Raspberry Pi

Cambridge Computing Education Research Symposium

Table of contents

Table of contents cont.

Poster presentations

Welcome from the organisers

The Raspberry Pi Foundation and the Department of Computer Science and Technology at the University of Cambridge are delighted to welcome you to the Cambridge Computing Education Research Symposium.

This symposium gives us an opportunity to bring together academics and educators with a common interest in computing education. For this symposium we are focusing on the theme of computing education for young people in non-formal and formal settings.

Research in computing education is a young field. There has been a steady stream of research focused on higher education, including how students at university learn computer science; and another substantial stream on how technology can and has impacted education more broadly. Less apparent is research on computing education for young people, which is our topic for today. Computing is now part of the school curriculum in many countries worldwide, and developing as an extra-curricular option in many more. In this context, further research on how best to teach, learn, and assess computing is desperately needed. We also need to investigate ways of inspiring and motivating all young people in an area which is increasingly important for their future.

We were looking forward to welcoming you to Cambridge, England, for our first research symposium in what we hope will be a regular series. Cambridge is a beautiful city and we had planned two days of networking and discussion. However, for reasons beyond our control an in-person meeting was not to be, and we are delighted to be able to offer this symposium to even more of you as a virtual event. We are very grateful to all the people who have worked with us to make this a success, and I'm sure you will appreciate there's been a learning curve along the way!

We were very pleased to receive a good number of submissions to this conference. We hope you will enjoy the nine presentations and twelve posters we have selected for today. In addition, we are delighted to welcome Dr Natalie Rusk from the MIT Media Lab as our keynote speaker. We hope you enjoy the day!

Dr Sue Sentance
Raspberry Pi Foundation

Professor Alastair Beresford
University of Cambridge

April 2020

Symposium organisation

Programme chair: Hayley Leonard

Local organisation: Diana Kirby

With special thanks to Janina Ander, Sophia Donovan-Spalding, Helen Drury, Tom Evans, Lizzie Jackson, Thom Kunkeler, Oliver Quinlan, and Carrie Anne Philbin.

Keynote speaker

Dr Natalie Rusk,
MIT Media Laboratory

Natalie Rusk is one of the creators of Scratch and is a Research Scientist in the Lifelong Kindergarten group at the MIT Media Lab.

She is the lead author of the Scratch Coding Cards and co-founder of the Computer Clubhouse, which has grown into an international network of 100 after-school centers where young people create projects that build on their interests.

Natalie researches youth motivation for learning in Scratch and other learning environments and recently co-authored the research paper, Youth perspectives on their development in a coding community.

Natalie earned a Masters in Education from Harvard and a PhD in child development from Tufts University. You can read some of Natalie's writing online, including her article: There's More Than One Way to Code a Cat.

How can we design research that supports young people as creators?

Millions of young people around the world use the Scratch coding environment to create animations, games, and other interactive projects. While many research studies have focused on learning with Scratch and related technologies, these studies differ widely in their questions, methods, and findings.

This keynote will highlight examples and stories from research that have advanced understanding of how to support young people developing as creators, including the role of peers and other community members. Building on these insights, Natalie will suggest ways to document learning experiences to help improve and expand opportunities for young people from diverse backgrounds to develop a broad range of skills as they create projects.

Paper presentations

Theme 1:

Teacher engagement in computing education research

Enabling school computing to respond to a skills mismatch between education and the ‘world of work’: teacher-researcher and academic voices report on work in progress

Alison Twiner (University of Cambridge), **Jo Shillingford** (Chellaston Academy), **Louis Major** (University of Cambridge), **and Rupert Wegerif** (University of Cambridge)

Background

The ‘fourth industrial revolution’, accelerated by advances in technology and computing, is widening a mismatch between skills enabled by traditional schooling and skills needed for the workplace. Employers in engineering, ICT, and other technical disciplines are clear about the competencies young people need: technical and practical skills (such as programming), alongside the transferable skills (including creativity and collaboration) that help individuals thrive in any organisation (Nicholls, 2018). School computing education is ideally placed to respond to this challenge, with national curriculum criteria indicating it “equips pupils to use computational thinking and creativity to understand and change the world” (Department for Education, 2013). The Virtual Internships Project (VIP) – a significant collaboration between the University of Cambridge, BT, and Huawei – is developing and evaluating a new approach to education that simulates meaningful encounters with the ‘world of work’ for young people aged 11-14. Initially working with computing and other teachers in ‘education opportunity areas’ in England, VIP facilitates 1) raising young people’s aspirations, awareness, and ambition in areas traditionally associated with low social mobility; and 2) developing complex competencies, in particular, collaboration and creativity.

Research focus

In this symposium we report on research in progress (2019–2021): developing and evaluating the VIP programme within KS3 Computing. We illustrate how this can meet curriculum criteria, whilst supporting authentic links to the world of work through a dialogic approach (building on strong evidence of the value of ‘dialogic education’, e.g. Howe, et al., 2019), in which collaboration and creativity around meaningful challenges to design useful technological solutions has an intrinsic and potentially societal value (Johnson & Adams, 2011). The programme is designed to take roughly 10 in-class hours over 6-10 weeks, where students in groups will research the challenge; design, model, or build a solution; and present their findings.

Method

VIP is informed by design-based research (DBR; Bakker, 2019). Driven by the need to ‘close the gap’ between educational theory and practice, alongside demand for rigorous, systematic ways of developing practically-relevant educational materials and environments, DBR features design as an integral part of the research process. Through systematic and iterative development and evaluation, we collected quantitative and qualitative data regarding what works, how, and why. Data from phase one, iteration one (for computing: seven classes in two schools, five teachers, 200+ students) includes: lesson observations, post-programme interviews with teachers, focus groups and surveys with students. In phase one, iteration two, and into phase two, we are incorporating more quantitative assessment. This data and assessment will inform ongoing programme development and evaluation.

Findings

In this symposium, we share significant outcomes from the ongoing work in progress in two 'lead-research schools'. For instance, through initial discourse analysis of the construct 'Collaborating2Create' as manifested in computing classrooms. Findings will also be presented from 50+ computing student retrospective pre-test (Drennan & Hyde, 2008) and post-test perceptions (analysed using related samples Wilcoxon signed rank test), as well as three focus groups on how attitudes toward learning and the curriculum changed, and whether aspirations, awareness, and ambitions were impacted. An additional focus will be the role of the teacher, as genuine teacher-researchers in DBR. One computing teacher-researcher and careers-lead will present findings around the programme as an innovative and meaningful way to meet curriculum requirements, with practical classroom examples.

Conclusion and implications

The reported work in progress offers an approach to computing education that capitalises on authentic links to the world of work through the facilitation of problem-solving and effective collaboration around meaningful challenges. Next steps will utilise the learning from phase 1 to develop and test a scalable model, which will include connections with other world of work partners. The development of such opportunities have the potential to make a real impact on schools, students, and educational practice.

References

Bakker, A. (2019) Design research in education: a practical guide for early career researchers. Abingdon, Routledge.

Department for Education (2013) Computing programmes of study: key stages 3 and 4. Available from: https://www.computingatschool.org.uk/data/uploads/secondary_national_curriculum_-_computing.pdf

Drennan, J. & Hyde, A. (2008) Controlling response shift bias: the use of the retrospective pre-test design in the evaluation of a master's programme. Assessment & Evaluation in Higher Education. 33 (6), 699-709.

Howe, C., Hennessy, S., Mercer, N., Vrikki, M. & Wheatley, L. (2019) Teacher-student dialogue during classroom teaching: does it really impact upon student outcomes? The Journal of the Learning Sciences. 28 (4-5), 462-512. Available from: DOI: 10.1080/10508406.2019.1573730

Johnson, L. & Adams, S. (2011) Challenge based learning: the report from the implementation project. Austin, Texas, The New Media Consortium.

Nicholls, B. (2018) Challenge based learning: A real-world approach for secondary students to solve complex problems using geoscience knowledge and skills. Terrae Didat. 14 (4), 369-372.

Codeveloping primary (K-5) programming design concepts with teachers

Jane Waite and Paul Curzon (Queen Mary University of London)

Background and context

In 2014, the national curriculum of England agreed that by the age of eleven, young people should know how to design simple programs (Department for Education, 2013). Yet there has been little research into what primary programming design is or how it should be taught (Falkner & Vivian, 2015; Waite, 2017). A survey of over two hundred primary teachers in England reported that teachers thought design in programming projects was important, but this was not converted into teaching practice (Waite et al., 2018). Reasons for this included pupil resistance to design, a lack of time, questions on pedagogy, confusion over what an algorithm is, and a lack of training on how to teach design (Waite et al., 2020).

Research focus

To create training on how to teach the design of programs, the concepts related to this area must be known. However, these concepts have yet to be defined (Rich et al., 2017). To address this, we are codeveloping a design toolkit with teachers to provide the underpinning concepts about primary programming design, and work towards a clearer definition of programming design. We describe here the approach taken, which draws on design-based research (Cobb et al., 2003) and participatory design (Muller & Kuhn, 1993).

Method

An initial version of the toolkit was created and piloted in professional development (PD). Feedback was used to update the toolkit, and the new version reviewed by an expert in the field.

The reviewed toolkit contained eleven concepts (including project genre, design component, design format, and a design refinement process) and was used with a group of expert primary teachers in a codevelopment process. Fifty-six teachers volunteered to take part in this process. Based on their answers in a participant survey, respondents were ranked by expertise and asked to interview in that order. The number of teachers interviewed was managed using a saturation of responses approach (Fusch & Ness, 2015).

Twenty-eight teachers, half male and half female, geographically spread from south-west England to Scotland were interviewed face to face. Interviews followed a protocol, repeated across concepts. First, the teacher read a concept description, then they observed the concept being applied to a description of a teaching activity. Next, the teacher applied the concept to an activity they taught. After this, the concept was discussed, including categorising how useful it might be for different types of teachers.

The toolkit will be updated based on analysis of the interviews, and a revised version will be shared online with interviewees for comments. These comments will be used to develop a final version of the toolkit. Interviews and comments will be analysed using Kukartz's (2014) qualitative thematic analysis approach.

Findings

During the interviews, all teachers successfully applied each concept to their lesson activity. A range of activities were reviewed, including unplugged, animations, quizzes, games, and physical computing activities. Teachers suggested many ways to improve the toolkit, including adding extra detail, changing definitions, adding examples, and radically changing one concept. There was broad agreement that some concepts were more useful to novice teachers than others, but generally that all concepts were useful to experienced teachers developing resources and PD for others.

Conclusion and implications

We have worked with teachers to develop a set of primary programming design concepts. These concepts can be used to develop an understanding of design in primary programming projects: to review and compare activities, to guide the creation of resources, and as underpinning content for PD. Further research is needed to evaluate whether these concepts are useful beyond the primary English context and to evaluate the impact of introducing these on pupil progress.

References

Cobb, P., Confrey, J., diSessa, A. A., Lehrer, R., & Schauble, L. (2003) Design experiments in educational research. Educational Researcher, 32 (1), 9-13. Available from: DOI: 10.3102/0013189X032001009

Department for Education (2013) National Curriculum in England: computing programmes of study - key stages 1 and 2. Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

Falkner, K. & Vivian, R. (2015) A review of Computer Science resources for learning and teaching with K-12 computing curricula: an Australian case study. Computer Science Education 25, 390–429. Available from: DOI:10.1080/08993408.2016.1140410.

Fusch, P. I., & Ness, L. R. (2015) Are we there yet? Data saturation in qualitative research. The Qualitative Report, 20 (9), 1408-1416. Available from: https://nsuworks.nova.edu/tqr/vol20/iss9/3

Kuckartz, U. (2014) Qualitative text analysis: a guide to methods, practice and using software. London, Sage.

Muller, M. J., & Kuhn, S. (1993) Participatory design. Communications of the ACM, 36 (6), 24-28 Available from: DOI: 10.1145/153571.255960

Rich, K., Strickland, C. & Franklin, D. (2017) A literature review through the lens of computer science learning goals theorized and explored in research. SIGCSE '17: The 48th ACM Technical Symposium on Computer Science Education, 495–500. ACM, New York, NY, USA. Available from: DOI:10.1145/3017680.3017772.

Waite, J. (2017) Pedagogy in teaching Computer Science in schools: a literature review (after the reboot: computing education in UK schools). Available from: https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf.

Waite, J., Curzon, P., Marsh, W. & Sentance, S. (2018) Comparing K-5 teachers' reported use of design in teaching programming and planning in teaching writing. Proceedings of the 13th Workshop in Primary and Secondary Computing Education. Available from: DOI:10.1145/3265757.3265761.

Waite, J., Curzon, P., Marsh, W. D. & Sentance, S. (2020) Difficulties with design: the challenges of teaching design in K-5 programming. Computers & Education 150, 103838,ISSN 0360-1315. Available from: https://DOI.org/10.1016/j.compedu.2020.103838.

Theme 2:

Assessment tools

Computational Thinking Challenge: a pilot study on reliability and usability

Rina Lai (University of Cambridge)

Background and context

The paucity of assessment tools for teachers is a central factor that impedes the promotion of computational thinking (CT) in the classroom (Rich & Hodges, 2017). Indeed, psychometrically-sound, reliable, and valid CT assessments are scarce, if not absent, in the current literature. Thus, among many unaddressed questions regarding CT, the development and evaluation of assessment tools ought to be owed a special emphasis in the literature.

Research focus

This pilot study consists of three stages that analyse the reliability and usability of a computerised competency-based assessment of CT for secondary school learners: Computational Thinking Challenge (CTC). CTC automatically scores and provides personalised formative reports to students surrounding five components: problem decomposition, generalisation/ pattern recognition, algorithms, abstraction, and debugging. They were selected based on our survey with results aligning with the systematic literature review conducted by Kalelioglu et al. (2016). The report includes students' strengths, explanation of CT components, and thinking points designed to foster metacognitive strategies in CT.

Method

The first stage of the study evaluates the internal consistency reliability of CTC; the second stage investigates the comparative reliability between the computer-based version and the paper-based version of CTC. Cronbach's alpha was used to establish the internal consistency reliability in these two stages based on Kline (2000) and George & Mallery (2003): excellent (α>.9), good (.7<α<.9), acceptable (.6< α<.7), poor (.5< α<.6), unacceptable (α<.5). Extending from the first two parts, the third stage reports the results of a Likert-scale survey regarding user's experiences for students regarding CTC items. The survey aims to address two areas in CTC: difficulty/clarity of items and motivation/enjoyment.

The sample in stage one was composed of a total of 19 children whose age ranges from 14.9 to 16.2 (Mage=15.51; SDage = 0.35). The gender ratio was fairly unequal, though albeit reflects the gender imbalance reported by The Royal Society (2017); 17 (89%) of the participants were males and 2 (11%) were females. The sample in stage two and three was composed of a total of 24 children whose age ranges from 16.9 to 18.4 (Mage= 17.5; SDage = 0.37). Among the participants, 21 (87.5%) of them were males and 3 (12.5%) were females. The majority of participants in this group have had some coding/programming experience prior to the study (except for two non-responses): 47.4% (N=3) over 3 years; 21% (N=14) over 2 years; 5.7% (N=3) over 1 year and; 5.7% (N=2) over 6 months.

Findings

Using the aforementioned criteria, results in stage one suggests that CTC demonstrates good reliability, Cronbach's α = .79 (Mscale =9.21; SDscale = 3.73; σ2=13.95). Results in stage two suggest that the paper-based version has relatively poor reliability, Cronbach's α =.59, compared to the computer-based version that demonstrates high reliability, Cronbach's α = .78. Stage three of the study suggests half of the participants found the items to be "neither easy nor difficult" (N=12; 52.2%) and understood the questions "very well" (N=11; 47.8%). Regarding the motivational aspect, almost half of the participants found CTC to be "moderately" stimulating/ interesting/ fun (N=11; 47.8%), and enjoyed CTC "a little" (N=7; 30.4%) to a "moderate" (N=7; 30.4%) extent.

Conclusion and implications

The three-stage pilot study provides initial supporting evidence to the feasibility and quality of CTC. It is hoped that the outcome of this study is a properly validated assessment tool that allows for robust and reliable data on students' CT skills, contributing to future research and practice. The main study will continue to evaluate the quality of CTC from a psychometric (item response theory) and educational data mining approach in a large sample.

References

George, D., & Mallery, P. (2003) SPSS for Windows step by step: a simple guide and reference, 11.0 update (4th ed.). Boston: Allyn & Bacon.

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016) A framework for computational thinking based on a systematic research review. Baltic Journal of Modern Computing, 4, 583–596.

Kline, P. (2000) The handbook of psychological testing. Abingdon, Routledge.

Rich, P., & Hodges, C. B. (Eds.) (2017) Emerging research, practice, and policy on computational thinking. New York, Springer International Publishing.

The Royal Society (2017) After the reboot: computing education in UK schools. Available from: https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

Automated marking of free-text questions in STEM

Meurig Thomas and Alastair R. Beresford (University of Cambridge)

As well as providing resources for introducing concepts, the ideal online learning platform would ask questions of the learner to detect misconceptions and provide timely, meaningful feedback to correct them. Such online questions need to consider the trade-off between the possible space of user answers and the accuracy of the automated marking. In this talk we discuss the use of short-form, free-text questions to facilitate a wide range of possible student answers on the Isaac online learning platform as used by 'isaacphysics.org' and 'isaaccomputerscience.org'.

To consider the possible answer versus accuracy trade-off, we can look at the case of multiple choice questions. It is trivial to accurately mark multiple choice questions automatically, however, learners using this type of question are aided in their submissions by constraining the set of possible misconceptions and by testing familiarity rather than recall. These questions are less effective on an open online platform where users can create multiple accounts or submit multiple times.

To increase the range of possible answers, free-form text response questions can be adopted. This question type also matches the style of exam questions in many STEM subjects in the UK. We will discuss how we have been able to facilitate the creation and marking of free-text questions on an online platform used by GCSE to A-level learners and outline a view toward future approaches to finding better compromises in the trade-off space.

We will discuss the integration of The Open University's OpenMark rule-based model (Butcher, 2006; Butcher & Jordan, 2010) which we have used to mark 14,300 answers from 1030 students of physics and computer science since early 2019. The model is built from rules using a keyword matching technique with word-level wildcards and phrase-level modifiers to match out-of-order words, ignore additional words, and accommodate simple spelling mistakes through evaluating edit distances. We describe the method's strengths and weaknesses, and provide an analysis of its use in practice.

A demo will be given of the tool we have built to help facilitate the creation of free-text questions with rule-based marking. Our web-based tool enables question creators to manipulate the marking rules, and add example student answers to a test harness to ensure the automated marker matches expected results.

Despite the benefits of control and interpretability provided by a rule-based marking system, the expressiveness of language means that the technique suffers from an undesirable rate of both false-positives and false-negatives. In addition, the rule writer needs a large corpus of sample student answers and many need to accommodate a wide range of ways to express a correct answer. We will then describe how several machine learning and natural language processing approaches may help overcome these limitations.

For example, inverting the test harness gives us a supervised learning classification problem where rules can be learnt from labelled examples. The problem with this approach is that supervised learning usually requires a large amount of training data and we cannot expect our question creators to create thousands of correct and incorrect responses for each of their questions.

We believe more promising possibilities can be found by leveraging recent advances in natural language processing. We will finish the talk by describing several new approaches which use sentence embedding to automatically mark answers.

References

Butcher, P. G. (2006) OpenMark examples. The Open University. Available from: http://www.open.ac.uk/openmarkexamples

Butcher, P.G. & Jordan, S.E. (2010) A comparison of human and computer marking of short free-text student responses. Computers & Education, 55(2), 489-499.

Theme 3:
Application of theoretical frameworks

Semantic waves: analysing the effectiveness of computing activities

Paul Curzon (Queen Mary University of London), **Jane Waite** (Queen Mary University of London), **and Karl Maton** (The University of Sydney)

Background and context

Computer science is being introduced at school level worldwide, but with little existing research into appropriate pedagogy, and with many teachers having little experience to build on. Different teaching approaches have emerged with differing degrees of success. Simple ways are needed to help teachers predict the effectiveness of activities and identify ways to improve them.

Research focus

The notion of 'semantic waves' forms part of Legitimation Code Theory, an educational theory by Maton (2013) that has been successfully applied in many disciplines. Its focus on changes in the context-dependence (semantic gravity) and complexity (semantic density) of knowledge offers a good way to think about what makes an effective learning experience. Its utility in understanding the teaching of computing has been argued for by Curzon et al. (2018), Curzon (2019), Waite et al. (2019), Curzon and Grover (to appear). The theory gives a way to think about why different teaching approaches work or do not. It can be used to evaluate individual or sequences of lesson plans, online resources, and to teach students how to write good explanations. To date it has only been applied to one computing activity (Waite et al., 2019) — we provide further evidence of its applicability in computing contexts.

Method

We applied semantic wave analysis to two activities / approaches. The first unplugged programming activity, 'box variables' (Curzon, 2014) was chosen as it was known to be very effective (from student feedback and peer review). The second activity followed the copy-code style (easy for novice teachers), but suggested to be an ineffective way to teach programming. For each activity, the combined changes in semantic gravity and semantic density (concrete, everyday concepts in everyday language versus abstract concepts in technical language) were plotted in a simple, coarse-grained way.

Findings

The profile of the box variables activity was found to follow a semantic wave of moves up and down in context-dependence and complexity, supporting the packing and unpacking of subconcepts. The theory predicts this to be a strong approach. However, analysis highlighted that the repacking section was weak and suggested an improvement of having the students summarise the points learned. By contrast, the copy-code activity had a stepped 'down escalator' unlinked structure, with no unpacking and unpacking support. The theory suggests this lack of linking and repacking would be a critical factor in its lack of effectiveness.

A workshop was delivered to in-service teachers, explaining the theory and having the attendees draw the semantic waves for the activities before discussing them. Attendees were able to draw semantic waves, and doing so led to a discussion about how the copy-code activity could be improved. Participant feedback was very positive about the approach (though this remains informal as yet).

Conclusion and implications

Semantic waves provide a powerful way to think about the effectiveness of a learning activity as well as its particular delivery, and so improve teaching. By looking at changes in semantic density and semantic gravity of an activity, the structure of the activity can be improved. The analysis helps explain how and why unplugged activities can be an effective form of teaching computing, and why ways to teach programming adopting a copy-code strategy may be ineffective. The analysis also suggests how to improve such approaches. This leads one from copy-code activity to code-predicting type approaches, such as PRIMM (Sentance & Waite, 2017), for example.

We believe the theory and analysis techniques behind semantic waves should become a standard part of teacher training for computing teachers both for initial teacher training and as part of continuing professional development.

References

Curzon, P. (2014) The box variable activity. Available from: https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-box-variable-activity/

Curzon, P., McOwan, P.W., Donohue, J., Wright, S., & Mars, D.W. (2018) Teaching computer science concepts. Computer science education: perspectives on teaching and learning in school, S. Sentance, E. Barendsen, & C. Schulte (Eds.), 91-108. Bloomsbury Publishing, London.

Curzon. P. (2019) Follow semantic waves, tip 9 of learning to learn to program. Available from: https://teachinglondoncomputing.org/learning-to-learn-to-program/ An informal blog on practical ideas about teaching programming

Curzon, P. & Grover, S. (to appear), Guided exploration for introducing programming concepts through unplugged activities, chapter in a forthcoming book.

Legitimation Code Theory (n.d.). Available from: http://legitimationcodetheory.com/

Maton, K. (2013) Making semantic waves: a key to cumulative knowledge-building. Linguistics and Education 24, 8-22.

Sentance, S. & Waite, J. (2017) PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. Proceedings of the 12th Workshop on Primary and Secondary Computing Education, 113-114. ACM. Available from: DOI 10.1145/3137065.3137084

Waite, J., Maton, K., Curzon, P. & Tuttiett, L. (2019) Unplugged computing and semantic waves: analysing crazy characters. UKICER: Proceedings of the 1st UK & Ireland Computing Education Research Conference. University of Kent, Canterbury, UK. ACM. Available from: DOI: 10.1145/3351287.3351291

Understanding conceptual transfer in second and subsequent programming languages

Ethel Tshukudu and Quintin Cutts (University of Glasgow)

As students learn computer science (CS), they will transfer skills and understanding from one programming language (PL) to another, many times. Curricula allow flexibility such that there is no dominant sequence of languages used. For example, students can get their first exposure to PL concepts in primary schools via block-based languages, such as Scratch or Alice, and then move on to a text-based language later in secondary school, such as Python and Java. Literature has shown that transitioning between programming languages remains a significant challenge for students' conceptual understanding (Kölling & Altadmri, 2015; Nelson et al., 1997; Powers et al., 2007; Walker & Schach, 1996). In line with code comprehension research (Schulte et al., 2010), we believe that for students to transition effectively, they have to first understand programs in the new language before they can write them.

We have undertaken three studies that support a model that we have developed of PL transfer via program comprehension. These are based on the notion of semantic transfer theory (Jiang, 2004) and mindshift theory (Armstrong & Hardgrave, 2007). Firstly, an in-depth qualitative study was conducted on five students transitioning from procedural Python to object-oriented Java via semi-structured interviews held fortnightly for a period of ten weeks. Students were asked to talk through as they carried out code comprehension exercises in Python and Java. The data was transcribed, coded, and analysed. The second study was a quantitative within-participant study which investigated 120 students by exposing them to the same experimental conditions as they transitioned from procedural Python to object-oriented Java. A Java pre-quiz on code comprehension was given on the third week of learning Java, followed by intervention on awareness of similarities and differences between the two languages by the instructor on the fourth week, and finally a post-quiz was given on the eighth week. The third study was a within-participant quantitative study conducted at a different university and investigated 277 students transitioning from object-oriented Python to object-oriented Java in their first lesson of learning Java. These students were asked to guess the results of hand executing the Java code given based on their Python knowledge.

The results showed several commonalities across the three studies. The findings support the model, indicating that during the initial learning stages, learners relied mostly on their syntactic matching between Python and Java and subsequent semantic transfer which affected their learning of Java concepts both positively and negatively. The semantic transfer is positive for learning on constructs that share similar syntax and semantics, negative on constructs that share similar syntax but different semantics, and lastly there is little or no transfer on constructs that do not share similar syntax although they have the same underlying semantics. Lastly results showed that instructional interventions on PL transfer helped students to significantly transfer their conceptual understanding from Python to Java.

In this paper, we will be exploring how these studies and our model carry over to the prevalent school context of block-based to text-based transfer. We will also argue about the role of the teacher in PL transfer and how explicit reference to similarities and differences between known and new PL during instruction might benefit students faced with transition challenges. Although significant work has been carried out in this area, it has not been via this code comprehension model of PL transfer, which has new insights to offer.

References

Armstrong, D.J. & Hardgrave, B.C. (2007) Understanding mindshift learning: the transition to object-oriented development. MIS Quarterly, 453–474.

Jiang, N. (2004) Semantic transfer and its implications for vocabulary teaching in a second language. Modern Language Journal, 88(3), 416–432.

Kölling, M., Brown, N.C.C., & Altadmri, A. (2015) Frame-based editing: easing the transition from blocks to text-based programming. Proceedings of the Workshop in Primary and Secondary Computing Education, 29–38. ACM.

Nelson, H.J., Irwin, G., & Monarchi, D.E. (1997) Journeys up the mountain: different paths to learning object-oriented programming. Accounting, Management and Information Technologies, 7(2), 53–85.

Powers, K., Ecott, S., & Hirshfield, L.M. (2007) Through the looking glass: teaching CS0 with Alice. SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, 213–217.

Schulte, C., Clear, T., Taherkhani, A., Busjahn, T., & Paterson, J.H. (2010) An introduction to program comprehension for computer science educators. ITiCSE-WGR '10: Proceedings of the 2010 ITiCSE working group reports, 65–86. ACM.

Walker, K.P., & Schach, S.R. (1996) Obstacles to learning a second programming language: an empirical study. Computer Science Education, 7(1), 1–20.

Theme 4:

Perceptions and attitudes

Exploring resilience for effective learning in computer science education

Tom Prickett (Northumbria University), **Tom Crick** (Swansea University), **Morgan Harvey** (University of Sheffield), **Julie Walters** (Northumbria University), **and Longzhi Yang** (Northumbria University)

Background and context

Many factors have been shown to be important for supporting effective learning and teaching — and thus progression and success — in formal educational contexts. While factors such as introductory-level computer science knowledge and skills — as well as pre-university learning and qualifications — have been extensively explored, the impact of measures of positive psychology are less well understood for the discipline of computer science. This preliminary work investigates the relationship between effective learning and success, and two measures of positive psychology: grit (Duckworth's 12-item grit scale) (Duckworth et al., 2007), and the Nicholson McBride resilience quotient (NMRQ) (Clarke, 2010). The subjects will be first year computer science undergraduates, to provide insight into the factors that impact on the transition from secondary to tertiary education.

Research methods

This quantitative study was conducted by incorporating two survey-based measures of positive psychology — grit and resilience — into the teaching of a first year core subject as part of a UK computer science degree programme in February 2019. Students were asked to complete the surveys using the university's electronic learning platform. The students were supported in the interpretation of their results and guidance was provided regarding strategies to adopt to improve their degree studies. The study was approved by the university's ethics board and student consent was explicitly obtained to use their data for research. Data on student performance was obtained at the end of the teaching year and consists of the results from five different subjects over both semesters of the academic year as well as attendance data over the year.

The data was analysed by a combination of correlation analysis and logistic regression. The intention of the logistic regression was to explore the potential strength of relationship rather than to develop a model for predictive purposes.

Findings

Data was captured related to grit (N=58) and resilience (N=50) questionnaires and coaching. Analyses demonstrate that resilience is statistically significantly (1% level) linked (correlation analysis and logistical regression) to attendance and performance for individual subjects and year average marks. However, this was not the case for grit.

Conclusion and implications

Promoting effective learning and student success remains a challenge in computer science, with high failure rates reported in foundation areas, such as introductory programming (Bennedsen & Caspersen, 2019; Watson & Li, 2014). The results of this preliminary study demonstrate that the 12-item resilience scale could be a factor in promoting success, but that the same was not true for the 12-item grit scale. The results of this single-institution study lead to a number of possibilities for future work analysing the transition from secondary education to tertiary education, such as providing insight into learner attitudes, behaviours, and dispositions, especially how this links to the teaching and assessment of key curricula concepts in computer science. For example:

i) Initiatives related to the active development of student resilience can be deployed and their effectiveness evaluated

ii) Replicating the study with larger cohorts and at other schools/colleges/universities to validate this study, increasing the sample size and strengthening the statistical basis.

iii) Using resilience in predictive models alongside other key factors in order to further augment and enhance the prediction of student success.

Alongside substantial national curriculum and qualifications reform across the four nations of the UK (Brown et al., 2014), as well as a significant socio-economic push to produce more graduates with 'high-value' digital, data, and cyber skills (Crick et al., 2020; Crick et al., 2019; Tryfonas & Crick, 2018), these changes are being monitored and replicated internationally.

With changes to curricula, as well as rethinking programmes, pedagogies, and practice, we thus recognise similar challenges and opportunities in a number of other jurisdictions, providing a platform for replicability, portability and extension of this work.

References

Bennedsen, J. & Caspersen, M.E. (2019) Failure rates in introductory programming: 12 years later. ACM Inroads 10(2), 30–36.

Brown, N.C.C., Sentance, S., Crick, T., & Humphreys, S. (2014) Restart: The resurgence of computer science in UK schools. ACM Transactions on Computer Science Education 14(2), 1–22.

Clarke, J. (2010). Resilience: Bounce back from whatever life throws at you. Crimson Publishing, USA.

Crick, T., Davenport, J.H., Hanna, P., Irons, A., & Prickett, T. (2020) Computer Science degree accreditation in the UK: a post-Shadbolt review update. Proceedings of Computing Education Practice, article 6, 1-4. ACM.

Crick, T., Davenport, J.H., Irons, A., & Prickett, T. (2019) A UK case study on cybersecurity education and accreditation. Proceedings of 49th Annual Frontiers in Education Conference (FIE 2019). IEEE.

Duckworth, A., Peterson, C., Matthews, M.D. & Kelly, D.R. (2007) Grit: Perseverance and passion for long-term goals. Journal of Personality and Social Psychology 9(6), 1087–1101.

Tryfonas, T. & Crick, T. (2018) Public policy and skills for smart cities: the UK outlook. Proceedings of 11th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'18), 116-117. ACM.

Watson, C. & Li, F.W.B. (2014) Failure rates in introductory programming revisited. Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE'14), 39-44. ACM.

How is programming taught in code clubs? Experiences, gender perceptions, and learning barriers experienced by code club teachers

Efthimia Aivaloglou and Felienne Hermans (Leiden Institute of Advanced Computer Science)

Background and context

Programming is a popular extracurricular activity for school children. International programs like CoderDojo or Code Club are supporting an increasing number of after-school programming clubs. Programming classes are also given in independent code clubs, coding summer camps, workshops in libraries and museums, as well as various other venues. Code club lessons are different from in-school programming classes in terms of setting and lesson material, whereas the learning environment in code clubs has already been found to affect student emotions (McKelvey & Cowan, 2017) and motivation (Butler et al., 2018).

Research focus

The goal of our research is to explore how programming is taught at code clubs through the experiences and perceptions of code club teachers. We investigate if the learning barriers that have been reported by school teachers in programming (Dorn et al., 2018), including motivation, commitment, and abstraction capacity, also apply to students of code clubs. We also aim to determine if the gender perceptions that have been found to apply to school teachers (Funke et al., 2015), for example about the structuredness, self-confidence, and scientific curiosity of their students, also apply to code club teachers.

Method

Towards this direction, we conducted an exploratory survey where we invited code club teachers to answer to a combination of closed and open-text questions. The survey received 98 complete responses from people teaching at CoderDojos, Code Clubs, various country-specific programs, and independant after-school programming clubs. We performed quantitative analysis of the data collected through the survey, as well as qualitative analysis of the open-text responses.

Findings

The results from the survey and the qualitative analysis of the open-text responses suggest that (1) the teaching material and practices vary between different code clubs and different programs, but (2) plenary sessions and summative assessments are rare. Also, (3) motivation and commitment are rarely identified as learning barriers for code club students, whereas debugging, error messages, and abstract thinking are the most commonly reported difficulties. Gender differences are perceived by code club teachers, with (4) boys in code clubs being perceived as more confident and having more familiarity and prior knowledge of programming, whereas (5) persistence, concentration, responsiveness to instruction, collaboration skills, grit, and structuredness are considered increased for girls.

Conclusion and implications

Our findings highlight the differences between code club lessons and after-school programming classes, with motivation and commitment being rarely reported for the former, but within the most commonly identified learning barriers in the latter. Within the perceived gender differences, we find that the increased familiarity and prior knowledge on programming of male students is a concerning factor, because it is known from existing work that prior programming experience strongly affects learning performance (Wilcox & Lionelle, 2018) and CS career attractiveness (Aivaloglou & Hermans, 2019).

References

Aivaloglou, E. & Hermans, F. (2019) Early programming education and career orientation: the effects of gender, self-efficacy, motivation and stereotypes. Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), 679-685. ACM, New York, NY, USA.

Butler, N., Flood, C., & Power, A. (2018) What motivates a Ninja? An exploration of students' CoderDojo experience. Cyberpsychology and Society: Current Perspectives, Andrew Power (Ed.). Abingdon, Routledge.

Dorn, N., Berges, M., Capovilla, D., & Hubwieser, P. (2018) Talking at cross purposes: perceived learning barriers by students and teachers in programming education. Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18), article 12. ACM, New York, NY, USA.

Funke, A., Berges, M., Mühling, A. & Hubwieser, P. (2015) Gender differences in programming: research results and teachers' perception. Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15), 161-162. ACM, New York, NY, USA.

McKelvey, N. & Cowan, P. (2017) Valence at CoderDojos: an exploration. Literacy Information and Computer Education Journal (LICEJ) 8(1), 2525–2533.

Wilcox, C. & Lionelle, A. (2018) Quantifying the benefits of prior programming experience in an introductory computer science course. Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18), 80-85. ACM, New York, NY, USA.

Survey of female A-level CS students: sense of social purpose, sense of belonging, and hedging

Lynne Blair, Lisa Thomas, and Emily Winter (Lancaster University)

Background and context

The percentage of women in the tech workforce stands at 17% (BCS, 2019) and, in education, 21.4% of females complete Computing at GCSE (JCQ, 2019-a), and 13.3% complete CS A level (JCQ, 2019-b). There is widespread concern how this gender imbalance and persisting image of CS as a masculine discipline puts off otherwise interested and competent women.

Research focus and method

A survey of 56 female A-level CS students from schools/colleges across the north-west (Lancashire) region sought insights into perceptions of CS and reflections on future study/careers. Our particular focus was better understanding the experiences of students in our local context. To recruit participants, contact was made with CS teachers at all A-level CS providers in our region. Teachers were asked if they would help distribute a questionnaire to their female CS students. Student participation was voluntary.

Findings

Thematic analysis uncovered two themes — sense of social purpose and sense of belonging — plus the frequent use of 'hedging', a sociolinguistic device commonly identified in female speech (Murphy, 2010).

Sense of social purpose: Regarding important aspects for their future careers, respondents ranked the "development of technical skills" first, followed closely by "improving people's lives", and "improving society", with weighted counts of n=105, 100, 97 respectively.

Open responses to projects showed students valuing "making a difference" (R22), and "real-world users" and "realistic situations" (R47). Examples included simulating "staff movement in a hospital" (R5), a "voting machine [hardware and software] to be used in school" (R7), an encrypted "booking system for a local music school" (R17), and "dyslexia screening" (R37).

However, some respondents felt a disconnect: a CS career was not "helping people in need" nor "a caring career" (R10). Similarly:

"I want to impact on people's lives (social action) and I don't think this can be done with CS and, if it can be done, I haven't been educated how to." (R15)

Sense of belonging: Identified as an important predictor of success and retention (Rainey et al., 2018, Sax et al., 2018), yet a realisation of not fitting in (R49) and wishing for "more people you can easily relate to" (R3) continues to persist.

Some students were concerned/surprised by the gender gap: "I wasn't expecting to be the only girl in my class" (R49, similarly R19, R21). Stronger emotions included: "intimidated" (n=4); "uncomfortable" or "worried" (n=3); or "disconcerted", "scared", "out of place" (n=2).

In contrast, positive comments included "it may be an advantage to be a woman in CS" (R44), feeling "more unique" (R30), and noting:

> "Girls who enter pave the way for decreasing the gap." (R27, similarly R35)

Other responses indicated determination "to take it because I love it" (R17) and resilience:

> "I have to work harder than men to gain men's respect. I will not let this prevent me from studying CS [...] and I'm not going to stop pursuing my ideal career." (R18, similarly R20)

Hedging: Faced with negative feelings, many respondents were quick to add diminishing clauses. The extent was surprising, with n=20 responses clearly exhibiting this linguistic device:

> "Uncomfortable but would get used to it" (R12, similarly R15, R39, R48)

> "Out of place but would still enjoy it" (R24)

> "Isolated at first but once I've made friends, I think I'd be ok" (R25)

Conclusion and implications

These female A-level students have already developed mechanisms of resilience and a 'way of being' in a predominantly male CS education environment. Yet the data shows a fragility regarding sense of purpose and sense of belonging. Many hedged responses embrace an individualistic perspective that stresses self-responsibility to fit in and succeed. This offers a challenge to all in CS — responsibility to adapt also lies at departmental/institutional and discipline levels.

References

BCS (2019) Almost half of women in IT think gender is their biggest barrier to promotion. Available from: https://www.bcs.org/more/about-us/press-office/press-releases/almost-half-of-women-in-it-think-gender-is-their-biggest-barrier-to-promotion/

Joint Council for Qualifications (JCQ) (2019-a) GCSE (full course) results summer 2019. Available from: https://www.jcq.org.uk/examination-results/gcses/2019/main-results-tables/gcse-full-course-results-summer-2019

Joint Council for Qualifications (JCQ) (2019-b) A level and AS results summer 2019. Available from: https://www.jcq.org.uk/examination-results/a-levels/2019/main-results-tables/a-level-and-as-results-summer-2019

Murphy, B. (2010) Corpus and sociolinguistics: investigating age and gender in female talk. Studies in Corpus Linguistics (vol. 38). John Benjamins Publishing.

Rainey, K., Dancy, M., Mickelson, R., Stearns, E. & Moller, S. (2018) Race and gender differences in how sense of belonging influences decisions to major in STEM. International Journal on STEM Education, 5(10).

Sax, L.J., Blaney, J.M., Lehman, K.J., Rodriguez, S.L., George, K.L. & Zavala, C. (2018) Sense of belonging in computing: the role of introductory courses for women and underrepresented minority students. Social Sciences, 7(8), 122–144.

Poster presentations

Learning graphs: a strategic approach to computing curriculum planning

George Boukeas, Andy Bush, Rebecca Franks, Ben Garside, Sway Grantham, Ben Hall, and Allen Heard (Raspberry Pi Foundation)

The National Centre for Computing Education (NCCE) is developing a comprehensive curriculum package of more than 500 hours worth of teaching resources, to support the delivery of the English national curriculum for computing across all Key Stages. The resources need to be planned down to the learning objectives and activities of each individual lesson, ensuring coverage of the national curriculum, consistency, and pedagogical robustness.

The scale and complexity of this planning problem is tackled by organising the content (concepts, knowledge, skills, objectives) into interconnected networks called learning graphs. Depending on the level of abstraction, the nodes in a learning graph could contain anything ranging from the contents of a curriculum strand across an entire Key Stage, to the learning objectives of a six-lesson unit. Nodes are connected if they represent two adjacent waypoints in the learning process and will often form clusters, corresponding to specific themes.

Learning graphs are similar to existing approaches used for describing learner journeys through knowledge, concepts, or skills: learning trajectories, learning progressions, and learning maps are common terms of similar flavour encountered in the literature (Achieve, 2015; Clements & Sarama, 2004; Kingston & Broaddus, 2017; Sztajn et al., 2012; Cambridge Mathematics, n.d.). There is variation in how these approaches are defined and to what purpose they are used. Learning graphs are different in that they directly inform our lesson planning decisions and are thus "translated into usable tools for teachers" (Daro et al., 2011). Also, learning graphs are (currently) empirical, instead of research- or evidence- based, since little is known about learning waypoints in computing (Guzdial & Morrison, 2016). There is recent work on learning trajectories for computational thinking concepts (Rich et al., 2018; Rich et al., 2017).

A collaborative, iterative process for producing learning graphs is followed. The content developers, teachers, researchers, and members of the NCCE academic board are involved in generating and reviewing the graphs. Initially, the graphs produced are in a 'fluid' state: they uncover the structure of the content and the possible journeys through it, without being bound to a specific teaching pathway. Eventually the graphs reach a 'solid' state, where the nodes are further organised and arranged to reflect specific suggestions on the order that the content could be delivered.

The use of learning graphs for planning has been evaluated through a series of interviews and discussions with the content developers, who report significant merits. In contrast to lists of curriculum statements or learning objectives, learning graphs reveal the non-linear structure of the content and lead to critical thinking about the relationships between different components, which directly impacts the structure and sequence of the lessons. They highlight possible gaps between learning waypoints. They are also instrumental in clarifying terminology and using it consistently.

On the other hand, learning graphs can get large, complicated, and interwoven. Structuring them in a clear way can be challenging and it is evident that a purpose-built tool is necessary for working efficiently with them.

Our aim is to further investigate research into similar approaches, to see how learning graphs can be put to other uses, such as informing assessment and pedagogy. Another focus is on how to capture and structure teacher feedback on learning graphs, for the purpose of improving and refining them. Finally, we would also like to use teacher feedback to understand how they can be used by teachers, such as for understanding the 'flow' of content through a unit's lessons, ensuring curriculum coverage, or justifying teaching decisions.

Our grand vision is that the learning graphs produced in the context of the NCCE will serve as the starting point for a comprehensive set of learning waypoints for computing education.

References

Achieve (2015) The role of learning progressions in competency-based pathways. Available from: https://www.achieve.org/files/Achieve-LearningProgressionsinCBP.pdf

Cambridge Mathematics (n.d.) An update on the Cambridge mathematics framework. Retrieved January 2020 from: www.cambridgemaths.org/Images/cambridge-mathematics-framework.pdf

Clements, D. & Sarama, J. (2004) Learning trajectories in mathematics education. Mathematical Thinking and Learning, 6, 81-89.

Daro, P., Mosher, F., Corcoran, T., Barrett, J., Battista, M., Clements, D., Confrey, J., Daro, V., Maloney, A., Nagakura, W., Petit, M., Sarama, J. (2011) Learning trajectories in mathematics: a foundation for standards, curriculum, assessment, and instruction. CPRE Research Report # RR-68.

Guzdial, M. & Morrison, B (2016) Growing computer science education into a STEM education discipline. Communications of the ACM 59(11), 31–33.

Kingston, N. & Broaddus, A. (2017) The use of learning map systems to support the formative assessment in mathematics. Education Sciences, 7(1), 41.

Rich, K.M., Binkowski, T.A., Strickland, C. & Franklin, D. (2018) Decomposition: A K-8 computational thinking learning trajectory. Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18), 124–132.

Rich, K.M., Strickland, C., Binkowski, T.A., Moran, C. & Franklin, D. (2017) K-8 learning trajectories derived from research literature: sequence, repetition, conditionals. Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17), 182–190.

Sztajn, P., Confrey, J., Holt Wilson, P. & Edgington, C. (2012) Learning trajectory based instruction: toward a theory of teaching. Educational Researcher 41(5), 147-56.

Mapping the use of physical computing at Key Stage 2 in England

Katharine Childs (Nottingham Trent University)

Background and context

Computing was introduced into the English national curriculum in 2014, replacing the disaggregated subject of ICT. After the reboot (Royal Society, 2017) evaluated existing provision and identified some significant challenges that must be overcome to fully embed the curriculum consistently across all schools. This research responds to a suggestion from Crick (2017) to map with greater clarity what is actually being taught in schools and focuses on physical computing, one of the four pedagogical contexts for teaching computer science identified by Waite (2017).

Research focus

At Key Stage 2 (ages 7–11), pupils are required to design and write programs including those to control or simulate physical systems (DfE, 2013). However, there is little available evidence about which devices teachers are (or are not) using in the classroom to deliver this curriculum requirement. Pedagogy starts with planning, and teachers are influenced by a number of different factors when they plan a scheme of work: their school, the curriculum, themselves as a practitioner and, specifically for computing, their community of practice (Sentance & Humphreys, 2018). This framework of four influencing factors was used to explore how teachers decide whether to include physical computing in their classroom, and if they do, which devices they are choosing to use.

Method

An opportunistic sample of teachers who use online communities was surveyed using an electronic questionnaire, which was first piloted and evaluated. Three online communities were selected for their ability to reach suitable Key Stage 2 teachers in England: Computing at School, Primary Rocks and TES. The survey questions asked teachers about the content of their 2018/2019 plans and schemes of work, and then used digital images to help them identify which devices they were planning to use, if any.

Findings

69% of respondents (n=54) were qualified teachers, with the remaining 31% reporting other job roles, ranging from a Specialist Learning Advisor (Computing) to an IT Technician / Teaching Assistant. Only 57% (n=54) of the respondents had the role of computing coordinator in their school. The most surprising aspect was that the 43% of respondents taught all four KS2 year groups. This subgroup (n=23) taught in all different sizes of school, and of these 23 teachers, 7 were not computing subject coordinators.

Of the four factors influencing curriculum design, teachers were most positive about the curriculum area. There was strong support for both the benefits of using physical computing and its use to deliver learning outcomes. As would be expected from a sample of teachers who use online communities, almost two-thirds of respondents felt they had opportunities to network with other teachers and share ideas.

However at school level, the time to prepare physical computing activities and access to enough equipment were both barriers to its use, which aligned with previous research in the field (Pryzbylla et al., 2017). At practitioner level, almost half the teachers surveyed felt they lacked access to high-quality written resources and training opportunities to use physical computing in their lessons, although a majority of teachers (74%) felt they had a good understanding of the learning outcomes that physical computing activities could deliver. Devices which can be seen as more time consuming were the most prevalent devices used in the classroom, specifically the BBC micro:bit and the Crumble microcontroller boards.

Conclusion and implications

Teachers need access to high-quality resources and training to support them to teach computing using the most prevalent physical computing devices. Further qualitative research is needed to produce case studies which evidence learning outcomes when using physical computing and which investigate practical solutions to the time problems faced by teachers.

References

Crick, T. (2017) Final draft: Computing education: an overview of research in the field. The Royal Society, London. Available from: https://royalsociety.org/-/media/policy/projects/computing-education/literature-review-overview-research-field.pdf.

Department for Education (DfE) (2013) Computing programmes of study: key stages 1 and 2. House of Commons, London. Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study#key-stage-2

Przybylla, M., Henning, F., Schreiber, C. & Romeike, R. (2017) Teachers' expectations and experience in physical computing. In: Dagienė V., Hellas A. (eds.). Informatics in Schools: Focus on Learning Programming. ISSEP 2017. Lecture Notes in Computer Science, vol 10696, pp 49-61. Cham, Springer.

Royal Society (2017) After the reboot: computing education in UK schools. Available from: https://royalsociety.org/~/media/events/2018/11/computing-education-1-year-on/after-the-reboot-report.pdf

Sentance, S. & Humphreys, S. (2018) Understanding professional learning for Computing teachers from the perspective of situated learning. Computer Science Education, 28(4), 1-26. Available from: https://DOI.org/10.1080/08993408.2018.1525233

Waite, J. (2017) Pedagogy in teaching computer science in schools: a literature review. The Royal Society, London. Available from: https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf.

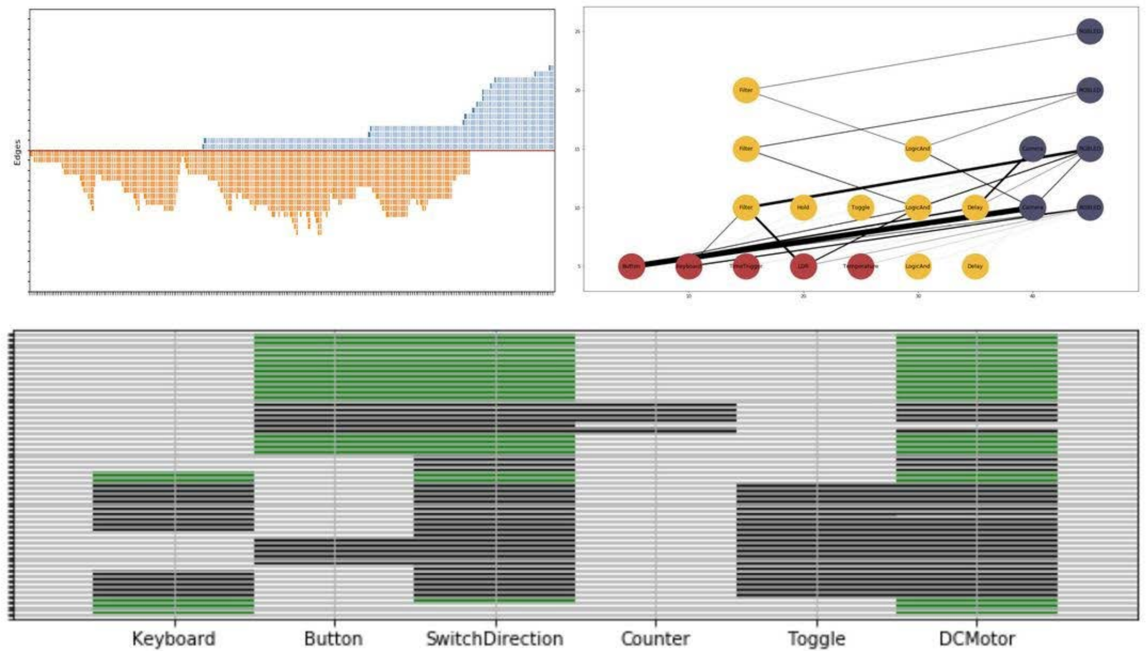
The journey is the destination: process-oriented data visualisations to explore appropriation in open-ended robotics

Veronica Cucuiat (University College London)

Open-ended project-based learning activities are rooted in constructivist, student-centred, growth-based pedagogies which emphasise a trial and error learning process over perfect end results (Berland et al., 2014; Cukurova et al., 2016; Papert, 1993).

In addition, open-ended projects offer students the chance to externalise their own interpretation of the ideas they are exposed to, associate them with previously existing concepts, and in that process, expand their corpus of personal understanding (Duckworth & Yeager, 2015; Polman, 2006). The open-ended aspect of project-based learning is explored in the educational theory literature via the concept of appropriation, as the self-constructive component of activity (Poizat et al., 2013; Wertsch, 1991). However current project-based learning assessments are usually based on the resulting end-product of the project, such as an artefact or a portfolio, overlooking the actual cognitive and development processes that occur in the process (Black & Wiliam, 2010; Resnick et al., 1970), such as appropriation.

The study uses trace log data which stores every version of the students' work to produce visualisations which showcase the experimentation process, during the project timeline, independent of final outcomes. Drawing upon quantitative ethnography (Shaffer, 2017) and the theory of appropriation (Rogoff, 1995; Wertsch, 1991), the trace logs data was structured across three dimensions of variety, validity, and complexity to characterise the students' actions (Fry, 2007). Aspects of students' appropriation of the SAM Labs kits are discussed as they emerge from comparisons between students' experimentation process.



The data was collected at a London school, during a real-life design and technology project spanning over six months, with 18 children aged 10–12. The children built intelligent board games, inspired by traditional games but enhanced with electronic behaviours built using the SAM Labs blocks, such as automatic dice, trap doors, light and sound effects as the players progress on the board. Each version of the virtual circuits was logged to record the modifications students make to their artefacts. These provide a versioning of the graphs throughout the project timeline.

From the visualisations, appropriation comes through in various forms:

- the different blocks used to achieve the same SAM behaviour
- the different experiments of achieving the same SAM behaviour
- the different contexts of using the same block
- the different experimentation volume towards achieving the same SAM behaviour
- the different final implementations of the same SAM behaviour

Three emerging themes emerges from initial explorations with teachers:

1. showing the construction journey in full has value in understanding individual students' nuances
2. cross-contextual applications of the same functionality can differ significantly
3. emerging tensions between complexity, variety and validity as students experiment

References

Berland, M., Baker, R., & Blikstein, P. (2014) Educational data mining and learning analytics: applications to constructionist research. Technology, Knowledge and Learning, 19. Available from: DOI: 10.1007/s10758-014-9223-7.

Black, P. & Wiliam, D. (2010) Inside the black box raising standards through classroom assessment. Available from: DOI: 10.1177/003172171009200119. http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=epidoc1.in]/?t2000=022921/(100)

Cukurova, M., Avramides, K., Spikol, D., Luckin, R. & Mavrikis, M. (2016) An analysis framework for collaborative problem solving in practice-based learning activities: a mixed-method approach. Proceedings of the Sixth International Conference on Learning Analytics & Knowledge (LAK '16), 84-88. ACM, New York, NY, USA. Available from: DOI: 10.1145/2883851.2883900.

Duckworth, A. & Yeager, D. (2015) Measurement matters: assessing personal qualities other than cognitive ability for educational purposes. Educational Researcher, 44(4), 237–251. Available from: DOI: 10.3102/0013189X15584327.

Fry, B. (2007) Visualizing data. Newton, O'Reilly Media.

Papert, S. (1993) The children's machine: rethinking school in the age of the computer. New York, NY, USA, Basic Books, Inc.

Poizat, G., Haradji, Y. & Adé, D. (2013) When design of everyday things meets lifelong learning. . . International Journal of Lifelong Education, 32(1). Available from: DOI: 10.1080/02601370.2012.734485.

Polman, J. (2006) Mastery and appropriation as means to understand the interplay of history learning and identity trajectories. Journal of The Learning Sciences, 15(2), 221–259. Available from: DOI: 10.1207/s15327809jls1502_3.

Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K. & Silverman, B. (1970) Digital manipulatives: new toys to think with, 281–287.

Rogoff, B. (1995) Observing sociocultural activity on three planes: participatory appropriation, guided participation, and apprenticeship. In: In J. V. Wertsch, P. del Rio, & A. Alvarez (Eds.), Sociocultural Studies of Mind, 139-164. Cambridge, UK, Cambridge University Press.

Shaffer, D. (2017) Quantitative ethnography. Madison, Cathcart Press.

Wertsch, J.V. (1991) Voices of the mind: a sociocultural approach to mediated action. Cambridge, Massachusetts, Harvard University Press.

Effective use of mathematical equations in an online learning environment

Andrea Franceschini, James P. Sharkey, and Alastair R. Beresford (University of Cambridge)

Mathematical analysis is an essential tool in the practice of science, technology, engineering, and mathematics, and consequently it is important for students in these subjects to demonstrate effective use of mathematics (Goldstone & Landy, 2009). In this talk we are interested in supporting the use of mathematical equations in an online learning environment; in particular, we require methods of supporting both the entry and automated marking of mathematical equations, in order to support immediate personalised feedback to the learner.

We report on our experience designing, building, and using Inequality: an open-source formula entry tool which works across all major browsers, supports both mouse and touch-based entry, and is usable by high school children and teachers. Inequality is composed of a graphical, drag-and-drop front-end interface to build expressions in response to a question and a back-end service. It automatically marks answers entered with model answers for the given question as specified by our team of content creators, with various degrees of flexibility in how two expressions are considered equivalent.

Inequality has been used for nearly three years to support over 20,000 students and 900 teachers of GCSE and A level Physics. Since May 2019, Inequality supported over 300 pupils and about 50 teachers with symbolic Boolean expressions as taught in A level Computer Science.

We describe the effect Inequality has on the behaviour and performance of students using our learning platform. We compared the behaviour of students who used approximately 350 physics and mathematics questions, in either multiple choice or symbolic format. Our analysis explored nearly 500,000 answer attempts and determined that 73% of the 350 questions required fewer attempts to answer correctly in symbolic format. Because the Boolean logic questions in our computer science platform were developed symbolically from the beginning, we currently cannot perform the same comparison.

We also looked at how formulae are constructed using Inequality across physics and computer science. We built action trees comprising actions such as dragging a symbol from the menu, attaching and detaching a symbol to another symbol, and so on. We found that, while there are a few recurring and efficient ways of building correct answers, many students arrived at correct answers in less efficient and sometimes more convoluted ways. For example, some students effectively used Inequality to manipulate an expression as they built it, adapting what they would otherwise do with pen and paper. We found very similar patterns in both physics and computer science, which tells us that students work in similar ways, and suggests that many of the benefits seen in the physics platform may translate to the computer science platform.

We asked some of our students from the physics platform to complete a feedback questionnaire that was essential in contextualising our quantitative analysis. We analysed 563 valid responses from students who were largely in Year 12 (typically aged 16–17) and started using the physics platform in the same year. The questionnaires produced three key findings:

1. Students found it more difficult to work with larger formulae — mainly due to some usability issues that we since fixed — but they generally do not avoid work that they think will require large formulae

2. Students have a slight preference towards Inequality as opposed to pen and paper when they need to manipulate formulae, and they find the editor helpful in working out solutions

2. Students do not find Inequality distracting or hindering in their workflow

References

Franceschini, A., Sharkey, J.P. & Beresford, A.R. (2019) Inequality: Multi-modal equation entry on the web. Proceedings of the Sixth ACM Conference on Learning @ Scale, 1–10. Available from: https://DOI.org/10.1145/3330430.3333625

Goldstone, R. & Landy, D. (2009) How much of symbolic manipulation is just symbol pushing? Proceedings of the Annual Meeting of the Cognitive Science Society, 31.

Communicating computer science in schools: a case study

Anandha Gopalan and Jackie Bell (Imperial College London)

In recent years there has been a drop in the numbers of students studying GCSE Computer Science in the UK (Cellan-Jones, 2019). This has been linked to a couple of factors, primarily the lack of support in schools for the new curriculum and the lack of awareness of the relevance of the subject in wider society. To support the learning of computing in schools, and help plug this gap a little, the Department of Computing at Imperial College London has developed a course titled 'Communicating computer science in schools'. In this course, undergraduate students go into schools to assist with and lead computer science lessons, spreading their knowledge and skills to the pupils, and helping to develop schools' competency in teaching modern computer science.

The new GCSE computing curriculum introduced in 2014 heralded a sea change in schools, as it replaced ICT with Computer Science, 2018 being the last year for the ICT GCSE. The ICT curriculum was very broad and consisted of fairly mundane activities concerned with 'data processing', such as spreadsheets and document preparation. The new GCSE curriculum, on the other hand, teaches all aspects of computer science, including programming and data presentation and analysis by computers. Unfortunately, there had been a decrease in the number of students taking the new GCSE as compared to 2016, as well as a decrease in the numbers taking ICT. If this trend continues, the numbers of secondary school students receiving a fair and representative course on modern computer science and its applications will be relatively small and there is a real concern that we will not have enough trained computer scientists in the future (Cellan-Jones, 2017).

'Communicating computer science in schools' (Imperial College London, n.d.) is an optional module which students can take either in the third or fourth year of their degree in lieu of another computing course in the department. Each student is placed with a different London-area school for a period of six to eight weeks (usually for around two hours per week). While taking this module, students support the teaching of computer science by gaining first-hand experience of the school environment.

At first, the students observe the host teacher and after they gain more experience and confidence, they take the lead and implement a special teaching project, which may be to teach a lesson themselves and/or plan some activities (such as code clubs, etc.). This project is planned directly in conjunction with a teacher at the school. To ensure the suitability of the student in terms of motivation and responsibility, we interview every student. We started with eight students and schools in 2013–14 and now have twenty students engaged in the programme. Many of the schools which started with the first cohort are still taking part in the programme and these schools are a mix of state, private, girls-only, boys-only, and co-ed schools.

Running this course has provided us with unique insights into the ground realities of teaching computing in schools. We observed that it was better to have a subject specialist teaching it, although a teacher who is passionate and willing to learn was more effective than an expert who was disinterested. Pupils seemed to be more engaged when shown the applications of what they were learning, especially when examples were tailored to them. Misconceptions about the subject abound and start very early on. Children love to learn concepts kinaesthetically, sometimes involving computer-based activities too early can put them off. It is very important to win the parents' support.

References

Cellan-Jones, R. (2017) Computing in schools - alarm bells over England's classes. BBC News. Available from: https://www.bbc.co.uk/news/technology-40322796

Cellan-Jones, R. (2019) Computing in schools in 'steep decline'. BBC News. Available from: https://www.bbc.co.uk/news/technology-48188877

Imperial College London. (n.d.) Communicating computer science in schools. Available from: http://www.imperial.ac.uk/computing/current-students/courses/322/

A look at conceptual differences between mathematics and programming

Tobias Kohn (University of Cambridge)

Background and context

Mathematics and computer science are closely related fields that share a lot of common ground and terminology. Indeed, the difference between mathematical and computational thinking is still being debated, and algorithms and programming are often taught as part of a mathematics course. Closer inspection, however, reveals some fundamental differences in how basic concepts and ideas are treated and how problems are approached. Even where mathematics guarantees the existence of a unique solution, for instance, finding that solution might be beyond the means of any computer.

In an educational setting, the differences between mathematics and computer science make a strong case for why the introduction of computer science alongside mathematics is well warranted. Moreover, when we introduce students to programming, say, we might find that their existing knowledge and skills in mathematics can both be an asset as well as a hurdle to overcome in order to master programming. In particular, students need to comprehend the difference between mathematical reasoning on the one hand, and following algorithmic computations, on the other hand.

Research focus

Functions are a well-known example of a seemingly shared concept between mathematics and programming, which, on closer inspection, reveals to bear significant differences. In particular, when seen as a mathematical expression, \`f(2)\` stands for a uniquely determined element, whereas the same expression as part of a computer program triggers a sequence of operations with possibly undetermined or varying outcome. Hence, a program containing lines such as \`if input() < 0 or input() > 9:\` indicates a conceptual model that is closer to the mathematical notion of functions, rather than to the notion effectively used in programming. When made the subject of discussion in classes, such differences can provide an opportunity to deepen the students' understanding. Without proper discussion, however, these differences can equally give rise to deep misconceptions, where students struggle to reconcile concepts from both fields.

Methodology and findings

Learning to program is well accepted as a difficult challenge and often frustrating experience. Errors and bugs of various kinds can be hard to see, especially for a novice programmer, and can keep a program from running or following the intended algorithm. Some of these errors might in fact stem from misconceptions where students apply mathematical reasoning instead of computational procedure to solve a problem. By inspecting about 40 programs from students in 10th grade, we have attempted to identify bugs and errors that suggest such a background.

Our inspection of variable usage, assignment, and sequential execution in students' programs suggests indeed an underlying, misapplied 'mathematical' approach of about a third of the students (Kohn, 2017). This is in accordance with findings from older, similar studies (Bayman & Mayer, 1983), with other research also pointing out the difficulties of variables and assignment, e.g. Kuittinen & Sajaniemi, 2004, Putnam et al., 1986, and Samurçay, 1985. Furthermore, a more recent study (Lister et al., 2004) has found that novice programmers have poor tracing skills, which is also in agreement with our findings.

Conclusion and implications

Understanding possible sources of misconceptions is a first step towards successfully addressing misconceptions in the classroom. Our findings suggest that care must be taken when emphasising the close relationship between mathematics and computer science during a programming course. During a trial run in a high school, we have instead discussed some of the issues discovered during our study, with promising results: explicit discussion in the classroom might indeed have a positive impact on the students.

References

Bayman, P. & Mayer, R.E. (1983) A diagnosis of beginning programmers' misconceptions of basic programming statements. Communications of the ACM, 26(9), 677-679.

Kohn, T. (2017) Variable evaluation: an exploration of novice programmers' understanding and common misconceptions. SIGCSE '17: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 345-350.

Kuittinen, M & Sajaniemi, J. (2004) Teaching roles of variables in elementary programming courses. SIGCSE Bulletin, 36(3), 57-61.

Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. (2004) A multi-national study of reading and tracing skills in novice programmers. SIGCSE Bulletin, 36(4), 119-150.

Putnam, R.T., Sleeman, D., Baxter, J.A. & Kuspa, L.K. (1986) A summary of misconceptions of high school basic programmers. Journal of Educational Computing Research, 2(4), 459-472.

Samurçay, R. (1985) The concept of variable in programming-its meaning and use in problem-solving. Educational Studies in Mathematics, 16(2), 143-161.

Early findings from Isaac Computer Science

Eirini Kolaiti (Raspberry Pi Foundation)

Background

Isaac Computer Science (CS) is a free, online platform for supporting students and teachers with A level Computer Science. It is part of the Department for Education's National Centre for Computing Education that aims to boost computing education across England. Isaac CS provides learning materials that cover every topic of the Computer Science curriculum and questions for students to practise with. The programme also offers a variety of training events for teachers and students.

Content informed by research

Both the content and features of the platform are informed by research in the field. A well-established area of computing education research stresses the importance of identifying and resolving misconceptions around programming (Sorva, 2013; Sirkiä & Sorva, 2012; Du Boulay, 1986). In order to address misconceptions, the Isaac CS platform includes carefully planned multiple-choice, text, and numeric questions. Feedback for common wrong answers is provided to the student which can help with resolving misunderstandings.

Research has shown that reading, explaining the purpose of, and tracing code fosters the ability of novices to write their own programs (Lister et al., 2009). In light of these findings, Isaac CS includes tasks such as asking students to fill in missing code, trace an algorithm to find an output, or identify the overall purpose of a snippet of code.

A popular type of question for the development of programming skills is Parson's Problems; answers are split into pieces and randomised which the learners then need to put in the right order. Parson's Problems are considered to be motivating (Guzdial, 2017) and beneficial to the learner as they can be used to model well-written code to encourage good programming practice (Parsons & Haden, 2006). On Isaac CS, Parson's Problems are used across multiple topics. Questions might involve creating a code snippet, ordering steps in a process, or constructing a valid expression or sentence.

The use of worked examples has been highlighted as a key practice for computer science because it enables students to develop problem-solving techniques and improve their performance on near transfer tests (Skudder & Luxton-Reilly, 2014). Fully worked examples are used to demonstrate an approach to solving a problem and for clarifying difficult concepts. Equivalent questions with three levels of hints function as faded-worked examples to consolidate learning.

Bespoke features

Research suggests that response systems (i.e. online platforms that auto-mark questions) facilitate learning when they provide instant feedback that both the students and the teacher can act upon (Kay & LeSage, 2009). The 'markbook' and 'my progress' pages offer real-time insights that help identify areas of difficulty. Bespoke features such as the marking of free-text questions and the boolean logic editor have opened up the opportunity for further research in the field.

Engagement

Even though it is still early to evaluate the impact of Isaac CS, data around user engagement suggest that Isaac CS has been received positively from the educational community. Taking into consideration the geographical spread of state schools who have signed up to Isaac CS, we can assert that the platform has reached learners across England.

To conclude, Isaac CS has made a very encouraging start. Nonetheless, there are many aspects that need to be researched further. Only 20% of the active users are female; a figure that reflects the gender balance in computer science education and needs to be addressed further. Moving forwards, an annual survey of the registered users will be carried out to help evaluate the impact of the programme. There is also scope to analyse the various learning paths that students follow, pinpoint challenging topics, and test interactive features to understand how to best support learners.

References

Du Boulay, B. (1986) Some difficulties of learning to program. J. Educational Computing Research 2(1), 57–73.

Guzdial, M. (2017) Balancing teaching CS efficiently with motivating students. Communications of the ACM 60(6), 10–11.

Kay, R. H. & LeSage, A. (2009) Examining the benefits and challenges of using audience response systems: A review of the literature. Computers & Education, 53(3), 819–827.

Lister, R., Fidge, C. & Teague, D. (2009) Further evidence of a relationship between explaining, tracing and writing skills in introductory programming, in ACM SIGCSE Bulletin, 41(3), 161-165. ACM.

Parsons, D. & Haden, P. (2006) Parson's Programming puzzles: a fun and effective learning tool for first programming courses, ACE '06: Proceedings of the 8th Australasian Conference on Computing Education, 52.

Sirkiä, T. & Sorva, J. (2012) Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. Proceedings of the 12th Koli Calling International Conference on Computing Education Research, 19-28. ACM.

Skudder, B. & Luxton-Reilly, A. (2014) Worked examples in computer science. Proceedings of the Sixteenth Australasian Computing Education Conference, 148, 59-64. Australian Computer Society, Inc..

Sorva, J. (2013) Notional machines and introductory programming education. ACM Transactions on Computing Education, 13(2), 1–31.

Investigating the impact that the Raspberry Pi online learning project has on teachers' self-efficacy in teaching computing

Alex Parry, Martin O'Hanlon, Mac Bowley, and Matt Hogan (Raspberry Pi Foundation)

Background

When the new computing curriculum was introduced in England in September 2014, many teachers found themselves without the subject knowledge and pedagogical skills required to teach the subject. The Royal Society's (2017) report 'After the reboot: computing education in UK schools' found the computing provision to be 'patchy and fragile', and identified concerns around a shrinking workforce and teachers' readiness to implement the new curricula.

A recent study of secondary school teachers in England from the National Foundation for Educational Research analysed the recruitment and retention problem faced by teachers of science, mathematics, and computing (Worth & Van den Brande, 2019). This report suggests that high-quality professional development is likely to help improve teachers' self-efficacy (belief in their own abilities), satisfaction, and their likelihood of staying in teaching. Moreover, the report found that technology (including computing) teachers are significantly more likely than other teachers to identify a need for professional development.

The Raspberry Pi online learning project aims to enhance subject knowledge, pedagogical skills, and confidence of computing teachers to enable them to successfully deliver the English computing curricula. The courses are free to access and available to all at FutureLearn.com, and are structured to support teachers who are new to computing and experienced teachers alike.

Method

The data gathered about learners on the online courses is taken from two places: course participation data provided by FutureLearn, and course surveys provided towards the end of the course.

FutureLearn provides anonymised quantitative data about learners participation on individual courses, which includes detailed information relating to enrolments, step completions, comments, video views, and peer assignments and reviews.

Learners are also presented with a course survey in the final week of the course. There are two different surveys: one for courses that include programming, and one for non-programming courses. These surveys ask the learners to rate how much they agree with statements about their understanding and confidence of the course content.

Findings

While the online courses are targeted to teachers in England, they are used by learners all over the world from many different professions and backgrounds. In 2019, 1251 teachers in England participated in 3205 combined course runs. In the same time period, 39,143 learners participated in our courses from over 200 different countries. From these participants, 37.1% are employed in teaching and education and 39.8% live in the UK. This means that from the people who take our courses, a minority are teachers in England.

For the non-programming courses, 83% of learners agreed that they have become more confident in their understanding of the course material since starting the course. 81% of learners also agreed that they had become more confident in explaining the concepts from the course to others.

For the programming courses, 81% of learners agreed that they have improved their programming skills, but only 64% also report increased confidence in teaching programming (this difference compared to the non-programming courses may be related to the way the question was phrased specifically as referring to 'teaching' rather than 'explaining').

Conclusions

The initial data that has been collected on teacher satisfaction and increased confidence is positive. However, it is important to note that the current data from the surveys does not account for learners who dropped out before reaching the final week of the course.

More research is needed to further understand teachers' self-efficacy and support them appropriately with delivering the computing curricula in England. Two questions that inform the next steps of research are:

- How do teachers implement what they have learned from online learning with their students in the classroom?
- What are the barriers to teachers taking our courses?

References

Royal Society (2017) After the reboot: computing education in UK schools. The Royal Society, London. Available from: https://royalsociety.org/~/media/events/2018/11/computing-education-1-year-on/after-the-reboot-report.pdf

Worth, J. & Van den Brande, J. (2019) Retaining science, mathematics and computing teachers. Slough, NFER

Investigating the relationship between programming and natural languages, and the impact of applying language learning tools within the PRIMM framework

Alex Parry (Raspberry Pi Foundation)

Background

The role of language in the field of computer science education is seen by some researchers as a particularly important aspect that is often overlooked when designing and implementing introductory programming courses (Lister et al., 2009; Portnoff, 2018). A recent framework for structuring programming lessons that actively encourages the use of language is PRIMM (Predict-Run- Investigate-Modify-Make), which aims to counter the known problems that novices encounter as they attempt to write programs before they are able to read them (Sentance & Waite, 2017). The initial large-scale study of PRIMM — involving nearly 500 Key Stage 3 students in England (aged 11–14 years old) — showed promising results on learners' attainment (Sentance et al., 2019). However, some teachers observed students spending hardly any time in the later code writing stages of 'modify' and 'make', which aligns with the majority of programming research that contends that writing code is more difficult than reading or tracing (Lopez et al., 2008; Qian & Lehman, 2017).

Method

This empirical study involved modifying the resources from the initial PRIMM study to include tools commonly used when teaching first and second languages to young people, such as colour coding words, cloze exercises, and fading worked examples. The aims of this research were to measure the impact that the modified resources had on student learning, and to explore how students perceive learning a programming language compared to a natural language. The study was conducted in a girls school in England with Year 8 students aged 12–13 years old. There were two non-randomised groups of 30 students each; one class was the control group who were taught with the original PRIMM material, whilst the other class was the experimental group taught with the modified resources. Each group took a baseline test and a post-test so their progress could be measured. The Mann-Whitney U test was employed for the analysis of the test scores since the data was being compared across two independent groups and it was not normally distributed. Furthermore, a focus group of 5 students from the experimental group was conducted a week after the intervention had finished to explore students' perceptions on whether learning a programming language was similar to learning a natural language. The focus group was audio recorded and transcribed before being categorised into themes for analysis.

Findings

The results of the two tests found that the performance of students on the baseline test was not significantly different between the two groups, yet the experimental group achieved a significantly higher score on the post-test than the control group. This indicates that the inclusion of language tools within the PRIMM materials made a positive impact on students' learning. Analysis of the focus group conveyed that most students made similar connections between both types of languages, however they were not always considered to be on the same level of difficulty. Some students thought programming was more challenging due to the syntax, whilst others found it easier because of the limited, yet more versatile vocabulary.

Conclusions

The analysis of the focus group indicates that a positive perspective of foreign or native languages may benefit students' belief in their ability to learn a text-based programming language, and negative preconceptions can discourage students. These findings correspond with the view that to be good at programming you must be highly skilled in your native language (Dijkstra, 1982, 129-131). Students found the colour-coded program statements helpful for understanding the grammatical structure of a new concept, and fading cloze exercises useful in the later stages of writing code. Further research is required to understand how natural language skills translate to learning a programming language.

References

Dijkstra, E.W. (1982) Selected writings on computing: a personal perspective. New York, Springer-Verlag. ISBN: 0–387–90652–5.

Lister, R., Fidge, C. & Teague, D. (2009) Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. In Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09), 161–165.

Lopez, M., Whalley, J., Robbins, P. & Lister, R. (2008) Relationships between reading, tracing and writing skills in introductory programming. In Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08), 101–112.

Portnoff, S. R. (2018) The introductory computer programming course is first and foremost a language course. ACM Inroads, 9(2), 34–52.

Qian, Y. & Lehman, J. (2017) Students' misconceptions and other difficulties in introductory programming: a literature review. ACM Transactions on Computing Education, 18(1), 1–24.

Sentance, S. & Waite, J. (2017) PRIMM: Exploring pedagogical approaches for teaching text-based programming in school'. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE '17), 113–114.

Sentance, S., Waite, J. & Kallia, M. (2019) Teachers' experiences of using PRIMM to teach programming in school. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), 476–482.

Could an integrated, student-centred approach to computing curriculum design have a positive impact upon students' problem-solving attitudes and behaviours in Key Stage 3?

Julie Price (Member of the Chartered College of Teaching, Associate Member of British Computer Society, NCCE CS Champion, CAS Master Teacher)

Background and context

Where once it was assumed that the vast majority of brain development took place during the first few years of life, advances in technology have allowed neuroscientists to demonstrate that the brain continues to develop through adolescence, potentially into one's twenties or thirties.

The brain area involved in higher-order cognitive processes changes most dramatically during adolescence. Meanwhile, the area involved in emotion- and reward-processing is hypersensitive. As the ability to inhibit inappropriate behaviour is still developing, adolescents become more inclined to take risks, moodiness, social sensitivity, and self-consciousness.

An understanding of the nature of adolescent development, therefore, should inform the design of the school curriculum. An engaging, inclusive, and motivational student-centred computing curriculum design can draw upon computational thinking as enabling students to understand the digital world in a deeper way. Learning needs to be real to the students' lives and relevant links forged with other subjects and real-life situations (Curzon et al., 2014).

Wing (2010) defined computational thinking as "the mental activity in formulating a problem to admit a computational solution". While identifying no conclusive definition, Roman-Gonzales, et al. (2017) corroborate the conceptualisation of computational thinking as a problem-solving ability.

Research focus

At the heart of the research project lies the question: could an integrated, student-centred approach to computing curriculum design have a positive impact upon students' problem-solving attitudes and behaviours in Key Stage 3? The project formed a requirement of the British Computer Society Certificate in (Secondary) Computer Science Teaching. The sample of the student population participating in this study comprised 28 Year 7 students: 10 female and 18 male. This gender imbalance extended to SEND for mathematics and reading.

Method

Having been introduced to the concept of problem-solving attitudes and behaviours (Bagge, n.d.), the students carried out a self-evaluation and responded to a real-life problem, a process which was repeated at the end of the project. The intervention took the form of an integrated study, applying skills from English and personal and social education (PSE), with new computing learning being hung on an unfolding crime-investigation narrative.

The qualitative data provided by the students' responses was processed quantitatively and analysed alongside observations made during the project, together with the tangible evidence provided by the students' endeavours.

Findings

An analysis of the data collected suggests that the students had become more confident at solving problems over the course of the project. With regard to the relative progress made by girls and boys, their responses indicate that the initial gender gap has been reduced.

While the boys admitted to being over-confident, girls commented that they had underestimated their level of confidence in each of the eight problem-solving attitudes and behaviours capabilities at the beginning of the project. However, both judge that their ability to assess accurately improved as the project progressed.

With regard to the application of computational thinking to problem-solving at the project's end, double the number of girls provided evidence of a significant improvement, whereas 80% of boys provided evidence compared with zero at the beginning.

Overall, the analysis of data indicates that the learning experiences have had an overall positive effect upon the problem-solving skills of the students and their confidence when presented with problems requiring solutions.

Conclusion and implications

This study was carried out over a brief period of time and with a limited number of students, and circumstances dictated that compromises caused deviation from the original plans for the study.

Nevertheless, the project has highlighted several interesting, tentative ideas worthy of further research, requiring the application of greater rigour.

References

Bagge, P. (n.d.) Problem solving attitudes & behaviours. Code-it. Available from: http://code-it.co.uk/attitudes/

Curzon, P., Dorling, M., Ng, T., Selby, C. & Woollard, J. (2014) Developing computational thinking in the classroom: a framework. Swindon, UK, University of Southampton.

Román-González, M., Pérez-González, J. & Jiménez-Fernández, C. (2017) Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior, 72, 678-691, Available from: DOI: 10.1016/j.chb.2016.08.047

Wing, J. M. (2010) Computational Thinking: What and why? Carnegie Mellon University. Available from: http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Challenges facing computing teachers in Guyana

Lenandlar Singh (University of Guyana), **Sue Sentance** (King's College London), **and Penelope De Freitas** (University of Guyana)

Background

Guyana is the only English-speaking South American country and has a population of approximately 750,000 inhabitants. Information technology is taught in most secondary schools using a national syllabus, including problem solving, algorithms, programming, hardware and software. Both information technology and computer science can be taken as an elective. Guyana's education system has a strong association with the Caribbean.

With the introduction of computer science into school curricular around the world in recent years, it is easy to forget the challenges that developing countries may experience in following suit. The shift from the teaching of basic digital skills to a knowledge-based curriculum that comprises of more computer science concepts, including programming, is particularly challenging for countries where a lack of resources and in-service teacher education makes change more difficult to effect. Teachers face a range of challenges introducing computing into schools, including subject knowledge and confidence (Sentance & Csizmadia, 2017; Yadav et al., 2016).

Research focus

Here we describe a study carried out to investigate the challenges faced by teachers in Guyana relating to resources, student engagement, and in-service teacher education (Sentance, Singh & De Freitas, 2020). A small-scale mixed-methods study with 48 teachers was conducted in two areas of Guyana, accompanied by a workshop covering a range of interactive activities. Teachers reported a desire to improve the opportunities for their students, but described challenges including lack of computers, subject knowledge, and support. They also reported that they found programming hard to learn and teach, with computer architecture easier in both these areas.

Method

In March 2018, two all-day workshops were held for teachers of IT and computing in two areas of Guyana. The sessions consisted of three sets of activities: a) teaching computing unplugged, b) using the Micro:bit, and c) pedagogical strategies for teaching programming. Alongside the workshop, a study was conducted to elicit teachers' needs and attitudes using a range of data collection methods, enabling quantitative and qualitative analysis. Data collection methods included free-text writing, a short survey, and focus group notes.

Findings

Teachers were asked which topics they found easy or hard to teach and which students they found easy or difficult to teach. The free-text task elicited a range of topics across information technology and computer science. The majority of teachers reported that programming was hard to teach and hard to learn, whereas the teaching and learning of computer architecture was thought to be easy.

In a questionnaire, 94% teachers said they would like to improve their subject knowledge in IT and computing. 63% said they would like more support in teaching the topics in the curriculum and 58% said that students are not always engaged in the subject matter. Only 15% said that they lacked confidence in what they were teaching. Through focus groups, teachers articulated a need for teacher training, support, and more resources to support teaching and to engage students. Teachers' confidence was not as much of an issue as we had expected. However, where confidence was described as an issue, it was around programming.

Conclusion

This poster describes the first study relating to computing education in Guyana. Although professional development opportunities and resources for the teaching of computing are emerging in many countries, developing countries have not had access to the same inputs as the western world. We need to hear the voices of teachers all around the world, and need more research into what computing education could look like globally, including how we can overcome some of the likely obstacles and challenges.

References

Sentance, S. & Csizmadia, A. (2017) Computing in the curriculum: challenges and strategies from a teacher's perspective. Education and Information Technologies 22, 2 (2017), 469–495.

Sentance, S., Singh, L., & De Freitas, P. (2020) Challenges facing computing teachers in Guyana. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 1323-1323.

Yadav, A., Gretter, S., Hambrusch,S., & Sands, P. (2016) Expanding computer science education in schools: understanding teacher experiences and challenges. Computer Science Education 26, 4 (2016), 235–254.

Singularity: a DataDrivenDance

Genevieve Smith-Nunes (University of Cambridge)

'Singularity: A DataDrivenDance' is a transdisciplinary design-based research (DBR) investigation into computing education, using classical ballet and biometric data to help teach computer science theory to disadvantaged/under-represented, secondary-aged students, and explore the ethical and social justice implications of future technologies upon this cohort. My practice of data-driven-dance views the systematic nature of ballet to that of a programming language. In ballet, to recreate different patterns associated with computer science theories and concepts. In 'Singularity', the ballet, physical computing, biodata, and educational resources combine to explore the story of interstellar travel, computing theory, and biodigital ethics through this proposed study.

Background

'DataDrivenDance' was born out of the lower numbers of female students opting for computing GCSE or A level during my secondary teaching career. Girls continue to be heavily under-represented in computing education, current figures for the UK show that only 10% of A- level computer science students are female (Kemp et al., 2018). DataDrivenDance involves classical ballet productions as a medium for delivering computing theory and concepts aimed at non-specialist audiences. An ongoing award winning project since 2012, it uses ballets as an R&D platform for developing creative computing educational classroom resources.

Research focus

The literature led to an awareness of two particular issues that are central for computing education in the current landscape. In their simplest form: One: How do you engage girls in computing? Two: How do you raise awareness of data ethics? These conjectures paved the way to DBR as the most appropriate methodology to answer these central issues in computing education. Why are so few girls taking computing pre-university exams? The precise attributes of classical ballet, with set structures, routines, rules, and terminology, marry with programming and computational thinking (Wing, 2006). Make available opportunities to address future ethical questions of biodigital ethics-cognitive privacy (Farahany, 2018).

Methodology

Not confined to a single approach but the amalgam of education, computing, and traditional ballet practice, materials, and pedagogy. This interdisciplinary DBR (Bakker, 2018) crosses the boundaries of computer science, dance, and education incorporating the research methods and practices of each domain. It is not a risk-free approach: each domain has particular frameworks relating to theory and methodology. From the world of dance, using 'studio as practice', studio-based research as a method for enquiry (Barrett & Bolt, 2014), and practitioner research to develop a methodology that applies the rigour, practice, and pedagogies of each domain. DBR is iterative in nature, similar to Agile software methodology, the proposed design framework will consist of three iterations. Each interaction will test the use of ballet and biodata, both as tools for creativity and a platform for discourse on (neuro) ethics (Le, 2019). Cyclic conjectures leading to a more refined design framework and design principles. Iteration one focuses on dancers participants, the second iteration on students, and thirdly trainee computing teachers; all co-creators within the study.

Proposed data analysis

Using an integrated data approach (Hesse-Biber, 2010) complemented with thematic analysis of interview data using stemming and lemmatisation techniques (Vallbé et al., 2007; Schütze et al., 2008) to reduce researcher subjectivity. The education components evaluated for subject suitability and impact on students' creative computing knowledge. Tools such as ARCS: attention, relevance, confidence, and satisfaction model, (Keller, 1987) and RIMMS (reduced instructional motivational materials survey) questionnaires (Loorbach et al., 2014). EGG data (aggregated and anonymised) collected from participants during the study and audience members during live performance will be used as open datasets for interactive live performance elements, and educational resources. Datasets will then be shared later for other researchers and educators to use.

References

Bakker, A. (2019) Design principles in design research: a commentary. In Unterrichtsentwicklung macht Schule,177-192. Wiesbaden, Germany, Springer VS.

Barad, K. (2007) Meeting the universe halfway: quantum physics and the entanglement of matter and meaning. Durham, USA, Duke University Press.

Barrett, E. & Bolt, B. (Eds.). (2014) Practice as research: approaches to creative arts enquiry. London, Ib Tauris.

Golz, P. & Smith-Nunes, G. (2015) [arra] stre: a data-driven ballet. In Proceedings of the Conference on Electronic Visualisation and the Arts (EVA '17), 90-91. Swindon, UK, BCS Learning & Development Ltd.

GOV.UK. (2018) National curriculum in England: computing programmes of study. Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/ national-curriculum-in-england-computing-programmes-of-study

Farahany, N. (2018) When technology can read minds, how will we protect our privacy?. TED. Available from: https://www.ted.com/talks/nita_farahany_when_technology_can_read_minds_how_will_we_protect_our_privacy?language=en

Hesse-Biber, S. N. (2010) Mixed methods research: merging theory with practice. New York, NY, Guilford.

Keller, J.M. (1987) Development and use of the ARCS model of instructional design. Journal of instructional development, 10(3).

Kemp, P.E.J., Berry, M.G. & Wong, B. (2018) The Roehampton annual computing education report. London, University of Roehampton.

Law, J. & Hassard, J. (1999) Actor-network theory and after. Oxford, Blackwell Publishers.

Loorbach, N., Peters, O., Karreman, J. & Steehouder, M. (2014) Validation of the instructional materials motivation survey (IMMS) in a self-directed instructional setting aimed at working with technology. British Journal of Educational Technology, 46(1), 204–21, Available from: DOI:10.1111/bjet.12138.

Schütze, H., Manning, C.D. & Raghavan, P. (2008) Introduction to information retrieval. Cambridge University Press, Available from: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

Smith-Nunes, G., Shaw, A. & Neale, C. (2018) [pain]Byte: Chronic Pain and BioMedical Engineering Through the Lens of Classical Ballet & Virtual Reality. In Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction (TEI' 18), 493-497.

Toussaint, M.J. & Brown, V. (2018) Connecting the arcs motivational model to game design for mathematics learning. Transformations, 4(1), 19-28.

Vallbé, J.J., Martí, M.A., Fortuna, B., Jakulin, A., Mladenic, D. & Casanovas, P. (2007) Stemming and lemmatization: improving knowledge management through language processing techniques. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.9503&rep=rep1&type=pdf

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