Preface

This year, the 9th International Conference on Informatics in Schools: Situation, Evolution and Perspective – ISSEP 2016 was held together with the 11th Workshop in Primary and Secondary Computing Education – WiPSCE 2016, from 12th to 15th October. This co-location gave the opportunity to bring together both communities and get a broader dissemination of the results.

ISSEP focuses on educational goals and objectives of Informatics or Computer Science as a subject matter in primary and secondary schools (K-12 education) and their different realization in compulsory and voluntary courses. It provides an opportunity for researchers and educators to reflect upon the goals and objectives of the subject, its curricula and various teaching and learning paradigms and topics. This year, the works presented have raised the issue of the improvement of informatics knowledge, and what is particular interesting, the change of perception and of attitude towards informatics and/or Computer Science.


The conference received 50 submissions. Each submission was reviewed by at least three reviewers and evaluated with respect to its quality, originality and relevance to the conference. The committee accepted 21 submissions for the presentation at the conference. Out of the 21, 17 were chosen to be published in a volume of Lecture Notes on Computer Science (LNCS). The four other papers are published here.

Moreover 12 posters were presented during the conference. Authors had the opportunity to get feedback on their work not yet published. ISSEP has also the tradition to offer workshop, that are meeting points where learning activities, learning tools or systems, good practices, and lessons are presented and discussed with the audience. This year, four workshops were proposed. Short presentations of the posters and the workshops are published in these proceedings.

We want to thank once more the members of the Program Committee and the additional reviewers for their great work for the selection process. We would like to warmly thank Jan Vahrenhold, chair of WiPSCE, and the members of the organizing committee from the University of Münster who managed to make the co-hosting of both conferences a real success.

September 25th, 2016
Andrej Brodnick and Françoise Tort, Chairs
Conference Organization

Program Committee

Marc Berges  University of München, Germany
Miles Berry  University of Roehampton, London, UK
Andrej Brodnik (Chair)  University of Ljubljana and University of Primorska, Slovenia
Javier Bilbao  University of the Basque Country (UPV/EHU), Spain
Andreas Bollin  University of Klagenfurt, Austria
Valentina Dagiene  University of Vilnius, Lithuania
Barbara Demo  University of Torino, Italy
Ira Diethelm  University of Oldenburg, Germany
Béatrice Drot-Delange  University of Clermont-Ferrand, France
Michalis Giannakos  Norwegian University of Science and Technology, Norway
Yasemin Gülbahar  University of Ankara, Turkey
Juraj Hromkovic  Swiss Federal Institute of Technology, Zurich, Switzerland
Ivan Kalas  UCL Institute of Education, London, UK - Comenius University, Bratislava, Slovakia
Peter Micheuz  University of Klagenfurt and Gynasium Völkermarkt, Austria
Christophe Reffay  University of Franche-Comté, France
Carsten Schulte  Freie Universität Berlin, Germany
Maciej Syslo  University of Wroclaw, Poland
Françoise Tort (Chair)  Ecole Normale Supérieure de Cachan, University of Paris Saclay, France
Mary Webb  King’s College London, UK

Poster

Georges-Louis Baron  Université Paris V René Descartes, France

Workshops

Peter Micheuz  Alpen-Adria-Universität Klagenfurt, Austria
**External Reviewers**


**Local Organization**

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holger Danielsiek</td>
<td>Westfälische Wilhelms-Universität Münster, Germany</td>
</tr>
<tr>
<td>Dana Glasmeyer</td>
<td>Westfälische Wilhelms-Universität Münster, Germany</td>
</tr>
<tr>
<td>Jan Vahrenhold</td>
<td>Westfälische Wilhelms-Universität Münster, Germany</td>
</tr>
<tr>
<td>Mirko Westermeier</td>
<td>Westfälische Wilhelms-Universität Münster, Germany</td>
</tr>
</tbody>
</table>

**Sponsoring Institution**

Ecole Normale Supérieure de Cachan - University of Paris - Saclay, France  
Westfälische Wilhelms-Universität Münster, Germany
Published papers:

- Khaled Asad - Teaching Computer Image Processing Subject to Middle School Students: Cognitive and Affective aspects
- Erik Barendsen and Tim Steenvoorden - Analyzing Conceptual Content of International Informatics Curricula for Secondary Education
- Valentina Dagiene and Sue Sentance - Its computational thinking! Bebras tasks in the curriculum
- Peter Hubwieser, Elena Hubwieser and Dorothea Graswald – How to Attract the Girls: Gender-Specific Performance and Motivation in the Bebras Challenge
- Dan Lessner - Attitudes towards Computer Science in Secondary Education: Evaluation of an Introductory Course
- Ebrahim Rahimi, Erik Barendsen and Ineke Henze -Typifying Informatics Teachers’ PCK of Designing Digital Artefacts in Dutch Upper Secondary Education
- Gabriele Stupuriene, Lina Vinikiene and Valentina Dagiene – Students’ Success in the Bebras Challenge in Lithuania: Focus on a Long-Term Participation
- Jiri Vanicek - What Makes Situational Informatics Tasks Difficult?
- Erik Barendsen, Natasa Grgurina and Jos Tolboom - A New Informatics Curriculum for Secondary Education in The Netherlands
- G. Barbara Demo - And Now What Do We Do with Our Schoolchildren?
- Natasa Grgurina, Erik Barendsen, Bert Zwaneveld, Klaas van Veen and Cor Suhre – Defining and Observing Modeling and Simulation in Informatics
- Hai Hong, Jennifer Wang and Sepehr Hejazi Moghadam - K-12 Computer Science Education Across the U.S.
- Juraj Hromkovic, Tobias Kohn, Dennis Komm and Giovanni Serafini - Combining the Power of Python with the Simplicity of Logo for a Sustainable Computer Science Education
- Natasa Mori and Matija Lokar - A new Interactive Computer Science Textbook in Slovenia
- Gabriel Parriaux and Jean-Philippe Pellet - Computer Science in the Eyes of Its Teachers in French-Speaking Switzerland
- Ira Diethelm and Melanie Schaumburg – IT2School – Development of Teaching Materials for CS Th rough Design Thinking
- Zsuzsanna Szalayne Tahy and Zoltan Czirkos – “Why Can’t I Learn Programming?” The Learning and Teaching Environment of Programming
# Table of Contents

## Papers

A National Engagement Model for Developing Computer Science Education in Wales ................................................................. 1  
*Tom Crick and Faron Moller*

Informatics at Secondary Schools in the French-Speaking Region of Belgium: Myth or Reality?  ................................................. 13  
*Julie Henry and Noemie Joris*

An Exploratory Investigation of Change in Students’ Subjective Perception of Informatics ......................................................... 25  
*Claudio Mirolo*

Bringing CS Innovations to the Classroom Using the Model of Educational Reconstruction ...................................................... 31  
*Mareen Przybylla, Andreas Grillenberger and Ralf Romeike*

## Posters

Teachers’ Attitudes towards CS High School Curriculum in Israel and Lithuania ................................................................. 41  
*Tamar Benaya, Valentina Dagiene and Ela Zur*

Cultivating Teachers Ability to Conduct Maker Activities ................................................................. 43  
*Ying-Jyun Chen and Greg C Lee.*

From Greenfoot to Java: Study of Students’ Object-Oriented Design Ability and Java Programming Skills ................................................................. 45  
*Jia-Yi Chen and Greg C Lee.*

Learning Informatics: Activities-based Model ................................................................. 47  
*Valentina Dagiene, Lina Vinikiene and Gabriele Stopuriene*

A Programming Circus for Primary Schools ................................................................. 49  
*Katharina Geldreich, Alexandra Funke and Peter Hubweiser*

A Playful Tool to Introduce Lower Secondary School Pupils to Recursive Thinking ................................................................. 51  
*Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo and Manuel Previtali*

Bridging Motivation Gaps with Physical Computing in CS Education ................................................................. 53  
*Mareen Przybylla, Patrick Israel, Julia Streichert and Ralf Romeike*

A Teaching Maturity Model for Informatics Teachers in Primary and Secondary Education ................................................................. 55  
*Elisa Reci and Andreas Bollin*

Learning of Robotics in Primary School, What Teaching Aid for Which Activities? ................................................................. 57  
*Micheal Spach, George-Louis Baron, Francois Villemonteix*
Uniting Computational Thinking Problem Solving Strategies with MIT Inventor ................................................................. 60

Bernhard Standl

Computer Science in the School Curriculum: Issues and Challenges .......... 64
Mary Webb, Tim Bell, Niki Davis, Yaacov J. Katz, Nicholas Reynolds,
Dianne P. Chambers, Maciej M. Syslo, Andrew Fluck, Margaret Cox,
Charoula Angelivalanides, Joyce Malynsmith, Joke Voogt, Jason
Zagami, Peter Micheuz, Yousra Chtouki and Natasa Mori

Computer Science in High school: Girls' Attitudes towards Gender Separation ............................................................... 67

Doron Zohar, Tamar Benaya and Ela Zur

Workshops

artEoz – Dynamic Program Visualization ........................................... 70

Martine Gautier and Brigitte Wrobel-Dautcourt

IT2School Project .............................................................................. 72
Ira Diethelm and Melanie Schaumburg

Physical Computing for Novices: Using the TinkerKit with Snap4Arduino .... 74
Mareen Przybylla

From Scratch to Patch: a "hands-on" Workshop .................................. 76
William Robinson

Sprego-Unplugged Functional Programming ..................................... 78
Maria Csernoch

Guess the Code ................................................................................. 81
Zsuzsanna Szalayné Tahy
A National Engagement Model for Developing Computer Science Education in Wales

Tom Crick\textsuperscript{1} and Faron Moller\textsuperscript{2}

\textsuperscript{1} Department of Computing & Information Systems
Cardiff Metropolitan University, UK
tcrick@cardiffmet.ac.uk

\textsuperscript{2} Department of Computer Science, Swansea University, UK
f.g.moller@swansea.ac.uk

Abstract Computer science education in the United Kingdom has undergone substantial scrutiny, and in England a new computing curriculum has just been introduced. However, in Wales – a devolved nation within the UK – political, geographical and socio-technical issues have hindered any substantive educational policy or curriculum reform for computer science over the past ten years. In this paper we present the activities of Technocamps, a university-based schools outreach programme founded in 2003 and its wider impact on computer science education and teachers in Wales. Furthermore, with imminent curriculum reform, we frame the wider opportunity for sustainably embedding both high-value digital competencies and computer science education – as well as changing the wider public perception and importance of computer science – as a prospective replicable case study of a national engagement model for countries with similar aspirations of becoming digitally confident and capable nations.

1 Introduction

In the 1980s, computer studies was a popular subject in schools across the UK. The ubiquitous presence of the popular BBC Micro\textsuperscript{3} – which was of little practical use unless you were able to program – saw a large proportion of school children learning the fundamentals of programming in a curriculum which included a variety of complementary topics such as hardware, software, Boolean logic and binary number representation.

By the 1990s, the emergence of pre-installed software – specifically office productivity software such as word processors and spreadsheet programs – meant that computers were no longer predominantly machines that needed to be programmed in order to do anything useful or interesting. Less and less time was being spent in the computer studies classroom on thinking about and writing programs, as basic digital literacies and IT user skills became regarded as the priority. However, as interest in viewing the computer as a creative tool waned in

\textsuperscript{3}http://en.wikipedia.org/wiki/BBC_Micro
favour of using it for more mundane tasks, various problems were being created, which were highlighted in two independent national enquiries in 1997. Both reports concluded that “Information Technology” in UK schools was in a primitive state and in need of attention and major investment. In line with the Stevenson Report, computer studies evolved into a new subject whose name was coined in that same report: Information and Communications Technology (ICT). Over the decade starting in 1997, the UK Government invested over £3.5bn in ICT in schools through various initiatives such as the National Grid for Learning and the New Opportunities Fund [1].

By 2000, ICT had permeated both primary and secondary school curricula. The emphasis was on developing the children’s IT skills and digital literacy in an honest attempt to address the increasing need for digital competencies amongst the general public. However, despite enormous government-funded ICT initiatives, various reports throughout the decade identified problems with implementing government policy on ICT educational reform [2–6]. Younie [7] summarises the problems identified by these reports into five key areas, including management, teacher training and competence, as well as impact on pedagogy. The ICT curriculum in Wales [8], while generally viewed to be more flexible and less prescriptive than the equivalent subject in England, exhibited many of the same issues [9, 10]. A full two-thirds of ICT teachers in the UK do not have a relevant qualification but may have moved into the role of ICT teacher simply by being sufficiently digitally literate [11]. The situation is worse in Wales, where this figure rises to 75% [12], with ICT perceived to be a low-priority discipline in many schools. Applications to study computer science at university slumped in the early part of the millennium – especially amongst females – and many of those who started a university computer science degree course found themselves dropping out during the first year, surprised at what computer science is and what studying it entails.

A decade later, two high-profile policy reports – one by Nesta [13], the UK’s innovation charity and one by the Royal Society [11], the UK’s premier science academy – made the very same observations. The report noted that ICT suffers from a poor reputation amongst pupils, parents and industry, who consider it dull and unchallenging and hence a low-value discipline, especially compared to other strategically-significant STEM subjects. With ICT embedded across the primary school curriculum, secondary school pupils found ICT in secondary school neither stimulating nor engaging [14]. The 2011 Wolf review [15] of vocational education for 14-19 year-olds in the UK further notes that the undemanding nature of ICT qualifications in secondary schools is readily exploited by schools: due to a disproportionately high national league table weighting associated with vocational qualifications, easily-achieved high results in ICT offer a welcome boost to a school’s league table position. Furthermore, as ICT is typically presented by schools as their “Computing” offering, students who might otherwise enjoy studying computer science are actively put off from what they are incorrectly but innocently led to believe is computer science [16, 17].
Technocamps\textsuperscript{4}, a university-based schools outreach programme based at Swansea University, was founded in 2003 to address these emerging problems in Wales. We have previously discussed its portfolio of activities in more detail \[18\]; in this paper, we consider the wider impact of the Technocamps project and its potential replicability as a case study of a national engagement model for other countries and regions. We evidence this through the consideration of its measurable effect on schools, teachers and pupils, contextualised by emerging educational (and economic) policy change in Wales, particularly with respect to reform of computer science and digital competencies.

2 The UK’s Four Education Systems

The UK consists of four nations ruled by one parliament, with an overall population of 64.5 million: England (population: 54.3 million), Scotland (5.3 million), Wales (3.1 million) and Northern Ireland (1.8 million) \[19\]. In 1997, Scotland and Wales held referendums which determined in both cases the desire for self-government. In the case of Wales, this led to the Government of Wales Act 1998 which created the National Assembly for Wales, to which a variety of powers were devolved from the UK parliament on 1 July 1999 (and again with the Wales Act 2014). In particular, education – which until then was a UK-wide government portfolio (minus Scotland, which for historical reasons has had a distinct legal and education system from England and Wales) – came under the control of the National Assembly for Wales.

Wales is a small nation to the west of England with an ancient Celtic culture and a thriving separate language (with c.20% of the population able to speak Welsh, a member of the Brythonic branch of the Celtic languages). Its south coast became pre-eminent during the Industrial Revolution due to coal mining and heavy industry; however, Wales is mostly rural and suffers from post-industrial poverty, seasonal employment and the dependence on the public sector for a significant proportion of jobs. The country is sparsely populated with resilience and interconnectedness of the transport infrastructure an issue. Hence its communities – specifically its schools and teachers – suffer from the perils of isolation, like other countries actively addressing the technology skills gap (such as New Zealand \[20\], Sweden \[21\] and Israel \[22\]). Apart from the south east corner (including its capital Cardiff) and the regions bordering England, the rest of the country is formally designated by the EU as a so-called “Convergence area”, meaning its per-capita GDP is less than 75% of the EU average.

Politically, Wales became a devolved nation within the UK in 1999. Prior to this, the education system in Wales was essentially identical to that in England and was in a healthy state, outperforming other regions in the UK in the years prior to and immediately following devolution. However, since devolution saw the education portfolio transferred to the National Assembly of Wales, it has suffered a rapid decline. Evans \[23\] presents a detailed analysis as to the cause

\[\text{http://www.technocamps.com}\]
of this, citing a multitude of policy changes and poor interventions, alongside a hard-hitting report from the OECD [24].

Whilst broadly maintaining the general educational system used in England, the Welsh Government embarked on a 10-year revolutionary plan including introducing the Welsh Baccalaureate\(^5\), an overarching qualification, with a purely practical-based assessment, incorporating transferable skills useful for higher education and employment, as well as explicitly using education as a lever to tackle socio-economic deprivation. Much of this plan was widely lauded by key stakeholders, being learner-focused and practitioner-led, placing an emphasis on skills development and ensuring that it is appropriate for the specific needs of Wales. However, since its implementation, it has been criticised for various reasons and by various stakeholders. The then Minister for Education and Skills appointed in June 2010, in looking for the causes of Wales’ failing education system, found cause to commission no fewer than 24 reviews before his resignation in February 2013 – almost one per month [23].

With devolved government comes fiscal autonomy; and the correlation between money and performance is an obvious target for critics, who point to a growing spending shortfall between Wales and England. The average spend per pupil in Wales in 2000-2001, just after devolution, was more than every region of England apart from the large metropolitan areas of London, the West Midlands and the North West, all of which benefit from their vast economies of scale. However, since then, the gap between the education budgets per pupil between Wales and England has steadily grown by about 1% per year.

When establishing a model for viewing school computer science education, it is apparent that there is substantial diversity between school education systems [25], and this can create obstacles when trying to understand progress made in one country and potentially replicate it in another [26]; this is also pertinent to the devolved (and diverging) educational systems of the UK.

\[3\] The Technocamps Initiative

Since 2000, Swansea University (as elsewhere across the UK) suffered a steady decline in the number of students enrolling in computer science, with the worst effect on the already-dwindling numbers of female students. In an attempt to address this worrying anomaly, the University reached out to local secondary school ICT teacher. However, there was positive resistance; for reasons explained later which did not apply to teachers in England, teachers in Wales felt overburdened and disinterested in exploring any perceptions of inadequacy in the curriculum and their delivery [16, 17].

As it appeared to be futile to influence schools and their ICT teachers directly, Technocamps was created in 2003 to promote computing amongst their pupils. This was a programme of engaging interactive computational workshops taking place on the university campus whose ultimate aim was to subtly re-introduce computer science into the ICT curriculum by generating the demand

\(^5\) [http://www.welshbaccalaureate.org.uk/](http://www.welshbaccalaureate.org.uk/)
from the students. Originally run only at Swansea University, Technocamps hubs have since been created at most universities throughout Wales, offering wide geographical coverage.

Teachers in Wales were happy to “treat” their classes to these “day out” activities; but they were then faced with the prospect of satisfying their pupils’ newly-discovered passion for computing, programming and computational thinking by introducing “Technoclubs” as lunch-time extra-curricular activities in the school. With generous help, resources and guidance from Technocamps – along with the fact that in many cases students appeared to be more technically informed and digitally literate than their teachers – these clubs have flourished, and the impact of Technocamps in changing attitudes in Welsh schools regarding ICT and computing has been widely acknowledged, both by the Welsh Government, as well as the teaching community in Wales. The wide spectrum of Technocamps activities is presented in further detail in [18]; here we assess its wider impact.

3.1 Measuring Impact: Wales Divided

In 2010, based on long-term empirical data regarding its effect on school children’s attitudes towards computer science and technology careers – as well as their teachers’ – Swansea University was awarded £3.9 million funding towards a £6 million four-year project (with the remaining £2.1 million generated through matched funding from the university) by the Welsh Government under the EU’s European Social Fund (ESF) Convergence Programme to run Technocamps with regional hubs at the Universities of Aberystwyth, Bangor and South Wales. Due to EU restrictions, Technocamps was prohibited from providing any support (specifically, resources for workshops, teacher sessions, Technoclub support, etc) to schools outside of the Convergence area – namely, the eastern region of Wales, including its capital city Cardiff, bordering England. Whilst an unfortunate artefact of the funding, a fortuitous side effect of this restriction was that it allows for a true assessment of the real impact of Technocamps, as the nation was invariably divided into two halves: West Wales received the full Technocamps experience, whilst East Wales (including its capital, Cardiff) did not.

Cardiff is the primary base of Computing At School (CAS) in Wales; CAS have been widely recognised for their role in reform of the Computing curriculum in England [27]. Since 2010, Technocamps has supported CAS in promoting their teacher-led initiatives (specifically the local/regional CAS Hub model and the CAS Network of Excellence [14,27,28]). In particular, in 2010 Technocamps and CAS jointly sent out an information pack to every secondary school in Wales, following similar initiatives in England and Scotland. Technocamps produced the packs and posted these out to all of the schools; CAS Wales provided the costs for sending the information packs to the schools outside of the Convergence area of Wales (in 2012, CAS Wales was awarded a grant of c. £70,000 from the Welsh Government to support the development of the CAS Network of Excellence model of teacher-led activity across Wales, supplementing the several millions of pounds granted to CAS by the UK Government for this activity across
England). The information pack included full details of the extensive resources being supplied on the Technocamps and CAS websites, which schools and teachers could freely download and use, in particular in support of extra-curricular computing clubs.

Despite the non-recurrent financial support of CAS Wales, and the support it offers teachers in Wales, the CAS model [28] – so successful in populous and geographically dense England – has never proven successful in Wales. For example, whilst CAS Hubs across the UK are generally run by schools for schools, abiding to the principle of the teacher-led initiative, virtually all of the CAS Hubs across Wales are led by university academics who also run Technocamps Hubs. Teachers have generally not been as self-organising in Wales compared to England to promote the wider CAS agenda to support curriculum reform and building a teacher-led community.

In contrast to this, an independent review [29] of Technocamps activity in the (socio-economically disadvantaged) Convergence region of Wales carried out for Welsh Government estimates that 5% of its secondary school-aged youths engaged with Technocamps through Workshops, and that more than a quarter of the secondary schools in the region have established Technoclubs. Furthermore, the new GCSE and A-Level Computer Science qualifications (taken at 16 and 18 respectively) – which has had patchy uptake in Wales due to the lack of curriculum reform – are now starting to be adopted by an increasing percentage of these schools, whilst schools outside of the Convergence area (and outside the reach of Technocamps) continue to deliver the ICT curriculum. Although it could not operate within the non-Convergence area of Wales, Technocamps promoted all of its extensive on-line resources which are freely available to schools outside the Convergence area of Wales, and supported the activities of CAS Wales to develop the CAS Network of Excellence model of teacher-led school-based activities throughout Wales. However, despite the sustained efforts of CAS Wales, there are very few active and sustained school-based computing clubs that are not inside the Convergence area and established due directly to Technocamps workshops and follow-up engagement.

In further support of this claim, consider the following example: the Annual Technocamps Robotics Competition has been open to all schools across Wales, promoted across all of Wales through Technocamps and CAS Wales networks, and even held on the outskirts of Cardiff in 2013. However, every single one of the 43 teams entered in the 2013 competition held near Cardiff travelled in from a Convergence area Technoclub formed on the back of Technocamps workshops and follow-up engagements with Technocamps initiatives. This provides clear evidence that the Technocamps model of intense direct engagement through campus-based workshops, in conjunction with teacher CPD and support, is crucial for success in promoting uptake of the discipline of computer science. The lack of confidence and isolation felt by the teacher community in Wales means that computing clubs have only arisen – and will likely only continue to develop – through direct involvement of and engagement with Technocamps.
3.2 Teacher Impact

In Spring 2015, as part of the Welsh Government’s *Learning in Digital Wales* programme, an anonymous on-line survey was carried out. A link to the survey was sent out to head teachers and ICT/Computing subject head teachers in every Secondary School across Wales. The survey set out to measure the extent to which schools and teachers: understood the (need for) proposed changes to the computing curriculum; felt the need for support to face these changes; and recognised the various organisations that were providing such support.

Responses to the survey were submitted from over a third of such schools, and these depict Technocamps in a particularly positive light. In particular, only one respondent claimed to be unaware of Technocamps, whereas over 85% of respondents were not only aware of Technocamps but were actively benefitting from its various activities. In contrast, only 60% were aware of and benefitted from CAS, whilst 19% were unaware of CAS. The lack of awareness and benefits of CAS is due, in no small part, to the Anglo-centric nature of CAS. However, even flagship facilities created by the Welsh Government’s Department of Education and promoted heavily within schools were not as well regarded: whilst every respondent was naturally aware of its online digital portal *Hwb*[^6], only 57% benefit from it; and a full 24% unaware of their regional educational consortium with only 51% benefitting from it.

3.3 Government Impact

The impact described above that the various Technocamps initiatives have had on changing perceptions in schools (both pupils and teachers) has also translated into impact on Welsh (and UK) Government thinking and policymaking. For example:

– In his speech at the 2012 Annual Technocamps Teachers’ Conference[^7], the Welsh Government’s Minister for Education and Skills acknowledged the importance of computer science education for all and how it addressed the key educational priorities in Wales, noting in particular the wide impact of Technocamps on pupils and schools; and expressed understanding of the wider educational and socio-economic impact that the government can make with educational reform in Wales. He also announced a variety of funded initiatives to support Technocamps’ aims of embedding computing within the school curriculum at all levels.

– One of the initiatives the Minister announced in his speech was the creation of a government oversight panel – the National Digital Learning Council (NDLC)[^8] – which would work on scoping the way forward for his department’s ICT strategy; and in his speech he appointed the Technocamps Director as an Expert Adviser to this panel.

[^8]: [https://hub.wales.gov.uk/pages/Community-NDLC](https://hub.wales.gov.uk/pages/Community-NDLC)
In 2013, the Minister commissioned an independent Review of the ICT Curriculum, citing the impact of Technocamps with its Director included amongst its members.

- Technocamps has been recognised by the UK Government as the driving force for computing education in Wales, through an invitation to appear at the Houses of Parliament in October 2014, hosted by the Chair of the House of Commons Science and Technology Select Committee.

- Technocamps’ impact on schools in the Convergence area of Wales has been recognised by the Department for Education and Skills (DfES) which has contracted Technocamps to deliver workshops at every state-sponsored secondary school throughout the whole country between September 2014 and March 2016 as part of their Learning in Digital Wales programme.

- Technocamps’ impacts on teachers has been recognised by the Department for Economy, Science and Transport (DEST), through the National Science Academy (NSA), which has contracted Technocamps to deliver teacher training between April 2015 and March 2018.

- Technocamps’ impact on primary schools has also been recognised by DEST, through the NSA, which has contracted Technocamps to deliver its Playground Computing programme between April 2015 and March 2018.

- In the run up to the May 2016 Welsh Assembly (devolved government) elections, Technocamps was cited as the cornerstone of developing digital skills in Wales in the UK national press\(^9\), heralding the importance of these skills for the economic future of the nation.

4 Education Policy Change in Wales

In light of the perceived failings within education in Wales there have been a number of reviews commissioned over the past five years to identify failures and make recommendations to rectify these; we reflect on two recent major reviews which are particularly pertinent to computing education.

ICT Curriculum Review (2013): In January 2013, the Minister for Education and Skills announced the formation of an ICT Steering Group to consider the future of computer science and ICT in schools in Wales. Its remit was to explore the issues that ICT in schools needed to be re-branded, re-engineered and made relevant to now and to the future; computer science should be introduced at primary school and developed over the course of the curriculum so that learners can progress into a career pathway in the sector; skills, such as creative problem-solving, should be explicitly reflected in the curriculum; with revised qualifications to be developed in partnership with schools, higher education and industry. It was initially envisaged to report back in Autumn 2013, with its

\(^9\) e.g. http://www.huffingtonpost.co.uk/carwyn-jones/skills-for-the-jobs-of-today-and-tomorrow_b_9767130.html
recommendations informing the wider review of assessment and 14-19 qualifications, with any necessary changes being considered as part of any revisions to the National Curriculum in Wales.

The ICT Steering Group published its recommendations [30] in October 2013, highlighting the importance of computing and digital literacy in a modern, challenging and aspirational national curriculum. Its headline recommendations were grouped into three main themes: curriculum and qualifications; teacher training and professional development; and infrastructure and monitoring. The report recommended that ICT be replaced from Foundation Phase onwards by a new subject named Computing. This subject would disaggregate into two main disciplines: Computer Science (CS) and Information Technology (IT); this new subject should be integrated into the curriculum as the fourth science, served by a mandatory programme of study, and receive the same status as the other three sciences. It recommended a clear distinction between cross-curricular digital literacies and the academic discipline of computing by proposing a statutory digital competency framework to work alongside existing frameworks for literacy and numeracy from Foundation Phase through to post-16 education. There was also a strong focus on supporting the ICT teaching profession in Wales, particularly around initial teacher education and incentivising routes into the profession, as well as raising the profile and importance of career-long professional development and in-service training.

In the context of the recently announced new Computing curriculum in England, the ICT Steering Group’s report was well-received, addressing the specificity of the educational challenges in Wales, as well as providing a broad and balanced curriculum, from digital competencies through to computer science. While aspects of the recommendations around digital competencies had been accepted, everything relating to curriculum and qualifications was preempted by the announcement in March 2014 of an independent review to provide recommendations to inform the development of a new Curriculum for Wales.

Independent Curriculum for Wales Review (2015): In March 2014, Professor Graham Donaldson, a former chief school inspector in Scotland, was appointed by the Welsh Government to conduct an independent review of curriculum and assessment of the entire curriculum in Wales. This continued on from a number of previous national consultations and reviews, such as the 2011-2012 Review of Qualifications for 14- to 19-year-olds in Wales (which aimed to ensure that qualifications in Wales are understood and valued and meet the needs of young people and the Welsh economy), as well as aggregating a number of independent subject-specific reviews, including the 2013 ICT review.

The Donaldson report (“Successful Futures”) [31], was published in March 2015 and proposed profound changes to the education system in Wales. While identifying strengths in the current education system, for example the early years Foundation Phase and the commitment to the Welsh language and culture, the report identifies significant shortcomings of the current curriculum arrangements, which essentially remain as devised in 1988 (when it shared a
national curriculum with England). The report argues that the curriculum has become overloaded, complicated and, in many parts, outdated. It identifies four purposes for the curriculum, recommending that the entirety of the school curriculum should be designed to help all children and young people to become: ambitious, capable learners, ready to learn throughout their lives; enterprising, creative contributors, ready to play a full part in life and work; ethical, informed citizens of Wales and the world; and healthy, confident individuals, ready to lead fulfilling lives as valued members of society. There are a number of similarities to Scotland’s *Curriculum for Excellence*, of which Donaldson was also involved.

With respect to computing education and the role of technology, the review identifies three cross-cutting, whole-schools “collective responsibilities”: literacy, numeracy and digital competencies. With the structure of Foundation and Key Stages disappearing, individual curriculum subjects would be replaced with six “areas of learning and experience”, in which subjects should “service the curriculum but not define it”. All teaching and learning would be directed to achieving the four curriculum purposes.

The Donaldson review recognises and adopts many of the recommendations of the 2013 ICT review, recognising the importance of separating digital competencies from the curriculum subject of computing, but providing clear pathways as well as significant opportunities for cross-curricular learning across science and mathematics. Computer science would thus sit within a new Science & Technology area, with a clear strand of learning from aged five through to qualifications at 16 and 18. Furthermore, it recommends a programme of professional learning to be developed to ensure that the implications of the review for the skills and knowledge of teachers are fully met, although no timescale for delivery were proposed (due to the required legislative changes). This curriculum review was cautiously well-received by the education community and the media in Wales, with significant detail remaining to be seen in implementation, resourcing and timescales. The publication of the Donaldson curriculum review was quickly followed by a review of initial teacher education in March 2015, alongside the Welsh Government’s “New Deal” for the Education Workforce, complementing the outcomes from the previous reviews, to reshape continuing professional development for teaching professionals to support them in shaping and delivering the new curriculum going forward into 2016.

5 Conclusions

As we have presented here and previously [18], Wales is at the cusp of implementing significant educational reform, with strategic importance given to digital competencies and computer science. The May 2016 devolved government elections have elected a new Welsh Government with refreshed ministerial portfolios to shape education and skills policy, as well as policy related to the digital economy for our prospective “agile digital nation”. We may see a number of testbed initiatives and activities useful to other nations reforming their curricula, especially in the context of high-value digital competencies. However, there remain
significant challenges, particularly around wider public perceptions of the disciplines and its inherent educational and economic value, and how to upskill the entire teaching community of Wales. This is the profound and long-term challenge that has to be recognised and addressed before we see the type of computer science education that is fit for purpose and does not actively dissuade students from progressing onto degree-level study or opting for diverse careers in the technology profession.

In England, despite the presence of a critical mass of computing teachers mobilised by the successful CAS initiative, there was still a profound and disruptive shift in attitude felt in the teaching community once the UK Government formally announced the new Computing curriculum would be introduced from September 2014. This momentum does not currently exist in Wales, and it is even more critical for the Welsh Government to influence the teaching community through its policy interventions. Furthermore, any new initiatives in this space have to address local/regional needs, but with strategic coordination at the national level; the previous funding model of Technocamps has clearly had an impact on engagement, upskilling and the wider perception change in the non-Convergence area of Wales.

Technocamps has been working through its Technoteach programme to create a small but critical mass of qualified teachers, necessarily through a programme of direct and intense intervention. Public pronouncements from Welsh Government regarding its intentions to follow England in fully adopting computer science education in schools will be needed to secure the schools’ buy-in to teacher CPD in readiness for the new curriculum. The Technoteach model of direct intervention will clearly remain necessary for some time after such government declarations; but in the fullness of time, and with a growing community of confident teachers, we will eventually arrive at a situation in which the teacher-led CAS model will be as effective in Wales as it has been in England. Furthermore, this hybrid practitioner model of “pioneers” and master teachers cascading best practice may be of relevance to other disciplines (such as mathematics and the sciences), as well as other nations reforming their computing and technology curricula.

References

Informatics at secondary schools in the French-speaking region of Belgium: myth or reality?

Julie Henry¹ and Noémie Joris²

¹ PRéCISE, Faculté informatique - Université de Namur (Belgique)
julie.henry@unamur.be

² CRIFA, Faculté de Psychologie, Logopédie et des Sciences de l’Éducation -
Université de Liège (Belgique)
oeemie.joris@ulg.ac.be

Abstract. In this article, a relation is made between: (1) the informatics skills that secondary students can (ideally) acquire (on the basis of curriculums), (2) the skills they possess after having passed the secondary education and (3) the topics actually taught in school, described by teachers themselves. These data offer an overview of the informatics place in secondary school in the French-speaking region of Belgium which is, in many respects, subjects to improvement.

Keywords: informatics curriculums content, informatics teachers’ viewpoint, informatics skills of university students, informatics in secondary education in Belgium

1 Introduction

If teaching informatics makes sense within the professional community, the non-initiated persons interpret it by many ways: discovering digital culture, mastering software, using technology in the classrooms, teaching about the basic functions of a computer and how to use it, coding, programming and so on.

However, the professional community agrees to say that informatics is now completely dissociated from the computer. It speaks about computational thinking (Wing, 2006; Wing, 2011). Before using a computer to solve a problem, the problem itself and the ways in which it could be resolved must be understood. Looking up a name in an alphabetically sorted list, cooking a gourmet meal, doing laundry or planning a route on a map: computational thinking helps us with all these tasks.

While its European neighbors (France, England…) have realized the importance of teaching it for some years, computational thinking is not present in curriculums in the French-speaking region of Belgium. Fortunately, informatics curriculums are offered to students in secondary schools. But the related courses are mostly optional and the taught topics are often software-related ones. In the same way, the current situation of K-12 informatics education in Flanders is also very chaotic: no informatics curriculum, optional courses and software-related topics.
In 2013, the informatics background of francophone students completing their secondary cursus was measured (Henry & Joris, 2015). This article proposes to complement these measures by adopting the posture of teachers. Who are the teachers of informatics courses? Which informatics do they teach?

The structure of the Belgian education is similar to that of many European countries. However, a summary of this structure will be made in section 2 to facilitate the reading of this article. The section 3 proposes a quick reading of the informatics curriculums. Topics covered in these programs will help to define a theoretical informatics background. In the section 4, the informatics background of students will be discussed. These results will be compared to informatics topics actually taught by teachers in classrooms. Last but not least, as this article should be considered as a draft definition for informatics education in Belgium, some lines of action will be discussed as outlook in section 5.

2 Education structure in Belgium

The Belgian school has a three-tier education system, with each stage divided into various levels: basic education including nursery school (children aged 3-6) and primary school (children aged 6-12), secondary education (children aged 12-18) and post-secondary education organized by universities or schools of higher education.

In Belgium, secondary education includes three cycles (usually abbreviated D1, D2 and D3). Each cycle lasts two years.

At the end of primary school, the pupils have to obtain their “Basic Education Degree”. If they succeed, they go to the “common D1” (1C-2C). This first cycle provides a broad general basis, with only a few options to choose from.

In the D2 and D3 levels, secondary school is divided into four types: general secondary education (GSE), technical secondary education (TSE), art secondary education (ASE) and vocational secondary education (VSE). TSE and ASE are divided into two groups of education: one focuses more on technical aspects (TSET, ASET) while the other focuses more on practical matters (TSEP, ASEP). Each type consists of a set of different directions that may vary from school to school. Its basic structure includes four components: a common basis, a mandatory option, a free option and a strengthening. The importance attached to each component depends on the type and not all schools offer every type. In this article, we wouldn’t take ASET, TSEP, ASEP and VSE into account because of the specificities of their curriculums.

In the French-speaking region of Belgium, schools are divided into three groups: the schools owned by Wallonia-Brussels Federation (WBF), the subsidized official schools (SO) and the subsidized free schools (SF).
Table 1. Secondary ordinary education system in the French-speaking region of Belgium

### Informatics in curriculums

Each of the three groups of schools proposes some curriculums in informatics. For all of these programs, a “content inventory” is provided below.

For easier comparison between the different curriculums, a standardized vocabulary and an abbreviation system were chosen to describe the topics mentioned inside these programs.

<table>
<thead>
<tr>
<th>Computer hardware</th>
<th>HW</th>
<th>Operating system</th>
<th>OS</th>
<th>Information retrieval</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word processing app</td>
<td>WP</td>
<td>Spreadsheet</td>
<td>SS</td>
<td>HTML</td>
<td>HTML</td>
</tr>
<tr>
<td>Presentation app</td>
<td>P</td>
<td>Image processing app</td>
<td>IP</td>
<td>XHTML/CSS</td>
<td>XCSS</td>
</tr>
<tr>
<td>Sound/video processing app</td>
<td>SVP</td>
<td>Databases</td>
<td>DB</td>
<td>email app</td>
<td>@</td>
</tr>
<tr>
<td>Web browser</td>
<td>WB</td>
<td>Search engine</td>
<td>SE</td>
<td>Network</td>
<td>Net</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coding</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mathematical logic</td>
<td>ML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Algorithmic</td>
<td>Algo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Programming</td>
<td>Prog</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Informatics and Society</td>
<td>I&amp;S</td>
</tr>
</tbody>
</table>

Table 2. Abbreviation system used in this article

Once the content is identified within each curriculum, a "theoretical" comparison is made between them, as a basis for discussion with the results of this article.

### Wallonia-Brussels Federation schools.

Four curriculums are proposed in the WBF schools. “Introduction to informatics” (1) is organized in the D1 level as a mandatory option. This curriculum becomes a free option in the (GSE and TSET) D2 and D3 levels. “Informatics” (2) and “Computer science” (3) are related curriculums that are organized respectively in the TSET D2 and D3 levels as free options. Then, “Informatics for information management” (4) is organized in the GSE D3 level as free option.
Table 3. Content inventory of WBF schools curriculums

Except for sound/video processing application and XHTLM/CSS, all the topics are present in the WBF curriculums. However, the majority of them is reserved to D2 and D3 level. Students have more chance to learn informatics in the TSET type. Furthermore, word processing and email application are the most taught topics.

### 3.2 Subsidized official schools

Only one curriculum is organized in some SO schools (but not in all of them). It is a free option proposed in the TSET D2 and D3.

Table 4. Content inventory of SO schools curriculums

In the SO curriculum, some topics are never taught: sound/video processing appli-
cation, HTML, XHTML and Informatics and society. Informatics is only present in the D2 and D3 of the TSET type. So, a student in the GSE type of a SO school will never learn any informatics.

3.3 Subsidized free schools

In the SF schools, informatics is considered as an educational discipline. Three curriculums are organized as mandatory option: one, “Education by ICT: introduction to informatics” (1), in the D1 level and two, both called “Informatics”, respectively in the TSET D2 (2) and D3 (3) levels.

<table>
<thead>
<tr>
<th>1 (All types)</th>
<th>2 (TSET)</th>
<th>3 (TSET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2</td>
<td>D3</td>
<td>D3</td>
</tr>
<tr>
<td>HW</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OS</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WP</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SS</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WB</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SE</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 (All types)</th>
<th>2 (TSET)</th>
<th>3 (TSET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2</td>
<td>D3</td>
<td>D3</td>
</tr>
<tr>
<td>IR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HTML</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>XCSS</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Net</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Code</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ML</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Algo</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Prog</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I&amp;S</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

X Topic taught in a mandatory option curriculum

Table 5. Content inventory of SF schools curriculums

In the SF schools curriculums, all topics are taught. Some of them, such as image, sound and video processing applications, database and XHTML/CSS are only taught in the D3 level. The others, such as computer hardware, operating system, word processing application, web browser, search engine and information retrieval are taught in every levels. However, most of the topics are only proposed in the TSET type.

3.4 Theoretical informatics background

The perusal of the different curriculums and the identification of their proposed topics lead to define the informatics skills that secondary students can ideally acquire. These skills are different between the three groups and the type of secondary schools (GSE and TSET).
The informatics skills to be acquired in the GSE type are zero for the SO group (no course). In TSET type, if a student chooses all informatics courses proposed by her/his school, she/he could acquire a good level in informatics. Nevertheless, all schools are free to organize informatics courses or not. In the WBF schools, only 75 (out of 118 proposing the GSE type – 63%) organize informatics courses. Only 10 (out of 39 proposing the TSET type – 25%) organize informatics courses. In the SO schools, 10 (out of 45 proposing the TSET type – 22%) organize informatics courses. Finally, in the SF schools, only 22 (out of 124 proposing the TSET type – 17%) organize informatics courses. This means that it’s easy for a student to pass her/his entire secondary cycles without having any informatics. So her/his informatics background can be totally non-existent.

4 Informatics in the classrooms

The curriculums census only gives a vision of which informatics could be present in secondary schools in Belgium. It is necessary to compare this informatics to the
one really taught in classrooms. To achieve this goal, two surveys were conducted: (1) the first one with the freshmen at the University of Liege who have just passed the secondary level and (2) the second one with the informatics teachers of secondary schools.

4.1 Methodology

In 2013, a survey was sent by email to 950 freshmen (17.2%) at the University of Liege (Henry & Joris, 2015). The questions were about the trajectory of students into secondary education (group, type) and their informatics background (attended informatics courses, acquired topics).

The second survey was sent by email to two discussion lists: CoP-PR-TIC and vi-saTICE. These lists were created as part of informatics-related projects in the secondary education. The study conducted here was a prerequisite for further study. So these lists appeared to be quick ways to make contact with informatics teachers (without really knowing how many of them are on the lists).

The survey included between 13 and 18 questions about the teachers themselves (initial training, experience), the schools where they work (group; type, cycle) and their courses (content).

4.2 Results and discussion

4.2.1 Informatics background of students

170 students out of 950 (17.9%) responded to the survey. 145 only matched the desired criteria, i.e. secondary education in Belgium and a type (GSE or TSET) which proposes informatics courses.

Out of these 145 students, 56 (38.6%) followed informatics courses. We deliberately haven’t taken into account students repeating their first bachelor (in order to avoid confusion with the informatics courses offered at University). So our “with informatics courses” sample was finally composed of 39 students.

9 students were in a WBF school (sub-sample 1), 3 in a SO school (sub-sample 2), 21 in a SF school (sub-sample 3) and 6 had too specific trajectories (change of group, etc.) to be taken into account. All were in a GSE secondary education.

For each informatics proposed topic, students had to say “yes”, “no” or “I don’t know” (abbreviated by “?”) about their classroom acquisition. The results are then compared to the theoretical informatics background (TIB) of GSE type.

Regarding the sub-sample 1 results, the WBF curriculums seem to be partially used by teachers. 6 out of 11 topics included in the TIB are acquired by more than 50% of the students. It is principally software-related topics. The others (hardware, operating system, databases, HTML and email application) have less success.
The results of the subsample 2 demonstrate the existence of courses not related to a curriculum. As the WBF group, the most cited topics are software-related: word processing application and spreadsheet.

Table 9. Sub-sample 2: informatics background of students vs TIB for SO group of schools

Regarding the sub-sample 3 results, the (D1 level) SF curriculum seems to be almost totally used by teachers. 5 out of 6 topics included in the TIB are acquired by more than 50% of the students. Some additional topics even complete the TIB: spreadsheet, presentation and image processing applications. It could demonstrate the existence of courses not related to a curriculum.
In the three groups, algorithmic and programming, not included in the GSE TIB, are logically unknown for a majority of students.

Now consider students who had no informatics courses (89, 62.4%). 36 students (40.4%) were in the WBF group, 2 (2.2%) in the SO group, 46 (51.7%) in the SF group and 5 (5.7%) conducted their studies in several groups. No group is spared by the lack of an informatics education.

### 4.2.2 Topics taught by teachers

36 teachers responded to the survey. Only one teachers out of 36 worked in the D1 level. 20 teachers worked in the D2 and/or D3 levels, in GSE and/or TSET types. So 21 responses were taken into account. This sample is absolutely not representative of the entire informatics teachers’ population but it is a good opportunity to make a first analysis, especially for the GSE type secondary education.

Before talking about informatics topics sociodemographic profile of informatics teachers can be drawn. The majority of the teachers is male (65.2%) and is more than 41 years old (73.9%). The average years of experience is 17.3 (SD of 10.3). The average years of experience as informatics teacher is 15.1 (SD of 8.85).

In terms of initial education, a wide diversity can be observed. 13 informatics teachers (out of 21, 60.8%) have completed a “Hard Sciences” oriented curriculum: Engineering (5), Mathematics (2), Sciences (4) or Informatics (2). 7 teachers have completed a “Soft Sciences” oriented curriculum: Economics (4), Accounting (2) and Business studies (1).

The majority of the participants teaches in the SF group (17, 80.9%). Only 4 of them teach in the WBF group.

### Table 10. Sub-sample 3: informatics background of students vs TIB for SF group of schools

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>66.7</th>
<th>14.3</th>
<th>19</th>
<th></th>
<th>Net</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>23.8</td>
<td>42.9</td>
<td>33.3</td>
<td></td>
<td></td>
<td>Code</td>
<td>19</td>
<td>42.9</td>
<td>38.1</td>
</tr>
<tr>
<td>SVP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>ML</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBB</td>
<td>4.8</td>
<td>61.9</td>
<td>33.3</td>
<td></td>
<td></td>
<td>Algo</td>
<td>0</td>
<td>71.4</td>
<td>28.6</td>
</tr>
<tr>
<td>WB</td>
<td>52.4</td>
<td>28.6</td>
<td>19</td>
<td>X</td>
<td></td>
<td>Prog</td>
<td>19</td>
<td>61.9</td>
<td>19.1</td>
</tr>
<tr>
<td>SE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td></td>
<td>I&amp;S</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Out of the 4 teachers of the WBF group, no one teaches in the D1 level, one teaches in the GSE D2 level and 4 in the D3 level.

Regarding the results of the WBF group, the teacher of the D2 level uses the official curriculum (OC). The taught topics are then word-processing application, spreadsheet and presentation application.

In the D3 level, 3 teachers work in the GSE type. One doesn’t use a curriculum. The other ones use one or both of the provided official curriculum. Most of the topics are taught by more than 50% of teachers.

Out of the 17 teachers of the SF group, one teaches in the D1 level, 11 teach in the D2 level and 14 in the D3 level.

Regarding the results of the SF group, the teacher of the D1 level doesn’t work with the official curriculum and proposes 4 topics: hardware, word processing application and information retrieval (out of 6 topics included in the official curriculum) and email application.

In the D2 level, 7 teachers (out of 11) work in the GSE type. There is no informatics curriculum for this type. But 6 out of them use the official curriculum of the TSET type. The last one doesn’t use any curriculum. Only 2 topics are taught by more than 50% of the teachers. There are software-related topics.

<table>
<thead>
<tr>
<th>HW</th>
<th>OC</th>
<th>IR</th>
<th>OC</th>
</tr>
</thead>
<tbody>
<tr>
<td>66.6</td>
<td>X</td>
<td>66.6</td>
<td>X</td>
</tr>
<tr>
<td>OS</td>
<td>66.6</td>
<td>X</td>
<td>HTML</td>
</tr>
<tr>
<td>WP</td>
<td>66.6</td>
<td>X</td>
<td>XCSS</td>
</tr>
<tr>
<td>SS</td>
<td>66.6</td>
<td>X</td>
<td>@</td>
</tr>
<tr>
<td>P</td>
<td>66.6</td>
<td>X</td>
<td>Net</td>
</tr>
<tr>
<td>IP</td>
<td>66.6</td>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>SVP</td>
<td>33.3</td>
<td>ML</td>
<td></td>
</tr>
<tr>
<td>DBB</td>
<td>100</td>
<td>X</td>
<td>Algo</td>
</tr>
<tr>
<td>WB</td>
<td>66.6</td>
<td>X</td>
<td>Prog</td>
</tr>
<tr>
<td>SE</td>
<td>66.6</td>
<td>X</td>
<td>I&amp;S</td>
</tr>
</tbody>
</table>

Table 11. Topics taught in the GSE D3 level of the WBF group vs topics included in the two GSE official curriculums

The results appear more positive than those obtained with students three years ago. Not only software-related topics are taught. But there are few teachers expanding official curriculum topics. So, students passing through GSE type education of WBF group have little chance of acquiring algorithmic and programming skills.
In the D3 level, 8 teachers (out of 14) work in the GSE type. There is no informatics curriculum for this type. But 5 out of them use the official curriculum of the TSET type. The other ones don’t use any curriculum. Taught topics are software-related topics.

The existence of courses not organized by the official curriculums is confirmed. Teachers use curriculums of TSET type. But all topics of these programs aren’t taught. Once again, the emphasis is on software-related topics. So, students passing through GSE type education of SF group have also little chance of acquiring algorithmic and programming skills.

The teachers’ results point in the same direction that the ones of students. However, they are slightly positive, reflecting perhaps a slow changing situation.
5 Outlook

Informatics curriculums are not sufficient to ensure a good informatics background to students. Indeed, it depends a lot on teachers. They seem to be more attracted by the software-related topics. They forget the “computational thinking”-related topics as code, algorithmic and programming. But curriculums are often limited to methodological recommendations and general objectives. These ones don’t constitute sufficient tags to establish an efficient course. The majority of the teachers are not trained to teach computer (Henry & Joris, 2013). So it is understandable that they teach only the topics with which they feel comfortable. If training the teachers might be a solution, it’s not the only one. Teaching of computational thinking to kids, standardizing curriculums and mandatory informatics courses for all… numerous possibilities should be considered. This study is only the beginning.

Regarding the results obtained in this study, only the ones of the GSE type secondary education have been treated. It would therefore complement these results using bigger (students and teachers) samples.

A more challenging improvement would be to expand the analysis on several years and to take into account Flanders and even another countries.

Regarding the methodology, interviews could be conducted with teachers and students to qualify their answers. A future work would be left a reflection of students on teachers’ results and vice versa. Observations in classrooms and analysis of the resources used during the courses would also allow to collect richer data.

Finally, the informatics skills of students would be quantitatively measured (Vandeput & Henry, 2012).

6 Bibliography


An Exploratory Investigation of Change in Students’ Subjective Perception of Informatics

Claudio Mirolo
University of Udine – Dept. of Maths, CS and Physics
via delle Scienze 206, 33100 Udine, Italy
claudio.mirolo@uniud.it

Abstract. This paper discusses an exploratory, small-scale investigation of students’ perception of informatics from an uncommon perspective, i.e. by addressing change between patterns representing frequencies of associations of keywords and ideas as they emerge from a questionnaire administered to particular groups of subjects. The analysis is aimed at identifying trends of change across subsequent instruction levels as well as in connection with extracurricular outreach programs.

Keywords: perception of informatics, secondary education, outreach

1 Introduction

It is often claimed that students hold misconceptions about the nature of informatics, that they tend to identify it with programming, and that outreach activities do have the potential to trigger some positive change of perspective. These issues have been investigated from different perspectives, but the results are not yet conclusive [13]. In the space of this paper we can only mention few such works and refer to the related literature for a broader picture. In particular, the alleged mismatch between the students’ and the computer scientists’ views of our field is the subject of [3, 6, 1, 13], and the perceived role of programming is considered in [6, 10]. As to the impact of outreach programs, most authors report successful outcomes, e.g. [4, 2], but others suggest some caution [13, 5].

In order to contribute to the debate on these topics, the present exploratory study tries to address the matter from an uncommon angle, namely by focusing on how the patterns of ideas spontaneously linked by students to the sphere of informatics evolve across different levels of general (i.e. offering a very limited exposure to computing) school instruction, as well as in which way they may be different from those of the freshmen who enrol in an informatics program at university. In addition, we analyze under the same perspective possible effects of the outreach programs offered to the classes involved in our projects. The data were collected through a compact questionnaire with two open-ended questions, asking for short definitions of informatics and programming according to the respondent’s subjective perception, together with five multiple-choice questions where one or more terms or statements could be chosen from a given list of options. The same questionnaire was administered as a pre-test, and also as a post-test for the school students who took part in the outreach activities.
2 Aims and Scope of the Investigation

According to the helpful terminology introduced in [13], here we consider aspects of students’ views, whereas much related work, e.g., [6, 1], is concerned with attitudes (manifestations of interest, motivation) and intentions (to study or work in the computing field). Explorations of students’ (pre-)conceptions are commonly run through interviews [14, 8]. A notable exception is [9], whose spirit and approach are close to ours but focused on programming. Since subjective perceptions—or views—are elusive, only similarity and diversity of patterns fall within the scope of this work. In particular, the analysis is by no means aimed at assessing the effectiveness of the outreach activities in terms of students’ learning. The observed patterns are however a precondition for more in-depth inquiry to explain the underlying phenomena. More specifically, we address the following (operationalized) research questions:

RQ1. How does the students’ subjective perception of informatics change across subsequent levels of (lower and upper) secondary instruction?

RQ2. Is there any diversity of perception between students attending general schools and university freshmen who choose informatics as their vocation?

RQ3. How central is programming for informatics in the students’ perception, i.e., how frequent is the association of programming with informatics?

RQ4. To which extent does the subjective perception of informatics change, at least provisionally, after exposition to short-term outreach activities?

The investigation was carried out in 2014–16 and involved two 7th-grade (K7: age 12–13) middle school classes; two 10th-grade (K10: age 15–16) and two 12th-grade (K12: age 17–18) classes of a general scientific high school (the last secondary grade is K13). It is worth noting that informatics is not a subject of study in these kinds of schools, where the students are only expected to learn some digital literacy through the use of ICTs across different curricular subjects. Some projects in cooperation with teachers, based on [12] (K7, K10) and [11] (K12), have indeed offered the opportunity to collect a set of data regarding the students’ perception of the field before and after the proposed experiences. In addition, for the sake of comparison we have also collected the answers of freshmen who have chosen to study informatics at university. The different group sizes are as follows: 39 students of level K7, 46 of level K10 and 47 of level K12 took part in the pre-test; of these 34, 42 and 39, respectively, took also part in the post-test. Of the university students, 43 come from general scientific high schools. Although not representative of a large population, the findings of this exploratory study may be of some interest in that they offer the opportunity to replicate similar experiments in different contexts and as a further step towards the development and validation of appropriate instruments to investigate students’ perception of informatics [7].

3 Analysis of the Answers

The aforementioned questionnaire has two sections. The first is about the subjective perception of informatics (open definition + 2 closed-ended questions);
the second is about the perception of programming (open definition + 3 closed-ended questions)—not proposed to middle schoolers. Given the space limits, in what follows we will mainly focus on the first two questions:

1. Based on your perception, provide a short definition of “informatics”.

2. What is informatics primarily about? Choose three terms that appear most relevant to you from the following list: algorithms, complexity, information, programming, applications, computer, models, simulation, automation, communication, multimedia, systems, calculation, data, problems, technology.

We first introduce some preliminary processing of the survey data. Then, we go through the research questions introduced in Section 2 and outline the most insightful findings. The treatment of the open answers is inspired by the phenomenographic analysis and is very similar to that found in [9]. The following steps summarize the inductive process to code key terms occurring in the students’ texts:

- Identification and annotation of relevant keywords;
- Removal of text copied from other items of the questionnaire;
- Revision of definitions to look for synonyms (to be assigned a unique code) and uses of a same word with different meanings (to be coded differently);
- Organization of key terms into areas with some relevant shared feature;
- Merging of sporadic codes into codes associated to broader ideas;
- Checks for consistency and further minor refinements of the coding.

The outcome of this process is a two-layer coding structure where 36 key terms (between parentheses) are organized into 12 areas:

1. problem-solving (problem-solving, problem approach, task complexity)
2. abstraction & modeling (abstraction, modeling & simulation, virtual machine)
3. automation & workflow (automation, task efficiency, data massiveness)
4. data & information (data/information, data collection & analysis, data processing)
5. algorithms & procedures (algorithms, algorithm logic, procedures & processes)
6. programming & language (programs & programming, task accuracy, formalism)
7. computation flow (computation, instructions & stepwise flow, input/output)
8. design & development (design & products, artifact function, artifact structure)
9. nature & evaluation (mathematical features, scientific features, evaluation)
10. computer-centered (computer, computer operating, hardware architecture)
11. I/C technology (information technology, applications, network & communication)
12. user-centered features (instrumental use, task-oriented tools, learning & sharing)

In addition to a direct examination of tables and histograms, the $\chi^2$-test is a suitable tool to analyze differences between patterns of counts. To this aim, we can construct two-column contingency tables representing pairs of patterns to contrast—e.g., the corresponding figures of the pre- vs. post-test, of K12 classes vs. university freshmen, etc. Since the $\chi^2$-model may be too inaccurate for figures capturing sporadic events, in order to build meaningful contingency tables we need to aggregate categories sharing related features into macro-categories. More specifically, for the open-ended definition of informatics the aggregation introduces a third coarser layer: technological tools (areas 11 and 12 above),
conceptual tools (areas 1, 2, 4, 5, 9), computer programming (areas 6, 7, 10), and engineering processes (areas 3, 8). Similarly, for the following multiple-option question the macro-categories are: technologies, problems & abstraction, data handling, and programming.

Research question RQ1. Only the data of the pre-test are relevant to get a picture of the usual state of affairs. Figure 1 shows the histogram of the percentages of students whose definitions of informatics refer to at least one key term in a given area. The relative frequencies of keywords referring to concrete aspects of the technology sphere (areas 10–12) appear to be quite stable over time, whereas we see a regular increase of terms relating to more conceptual topics (notably, in the areas 1, 4, 5, 8, and 9), including programming (area 6).

Research question RQ2. The answers to the first two survey questions consistently indicate that the weight of programming in the perception of informatics rises across instruction levels. However, this trend is broken (relative frequency halved) when we consider the perception of university freshmen coming from general scientific high schools. Said otherwise, the emphasis on programming is significantly reduced in the perception of those students who think of informatics as their vocation. Another differentiation lies in the higher incidence of the abstract concepts of algorithm and information in the freshmen’s answers to question 2 (+68%). A χ²-test on the contingency tables built as outlined above confirms that the evidence of diversity is strong (p-value = 0.005) relative to the options selected to answer question 2 by the K12 vs. the freshmen groups.

Research question RQ3. The options selected to answer question 2 reveal, even more strikingly, the central role that programming plays in the students’ perception, especially in the high school (about 93% of K12 respondents!). Further clues in support of the relevance attributed to programming come into view.
by contrasting the outcome of the post-test against the pre-test. We can indeed observe that the choice of programming is highly stable if compared to the other options, i.e., it is confirmed by 79% of the students who selected it in the pre-test. Similarly, a reference to key terms related to programming is also stable in the definitions provided by high school students (77% for K10 and 65% for K12). The answers to questions 5–7 (not reported here) give also some insight about the perceived nature of programming—mainly an engineering view.

Research question RQ4. Based on the options chosen to answer question 2, a $\chi^2$-test provides no statistical evidence of significant change of perception for K7 ($p$-value = 0.484) and K10 ($p$-value = 0.472) classes. On the other hand, the evidence of change is strong for the students of the K12 level ($p$-value = 0.003). The influence of outreach activities is then unclear at least in the case of young students. However, by focusing on the categories problems & abstraction vs. technologies, the increase of the options falling in the former and the decrease of those falling in the latter emerge as consistent traits for all the considered groups. Since a similar trend can be observed in the pre-test across subsequent instruction levels, the exposition to outreach programs seem to have the effect of anticipating the recognition of some conceptual aspects of the computing field.

Moreover, the patterns observed in the post-test are in some sense closer to the patterns relating to freshmen than those observed in the pre-test. The value of $\chi^2$ can be interpreted as a distance of the patterns in the two columns of a contingency table—the lower the value of $\chi^2$ the closer the patterns—and such a distance is always smaller in the post-test. As a final point, the answers to the first two survey questions have also been subjected to cluster analysis by applying the standard general model, but no remarkable clusters have been revealed. This may mean that the students didn’t share stereotypical views of informatics conveyed by their teachers.

4 Conclusions

The main observations resulting from the analysis of the answers of the students who took part in this work can be summarized as follows:

– Perhaps not surprisingly, independently of their engagement in specific programs, the students’ views of informatics get enriched with new associations of ideas across subsequent school levels—in particular of abstract ideas.

– There is some significant evidence, although not unequivocal, that the subjective perception of informatics by 12th-grade students does not match those of the freshmen who choose to study informatics at university.

– Programming is regarded as a core activity in informatics and this perception is stable and especially strong in the last high school years. Informatics freshmen, on the other hand, assign a less prominent role to programming. These findings may indirectly support the common belief that the identification of informatics and programming may be discouraging to several students.

– The potential of outreach programs to impact students’ view of informatics cannot be clearly assessed. The only piece of significant evidence of change of
perception has been found in connection with the extracurricular activities proposed to 12th grade classes. However, some exposition to such programs seems at least to have the effect of anticipating the recognition of some conceptual aspects of the computing field.

The extent to which the above results can be generalized to other contexts is still to be understood; they are, nevertheless, of some interest for the reasons mentioned in Section 2. Possible directions of future work include the analysis of gender differences, the exploration of connections between the pattern observed and the content of the extracurricular programs, the investigation on and comparison with the perception of teachers; the improvement of the questionnaire.

References

Bringing CS Innovations to the Classroom: a Process Model of Educational Reconstruction

Andreas Grillenberger\textsuperscript{1,3}, Mareen Przybylla\textsuperscript{2}, Ralf Romeike\textsuperscript{3}

\textsuperscript{1,3} Computing Education Research Group, University of Erlangen-Nürnberg, Germany
{andreas.grillenberger, ralf.romeike}@fau.de
\textsuperscript{2} Didactics of Computer Science, University of Potsdam, Germany
przybyll@uni-potsdam.de

Abstract. Computer Science continuously brings forth innovations that also contain new methods, ideas and principles. When considering these innovations from a CS education point-of-view, not only the scientific content, but also students’ and teachers’ perspectives and social demands should be considered to prevent just chasing trends. A promising way to prepare them for teaching is the model of educational reconstruction. In this paper, we illustrate the application of an adaptation of this model in two projects on data management and physical computing. By examples, we show how it is used to develop general guidelines and learning environments as well as concrete lessons and courses.

Keywords: Educational Reconstruction • Data Management • Physical Computing • Research Framework • CS Education • Secondary Schools

1 Introduction

Computer Science (CS) is a highly innovation-driven field. There are many developments that have obvious implications on CS practice and that thus should also play a role in formal CS education. For example, the miniaturization, efficiency and inexpensiveness of modern microcontrollers allows new use cases that are present everywhere in forms of embedded and ubiquitous computing systems. Innovations in data management allow storing and analyzing immense amounts of data and therefore open completely new possibilities, not only for business. Also, such developments are often related to one another: for instance, current innovations in networked embedded systems that can be summarized under the term “Internet of Things” (IoT) are impossible without proper methods to manage all the data that are involved in IoT applications.

In order to provide up-to-date CS education in schools that integrates every-day experiences of students and thus also promotes their motivation, current developments and innovations in CS must not be neglected. At the same time general CS education needs to focus on central ideas and concepts of the science [2, 17]. Thus, CS education research is in the tension between these innovations on the one hand and requirements for general education on the other.
When preparing the contents of innovative topics for schools, merely reducing the complexity and perceived difficulty of the subject matter is not enough. Instead, the field needs to be thoroughly examined. The central question is: How can innovations in CS be didactically prepared for teaching? One way to face these challenges is provided with the model of educational reconstruction (cf. [9]). This model, however, was developed from a natural sciences perspective and according to Diethelm et al. lacks some important aspects that are relevant in particular for CS education, such as the selection of proper phenomena to teach relevant contents [3]. They extended the original ideas with missing aspects and also take into account the general educative nature of CS education in schools. In this paper we will shortly summarize the ideas behind the model and discuss a way how it can be applied as a research framework. Finally, we will illustrate the application of this framework with two example projects on data management and physical computing, both representing current innovations in CS.

2 Educational Reconstruction

The preparation of learning contents and competence goals requires a thorough examination of the content structure and the inclusion of different perspectives on the topic. Kattmann et al. argue that central aspects of lesson planning such as the perspectives of learners are often only considered after the clarification and analysis of the science subject matter, if considered at all [9]. They see a clear gap between science education research and science instruction practice, which they seek to close with the model of educational reconstruction (MER). Here, the clarification of the science subject matter and the investigation of student perspectives both influence the design and evaluation of learning environments. This way, students’ conceptions are considered and contents are related to everyday ideas and experiences of the learners.

However, as Diethelm et al. [3] point out, CS differs from other subjects in goals, knowledge structure and teaching methods. They have therefore adapted and extended the MER for CS education (MER-CSE) and illustrated some of the components with examples. In addition to the aspects mentioned in the science models, they highlight the role of context and phenomena “to motivate the students, to open connections to prior knowledge or to show application situations of the intended knowledge.” [3]. This approach also ties in with the ideas of Piaget’s constructivism, i.e. that learning means to build knowledge structures from interpreting new information (e.g. acquired through playing with things or reading books) based on existing knowledge and experience. Further, in the MER-CSE, social demands are analyzed for verifying the educational significance of the intended learning content.

3 Application of the MER for CS Education

The goal of research using the MER-CSE is to analyze the subject matter and develop lessons and courses or design principles for such. For this, students’ and teachers’ perspectives are investigated in addition to the science content structure. Diethelm et al.
highlight the role of context and phenomena for motivation and constructivist learning [3]. Further, social demands are analyzed to verify the educational significance of the intended content. We rearranged and slightly adapted the components in the graphical representation of the MER-CSE and this way transferred it to a process perspective, which is illustrated in Fig. 1. The four boxes on the left show the different perspectives (science, students, teachers and society) that are investigated in order to derive contents, contexts and phenomena suitable for CS teaching (box in the middle) as well as more general design principles (box on the right). It is noteworthy that there is no clear starting point, however all boxes should be processed at least once to design and arrange lessons and courses tailored to the needs of the particular learning groups. These learning units are evaluated together with teachers and students, so that through the reflection of their experience the single steps of the overall process can be repeated in order to adjust the resulting learning environments to the particular demands of a given setting. This is similar to design-based research in that it involves iterations with various projects in various contexts to create and constantly refine design principles and best practice examples for lessons and courses.

3.1 Underlying Perspectives

The aim of the analysis of the science content structure is to make clear which elementary ideas underlie the content in question, e. g. great principles [2] or fundamental ideas [17] and their relations; thus to provide a science perspective on the topic. The analysis includes the critical and methodologically sound investigation of the science content, theories, methods and technical terms. This builds the foundation to outline the field of research, to identify gaps to CS education and to illustrate the contribution to the aims of CS teaching.

Investigating social demands helps to identify contexts that are relevant for students to cope with requirements that society puts on them in their everyday lives. The significance of CS for general education is underlined from a societal perspective: How are jobs, everyday life and education affected? In addition to stakeholder interviews, documents that mirror society as a whole are regarded, e. g. newspapers, policy papers or project proposals. Given this general approach, social demands only need to be exemplified and reflected in larger time intervals or when outer circumstances change.

As suggested both by Kattman et al. [9] and Diethelm et al. [3], the students’ cognitive and affective perspectives should be pervasive in all planning steps. According to

![Fig. 1. Application of the MER for CS Education](image-url)
Duit et al. [4], this includes pre-instructional conceptions, general cognitive abilities, interests, self-concepts and attitudes. The aim is to find out about more general perspectives of certain groups of learners and about different conceptualizations that students have when explaining CS phenomena, concepts or methods. This can be done in interviews, surveys, written tasks, etc. but also as meta-analyses of existing literature. Through reflecting learners’ perspectives during several iterations of teaching, the resulting teaching guidelines, lessons and courses will be improved over time. This way, the investigation of students’ perspectives helps to tailor lessons and courses to their needs in order to support learning.

Closely related to this are teacher perspectives, particularly focusing on their ideas about teaching, lesson planning, students’ conceptions of certain CS phenomena and about their own conceptions of those phenomena. Introducing new topics to CS teaching affects three dimensions: content, tools and pedagogy. Teachers need support in acquiring new knowledge and skills in a so far unfamiliar content domain, they might have to familiarize themselves with new tools and possibly adapt to new teaching methods. This means that we have to investigate their personal attitudes towards those new elements. When it comes to implementing the ideas in lessons and courses, teachers will be able to provide answers to relevant research questions, e.g.: Which difficulties/problems can be expected and what are possible solutions? With the help of teachers, we are able to find out the most frequently used, reasonable and successful solutions.

### 3.2 Educational Content Preparation

The selection of concepts helps to focus on aspects that are interesting for students and at the same time fundamentals of the subject. Students’ perceptions are important here because they tell us about their ‘mental constructions’ with regard to the content in question, which will affect the choice and preparation of concepts for contextualized learning. When social demands are analyzed, the outcome should be a catalogue of societal norms and requirements for CS education.

Together with the analysis of students’ interests, perspectives and conceptions, an appropriate selection of contexts can be chosen. The aim of contextualizing learning is broadly accepted by the CS education community. In particular, with Informatics in Context (InIK), students learn in authentic contexts that help to motivate them, that show real-world applications and that offer anchor points to build on prior knowledge (cf. [3]).

A central question of the MER-CSE is, which CS phenomena can be explained with contents and methods from the subject. In the following, phenomena are understood as “occurrences of informatics in everyday life and society” [8]. They can be directly or indirectly linked to informatics systems or “have an inherent informatical structure or suggest informatical reasoning” [8]. Useful phenomena, according to Diethelm et al. [3], are all events or occurrences that are related to a specific topic and can be experienced by the learners. In our intuitive understanding, we would add that useful phenomena for CS teaching can be perceived with senses and ideally have something surprising or mysterious that is not immediately explicable by the learners and thus
triggers their curiosity. We propose deriving phenomena in relation to concepts and contexts relevant for students.

3.3 Design and Implementation

The overall aim is to identify ideas and concepts relevant for teaching, to develop design principles and, using those, to put lessons and courses into practice. Through the reflections and feedback of all people involved, i.e. teachers, students and researchers, the implementations are revised and the guidelines and suggestions for learning environments improved over time. This way, practically usable examples, activities and materials that have been evaluated in real classroom situations are developed.

4 Research Examples

In the following, we will describe two different ways for applying this model. Both projects have in common that they introduce new, innovative topics to CS education at (secondary) school level, which entails that curricula and educational standards cannot be used for planning lessons and courses. Thus, the MER-CSE approach is very well suited in both cases, however, the projects set their focus on different aspects.

4.1 Physical Computing

This project deals with “Physical Computing” and the potentials it brings for the CS classroom. Physical computing covers the design and realization of interactive objects and installations and allows students to develop concrete, tangible products of the real world. In contrast to other hardware-centred approaches, e.g. robotics activities, it encourages learners to become creative inventors [14]. It shares concepts with embedded systems and similar technologies that are pervasive in students’ everyday lives [10, 12].

Underlying Perspectives. When investigating the science content structure of physical computing, the perspectives of makers, interaction designers, embedded systems engineers and several more were taken into account. Physical computing integrates methods and ideas of embedded systems, cyber physical systems, interactive systems and smart objects and combines these topics with arts, crafting and engineering. In interviews with experts in the field we investigated their point of view on social impacts of embedded and ubiquitous computing systems and social demands that are connected to these. There was a clear tenor that in order to fully participate in our current and future society, at least a basic understanding of data collection and processing with hardware and how it influences the environment is required. This helps to understand privacy issues and to make informed decisions in many areas of our everyday lives.

To find out about student perspectives, we conducted a study among 115 students and found that embedded systems are not in their focus. None of them had encountered any physical computing activities in CS lessons and their interests in such activities vary a lot depending e.g. on gender [cf. 15]. We can clearly see a gap between what
happens in the real world and what is taught in school. Based on these findings we developed learning environments and implemented project courses where students learn how devices interact and how to get, process and interpret real world data with sensors and actuators. In these projects we gathered additional data, such as concept maps, learner reports or video and voice recordings to extend our understanding about students’ perspectives. They have only a vague understanding how hardware can be used in combination with software in order to perceive changes in the environment and interact with it (cf. [13]). To include teachers’ perspectives regarding obstacles and concerns as well opportunities and motivational aspects, we integrated them into this research from the very beginning. We conduct professional development on physical computing, observe how they cope with the challenges, provide materials and support them in their pilot projects to analyze how their perspectives change when they gain practical experience [16].

**Educational Content Preparation.** Basic concepts in physical computing are sensing and reacting to events, usually changes in the environment. A context in which this is essential and that also affects students is assisted living. Although students may not directly be impacted, they can very well understand why older or impaired people need assistance. Interesting phenomena in this context are automated reactions to different events, e.g. calling keepers, when a dementia patient leaves his familiar environment or turning off household devices when no-one is at home.

Another example is wireless communication. A relevant and familiar context for all students alike, regardless of their age or gender, is traffic. In this combination of concept and context, the phenomenon of communication between cars is not explainable by using sensors alone. For instance, the questions might arise how a car can warn its driver of cross-traffic or emergency vehicles, that are not within sight.

A third example for very powerful concepts used in physical computing are timers and interrupts in the context of security in human-machine-interaction. Use cases and related phenomena are not only found in factories (e.g. assembly lines for cars), but for example also in automatically operated train systems or in arts projects like the *knife.hand.chop.bot*, a machine that plays the game of “five finger fillet” against the user. In all those cases it is crucial that the systems react immediately when body parts are in danger.

**Design and Implementation.** In this research, we started bottom-up with My Interactive Garden (cf. [14]), which encountered a lot of positive feedback from many experts. It has been implemented and adapted by many teachers, so that we were able to gather data in practical implementations running in parallel to the rest of the research process. This way we could investigate students’ and teachers’ perspectives intensively. At the same time the findings are used in addition to further systematic analyses on contents, contexts and phenomena, to improve existing lesson series, but also to develop new
teaching units with learning material following typical methods of physical computing that are influenced by teachers’ and students’ needs.

4.2 Data Management

The topic of the second project is “Data Management” and especially the changes that come with many innovations in context of “big data”. Data management evolved from the field databases and, in addition to traditional topics, incorporates recent changes and innovations. Many of these, even such that have been central to databases for years, were not examined from a didactical perspective yet. Thus, the focus of this project is on the content-oriented side of the model, i. e. to identify concepts and ideas of data management that are suitable for secondary CS education.

Underlying Perspectives. Many of the typical textbooks on databases are also fundamental to data management, which is classified in detail in the Guide to the Data Management Body of Knowledge [1]. Our analysis of the science content structure shows that new concepts have evolved (e. g. new NoSQL databases or data analysis methods), existing ideas are seen under a different light (e. g. redundancy, consistency or data quality) and the relevance of traditional topics like meta data, data privacy or data security is changing with the increasing potentials of data gathering, storage and processing. As teaching still focuses on traditional aspects like relational databases and SQL [5], changes in data management lead to a gap between what is taught in CS classes and the state-of-the-art in CS.

There is not only a gap between science and school, but also between our daily life and school: Nowadays, data analyses are often used in areas that affect us, e. g. analyses of credit card transactions to prevent fraud or user data in social media to suggest friends. We do not only come into contact with data regularly, but also produce large amounts of data. This builds a bridge to social demands: competencies for handling data are important, but people also need deeper knowledge about central concepts of data management to be able to manage and use data in a self-determined way (cf. [7]).

For analyzing students’ perspectives, we used data from a study on teenagers’ use of media and information [11]. It confirms the relevance of data-based technology in their life and shows that they typically use their own devices. They deal with various challenges and use a range of competencies, e. g. synchronize data between devices. Also, they are aware of risks involved when sharing information, but seem to underestimate the consequences, and they are undetermined how secure their data are in the services they use. This underlines the importance of relating central ideas of data management to the students’ daily life.

Also for teachers, innovations and changes in data management are often hard to grasp because of their complexity and of the continuing developments in this field. We found in teacher-training workshops that depending on their experience, teachers need support in gaining expertise in this field or need materials and tools. Despite these dif-
ferences, all participants confirmed the relevance of data management for CS education, even if they are complex. Also, they emphasized the need for teaching units they can flexibly adapt, as these topics are not yet part of curricula (cf. [5]).

**Educational Content Preparation.** Many of the concepts that are already part of database education continue to be relevant in data management teaching, but need to be considered from a wider perspective: redundancy, data schemes, relations between data, data analyses etc. But there are also concepts that are often addressed in other aspects of CS education, e.g. parallelization is typically considered in programming lessons, meta data are mainly discussed in relation to data privacy.

Various contexts can be used for linking these concepts with the students’ experiences. Such contexts include, e.g. smartphones, smart homes, social networks or loyalty cards. Interesting questions in these contexts are: Why are we offered discounts when using a company’s loyalty card? Why are data collections often focused on meta data instead of content? Also, there are various phenomena like potential data loss or duplicates when synchronizing data that can only be understood with knowledge about concepts like parallelization and redundancy.

**Design and Implementation.** This project has not been implemented as a course yet, however, a tool for analyzing a data stream using the block-based programming environment Snap! [6] was developed together with accompanying material. Using this tool, students analyze the Twitter stream and thus gain real insight into data stream analyses. Reactions have shown, that the approach to prepare smaller distinct parts of data management for teaching meets teachers desires, as they can use these modules flexibly. Also, they expected this tool to be suitable for raising students’ awareness for the possibilities and threats of data analyses, e.g. in the context of social media.

5 Conclusion

In this paper we described the adaptation and application of the MER-CSE as a research framework and illustrated two different approaches for preparing new and innovative contents of CS for teaching. With a top-down approach the design and implementation of lessons and courses is clearly seen as the result of the process. While this approach is very systematic, it brings the challenge to obtain students’ and teachers’ perspectives and to consider the feasibility of the scenarios that are derived from the research findings. The bottom-up approach on the other hand requires a very good and usable first implementation, which is the challenge in this case to avoid random experiments in class. In turn, students’ and teachers’ perspectives as well as practicability can be investigated very well. The research results from the different aspects investigated with this framework need to be carefully evaluated and discussed with experts both in theory and practice and have to be considered when designing guidelines, lessons and courses. We are aware that with the examples given we have not systematically validated the
framework. Nevertheless, it was shown that it is suitable for developing general guidelines and learning environments as well as concrete lessons and courses.

References

Posters
Teachers' Attitudes towards CS High School Curriculum in Israel and Lithuania

Tamar Benaya\textsuperscript{1}, Valentina Dagiene\textsuperscript{2}, Ela Zur\textsuperscript{1}

\textsuperscript{1}The Open University of Israel, Faculty of Mathematics and Computer Science
tamar, ela}@openu.ac.il
\textsuperscript{2}Vilnius University, Institute of Mathematics and Informatics
valentina.dagiene@mii.vu.lt

Abstract. Nowadays there has been considerable activities surrounding Computer Science (CS) education on all levels therefore we suggest to take a look at the experience of Israel and Lithuania who put a lot of effort in developing CS curricula. We conducted a survey among high-school teachers in both countries regarding teachers' attitudes towards the CS curriculum. We display the survey results and conclude with some insights.

Keywords: Computer Science Education, K-12, Computer Science Curricula.

1 Introduction

At present, all evidence points to a significant boom in Computer Science (CS) education at the high school level. Education policy makers are becoming inspired by the challenges posed by the CS Teacher Association in USA [1], Computing at School in the UK [2, 3] and the Computing Curricula 2013 [4]. The new Computing curriculum in UK [2] puts the subject on an entirely new footing, as the "fourth science" at school. It offers new opportunities for professional development for teachers and better education for students. In light of the recommendations presented, different countries developed unique curriculums for high school CS education [5]. We would like to share Israel's and Lithuania's long experience in teaching CS at high school. Our previous paper [6] presented the CS curricula in both countries and here we focus on teachers' attitudes towards the CS curriculum.

2 The Research

The teachers' attitudes towards the CS curriculum are vital for the success of CS education. Thus, we conducted a preliminary study among the CS teachers in both countries, in an attempt to learn about their backgrounds, attitudes and opinions towards the CS curriculum. We posed a 12-question questionnaire to CS teachers in both countries. In Israel, the questionnaire was sent to 137 CS teachers, and was answered by 25
of them (18%). In Lithuania, the questionnaire was sent to about 650 IT/CS teachers and was answered by 337 of them (about 52%).

3 Discussion and Conclusion

Both countries focus on the fundamentals of algorithms and programming and data structures. From the survey results we found that most of the respondents in both countries are senior teachers and about half of the teachers in both countries have an undergraduate degree in CS and good number of teachers also has a graduate degree. Therefore we value their responses which are based on many years of experience and insight on CS education.

In Israel, 79% claimed that they were satisfied or very satisfied with the informatics curriculum while in Lithuania only 23% claimed that they were satisfied/very satisfied with the informatics curriculum. They would like more emphasis on programming, modern technologies and new learning materials. Most of the respondents in Israel felt that they have enough professional support while in Lithuania only about half felt the same. They mentioned that they would like more practical advanced courses.

In Israel most of the respondents support Java and C# as an appropriate programming language for teaching CS while in Lithuania most of the teachers support C++.

In Lithuania most of the respondents claimed that basic knowledge on Informatics should be compulsory for every student in the education system while in Israel only about half felt the same. Half of the respondents in Israel and about 40% in Lithuania thought that algorithms and programming should be introduced as early as the 5th or 6th grade and the preferred programming language for this level in Israel was Scratch while in Lithuania the preferred languages were Logo, C++ and Scratch.

From this research we can conclude that the teachers feel that it is important that the curriculum be continuously updated and developed and that it is important to develop teacher support programs and to introduce algorithmic thinking in 5th/6th grade.

4 References

5. ACM Transactions on Computing Education (TOCE), special issue on Computing Education in (K-12) Schools, Volume 14 Issue 2, June 2014 (2014)
From Greenfoot to Java: study of students’ object-oriented design ability and Java programming skills

Jia-Yi Chen¹ and Greg C Lee²

¹ National Taiwan Normal University
Taipei, Taiwan
60347052S@ntnu.edu.tw

² National Taiwan Normal University
Taipei, Taiwan
leeg@csie.ntnu.edu.tw

Abstract. In this study, we tested a two-stage approach to teaching of Java programming to high school students. As oppose to traditional approach, during the first half of a semester long course, object-oriented concepts were first taught using Greenfoot. Java programming follows in the second half of the semester. A quasi-experimental study is being conducted to test the learning achievement of students using this two-stage approach to learn Java programming. Preliminary results showed that students developed better object-oriented programming concepts while maintaining same Java programming skill as those students whom learn Java programming for the entire duration of the semester.

Keywords: Java Programming, Greenfoot, Computer Programming

1 Introduction

Teaching Java programming to high school students presents many challenges. Aside from the common problems that all novice programmers encounter, students face additional difficulty in having to grasp object-oriented programming concepts in learning Java [1]. In this ongoing study we experimented with teaching object-oriented programming concepts through Greenfoot prior to teaching Java programming to high school students.

Greenfoot is a programming environment that aims at making the learning of object-oriented programming easy and fun. Greenfoot allows learner to create ‘actors’ that live in ‘worlds’ to build games, simulations, and other graphical programs, using the visualization and interaction tools built into the programming environment [2]. The actors’ actions (methods) are programmed using Java. With Greenfoot, learners can concentrate on object-oriented design with visual interface while learn to code methods in traditional text-based code.

The hypothesis of this research is that the time spent to acquaint students with object-oriented programming concepts can shorten Java programming learning time. Furthermore, it can help students become better Java programmers in the long run.
2 Research

240 students from six classes are participating in this study. The students are all first year student from one of the top all-girl high schools in Taiwan. Therefore, the students’ academic achievement in all subjects are above average and their learning motivation in general is high. The six classes are randomly divided into two groups, three classes are designated as the control group and the other three classes are designated as the experimental group. In the first semester of this study, each group of students received one-hour of programming lecture/practice per week for 16 weeks, with midterm and final exams after 8th and 17th week of study, respectively. The second semester of classes has yet to commence.

For the control group, Java programming syntax and concepts were taught in the first 16 weeks. The object-oriented design was incorporated into learning of programming concepts throughout the semester. For the experimental group, object-oriented programming concepts was specifically taught and practiced in the first 8 weeks, followed by Java programming syntax/concepts in the remaining 8 weeks of the semester. The two exams were specifically designed to evaluate students’ conceptual understanding of object-oriented programming concepts, students’ object-oriented design ability and programming skills. In particular, the exams aimed to test students’ ability to establish correct programming models, to analogize the use of inheritance, to understand hierarchies and their role in inheritance.

3 Current Findings

This study is at the half-way point. The results from the two exams showed that the experimental group has a better showing on the object-oriented programming concepts. Furthermore, although the experimental group has only learned to program in Java for eight weeks, the end of the semester exam showed that experimental group students have achieved equal Java programming competency as those students in the control group. As the experiment extends into the second semester, we will test how the two groups differ in object-oriented programming skills.

References

Cultivating Teachers Ability to Conduct Maker Activities

Ying-Jyun Chen¹ and Greg C Lee²

¹ National Taiwan Normal University
Taipei, Taiwan
smile69188@gmail.com

² National Taiwan Normal University
Taipei, Taiwan
leeg@csie.ntnu.edu.tw

Abstract. Maker activities allowed students to have hands-on experience in exercising creativities and to use interdisciplinary knowledge to solve real problems. However, very few teachers have any hands-on maker experience. Even fewer teachers have experience in designing and conducting maker activities in class. In this study, a three-stage teacher training program is created to help cultivate teachers’ maker activity designing and implementing abilities. Experiment with 16 teachers showed that the training program can indeed help many teachers enhance their ability and confidence in conducting maker activities.

Keywords: Maker Education, Teacher Training, Instructional Design

1 Introduction

Maker movement has make waves internationally. Even the U.S. White House has hosted Maker Faire to emphasize the spirits of learning by doing [?]. In education, creative maker projects allowed students to use learned interdisciplinary knowledge to solve real problems [?]. To do so, teachers must be familiar with the idea, the implementation and most importantly, designing of maker activities, so as to encourage students to learn by being a maker. We have found that although many teachers are familiar with the Maker concepts, the teachers lack the skill set to incorporate maker activities into their curriculum. In this study, we have designed a three-stage teacher training program, aimed to help teachers to have hands-on experience with Maker activities, and to design and to incorporate Maker activities into their curriculum. Since the study is on-going, we will report preliminary findings in this poster.

2 Research Setup

The teacher training program consists of three stages. In stage 1, teachers learned about the basics of Arduino board and electronic. In stage 2, teachers were
guided to complete an Arduino-based smart car project. In stage 3, teachers use their ingenuity and acquired skill from the first two stages to create an Arduino-based smart music box on their own. For the experiment, sixteen K-12 teachers participated in a 8-week long teacher professional training program. The timeline and activities of the training program is depicted in Figure ???. The teachers meet once a week for a 3-hour hands-on training. Stages 1 and 2 each required two weeks (Weeks 1, 2 and Weeks 3, 4) of training time. Stage 3 also required 2 weeks (Weeks 5 and 8) of training, but there was two non-meeting weeks in-between. The non-meeting weeks were time designed for the teachers to work on their smart music box. Questionnaires were completed by all 16 teachers at the end of each stage. Results from the questionnaires were analyzed to form the basis for the conclusion of this research.

Fig. 1: Timeline showing the 3 stages of teacher training.

3 Preliminary Findings

All 16 teachers have completed the three stages of training. Before the training, only 3 teachers are very confident about carrying out maker project, while at the end of the training, 15 teachers have become confident about their maker skill. In addition, no teachers were confident about designing and implementing maker activities before the training. By the end of the training, 11 teachers have developed self-confidence in designating and implementing some kind of maker activities. Five of those 11 teachers have expressed their readiness in carrying out maker activities in the coming semester.

References

Teaching Informatics: Activities-based Model

Valentina Dagiene, Lina Vinikiene and Gabriele Stupuriene

VU Institute of Mathematics and Informatics, Vilnius, Lithuania
{valentina.dagiene, lina.vinikiene, gabriele.stupuriene}@mii.vu.lt

Abstract. Inspiration, brought by more than 10 years’ experience, has developed Bebras challenge from a single contest-focused annual event into a multifunctional challenge and an activities-based educational community building model. The Bebras community aspiration is to wrap up serious scientific problems of informatics and the basic concepts into playful tasks, inventive questions in the way attracting students’ attention. The other goal is to help teachers explain this informatics concepts in an appropriate way for students. The Bebras community from Lithuania started to organize teacher training workshops in order to improve our teachers’ competencies.

Keywords: Bebras challenge, informatics concepts, teacher training

1 Introduction

The Bebras challenge is based on Bebras tasks, which are short, take a few minutes to answer using a computerised interface (or printed version, e.g. playing card), and requiring deep-thinking skills in the field of informatics [1]. Not all teachers can explain informatics concepts for students, so we decided to organize teachers training in which teachers are as students and learn informatics by using activities-based model.

2 Method

A long-standing problem, discussed by several researchers, is how to teach an introduction to theoretical informatics to secondary school teachers, including preservice and in-service teachers, lacking informatics knowledge and sufficient mathematical background. A preferred method of solving tasks from the Bebras challenge is suggesting that teachers participate in workshops based in Lithuania during the practical teacher training. Teachers were asked to complete the task, and while doing it they were guided to discover and understand informatics concepts. Each new informatics concept is connected to realistic situations of a particular task. This method is based on constructionist learning approach, when teachers could learn in both ways: 1) through developing (constructing) tasks, and 2) through analysing their solutions and explaining the essence of these tasks and why it’s informatics (deconstructing) [2]. We choose the set of
informatics concepts to be introduced from 5 to 9 grades. Concepts are divided into 5 categories according to the draft of new informatics curricula in Lithuania (Fig. 1). The list of concepts can be found in Lithuanian Bebras website.

<table>
<thead>
<tr>
<th>INFORMATION</th>
<th>DATA</th>
<th>DIGITAL TECHNOLOGIES</th>
<th>DIGITAL SYSTEMS</th>
<th>SOFTWARE</th>
<th>SECURITY AND ETHICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary numbers; compression, classification, parity</td>
<td>Pools</td>
<td>Algorithms; array, branching; finite state, command; search, finite-state machine; graph, set, operator, loop, optimization, parallelization, programming; public key cryptography; robot, search, shortest path; sorting, stack, tree, variable</td>
<td>Digital system</td>
<td>Software</td>
<td>Security and ethics</td>
</tr>
</tbody>
</table>

Fig. 1: List of proposed informatics concepts for teachers

Fig. 2: Informatics activities-based model

Concepts are provided with short description and Bebras task examples, which could be downloaded as playing cards. These cards can encourage teachers and students not only to think about the correct answer or how to teach/learn certain concepts, but they inspire to work together (teachers and students, students and students, etc.), collaborate in decision-making and try to find the best task solution (Fig. 2).

3 Conclusion

The model is based on finding the concepts in task and explanation of how it really works as well as to motivate students to share their own ideas, experience and understanding. We hope that informatics activities based on the concepts searching in the concepts map with playful examples will motivate teachers to rethink the teaching process, not forgetting that all attention will be based on students’ collaboration, understanding and analysis.

Acknowledgement: thanks Google CS4HS for support Informatics activities-based teachers training workshops.

References

A Programming Circus for Primary Schools

Katharina Geldreich, Alexandra Funke and Peter Hubwieser

TUM School of Education
Technical University of Munich, Germany
{katharina.geldreich, alexandra.funke, peter.hubwieser}@tum.de

Abstract. It is well known that students often hold many misconceptions and stereotypes towards computer science (CS). To prevent students from establishing these false ideas, it seems advisable to counteract at a very early stage in school. To face this, we designed an introductory programming course for children at primary schools. This poster describes the design of the extracurricular course as well as the pilot study we will conduct.

Keywords: Computer Science Education, Programming, Primary School

1 Introduction

Schools and universities have to deal with misconceptions and stereotypes towards CS [2] which are developed at a very young age [4]. One approach to prevent students from forming false and mostly negative attitudes is to introduce computer science concepts like programming at an early stage in school.

To create opportunities for making unadulterated experiences with technology and computer science, it is recommended to expose students to programming at primary or even kindergarten level. In order to provide such an experience, we planned a programming introduction course for grades 3 and 4. Two of our research questions are

- Which programming concepts can the students learn during this course?
- How differ the interactions and programming results between the students?

2 Course Development

2.1 Design

The course takes place over three days, on which we will spend four hours a day with the children. The students are supposed to learn basic principles of programming. To attract both girls and boys, we designed the course under the motto "programming circus", which should be appealing to both genders.

Day 1. The aim of the first day is to give the students a basic idea how computer programs work. To achieve this objective without the distraction of learning programming at first, we decided to use the unplugged approach [1].
To take up the circus theme, we will let the students program each other to find missing items and animals in a symbolic circus tent.

**Day 2.** On the second day, we want to enable the students to create simple multimedia products using the block based language Scratch [3].

We composed a learning circle in which the basic operations of Scratch get gradually introduced. For example, the children have to program the welcome greeting from the circus director, a joke telling clown and a dancing bear. The students first follow handed-out instructions and solve tasks afterwards.

**Day 3.** Our goal for the third day is getting an impression of what the students have learned so far and what they can apply in more open tasks.

They have to create their own circus story following several specifications like using a variety of characters or a repetition. At the end of the course, we will take a look at all projects and talk about the students’ experiences.

### 2.2 Pilot Study

To test the course design and effectiveness we will run a pilot class. During the pilot study of this course we want to collect data with the following methods:

- **Videography.** To analyze the interactions of the kids with each other and with the teacher, we will record the whole course on video.

- **Group interview.** We will use a variety of interviewing and reflection methods to get an idea of the students’ prior knowledge, what they think about programming and how they feel before and after the course.

- **Screen and audio capturing.** Screens and audio of every student will be captured to get an image of their working methods.

- **Scratch programming products.** We save all programming products the students produce during the three days to analyse them afterwards.

### 3 Conclusion and further research

After evaluating the results of the pilot study we will revise our course design. More courses with the revised curricula will be held.

We hope to get an insight of what concepts the students have of computer science and what prior knowledge they have. Furthermore we want to understand what they can learn during the programming course and how they learn.

### References

A playful tool to introduce lower secondary school pupils to recursive thinking

Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, and Manuel Previtali

Università degli Studi di Milano, Italy, http://aladdin.di.unimi.it

Abstract. To introduce lower secondary school pupils to recursive strategies, we developed a software tool for supporting active guided explorations of the execution of a recursive algorithm.

Keywords: recursion, K12 CS education, tool for algorithm exploration

1 Introduction

Recursion is a fascinating topic that can provide a powerful approach to problem solving, but is often considered to be out of reach for most educational contexts. Thus, we aimed at supporting a proper demystification of recursion in lower secondary schools, in order to suggest it as one of the many tools even younger people can bring on the table of problem solving. We designed a piece of software around a sort of little people metaphor [3] in which recursion is presented as a delegation of self-similar sub-tasks to “helpers”, that we called “fairies”.

2 A tool to visualize recursion

The tool, called Fatine (the Italian word for ‘little fairies’), visualizes the computational steps of a recursive algorithm that computes the reverse of a string chosen by the user, and is designed to support a work of analysis and abstraction about how the algorithm works. Fatine displays a “computer” whose monitor shows some moving objects, and the role of the latter is to represent the underlying process in some concrete way:

– a sequence of circles represents the recursion stack, each circle corresponding to a call of the recursive function (as many circles as characters in the input string);
– the substring passed as argument of a call is represented by a tower under the circle corresponding to the call and its height is proportional to the substring’s length;
– in the first phase of the process, when calls are made, the tower shrinks while moving from left to right, and when subsequently values are returned and composed into the solution, the tower moves back to left and it grows;
– the function is carried on by a little fairy: she detaches a piece from the top of the tower, passes on the tower, receives the tower, attaches a piece at the bottom of the tower, waits, sleeps. Fatine is organized in “levels” as common in video games. The pupils can observe the execution of the algorithm from view points of increasing depth. In the first level
the computer is a “black box” simply receiving the string as input and returning its reversed form. In the next levels pupils can conduct experiments by pausing the process execution: to promote abstraction and generalization, they are not allowed to pause and observe the behaviour of consecutive function calls. When an experiment ends, they can try again with different strings.

3 Experiment and evaluation

We tested the tool within a learning unit devoted to 8th graders and having the following goals: (1) know the fundamental features of recursion; (2) be able to execute a simple recursive procedure described in natural language (e.g. raise 2 to a given power); (3) be able to identify the fundamental features of a recursive algorithm, given its description or the possibility to observe its execution; (4) understand that a problem can be solved by solving subproblems linked to one another. We designed our learning unit according to our algomotricity [1,2] approach, thus we provided a proper environment to foster mental models of the topic under investigation through “unplugged”, motoric activities and the use of tangible objects and then engaged pupils in a “plugged” (software supported) activity to foster abstraction and conceptualization. In the “unplugged” activity pupils executed a recursive algorithm to compute the length of a string represented by a LEGO tower, each pupil executing a function call by following the instructions on a note. The plugged activity was based on Fatine: working in pairs on a computer, pupils saw how Fatine runs the string-reversing algorithm, and they could pause the execution in order to observe characteristic aspects of the process, more and more specific as they proceeded. The tool supported them in understanding what is going on during the execution of the recursive algorithm, by getting them aware of what happens at each single step of the process and meanwhile grasping, at a higher level, how such steps are interrelated. Then the two algorithms (string length and reverse) were compared by reviewing their key features and highlighting their commonalities. During the discussion, all the basic features of structural recursion emerged. In particular, the pupils inferred and told the teacher that the instructions in the two activities, the ones on the notes handed out and the ones executed by the fairies but not visible, had to be similar. The pupils had also to work out what these instructions were and write them down, a task which required them to rework on what they had learned.

References
Bridging Motivation Gaps with Physical Computing in CS Education

Mareen Przybylla\textsuperscript{1}, Patrick Israel\textsuperscript{1}, Julia Streichert\textsuperscript{1} and Ralf Romeike\textsuperscript{2}

\textsuperscript{1} University of Potsdam, Germany
\{przybyll,pisrael,jstreich\}@uni-potsdam.de
\textsuperscript{2} Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany
rafl.romeike@fau.de

Abstract. Intrinsic motivation is a key element of successful and effective learning. In computer science (CS) lessons, some students lack this kind of motivation. Physical Computing offers opportunities for students to recognise the relevance of CS and learn in motivating ways. With the aim of testing the validity of this statement, the KIM-test is used. Initial results show that in the given setting, the above claim is confirmed so that hypotheses for further research can be generated.

Keywords: Physical Computing, intrinsic motivation, CS education

1 Introduction

Students who see themselves as outsiders in terms of computer use often perceive school contents as useless, incomprehensible or nebulous \cite{7}. Thus, their intrinsic motivation for learning in CS is often low. Physical Computing is the development of interactive objects that communicate with their environment through sensors and actuators. It blends the virtual and the physical world and makes artifacts of learning about embedded and interactive systems visible, tangible and shareable. It requires learner initiative and activity, places high demands on their skills for self-determined learning and appeals to different senses. All these characteristics suggest that Physical Computing activities are suitable for promoting intrinsic motivation. With the aim of investigating motivational aspects of Physical Computing activities in various classroom settings, the KIM test for student motivation was adapted and used in a test case. We will describe the methodology and setting, before initial results are presented and discussed.

2 A Short Scale of Intrinsic Motivation

The short scale of intrinsic motivation (KIM, cf. \cite{7}) is a standardised test for learner motivation. It is suitable for making time-stable, objective, reliable and valid statements concerning students’ intrinsic motivation in contexts where self-determined and competent actions are in focus. Therefore KIM questions are also suitable for measuring intrinsic motivation in Physical Computing projects.
For the current project, where inventions take place in regular classrooms, KIM is used because it is very time-economic and measures action-based intrinsic motivation rather than more general academic intrinsic motivation. The test items of this scale were slightly adapted for the purpose of the study (e.g. “in the exhibition” was replaced by “in the classroom”).

3 Survey in a School Project

The Physical Computing project “My Interactive Garden” (MyIG, [?]) was conducted in a randomly divided elective CS course with 32 students of a tenth grade in a high school in Berlin. The project was implemented in the two course groups. One of the aims of MyIG is to make the connection between hardware and software components in embedded and interactive computing systems visible, tangible and understandable. Before and after this project, the students filled in a questionnaire that i. a. contains the KIM test (testing their motivation in the previous lesson series and the Physical Computing lesson series) and items to identify learners who perceive themselves as insiders or outsiders concerning their computer experience, based on the results of the work by Knobelsdorf [?].

4 Initial Results and Further Steps

When comparing the KIM results of the participants, we found that in these two groups, students who perceive themselves as outsiders were substantially lower motivated in the pre-test compared to their classmates who see themselves as insiders, no matter what the prior lesson series was about (text-based programming in Python vs. graphical programming in Scratch). In the post-test, for both groups and all students alike we found increasing motivation, especially on the sub scale perceived choice and within the group of the outsiders also on all other sub scales. Based on these findings we generate hypotheses for further research: (H1) Physical Computing is suitable for motivating students, (H2) outsiders benefit more from Physical Computing activities than insiders and (H3) insiders’ motivation remains equal or increases in such projects.

References

A Teaching Maturity Model for Informatics Teachers in Primary and Secondary Education

Elisa Reçi¹ and Andreas Bollin²

¹ Institute of Informatics Didactics, Alpen-Adria Universität
Klagenfurt, Austria
elisa.reciaau.at

² Institute of Informatics Didactics, Alpen-Adria Universität
Klagenfurt, Austria
andreas.bollinaau.at

Abstract. In order to improve the teaching quality of informatics in primary and secondary schools, we suggest the introduction of a Teaching Maturity model. Comparable to the Capability Maturity Model Integration (CMMI), we propose the use of Teaching Process Areas (PAs) and to address their implementation via Capability and Maturity Levels. This poster presents the usability of our model and first results of two case studies.

Keywords: CMMI, Teaching Maturity Model, Primary and Secondary Education

1 Introduction

Feedbacks and inspectors are the usual forms when assessing the improvement of the teaching quality. Sometimes, those results are biased and not objective, depending on personal feelings versus informatics teachers. There are no standards to assess their preparation efforts and their teaching process. Similar problems were to be found in the field of software engineering, and it turned out that the introduction of a maturity model helped with improving the situation in respect to quality. Like Chen et al. [1] or Montgomery [2], we believe that teaching is related to processes (and services to some extent), and thus we suggest to introduce a Teaching Maturity (TeaM) model, borrowed from SEI’s CMMI [3].

2 The TeaM Model

One aspect of the TeaM model is that teaching is treated as a service where quality is of high relevance. It thus determines the factors related to teaching quality and refers to them with the term Process Areas (PAs). The implementation of a PA is then assessed by two representation paths: (a) a continuous representation via Capability Levels (CL), improving the process by implementing one individual PA, and (b) a stage representation mapped to Maturity Levels (ML), improving the process by implementing a set of related PAs.
Each PA consists of Specific Goals (SG) which include Specific Practices (SP) and Generic Goals (GG) which include Generic Practices (GP). As Fig. 1 shows, a set of Specific Goals/Practices needs to be fulfilled with a specific level of capability in order to reach a specific level of maturity.

Such a model can now be (and is) used for either assessment or improvement activities. The education institution is responsible for the assessment, which contains a list of standards in form of a check-list. Moreover the teachers can use the model to check the improvement of their performance. For example, when a secondary teacher knows that she is at ML1 (Chaotic), and she wants to improve her teaching process by one stage, the model tells her that she has to implement the following PAs first: Teaching Unit Delivery (level 2), Environment/Infrastructure Management (level 1) and Course Design (level 2). And, improving even one more stage means to deal with other PAs as well.

3 Conclusion and further research

At the poster session we will present case studies that demonstrate the model and plan to get feedback for a more holistic study of the TeaM model.

References


A more detailed description of all areas can be found at the web-site of the TeaM project: http://iid.aau.at/bin/view/Main/Projects.
Learning of robotics in primary school, which studied concept during which activities?

Michel Spach¹, Georges-Louis Baron¹ and François Villemonteix²

¹University of Paris 5 Descartes (EDA)
michel.spach@live.fr,
georges-louis.baron@parisdescartes.fr

²University of Cergy-Pontoise (EMA)
fvillemonteix@gmail.com

Abstract. New educational robots ground, such as Bee-Bot and Thymio, re-
vived interest in learning computer through educational robotics. How teachers
that are non-specialists in this topic can seize this learning object in primary
school? What learnings can then be developed and what concepts can be ap-
proached?

Keywords: computer thinking, concept, education, ground robot, learning

1 Introduction

This research is part of a selective project led for the Research Agency (ANR):
"Teaching and learning computer at school" (DALIE), which aims to address issues
related to Computer Science teaching at that level and to characterize the ability of
students to conceptualize the objects on which they act.
It focuses both on teaching practices in Computer Science and on the creation of
teaching situations mobilizing ground robots and leading students to solve various
problems through direct manipulation.
We analyse how these teachers have coped with computational thinking in action and
how they develop this form of thinking among students. Our research has been con-
ducted in two classes of primary schools in the Paris region, between November 2015
and June 2016.

2 Theoretical Context

In the 1980s, LOGO was presented as an opportunity to implement situations allow-
ing develop creative forms of thought with notions promoted by Seymour Papert [6].
LOGO has received considerable attention from innovative teachers, and some critics,
on the ground that it was very specific to procedural thinking and could not bring
much outside of this field [3]. New ground robots smaller and more independent were
designed in the early 2000s. Researches have shown that these new robot have a real cognitive potential for the development of skills in the field of mathematical and algorithmic thinking, even for pre-school children. Our research is based on a framework of didactic and ergonomic psychology, using various works such as Mitchell Bers [2], the French theory of conceptual fields [9], the instrumental approach [8] and the didactics of informatics [1].

3 Methodology

Our research is exploratory and longitudinal. Observation sessions, focus groups with two teachers and focus groups with students have been led during eight months. Data survey grids related to our questioning and our theoretical framework were developed and allowed us to categorize (notions and concepts, pedagogical scenario, pedagogical tools) the data collected. Our analysis focuses on three segments of data from traces of activity sessions, videos and interviews with students and teachers. We favour an inductive approach by adopting a peripheral participant observation approach, trying to avoid causing any change in the classroom.

4 First Results and Perspectives

The teachers we observed were able to integrate, in a pragmatic way, ground robots, such as Bee-Bot and Thymio, in the learning sessions they designed. These sessions generally had the form of a situational problem for which pedagogical tools (coding table, observation grid, programming grids, etc.) used all along the programming process, helped students to program robots and also to develop their own proficiency of the concepts. These robots seem pertinent for trial-and-error-activities (“tâtonnement expérimental” as said the French pedagogue Célestin Freinet). Notions and concepts related to computing, such as algorithm, programming and procedure, has been approached. Our research reveals that teachers were able to foster learning of a computer literacy. It shows the existence of various kinds of instrumental genesis, artefacts becoming instruments, symbolic or material, supporting the manipulations and strategies developed. Now, our analysis will focus on the influence of teaching scenarios teachers follow and on how teachers analyse feedback of their own scenario.

5 References

Uniting Computational Thinking Problem Solving Strategies with MIT App Inventor

Bernhard Standl
Vienna University of Technology
Institute of Software Technology and Interactive Systems
Vienna, Austria
bernhard.standl@ifs.tuwien.ac.at

Abstract. This poster describes a computational thinking problem solving process helping students developing mobile applications with the MIT App Inventor. The chosen computational thinking problem solving process emerged from literature and is twofold. It consists the actual steps for solving a problem and soft-skills a student should hold in order to be successful in solving a task. We assume that the applied process provides students a feasible and structured framework for solving tasks with the MIT App Inventor. Furthermore, we designed teaching resources as worksheets, presentations, and lesson plans for a practical implementation of this approach into classrooms.

Keywords: computational thinking, problem solving, MIT App Inventor

1 Introduction

The term Computational Thinking has become popular in computer science education since Wing defined in [11], with the core message that it is not about thinking like a computer but rather to think like we humans think when we design problem-solving processes for computers. In Lee et al. [6] computational thinking is described as a process that involves defining, understanding, and solving problems, whereas Curzon [3] is reasoning at multiple levels of abstraction, understanding and generalization, while evaluating the appropriateness of the abstractions made. Again, Wing [12] suggested that the most important and high-level thought process in computational thinking is the abstraction process, which is used to let one object stand for many. Based on this and further publications by Barr et al. [2], Curzon et al. [3], Selby et al. [8] and Kafura et al. [5] we derived a problem solving process for combining it with MIT App Inventor for providing students a framework to create mobile applications.

2 Problem solving process

Since Wing [11] initiated the discussion on computational thinking, a variety of contributions aimed at finding a concrete definition for computational thinking
were published as for example Lee et al. [6] and Grover et al. [4]. Without going into all related literature here, recently Voogt [9] examined these definitions and Selby et al. [7] identified a problem solving process we are partly using here. Therefore, we see the actual process of solving a problem and finding a solution is described in six steps followed by students’ attitudes to make a successful solving process more likely. The actual problem process includes: **Understanding** the problem as whole and restate the problem to unveil new perspectives to support the solution process. Also, state clearly what should be achieved with the solution. **Decomposing** the problem to break a hard problem up into smaller, easier ones. Decomposition involves finding structure in the problem and determining how the various components will fit together in the final solution. **Abstracting** a problem in a way that gives a reduced view on the problem. **Designing** an algorithm to develop the step-by-step instructions for solving the problem. **Evaluating** whether a solution meets the criteria. **Generalize** and transferring the solution to a wider variety of similar problems. Furthermore we include these attitudes, described by Warmedal [10] and initially suggested by the Barr et al. [1], which promote a successful problem solving process: **Confidence** in dealing with complexity. **Persistence** while working with difficult problems, **Tolerance** for ambiguity. The ability to deal with **open-ended problems**, and he ability to **communicate** and cooperate with colleagues.

### 3 Classroom Integration with MIT App Inventor

The MIT App Inventor is a well-known tool for creating mobile apps for Android phones. It is based on block programming similar to Scratch. Experience we made, showed us, that the tool is easy accessible for students without much prior knowledge required about programming. Beyond creating simple games or tools, it is also possible to communicate with phone sensors and it can be connected to a database. Even more interesting is, that algorithms can be easy implemented and integrated comprehensible by using the block based language. This fact provides a suitable environment to integrate problem solving strategies described above without having code boundaries hindering students in unfolding their creative potential during the problem solving process. In particular, our idea is, to map the problem solving process into real classroom activities using the MIT App Inventor. Therefore we developed a worksheet where students are guided through this process.

#### 3.1 Example

The example worksheet below is a problem solving task for students to solve with MIT App Inventor using the proposed problem solving approach with an example of the App Inventor tutorials [13]. The worksheet describes the task, gives hints and is used by students to write down notes and insert screen shots of the process.
It can be seen at the pictures above, that students go through the process steps of the problem solving approach while trying to solve the task in computational thinking, from understanding the problem, decomposing it, abstracting, and then designing and evaluating the application before generalizing it into another context. We already have tested the worksheet in practice and experiences we made suggest that an integration of it into a virtual learning environment as Moodle would provide a more flexible structure for the teacher and the students. Furthermore, we identified some problems students had to go through the structured problem solving process and tend to just create the app without thinking about each step. Therefore a more student related specification and description of the process will be necessary for further steps.

4 Poster

This poster presents an approach a computational thinking problem solving approach, which emerged after literature review and experiences made can be combined with the MIT App Inventor. It displays the intended project where besides the the practical classroom activities. In particular the poster will include a description of the problem solving approach, lesson plans, teacher resources and worksheets for students. It is a practical implementation for K-12 computer science education of the broadly discussed term computational thinking.
References

Computer Science in the school curriculum: issues and challenges

Mary Webb1, Tim Bell2, Niki Davis2, Yaacov J. Katz1, Nicholas Reynolds3, Dianne P. Chambers4, Maciej M. Sysło5, Andrew Fluck6, Margaret Cox1, Charoula Angeli7, Joyce Malyn-smith8, Joke Voogt9, Jason Zagami10, Peter Micheuz11, Yousra Ch'touki12 and Nataša Mori13

1King’s College London, UK
mary.webb@kcl.ac.uk, mj.cox@kcl.ac.uk
2University of Canterbury, Christchurch, New Zealand
tim.bell@canterbury.ac.nz, Niki.Davis@canterbury.ac.nz
3Michlala - Jerusalem Academic College and Bar-Ilan University, Israel
yaacov.katz@biu.ac.il
4University of Melbourne, Australia
nreyn@unimelb.edu.au
d.chambers@unimelb.edu.au
5UMK Toruń, University of Wrocław, Poland
syslo@mat.umk.pl
6University of Tasmania, Australia
andrew.fluck@utas.edu.au
7University of Cyprus
cangeli@ucy.ac.cy
8Education Development Center, USA
jmsmith@edc.org,
9University of Amsterdam, Netherlands
J.M.Voogt@uva.nl
10Griffith University, Australia,
j.zagami@griffith.edu.au
11Alpen-Adria-University of Klagenfurt, Austria
peter.micheuz@aau.at
12Al Akhawayn University in Ifrane, Morocco
Y.Chtouki@aui.ma
13University of Ljubljana, Slovenia
natasa.mori@fri.uni-lj.si
Computer Science in the school curriculum: issues and challenges

Abstract. Analyses and discussions were undertaken over several years by researchers, policymakers and practitioners from a range of countries that vary in their approaches to the curriculum in relation to Computer Science. These analyses were undertaken predominantly within the International Federation of Information Processing (IFIP) by the IFIP Curriculum Task Force and at EDUsummIT 2015 and were motivated by a need to examine the rationale, issues and challenges following some concerns across the globe about the position and nature of Computer Science in the school curriculum. We summarise our findings and focus specifically on challenges for the computer science education community in communicating, clarifying needs and promoting curriculum change in order to encourage Computer Science in the curriculum both theoretically and practically.

Keywords: curriculum∙ Computer Science∙ rationale∙ international perspectives

1 Introduction and background

The situation of the curriculum for Computer Science varies between countries. In some, e.g. Cyprus, Poland and Israel, Computer Science has existed as a curriculum subject for many years. For others the curriculum for Computer Science has recently been substantially revised after a period of neglect followed by calls for reform. Even in those countries where Computer Science in the curriculum has a long history, there are differences in approach and in the importance of various factors that affect curriculum design and implementation. This poster will explain and clarify our process, analysis and findings that are outlined below.

2 Research and Analysis

Our analysis and discussion, based on examination of curriculum change in ten countries, as well as a review and content analysis of curriculum reports, led to a rich range of issues and considerations and a set of questions.

1 IFIP is the leading multinational, apolitical organisation in Information and Communication Technologies and Sciences; IFIP was originally set up under the auspices of UNESCO and continues to have a formal consultative status within UNESCO.

2 EDUsummIT is a global community of researchers, policy-makers and educators committed to supporting the effective integration of Information Technology (IT) in education by promoting active dissemination and use of research.
3 Findings

Eight key questions have emerged [1] and we have made some progress in answering four of these:

• What is the range of skills and understanding that should be developed in Computer Science?
• Are such skills and understanding necessary for everyone?
• At what age should Computer Science education commence?
• What pedagogical approaches are likely to be appropriate?

There is a consensus across curriculum reports about the key concepts and techniques of the discipline. However there is as yet no consensus about the importance of more general intellectual practices such as persistence in working through problems and tolerance for ambiguity as well as the importance of collaborative learning and group work. Furthermore, there is an emerging consensus regarding the best starting age for Computer Science being young, about five years old. The availability of software and other approaches designed to support younger learners in learning programming was identified as one of the key factors that have supported this early development of Computer Science learning.

One of the constraints for curriculum design is the need to introduce, early in the curriculum, all three major types of knowledge: concepts, propositions and know-how because these are dependent on each other. One promising approach is a spiral curriculum, such as that developed in Poland, where at each level unified aims are addressed but the spiral structure can allow for a range of aspects of progression that are critical for Computer Science including: increasing difficulty of problems; enabling students to tackle more of the problem-solving process as they progress, and consideration of the move from turtle/block-based programming environments to text-based.

Other questions remain as research challenges for which we have identified key issues. Furthermore we have defined challenges and solutions for advancing understanding of the roles of Computer Science / Informatics in the curriculum challenges that need to be addressed by policymakers, educators, industry partners and researchers [2]. A next step for the IFIP Curriculum Task Force is to examine frameworks and approaches that may provide support in addressing the challenges we have identified.

References

Girls' Attitudes towards Gender Separation in Computer Science in High school

Doron Zohar, Tamar Benaya, Ela Zur
The Open University of Israel, Computer Science Department
108 Ravutzky st. Raanana, Israel 43107
{doron.zohar, tamar, ela}@openu.ac.il

Abstract. Computer Science (CS) seems to be one of the few remaining disciplines almost entirely dominated by men, especially among university faculty and in the hi-tech industry. This phenomenon is prevalent throughout the western world. In Israel, we observed it in high schools where only 32% of the students who choose to major in CS are women. We describe a study in which effort was put in to encouraging girls to select to study CS. This effort included an "All girls' CS class. We display the results of a survey aimed to evaluate the girls' attitudes towards this separation.

Keywords: Computer Science, Education, Gender, High School.

1 Introduction

The shortage of women in CS studies and in the hi-tech world is a fact in many western countries. The percentage of women earning a bachelor’s degree in CS in the US fell from 37% in the mid-1980s to 18% in 2012 [1]. In the US, women comprise 26% of CS and Mathematical science professionals. Many studies discuss problems encountered by women in academia. Studies also suggest different approaches to increasing the percentage of women in CS programs [2, 3, 4, 5]. It seems that the first signs of this phenomenon can be detected in high schools. In the U.S. and in Israel, the percentage of females taking CS in high school is about 32%. The fact that so few females study CS motivated this study.

2 The Research

In several schools in Israel, in the year 2014/15 a lot of effort was put in to encouraging the 9th grade girls to select to study CS. The encouragement included workshops, visits to high-tech companies, teacher training workshops and parents' involvement. We found that all these activities brought an increase (from 22% – 25% to about 37%) in the number of girls who selected to study CS in high school.
In one of the schools, in which the activities above took place, we created an "All girls" CS class and conducted a study to check the attitudes of the girls towards this class. The girls in the school who selected CS were given a choice between an "All girls" CS class and a mixed CS class with a majority of boys. The population of the study included 24 girls who selected the "All girls" class in the year of 2015/16. A twelve question questionnaire was administered to the girls in the "All girls" class. The questionnaire was administered twice: once in the beginning of the school year and once in the second semester.

3 Discussion and Conclusions

A large majority of the girls found some advantages in the "All girls" class (smaller class, easier to study, less disturbances, easier to participate in class and that the class environment feels safer). These advantages were even more appreciated in the second semester except for the feeling that it is easier to participate in class, which decreased in the second semester. Another advantage mentioned by some girls was that it is easier to communicate and create friendships with girls. This advantage was more appreciated in the second semester. About half of the girls did not find any disadvantage in the "All girls" class. The main disadvantage mentioned by about 25% of the girls was that the atmosphere is boring and that it is not as much fun as with the boys. A large number of girls did not feel that there is more competition in the "All girls" class. We found that this feeling was more prominent in the second semester. Most of the girls mentioned that the boys have a negative influence on the class conduct (The boys disturb the class conduct and therefore it is hard to concentrate; The boys do not give the girls a chance to express themselves; The boys are arrogant and lower the girls self confidence). Few girls mentioned positive influence (The boys make the class more interesting and contribute to the educational atmosphere).

All in all we feel that this experiment was successful because the number of girls selecting CS has increased and most of the girls are generally satisfied with the "All girls" class environment. We plan to expand the experiment next year with additional "All girls" CS classes in other high schools.

4 References

4. Camp, T.: 'Computing, we have a problem....' ACM Inroads, 3(4), (2012), 34-40
Workshops
artEoz – dynamic program visualization

Martine Gautier (martine.gautier@loria.fr)
Brigitte Wrobel-Dautcourt (brigitte.wrobel-dautcourt@loria.fr)
LORIA - Université de Lorraine - FRANCE

Keywords: Dynamic program visualization, object programming, education, graphic object representation, memory diagram, software.

Abstract: artEoz software aims at supporting students in their learning computer programming. artEoz original design stems from the authors’ long term experience in teaching object oriented programming. It is grounded on offering the students a pedagogical view of the memory state, that is dynamically updated while the user’s program runs.

Understanding the programs runtime dynamics requires a mental abstraction of what happens in the memory. A teacher’s role is to help the students build this mental representation, which can be used to address any problem. The aim of artEoz is to provide the user with a visual representation of object programming paradigms, from variable declarations to function calls. In addition, we made its use easier than classical debugger tools. artEoz software addresses both beginners and experimented students thanks to its ability to visualize complex data structures.

artEoz software is published under the APP (French agency for software protection) license. Free download for educational purpose is available at arteoz.loria.fr.
Contents and practical implementation: The tutorial will introduce the basic usage of artEoz, namely to visualize the execution of your program code written in Java or Python. Sequences of exercises will address:

- the basic mechanisms: assignment, instantiation, instantiation per copy, cloning, ...
- more complex paradigms:
  - collections management: arrays, tables, lists, stacks, hash tables, trees, ...
  - memory management: unreferenced objects
  - function calls: parameters, receiver, return value, call stack, recursion, ...
  - declaration scopes, nesting scopes, ...
  - inheritance, dynamic binding.

About 20 sequences of exercises are currently available. They can be customized to fit different students’ needs. These sequences were developed, tested and progressively improved during our courses (2nd year undergraduate students in computer science, engineering school, ...) and workshops (secondary school teachers). Some are freely available on arteoz.loria.fr website. Alternative / new ideas will be proposed during the workshop based on questions / wishes / goals from participants.

The objectives of this workshop are to show the possibilities of artEoz software. In particular, it will emphasize its ease of use to teach beginners students. Also, we will show how to create educational sequences adapted to the learners’ profiles and teacher’s objectives. artEoz can either be used in a classroom or independently for self-learning.

Duration: Between 45 minutes and 2 hours, depending on the application and the participant objectives. Anyone can create their own teaching sequence using artEoz educational tool.

Targeted audience: code programming teachers, who wish to use artEoz as an educational tool during their lessons for beginners or for more experienced students; and more generally, anyone curious of how this software works.

Resources: Each participant must have a personal laptop computer with administrator rights (artEoz install) and internet access. However, a web version of artEoz is currently under active development. artEoz will be available for use via an internet browser.

artEoz software and the website arteoz.loria.fr are currently available in two languages: English and French.

Short Biographies:

Martine Gautier, assistant professor, Université de Lorraine, LORIA (Laboratoire lorrain de recherche en informatique et ses applications). Her works are focused on the development of educational tools.

Brigitte Wrobel-Dautcourt, assistant professor, Université de Lorraine, LORIA (Laboratoire lorrain de recherche en informatique et ses applications). Her involvement in teaching in computer science is behind the creation of artEoz.
IT2School-Workshop

Ira Diethelm and Melanie Schaumburg
Carl von Ossietzky University
Computer Science Education
26111 Oldenburg, Germany
ira.diethelm@uni-oldenburg.de, melanie.schaumburg@uni-oldenburg.de

Keywords: teaching materials, introduction to CS/Informatics, understanding IT

Abstract: This workshop will give a deeper insight into the materials described in the corresponding ISSEP-paper [3] about the IT2School project which aims at supporting CS / Informatics at school in grades 4-10.

The German association 'Wissensfabrik' is a widespread and stable network of over 120 different big and small companies with long tradition of cooperation between companies and schools for supporting STEM and economical thinking inside the regular school schedule. As their scientific partner, our aim was to create usable and meaningful materials that most teachers would like to teach with in their classes, regardless of their background, experience or knowledge of CS. Therefore, it had to be interesting and powerful and at the same time easy to use and understand. And it also had to be motivating and inspiring for students aged 10 to 16. For the final package of modules, we used many well-known ideas and compiled them to a flexible set that aims at fitting for all teachers in grades 4-10.

In this workshop we will give a hands-on experience on the modules of the IT2School project and its concept. We will give a brief introduction and afterwards there will be much time to try out in detail. In addition, time for discussion and reflection will also be provided.

The five basic modules to try out in this workshop are:

1. Communication from blinking to encryption: This module starts at coding information via blinking, adapted from cs4fn [2], and at the historical dimension of transporting information over large distances by Morse code. Building a string telephone and submitting pictures are also part of the CS unplugged experience [1, 4]. It ends with some easy Caesar's encryption.

2. Understanding the internet: How does such a big world fit into such a tiny box? This question leads to a setting that explains how the internet works, using paper and other craft materials only, but at the same time incorporating terminology like IP-addresses.
3. Codes in a supermarket: How does the checkout know the price? What happens if we manipulate barcodes and how are these used to run a supermarket? The answers to these questions are discovered in this module. Here, students and teachers should visit a local supermarket and get a view behind the scenes. Afterwards students think about useful innovations using QR-Codes in their school, e.g. for a game.

4. How to program, using Scratch, is a must-have module in a setting like this and is also suitable for all kinds of personas.

5. My very special input: ‘What happens if we connect bananas or play-doh to the computer and use them as a keyboard?’ is discovered in this module using MakeyMakeys [5] or similar controllers.

The project is (and maybe always will be) work in progress. The project material is published under Creative Commons. It is written in German at the moment. English versions are under construction. For this workshop we will provide summaries for all modules in English language.

References

Resources: All documents of the project including suggested lesson plans, work sheets etc. (in German) can be downloaded at the webpage www.it2school.de. English summaries will be provided at the workshop.

Ira Diethelm She is professor for computer science education and responsible for teacher education for the subject informatics / computer science at the Carl von Ossietzky University of Oldenburg, Germany.

Melanie Schaumburg She is a scientific assistant at the research institute OFFIS in Oldenburg and at the pedagogical department at the Carl von Ossietzky University of Oldenburg, Germany.
Physical Computing for Novices: Using the TinkerKit with Snap4Arduino

Mareen Przybylla

University of Potsdam, Didactics of Computer Science
August-Bebel-Str. 89, 14482 Potsdam, Germany
przybyll@uni-potsdam.de

Keywords: Physical Computing, Arduino, TinkerKit, Snap4Arduino, block-based programming

Abstract: Physical computing is the development of interactive objects or installations that communicate with their environment through sensors and actuators. With the Arduino TinkerKit and Snap4Arduino this can easily be integrated in schools. Preassembled sensors and actuators and a modification of the Snap visual programming language reduce the complexity of using and programming Arduino. In our workshop, participants will get to know these tools and different approaches to using them in the classroom.

Contents and practical implementation:

Physical computing blends the virtual and the physical world and makes artifacts of learning visible, tangible and shareable. It promotes learning in a creative and practical fashion and perfectly matches with the ideas of constructionism, which has the creation of personally relevant artifacts in its core.

With the Arduino TinkerKit and Snap4Arduino those benefits can easily be integrated in schools. The Arduino TinkerKit is an educational physical computing construction kit with preassembled sensors and actuators that reduces the complexity of using Arduino (Figure 1). It allows for immediate tinkering and trial without the need of elaborated skills in physics or principles of electrical engineering, such as soldering.

Snap4Arduino is a modification of the Snap visual programming language that lets you seamlessly interact with almost all versions of the Arduino electronic prototyping

Figure 1: TinkerKit Sensor Shied and modules
board. With Snap4Arduino in connection with Arduino already at a low age level immediate tinkering and the construction of interactive objects is possible even for novices (Figure 2).

![Figure 2: Using the TinkerKit with Snap4Arduino](image)

In this workshop, participants will get to know these tools and different approaches to using them in the classroom. With the example of My Interactive Garden [1] they are introduced to an award-winning physical computing teaching concept that was tested and evaluated in different contexts (schools, university, professional development). The aim of a My Interactive Garden project is to collaboratively work on an exhibition of a futuristic interactive garden. This way, learners are encouraged to not just copy or rebuild systems, which they already know, but to use their imagination and creativity in order to develop personally relevant interactive objects. Hands-on activities and examples will show the participants, how the provided tools can be used in class in order to inspire students to create meaningful interactive objects.

Resources:

All materials that will be used in the workshop are available online at www.tangible-cs.de and are currently being translated from German to English. Additional material explaining the tools as well as discussion forums can be found at www.arduino.org/tinkerkit and www.snap4arduino.org respectively.

References:


Short Biography:

Mareen Przybylla is research assistant and PhD candidate at the professorship for Didactics of Computer Science at the University of Potsdam, Germany. Her main research interest is on physical computing in CS education, especially concerning the design of concrete lessons and courses in terms of contents, learning material and methodology and the analysis and evaluation of such. In 2016 she started working at Arduino SRL where she is project manager of the TinkerKit and education.
From Scratch to Patch: a "hands-on" workshop

William Robinson, Glebe House School & King's College London

Keywords: novice programmers, block-text transition, algorithmic thinking

Abstract: A significant consequence of the widespread adoption of block-based languages such as Scratch as a means of introducing young learners to computer programming, coupled with the fact that "real world" programming is generally performed via textual languages, is that educators are increasingly forced to address the question of how one may best facilitate the transition. This workshop will give participants the opportunity for hands-on experience with Patch, an extensively modified version of Scratch which addresses this issue and which is based upon professional practical classroom experience gained through teaching Computing to children aged 6 to 13 in the UK using both Scratch and Python over the last 3 years. Participants will be supplied with a copy of Patch and invited to explore its features through a number of programs and discuss the argued pedagogical benefits of the modifications with the author.

Contents and practical implementation: A beta copy of the latest version of Patch will be made available to workshop participants, as well as several short programs which demonstrate its features (some of which are outlined below). Participants will be invited to explore these and discuss their pedagogical rationale.

True single-step tracing / debugging

User-editable (type-in) pseudocode blocks

```
# player guessed the secret word
if they've won
congratulate the player
show 'game over' animation
```

```
# player guessed the secret word
if they've won
say Congratulations for 2 seconds
show 'game over' animation
```
User-defined functions supporting return values, Python-like invocation syntax

Python-like iteration blocks & list operation syntax, emphasis of block indentation

Local / scoped variables indicating 'undefined' status where appropriate

Resources:

Participants are asked to bring their own laptop. Patch is a 32-bit cross-platform application based on Squeak, and a download link for the software will be supplied.

About the author:

The author and developer is a UK Computing teacher responsible for UK Curriculum Key Stages 1, 2 and 3 (i.e. primary and middle school) age groups, and is currently studying for the MA in Computing in Education at King's College, London. Patch was developed by the author as a result of discussions on that course with Dr Sue Sentance, and draws upon the author's own practical experience in the classroom of the issues surrounding the learning of computational thinking, particularly those aspects which pupils find most challenging as they move from blocks-based to textual programming. Patch runs on Windows, OSX and Linux and was originally developed in early 2016, based upon MIT's original Scratch 1.4 codebase. It has now been ported by the author to the most recent Squeak VM as a modification of the extensively redesigned and optimised version of Scratch created by Tim Rowledge for the Raspberry Pi Foundation.
Sprego Is Unplugged:
Functional Programming for Young Children

Maria Csernoch
Faculty of Informatics, University of Debrecen, Hungary

Theoretical background

Sprego – Spreadsheet Lego – is a programming tool in spreadsheet environment. Similar to Logo, it utilizes a limited set of instructions for real world problem solving, but in a functional language. As such, it is a concept based approach to programming in a widely used and accepted environment. Sprego would serve both as an introductory programming tool and a programming approach to spreadsheet management. The concepts and tools which Sprego applies are the following:

- functional modeling,
- schema construction, association, and accommodation.
- concept of function
- n-ary and composite functions,
- concept of n-dimensional vector,
- the set of Sprego functions: a dozen general purpose functions,
- authentic contents, matching the students’ interest, making them motivated by the content, connection to other subjects,
- limited spreadsheet tools, switching the focus from the interface to the programming aspect.

The concept of function plays a crucial role in Sprego programming. We claim that with this novel approach we can help building and/or strengthening the students’ concept of function and introducing n-ary and composite functions, providing practicing opportunities for building concepts which at present are considered higher mathematics. On the other hand, we can provide a simple tool for other sciences and subjects for data and information dissemination, retrieval, analysis, evaluation, and discussion in spreadsheet environment.

Description of the workshop material

In the workshop, authentic tables of various contents and data types will be provided to show what types of programming tasks can be carried out in Sprego environment and how these tasks can be related to introductory programming and data management. The selected data sources are also planned to demonstrate that text-based problems suit young children, since their knowledge in mathematics at this age is limited, and their lack of concepts, usually relied on in programming, will not be an
obstacle in building the models. Furthermore, one selected table is planned to be analyzed thoroughly, and based on this table real world problems to be solved.

In the workshop we present tools – printed materials, toys, stickers, scissors, 3D printed objects, colors, black board/interactive board (if it is available), etc. – which allow us to introduce functional modeling and programming for young children. With these tools we can demonstrate how a simplified functional programming environment would help building the concept of function, how we can utilize composite and n-ary functions, and n-dimensional vectors. The tools of unplugged Sprego allow us documenting the model and the coding process with minimized writing duties, which plays a crucial role considering young children.

With reducing the number of handy tools, we can demonstrate how the approach would work with older and more advanced students in secondary and tertiary education who are new to programming and open to algorithmic based end-user computing.

Tentative agenda

– The prepared tables for the workshop can be downloaded from a reserved website. These tables of various contents have been tried out with our students and they seemed highly interested in the selected contents. With experienced teachers of the workshop, analyzing the tables and call attention to the opportunities which lie within them takes about a couple of minutes.
– The main section of the workshop is solving tasks, based on one selected table, mainly focusing on handling strings, conditions, and counting.
  – To each task, we build the functional model then the algorithms, in both phases using the tools which would help the understanding of the problems. These two phases go on unplugged, where the software is only used to present the table and the problem.
  – Based on the algorithm, still unplugged, we build the composite function starting from the innermost function and show how it can be expanded.
  – Depending on the age of the students we have the option to choose (1) staying completely unplugged or (2) using the unplugged tool parallel with coding, or (3) completing the unplugged phase and then do the coding.
– In the final phase of the workshop, we plan the discussion of the expansion of the composite functions, depending on the students’ age and their level of computational thinking and on the discussion of the problem and the output.

Biography of the workshop leader

Maria Csernoch, associate professor at the Faculty of Informatics, University of Debrecen, Hungary (2009-). Teacher of Mathematics and Supervisor and Leading
Teacher of Informatics at the Teacher Training Secondary School of the University of Debrecen (1986–2009).

Degrees in Teacher of Mathematics, Descriptive Geometry, Informatics, and English, BSc in Software Engineering, PhD in Mathematics and Computer Sciences and Habilitation in Applied Linguistics.
Guess the Code
Zsuzsanna Szalayné Tahy
Eötvös Lorand University, Faculty of Informatics

Keywords: teaching practice, computational thinking, application, programming

Abstract:
Guess the Code is a classroom project and a method of teaching computational thinking, programming and using application. While we learn basic knowledge of how to use application, we try to guess the data structure, the algorithms of that application. We test the program, we discuss about the application as we were the programmer or the developer.

Contents and practical implementation (exhibition, discussion, …):
On the workshop we imitate a lesson. Participants get some problem and they try to solve it. In the second half of the workshop we discuss the programming aspects of the problems.

Some example of problems:
- Can you write $2^2$ into a spreadsheet’s cell? What are there in these cells?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^2$</td>
<td>$2^2$</td>
<td>$2m^2$</td>
</tr>
</tbody>
</table>

- If you copy a text with links to a spreadsheet, the links are saved. How can you extract the links?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video clips Working group: MST.COE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topic</td>
<td>Type</td>
<td>Rating</td>
<td>Content</td>
<td>Classroom use</td>
<td>Relevant courses</td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
<tr>
<td>Modern Physics</td>
<td>Advanced</td>
<td>Excellent</td>
<td>Lecture and lab work</td>
<td>Advanced students</td>
<td></td>
</tr>
</tbody>
</table>

- How does the MATCH() function find the searched value?

- How are my working hours counted?

… And when does the Coffee break start? Is there a 20 minutes presentation on the ISSEP conference?
• What kind of classes and groups are there in a school?

<table>
<thead>
<tr>
<th>11:00 a.</th>
<th>11 b</th>
<th>11 c</th>
<th>11:00 a.</th>
<th>11 e</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:30 a.</td>
<td>11 b</td>
<td>11 c</td>
<td>11:30 a.</td>
<td>11 e</td>
</tr>
</tbody>
</table>

• To be or not to be?
What is the difference between the calculated money and the amount in the pocket.

<table>
<thead>
<tr>
<th>Full time</th>
<th>Part time</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,89</td>
<td>3,1974E-14</td>
</tr>
<tr>
<td>8,89</td>
<td>3,1974E-14</td>
</tr>
</tbody>
</table>

Resources:
Every participant need a PC or laptop and an office application on it. Participants’ own tools could be used too. Different softwares give more experience. We need network to spread and collect solutions. A projector could be useful too.

Short Biographies
**Ms. Zsuzsanna Szalayné Tahy** has been a teacher for 28 years and a PhD student for 3 years. She teaches programming and using office applications. Her students are successful in problem-solving, they are on the top of IT competitions. Her research topic is how to make programming and computational thinking learnable for everyone.

More: [http://sztzs.szigbp.hu/sztzseng.html](http://sztzs.szigbp.hu/sztzseng.html)