Recent developments in computer science education research

Sue Sentance
King’s College London & Raspberry Pi Foundation

sue@raspberrypi.org
sue.sentance@kcl.ac.uk

University of Manchester
Computer Science Department Seminar
July 23rd 2018
Today’s talk

- Overview of field
- Key areas where developments are taking place
- Some King’s projects
- Programming research summary
- Questions

My focus is on research around computing in school but this mostly draws from CS Ed research in HE or has implications for it

About me:

Senior Lecturer in Computer Science Education at KCL (2014-2018)
Chief Learning Officer, Raspberry Pi Foundation, 2018-...
PhD in AI & ED, 1993
PGCE 1999
Taught in schools, 1999-2010
Royal Society Advisory Committee
Computing At School/ BCS Boards etc,
Where to start?

**Journals**
- Computer Science Education
- ACM TOCE
- BJET
- Computers & Education
- Computers in Human Behaviour

**Conferences**
- ICER (more theoretical)
- ITICSE (Europe)
- SIGCSE (US)
- WIPSCE (School-focus)
- ISSEP (School-focus)

**Pedagogy in teaching Computer Science in schools:**
A Literature Review
Jane Waite, QMUL & KCL

**Assessment in Computer Science courses:**
A Literature Review
Maria Kallia, KCL
Our context: interest in teaching computing in school is growing

1. Internationally – a snapshot from 6 countries (Webb et al, 2018)

<table>
<thead>
<tr>
<th>Theme</th>
<th>Australia</th>
<th>Israel</th>
<th>NZ</th>
<th>Poland</th>
<th>Slovakia</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entitlement - who is the COMPUTER SCIENCE curriculum for?</td>
<td>New curriculum for all</td>
<td>All must learn Computer Science and technology literacy incorporating computation-al thinking</td>
<td>High school subject for seniors from 2011; all children in primary schools from 2018</td>
<td>High and middle school subject for 20 years. All from 7-11 and an advanced option from 8-12.</td>
<td>All from elementary school upwards</td>
<td>All from elementary school upwards</td>
</tr>
<tr>
<td>Starting age for COMPUTER SCIENCE</td>
<td>From the first year of school (about 5 years old)</td>
<td>Elementary School</td>
<td>From 2018 from the first year of school</td>
<td>7 years old</td>
<td>8 years old</td>
<td>5 years old</td>
</tr>
</tbody>
</table>

2. In England, Computing is a mandatory school subject

Key questions for researchers
How do we teach CS in schools and in HE?
How do we increase engagement and motivation?

Key issues for researchers
In HE, CS Ed research is under-valued and under-funded
For K-12, CS Ed research is in its infancy, under-funded and not prevalent in education departments
Mapping the field

**Computer Science Education Research**

- **Pedagogy and assessment**
  - Peer instruction
  - Programming misconceptions
  - Tracing and reading code
  - Sub-goal modelling
  - Block-based environments
  - Automated assessment
  - Visualisation tools
  - Physical computing
  - Dual-modality environments

- **Tools, software, devices, etc.**
  - Pair programming
  - Computational thinking
  - Unplugged computing
  - Inclusion
  - Courses & curricular (inc theoretical underpinning)
  - Papert and constructionism
  - Sub-goal modelling

- **Societal, attitudinal factors**
  - Growth mindset
  - Diversity
  - Contextualisation
  - Mentoring

- **Outreach**
  - Teacher professional learning
  - Progression
  - Motivation and interest
  - Informal learning
Mapping the field

- Pedagogy and assessment
- Unplugged computing
- Inclusion
- Courses & curricular (including theoretical underpinning)
- Papert and constructionism
- Societal, attitudinal factors
- Outreach
- Progression
- Teacher professional learning
- Teacher professional learning
- Motivation and interest
- Informal learning
- Mentoring
- Diversity
- Growth mindset
- Contextualisation
- Dual-modality environments
- Physical computing
- Visualisation tools
- Automated assessment
- Block-based environments
- Tracing and reading code
- Sub-goal modelling
- Pair programming
- Tools, software, devices, etc.
- Programming misconceptions
- Peer instruction

Computer Science Education Research
Themes from CS Ed articles 2004-2014

How is this changing now?

Survey looked at papers in:
Journals:
- Computer Science Education
- ACM TOCE
Conferences:
- WIPSCE, ITICSE, ICER, ISSEP
and SIGCSE

Numbers of papers:
- All – 2225
- University - 1285
- School -420
Some particular areas to look at

A. Programming environments for beginners
B. Learning programming: pedagogy (inc tracing, threshold concepts, PRIMM and misconceptions)
C. Computational thinking
D. Physical computing
E. Inclusion
A. Programming environments
A. Programming environments for beginners

Increase in numbers of papers about Scratch since 2007
(in our survey)

Some key papers
Learning computer science concepts with Scratch (Meerbaum et al, 2013)
- Focus on concepts learned (not skills)
- Tested > 200 Year 9 students taught Scratch systematically
- Testing showed difficulties with loops, variables and concurrency

Habits of programming Scratch (Meerbaum-Salant et al, 2011)
- block-based environments increase extremely fine-grained programming (too bottom-up) – bad habits for future

Recent developments
Recent developments: comparing understanding in block-based and text-based programs (Weintrop and Wilensky, 2015)

Another paper last year found that although attitudes and perceived difficulty was the same with block-based and text-based programming, pupils achieved goals more quickly (less idle time) with block-based (Price & Barnes, 2015).
A: Dual-modality programming environments

David Weintrop and Nathan Holbert, 2017

From Blocks to Text and Back: Programming Patterns in a Dual-modality Environment

23 students – 13 high school, 10 in HE
Initially all students worked in blocks
Students switch modes – not one-way
Correlation between self-efficacy and use of text-based

Frame-based editing with Stride

A different approach is being taken by the Greenfoot team at KCL (Michael Kölling, Neil Brown et al) who use frame-based editing as a way of programming using text but a drag-and-drop highly structured interface
B. Learning programming
B. Learning programming: reading code

Some research highlights

2004: Multi-institutional study of reading and tracing skills shows better performance in programming by those able to trace code (Lister et al., 2004)

2011: Use of neo-Piagetian framework to establish stages that novice programmers go through (Lister et al., 2011)

2014: Exemplification of framework through case studies, using think-aloud to find out more about students’ thought process while programming (Teague and Lister, 2014).

Work in this area suggests that student pass through neo-Piagetian stages: sensorimotor, preoperational, and concrete operational stages, before eventually reaching programming competence at the formal operational stage.
B. Learning programming: Common misconceptions

The 1980s

Early work on misconceptions and novice programmers (Ben du Boulay, 1986, Bayman and Mayer, 1983, Bonar and Solway, 1985)

- Loops are difficult
- Differentiating between a string and a number
- Confusion of equality and assignment
- Inputting data (from where?)
- etc...

One recent paper: Exploring programming misconceptions (Sirkia and Sorva, 2014)

Based on analysis of 24,000 log files from UG students first learning programming:

Research found:
- Inverted assignment (first = second) - wrong way round
- If X == Y: (execute then part even if false)
- Returning False from function even though condition does not hold
- Not storing return value of function
- Etc....

Steady stream of work in this field, including Juha Sorva’s PhD work

New chapter by Sorva (2018) describes 41 different programming misconceptions (in my book)
Maria Kallia is working on a project to understand which concepts in computer programming are particularly difficult and could be identified as “threshold concepts”.

- Focus on functions and parameters in programming
- Latest study looked at liminal space – the confused state you are in before you reach an understanding of something (go over the threshold)
- Maria developed a test of programming performance and compared to attitudinal factors including self-efficacy, motivation and interest

Findings (all statistically significant):
- There is a significant relationship between liminal space and CS identity
- Troublesome knowledge impacts sense of belonging and motivation, but not identity
- Girls in the liminal group experience troublesome knowledge more intensely than boys and this influences their sense of belonging, motivation and identity while boys experience an impact only in the sense of belonging.
- She has developed a model for predicting students are in liminal or post-liminal space from their self-efficacy, motivation, identity and self-evaluation which explains 78.6% of the variance.
A framework for working with beginners using text-based programming:

**Predict** – given a working program, what do you think it will do? (function)

**Run** – run it and test your prediction

**Investigate** – get into the nitty gritty. What does each line of code mean? (structure). Lots of activities here: trace, annotate, explain, talk about, identify parts, etc.

**Modify** – edit the program to make it do different things (function)

**Make** – design a new program that uses the same nitty gritty but that solves a new problem (function).

**PRIMM Study (to appear!)**

Mixed-methods study

**Part A: Quasi-experimental design**

- 14 KS3 teachers
- 180 students in control group (11-13)
- 493 students in experimental group (11-13)

Pre-test and post-test

Findings:
- Statistically significant difference between post-tests of control and experimental groups

**Part B: Co-generation, design-based research with teachers**

- 10 interviews, 1 focus group, and teacher journals

Understanding why the approach works

Themes emerged around:
- Differentiation and increased accessibility
- Influence of language and talk

**Theoretical framework**

- Drawing on Vygotsky and social constructivism
- Concept of mediation is used to explain how the program begins on the social plane then moves to cognitive plane and to being understood by the learner
- This is similar to the Use-Modify-Create approach it builds on

**PRIMM Materials**

The materials we used in our study will shortly be available online with lesson plans for adaption by teachers
C. Computational Thinking
C. Computational thinking

CT has become a popular research topic

Graph shows number of papers published with CT in title
44 already in 2018

Different views (this analysis is simplistic!) generating some interesting debates

Jeannette Wing (2006)

- CT is a fundamental skill
- For everybody, not just computer scientists
- Human thinking, not just computer thinking
- Ideas and problem solving

“[Computational thinking] ... represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.”

Tedre & Denning (2016)

- History of algorithmic thinking – we’ve been here before
- CT is not a superior way of thinking
- We should emphasise design and modelling
- Risk of exaggerated claims
- Risk of narrowing view of computing

“The new CT movement aimed to include also those who use computational tools and those who engage in step-by-step procedures. The attempt to broaden the CT audience moved into unchartered territory, where there is less certainty that tool users and procedure followers need CT or benefit from it.”

Data from https://csedresearch.wordpress.com/computational-thinking/
## Some reviews of CT literature

<table>
<thead>
<tr>
<th>Author</th>
<th>Number of papers</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalelioglu, Gulbahar and Kukul (2016)</td>
<td>125 papers on CT</td>
<td>Lack of theoretical framework – game-based learning and constructivism primarily Immature field – not many papers No consistent definition</td>
</tr>
<tr>
<td>Lye and Koh (2014)</td>
<td>27 intervention studies</td>
<td>Focusing on CT and programming only Need to explore more classroom-based interventions</td>
</tr>
<tr>
<td>Shute, Sun and Asbell-Clarke (2017)</td>
<td>45 papers reviewed</td>
<td>Considered papers researching CT in robotics, game design and range of environments Comparison of frameworks and proposed new one</td>
</tr>
</tbody>
</table>
C: Example projects in the area of CT

Recent work by a group of researchers in Madrid:

- Can computational talent be detected? Predictive validity of the Computational Thinking Test
- Extending the nomological network of computational thinking with non-cognitive factors
- Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch

Possibly the most rigorous work being conducted in the area of computational thinking

Recent results:

- Development of a model to predict computational talent in school students
- Evaluated with 314 middle school students
- Distinguishes between computational regular thinkers and computational top thinkers
- Implications for computing curriculum development

NB Their definition of CT is more like programming (warning to Wing-lovers)
D. Physical computing
D. Physical computing: more and more devices

and the rest
D. Physical computing

Closely linked to constructionism in the literature:

“Constructionism – the N word as opposed to the V word – shares constructivism’s connotation of learning as “building knowledge structures” irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it’s a sand castle on a beach or a theory of the universe.” (Papert, 1991)

Research on pedagogical benefits

Tangible nature may facilitate understanding (Marshall, 2007)
Concrete easier to understand than abstract (Papert, 1980)
Lends itself to collaborative working (Sentance & Schwiderski-Grosche, 2013)
Can learn directly about how computers work and mathematics (Papert, 1980)
Facilitates creativity (Kafai, 2015)
Hybrid interfaces can be used to facilitate progression in programming (Horn et al. 2012).

Examples of pedagogical approaches:

Activity-media design (Jin et al, 2016)
Method developed to facilitate the development of physical computing learning activities which minimise cognitive load.

Use-modify-create (Lee et al 2011)
Move from “not mine” to “mine”
Use existing projects first, then modify and build new ones
E. Inclusion
E. Inclusion

Key question: How can we build a Computing curriculum that is accessible to all learners?


Design-based research methodology used to iteratively inform the development of the curriculum, programming environment, and research - involves researchers and practitioners collaborating in real-world settings with the aim of improving educational practices.

Catherine Elliott, presenting at LCERS

65 special-needs teachers reported on opportunities and barriers of computing in the SEND classroom

<table>
<thead>
<tr>
<th>Issues with the technology</th>
<th>43.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of computing hardware</td>
<td>32.3%</td>
</tr>
<tr>
<td>Lack of confidence in teaching the curriculum</td>
<td>35.5%</td>
</tr>
<tr>
<td>Lack of training for using the technology</td>
<td>27.4%</td>
</tr>
<tr>
<td>No time allocated for teaching Computing</td>
<td>11.3%</td>
</tr>
<tr>
<td>Lack of resources for SEND</td>
<td>71%</td>
</tr>
</tbody>
</table>

“Many of our students are very engaged by ICT. It can be incredibly motivational for ASD and SEMH learners. It empowers students who struggle with social interaction to present and share their work widely.”

Need for more research!
Torino is a physical programming language that was developed to be inclusive of learners with visual impairments. It was designed to teach programming concepts to children ages 7-11 regardless of level of vision.

To create programs with Torino, physical ‘command pods’ are connected together, which produce sound in the form of music, stories and poems. There are four main types of command pods: play, pause, loop and selection, each of which represents a line of code in the program.

Alex Hadwen-Bennett is looking at the way that visually-impaired learners use Torino to learn programming

**Initial findings:**
- Blind and partially-sighted students use the tool in different ways to learn programming
- Exploratory procedures (type of gestures) are used to trace the flow of control in the program
- Different types of gestures and exploratory procedures are used to demonstrate understanding

**Potential implications for sighted students:**

The focus on control flow as a separate process to tracing is being explored through this work -> implications for work in schools on teaching programming.
Programming research summary

We have already seen some research in the area of learning programming:
- Effectiveness of reading code
- PRIMM
- Threshold concepts
- Block-based and dual-modality programming environments

Other research (with no time to cover) includes:

- Stepwise self-explanation
- Sub-goal modelling
- Worked examples
- Pair programming
- Use of Parson’s Puzzles
- Tinkering and Bricolage

See Caspersen (2018) for an overview

Continuum of approaches for teaching programming

Continuum of approaches for school education by Jane Waite
(https://blogs.kcl.ac.uk/cser/2018/01/05/a-continuum-of-scaffolding/)
# What next?

## Upcoming meetings and conferences around computing education research

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICER 2018</td>
<td>Aug 13-15, Helsinki</td>
</tr>
<tr>
<td>WIPSCE 2018</td>
<td>Oct 4-6, Potsdam</td>
</tr>
<tr>
<td>ISSEP 2018</td>
<td>Oct 10-12, St Petersburg</td>
</tr>
<tr>
<td>CAS Research meetings</td>
<td>October half-term, Feb half-term, etc.</td>
</tr>
<tr>
<td>Computing Education Practice conference</td>
<td>Jan 9, Durham</td>
</tr>
<tr>
<td>SIGCSE 2019</td>
<td>Feb 27-Mar 2, Minneapolis</td>
</tr>
<tr>
<td>LCERS at King’s College London</td>
<td>June</td>
</tr>
<tr>
<td>ITICSE 2019</td>
<td>July 15-17, Aberdeen</td>
</tr>
</tbody>
</table>

Find this and more information at the UK-ACM SIGCSE website: [https://uki-sigcse.hosting.acm.org/](https://uki-sigcse.hosting.acm.org/)
Take-away thought

This should be an exciting time to be involved in computing education research due to the emergence of the subject in schools. The opportunities provided by new computing curricula coupled with advances in technologies and analytical tools with which to mine big datasets, and the increasingly interdisciplinary nature of educational research, offer enormous scope for advancing computing teaching and learning.

The Royal Society’s After the Reboot report, 2018
References


WIPSCOE '12. ACM, New York


