progress since the implementation of IT lessons in 1988. In those 30 years, life of an ordinary pupil changed a lot – obviously Austrian schools did not. The fact is that in Austria there is no subject "computer science" that is taught consistently throughout the school career, besides the one of IT-education in year five. Of course there exist numerous autonomous solutions in various schools but nothing standardized.

The new subject covers digital competences, media competences as well as civic education. According to the curriculum, those three topics should not be taught separately but must be connected to each other. The BGBLA ("Bundesgesetzblatt der Republik Osterreich" – federal law gazette) states that the¨ main aim is to improve the pupils dealing with media and technology to a more well-briefed and responsible one.

Schools can decide on their own if "basic digital education" is implemented as a stand-alone subject or integrated in Maths, English or else. Most schools just do not have the possibility to install a separate subject because of a lack of teaching staff or available teaching hours. While teachers play an important role in both scenarios, the type of support and/or training provided to them is diverse. As a consequence, the new curriculum has to be combined with the topics of the original subject, even if teachers have no training in digital education.

3 Integrated Digital Education Using Diagrams

With the idea of introducing computational thinking into daily school lessons, the problem of less-experienced teachers in the field of computer science could be reduced. Computational thinking covers solving problems, designing systems, and understanding human behavior by drawing on the concepts of computer science [5]. It also uses abstraction and decomposition when handling complex tasks or designing complex systems [5]. Modeling and visualization like creating diagrams is a common approach in the field of computer science, particularly when organizing and structuring complicated content. Modeling can be used as teaching and learning strategy, similar to concept mapping, in order to structure knowledge, visualize situations, plan processes and train key competences.

First a short text about a city tour bus is presented. Pupils should now extract the important information of the text passage and represent it using a diagram. To help with the process, pupils could work with different colors and shapes to highlight essential words. In this case, a blue rectangle represents a noun, a green rhombus a verb and a yellow ellipse different attributes.

The resulting entity relationship diagram (ER diagram) is shown in figure 1 using the highlighted terms from the text. It illustrates the relationships between individual objects. Such a concept can be introduced easily in any language lesson while learning the theory of text production.

As a consequence, one can summarize that a major benefit of the ER diagram is, that even complex structures and coherence can be simplified and made understandable for everyone. It should be mentioned, however, that it is not our main aim that the pupils design strictly correct ER diagrams but to impart the curriculum of the language lesson with computational thinking concepts operating in the background. Many more of those examples are currently developed, to help teachers covering these unfamiliar topics.

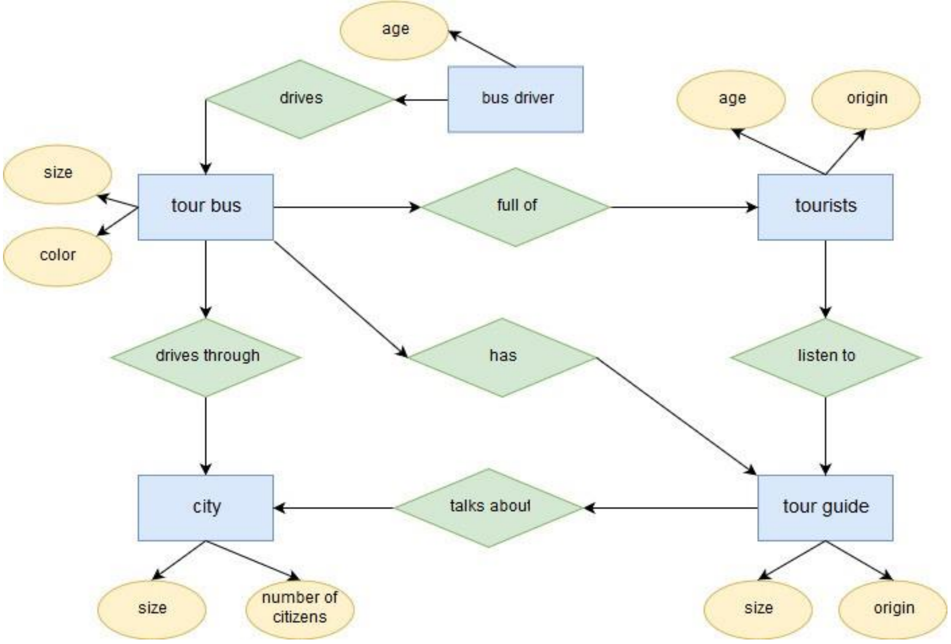


Fig. 1. Modeling as a tool for text work: ER diagram

4 Pilot Study “Integrated Digital Education”

4.1 Methodology

In order to gain better understanding if the implementation of the new subject “basic digital education” caused any problems, we designed a pilot study. The main research question is if the teachers need additional help and/or materials. The survey was conducted in various Austrian schools and was sent to teachers who took part in workshops or teacher training at our institute. As the number of test persons is quite low (*n* = 27), the results are not statistically relevant and have to be interpreted with caution.

5 Results

The majority of the sample group supported the idea of teaching “basic digital education” in an integrated way in their own subject. However, experience has shown that it is not that easy to design satisfactory lessons with integrated “basic digital education” concepts in each and every subject. For those who are already used to the integration of technology, like in Mathematics, Physics, Chemistry or the like, it is not that hard to meet the challenge. The part that worries teachers most when new structures are introduced, is if they still have enough time to fulfill their own curriculum. This could be solved if at least a part of "basic digital education" is taught in an integrative manner.

5.1 Discussion

An interesting fact is that 91% of the teachers claimed that they had no problems with the implementation of the new curriculum. The open comments section of the survey and practise reflects precisely the contrary. Teachers wrote that they do not have enough knowledge nor training or instructions to teach "basic digital education" competently. Others claim that especially those with lots of years in service have no interest in teaching something new or simply do not have the courage to leap the hurdle.

6 Outlook

While our survey cannot provide definite conclusions, the results do show a trend that there is need of additional teaching material and teacher training. We already developed various teaching materials that need testing and started with school intern teacher training to provide help. In order to better identify and, consequently, offer successful support for our region, we will now refine the survey and send it to each Upper Austrian school with help of the Austrian education authorities, and additionally make proposals for feasible and effective lessons of basic digital education across all subjects.

References

1. BGBLA (Bundesgesetzblatt der Republik Osterreich):¨ 71. Verordnung, 1- 24 (2018). https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2018_II_71/ BGBLA_2018_II_71.html. Last accessed 13 June 2019
2. Conrads, J.; Rasmussen, M.; Winters, N.; Geniet, A. et al. (2017). Digital Education Policies in Europe and Beyond. http://publications.jrc.ec.europa.eu/ repository/bitstream/JRC109311/jrc109311_digedupol_2017-12_final.pdf. Last accessed 18 June 2019
3. EU ICT Cluster (2010). Learning, innovation and ICT: Lessons learned by the ICT cluster Education & Training 2010 programme. https: //erte.dge.mec.pt/sites/default/files/Recursos/Estudos/key_lessons_ ict_cluster_final_report.pdf. Last accessed 18 June 2019
4. Eurydice (2011). Key Data on Learning and Innovation through ICT at Schoolin Europe. https://www.csee-etuce.org/images/attachments/ictkeydata_on_ learning_and_innovation_through_ict_2011_summary.pdf. Last accessed 6 June 2019
5. Wing, J. M.: Computational Thinking. Communications of the ACM (49/3), 33-35(2006), http://www.cs.cmu.edu/~wing/publications/Wing06.pdf. Last accessed 13 June 2019