24th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)

Aberdeen, Scotland, UK

15-17 July, 2019

SIG CSE

Program Overview

Time	Monday 15th July			
08:00 - 08:30	Registration			
08:30 - 09:00	Welcome/Opening			
09:00 - 10:00	Keynote 1: Kate Stone			
10:00 - 10:30	Coffee/Posters			
10:30 - 12:00	Parallel Session 1			
	Identity	Novice Programmers	Curriculum	Pedagogy
12:00 - 13:00	WG1	Lunch		
13:00 - 14:00	GitHub Education Presentations			
14:00 - 15:30	Parallel Session 2			
	Expectation and Experience	Programming	Curriculum	Pedagogy
15:30 - 16:00	Coffee/Posters			
16:00 - 17:30	Parallel Session 3			
	Panel 1	Pair Programming	CS1	Student Reasoning

Time	Tuesday 16th July		
08:00 - 09:00	Lean Coffee		
09:00 - 10:00	Parallel Session 4		
	Mendix Presentation	Panel 2	Tips, Techniques and Courseware 1
10:00 -	Networking		Networking
10:30 - 11:00	Coffee/Posters		
11:00 - 12:30	Parallel Session 5		
	Working Groups Presentations 1	Professional Development	Tips, Techniques and Courseware 2
12:30 - 13:00	Collect Packed Lunches		
13:00 - 18:00	Excursions		
19:00 - 23:00	Conference Banquet		

Time	Wednesday 17th July			
08:00 - 09:00	Lean Coffee			
09:00 - 10:00	Keynote 2: Marian Petre			
10:00 - 10:30	Coffee/Posters			
10:30 - 12:00	Parallel Session 6			
	Panel 3	Working Groups Presentation 2 and Doctoral Consortium Presentations	Programming	Experience reports
12:00 - 13:00	Lunch			
13:00 - 14:30	Parrallel Session 7			
	School Education Issues	Computational Thinking	Programming	Software Architecture and Algorithms
14:30 - 15:00	Coffee/Posters			
15:00 - 16:30	Parrallel Session 8			
	School Education Issues	Operating Systems and Security	Learning Environments	Evaluation and Analytics
16:30 - 17:00	Closing Session			

Table of Contents

ITiCSE 2019 Conference Committee

Conference Chairs
Bruce Scharlau *(University of Aberdeen, UK)*
Roger McDermott *(Robert Gordon University, UK)*

Programme Chairs
Arnold Pears *(KTH Royal Institute of Technology, Sweden)*
Mihaela Sabin *(University of New Hampshire, USA)*

Working Group Chairs
Guido Rößling *(Technische Universität Darmstadt, Germany)*
Michail Giannakos *(Norwegian University of Science and Technology (NTNU), Norway)*

Database Coordinator
Janet Carter *(University of Kent, UK)*
Simon *(University of Newcastle, Australia)*

Web Page Coordinator
Dennis Bouvier *(Southern Illinois University Edwardsville, USA)*

Tips, Techniques & Courseware
Jane Sinclair *(University of Warwick, UK)*

Posters and Panels
Bedour Alshaigy *(Oxford Brookes University, UK)*

Doctoral Consortium
Mark Zarb *(Robert Gordon University, UK)*
Angela Siegel *(Dalhousie University, Canada)*

Student Coordinator
David Corsar *(Robert Gordon University, UK)*

Proceedings
Stan Kurkovsky *(Central Connecticut State University, USA)*
Jim Paterson *(Glasgow Caledonian University, UK)*

Event Organisation and Evaluations
Julie Dixon *(University of Aberdeen, UK)*

ITiCSE 2019 Conference Committee

Associate Programme Chairs

Michal Armoni *(Weizmann Institute of Science, Israel)*
Brett Becker *(University College Dublin, Ireland)*
Bo Brinkman *(Google, Inc, USA)*
Kevin Buffardi *(California State University – Chico, USA)*
Sridhar Chimalakonda *(Indian Institute of Technology Tirupati, India)*
James Davenport *(University of Bath, UK)*
Nick Falkner *(The University of Adelaide, Australia)*
Cruz Izu *(The University of Adelaide, Australia)*
Meena Jha *(Central Queensland University, Australia)*
Carsten Kleiner *(University of Applied Sciences & Arts Hannover, Germany)*
Daniel Krutz *(Rochester Institute of Technology, USA)*
David Levine *(St Bonaventure University, USA)*
Violetta Lonati *(Università degli Studi di Milano, Italy)*
Andrew Luxton-Reilly *(University of Auckland, New Zealand)*
Larry Merkle *(Air Force Institute of Technology, USA)*
Gabriela Moise *(PG University of Ploiesti, Romania)*
Mattia Monga *(Università degli Studi di Milano, Italy)*
Christopher Moretti *(Princeton University, USA)*
James Paterson *(Glasgow Caledonian University, UK)*
Keith Quille *(Technological University Dublin, Ireland)*
Saquib Razak *(Carnegie Mellon University, Qatar)*
Guido Rößling *(Technische Universität Darmstadt, Germany)*
Claudia Szabo *(The University of Adelaide, Australia)*
Cynthia Taylor *(Oberlin College, USA)*
Troy Vasiga *(University of Waterloo, Canada)*

Welcome to ITiCSE 2019 in Aberdeen, Scotland, UK

Welcome to ITiCSE 2019 at the University of Aberdeen! The university was founded in 1495 and is Scotland's third oldest university, as well as the fifth oldest university in the English-speaking world. This year it was acclaimed as the 'Scottish University of the Year' by the Times and Sunday Times Good University Guide.

Aberdeen is a medieval university city of about 250,000 people, which sits on the east coast of the North Sea between the Rivers Dee and Don, and is Scotland's third largest city. The city, known for its energy related business, sits at the heart of an area of outstanding natural beauty which includes Royal Deeside, the Grampian mountains, and the Aberdeenshire coast. It is also less than 50 miles from the heart of the Scotch malt whisky industry, and a departure point to visit the historic and prehistoric sites on the islands of Orkney and Shetland. We also have the world's busiest heliport, which annually ferries 500,000 staff to and from the oil rigs in the North Sea. This is why you'll regularly hear helicopters during your visit.

This year, an ITiCSE record of 243 papers were submitted, of which 66 were accepted, giving an acceptance rate of 27%. Of these papers, just over half had an author from the United States or Canada, while European authors were represented in about 40% of the papers. We also accepted papers with authors from Central and South America, China, Japan, Australia, and the Middle East, giving us a truly international flavour of current Computer Science Education research and practice.

In addition to the Paper, Poster and Panel submissions, and Tips, Techniques and Courseware presentations, we have ten Working Groups investigating these topics: the pacing of introductory CS courses; fostering program comprehension for novice programmers; exploring pass rates in computing and other STEM subjects; sustainability issues in CS; diversity in the cybersecurity field; data science education; benchmarking K-12 CS education in schools; developing a model curriculum for cloud computing; and designing better compiler error messages. The reports from these groups will be published in a companion volume to the final proceedings, but we look forward to the working groups presenting preliminary findings during the conference.

We are introducing a Doctoral Consortium over the weekend ahead of the conference in order to provide PhD students studying computing education an opportunity to explore and develop their research interests in a workshop environment with a panel of established researchers. We hope this encourages new members into our academic community where they can feel welcome, and contribute to ongoing work in our area.

We have two keynote speakers. Dr Kate Stone, the founder of Novalia, will talk about her work making technology disappear though the power of digital ink. Marian Petre, Professor of Computing at the Open University in the UK, will reflect on long-term empirical research on software design and consider how expert practice can inform how we teach the subject.

The conference banquet will be held in the Beach Ballroom, which an art deco building on the seafront famous for its steel sprung dancefloor. Naturally, for 'castle country' you'll have the chance to visit a castle, and taste some whisky too as part of the excursions.

We look forward to talking to you while you're here. We know you'll have fun making new friends, and reacquainting yourself with old ones too, as you fill up on new ideas and

innovations to take home to your classroom. We hope you'll also take home some wonderful memories of our lovely surrounding countryside and the beautiful Aberdeenshire coast.

Slàinte

Bruce Scharlau and Roger McDermott
ITiCSE 2019 Conference Chairs

Arnold Pears and Mihaela Sabin
ITiCSE 2019 Programme Chairs

Sponsors

We thank all of our sponsors for helping to make ITiCSE 2019 in Aberdeen possible. They helped us with general funding, plus venues and catering, and your bags for extra souvenirs, as well as funding for PhD students to launch the Doctoral Consortium. Thank you all.

GitHub Education

GitHub Education - Platinum Sponsor

Real world tools for engaged students. GitHub Education helps students, teachers, and schools access the tools and events they need to shape the next generation of software development. Over 1 million teachers and students at schools around the world use GitHub to accomplish their learning goals.

mendix

Mendix - Gold Sponsor

Where thinkers become makers. Mendix is the only platform that empowers both business and professional developers to make apps that get to value sooner. Mendix combines Speed, Collaboration, and Control. The result is business-changing software made easy.

SIGCSE acm Association for Computing Machinery sicsa The Scottish Informatics & Computer Science Alliance

acm Europe Council INFORMATICS EUROPE Bloomberg Engineering

Springer 1495 UNIVERSITY OF ABERDEEN BON ACCORD ABERDEEN CITY COUNCIL

ITiCSE Code of Conduct

The open exchange of ideas and the freedom of thought and expression are central to the aims and goals of the ACM SIGCSE Organization and ITiCSE Conference; these require an environment that recognizes the inherent worth of every person and group, that fosters dignity, understanding, and mutual respect, and that embraces diversity. The ACM Code of Ethics embraces the "values of equality, tolerance, respect for others, and the principles of equal justice". For these reasons, the ACM SIGCSE Organization and ITiCSE Conference are dedicated to providing a harassment-free conference experience.

Harassment is unwelcome or hostile behavior, including speech that intimidates, creates discomfort, or interferes, in an ACM SIGCSE Organization event. Harassment in any form, including but not limited to **harassment based on race, gender, religion, age, color, national origin, ancestry, disability, sexual orientation, or gender identity, will not be tolerated**. Harassment includes the use of gratuitous language or sexual imagery in public presentations and displays, degrading verbal comments, deliberate intimidation, stalking, harassing photography or recording, inappropriate physical contact, and unwelcome sexual attention.

Conference participants violating these standards may be sanctioned, expelled from the conference or asked not to attend future conferences or conference events, at the discretion of the conference organizers and the SIG executive committee.

If you believe you have been harassed or notice that someone else is being harassed, or have any other concerns, you are encouraged to report the incident in confidence to either of the conference chairs.

https://www.acm.org/about-acm/policy-against-harassment

Best Paper Award

The Best Paper Awards at ITiCSE 2019 is sponsored by the ACM Europe Council.

The ACM Europe Council Best Paper Awards recognise authors of outstanding technical contributions to ACM conferences taking place in Europe. The award acknowledges groundbreaking research in each conference's respective field for its importance and contribution to the area, to highlight theoretical and practical innovations that are likely to shape the future of computing both in Europe and worldwide.

Award

Each of the authors of the winning paper will receive a plaque. In addition, there will be a 1000 Euro honorarium to be split among the authors. Moreover, the recipient's paper will be provided online through the ACM Europe Council Awards page.

Doctoral Consortium

The Doctoral Consortium introduced this year has the following objectives:

- Provide a supportive setting for feedback on doctoral research and research direction.
- Offer each candidate comments and fresh perspectives on their work from researchers and other candidates outside their own institution.
- Promote the development of a supportive community of scholars.
- Support a new generation of researchers with information and advice on research and academic career paths.
- Contribute to the conference goals through interaction with other researchers and conference events.
- Furthermore, candidates attending the DC are also required to present a poster at the main conference, allowing them to start networking with the wider ITiCSE community. Further details on this will be given following acceptance.
- The DC will be held on Saturday 13th and Sunday 14th July 2019, but accepted candidates are expected to participate in the rest of the conference until Wednesday 17th July 2019. Candidates at any stage of their doctoral studies are welcome to apply and attend. The number of participants is limited to ten.

Doctoral Consortium Submissions

Mohammad Tahaei (University of Edinburgh)
'I Don't Know Too Much About It': On the Security Mindsets of Future Software Creators

David Williams (Pulse College)
Utilising Game Design To Create Engaging Education

Madeleine Lorås (Norwegian University of Science and Technology)
From Studying to Learning Computer Science - A study of the first-year experience of computer science education at university

Ioannis Karvelas (University College Dublin)
Investigating Novice Programmers' Interaction with Programming Environments

Justyna Szynkiewicz (NORD University)
Learning through Construction's Influence on IT Students' Identity Formation

Amanpreet Kapoor (University of Florida)
A Grounded Theory of Computing Professional Identity Formation

Marialaura Moschella (Free University of Bolzano)
Computational Thinking At Primary School: Didactical and Psychological Aspects

Tony Lowe (Purdue University)

A Theory of Applied Mind of Programming

Liat Nakar (Weizmann Institute of Science)

Pattern-Oriented Instruction and its Influence on Meaningful Learning of Algorithmic Patterns and Acquiring Fundamental Skills in Computer Science

Léon McGregor (Heriot-Watt University)

Gamification and Collaboration to Evaluate and Improve the Security Mindset of Developers

Lean Coffee

There will be a Lean Coffee session each morning on Tuesday and Wednesday from 08:00 - 09:00 in Elphinstone Hall. This provides an opportunity to discuss CS education topics in a short-format, small group exercise. We'll set up groups as numbers require.

Grab a coffee, or other beverage, from the bakery on the High Street before coming to join us. They open at 07:00.

"*Lean Coffee is a structured, but agenda-less meeting. Participants gather, build an agenda, and begin talking. Conversations are directed and productive because the agenda for the meeting was democratically generated*." http://leancoffee.org

Conference Venues

ITiCSE 2019 is being held at the University of Aberdeen King's College campus:

- Registration check-in: Elphinstone Hall
- Helpdesk: Elphinstone Hall
- Plenary Talks: William Guild Building, Arts Lecture Theatre (ALT)
- Parallel Session 1: New King's lecture theatre 6 (NK 6)
- Parallel Session 2: New King's lecture theatre 10 (NK 10)
- Parallel Session 3: Regent Lecture Theatre (Regent)
- Parallel Session 4: New King's lecture theatre 1 (NK 1)
- Posters/Exhibitors: Elphinstone Hall
- Coffees and Lunches: Elphinstone Hall (weekend: Linklater Rooms)
- Lean Coffee: Elphinstone Hall
- Conference Banquet: Beach Ballroom, Beach Promenade, Aberdeen AB24 5NR

Morning Sessions

8:30-9:00 **Room: ALT**	**Welcome and Opening Session**
09:00-10:00 Chair: Mihaela Sabin **Room: ALT**	**Keynote 1** Kate Stone **Designing for a Connected World**

10:00-10:30 **Elphinstone Hall**	**Coffee/Posters** Aparna Mahadev **Insights from using Supplemental Instruction (SI) in Data Structures course to increase retention** Juan J. Olarte, Cesar Dominguez, Arturo Jaime Elizondo, Francisco J. Garcia **Organization of part-time internships in Computer Science Engineering** Abdurrahman Arslanyilmaz, Kendra Corpier **Eye Tracking to Evaluate Comprehension of Computational Thinking** Tom McKlin, Taneisha Lee, Dana Wanzer, Brian Magerko, Doug Edwards, Sabrina Grossman, Emily Bryans, Jason Freeman **Exploring the Correlation Between Teacher Pedagogical Content Knowledge and Student Content Knowledge in Computer Science Classrooms** Barry Fagin **Idempotent Factorizations: A New Addition to the Cryptography Classroom** Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, Yasushi Kuno **How Children Express Intents with Their Codes? Analysis of Viscuit Programs from Kindergarten Module** Hillary Dawkins, Grant Douglas, Kevin Glover-Netherton, David Hudec, Sean Lunt, Dalton Polhill, Mostafa Rashed, Matthew Sampson, Alliyya Mohammed, James Mosley, Rhys Young, Judi McCuaig **Development of a checklist tool for teaching problem-solving Skills** Ernestina Menasalvas, Ana M. Moreno, Nik Swoboda **A Proposal for Recognizing Skills in Data Science Using Open Badges** Rebecca Reuter, Marco Knietzsch, Florian Hauser, Jürgen Mottok **Supporting Abstraction Skills Using Augmented Reality?** Ryan Crosby, Marie Devlin, Lindsay Marshall **Computing: Ipsative Assessment. Improving the student experience in higher education using personalised assessment and feedback**

Monday Keynote

Kate Stone

Designing for a Connected World

The march of the digital frontline seems to have relentlessly vaporised so much of what is around us and sent it to the cloud. With our heads now so often stuck in those clouds we

seem to be losing touch of what is physically around us; including each other. What if we could add a digital soul to everyday objects and design, prototype and manufacture physical products almost as quickly as digital experiences? My work is a journey towards attempting to let some of this happen.

Biography

Kate has spent the last decade on a journey of discovery from the world of science to creative design. Kate and her team at Novalia have developed a new technology platform with which they create products that are a delightful blend of being magical, old fashioned and futuristic. Kate believes the future will look more like the past than the present due to our natural mix of nostalgia and futuristic stargazing. Over the last few years with her team she has created experiences for large brands as part of advertising campaigns, working with Pizza Hut, McDonalds, Bud Light, Hersheys and IKEA. They are presently working on a children's toy called 'Touchscape' that is a board game like surface that connects to Amazon's Alexa.

Kate is from the UK however now lives in Woodstock, New York and has a degree in electronics from Salford University and PhD in physics from Cambridge University but believes the most useful things she learnt in life were discovered during four years of travel through Australia and Asia and in particular working on a sheep farm for two years in the Australian outback.

Kate has spoken at TED twice and both talks can be found online on TED.com.

Sessions	10:30	11:00	11:30
1A: Identity **Chair: Anne-Kathrin Peters** **Room: Regent**	Ella Taylor-Smith, Sally Smith and Colin Smith **Identity and Belonging for Graduate Apprenticeships in Computing**	Amanpreet Kapoor and Christina Gardner-Mccune **Understanding CS Undergraduate Students' Professional Identity through the lens of their Professional Development**	Mikko-Ville Apiola and Mikko-Jussi Laakso **The Impact of Self-theories to Academic Achievement and Soft Skills in Undergraduate CS Studies: First Findings**
1B: Novice Programmers **Chair: Briana Morrison** **Room: NK6**	Ricardo Caceffo, Pablo Frank-Bolton, Renan Souza and Rodolfo Azevedo **Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory**	Tony Lowe **Explaining novice programmer's struggles, in two parts**	Kathryn Cunningham, Mark Guzdial, Shannon Ke and Barbara Ericson **Patterns, structure, and persistence: Implications for interactive program visualization from novice sketching and tracing**
1C: Curriculum **Chair: Katrina Falkner** **Room: NK 10**	Henrik Nygren, Juho Leinonen and Arto Hellas **Non-restricted Access to Model Solutions: A Good Idea?**	Simon, Brett Becker, Sally Hamouda, Robert McCartney, Kate Sanders and Judy Sheard **Visual Portrayals of Data and Results at ITICSE**	Brett A. Becker **A Survey of Introductory Programming Courses in Ireland**
1D: Pedagogy **Chair: Richard Glassey** **Room: NK 1**	Grégoire Fessard, Patrick Wang and Ilaria Renna **Are There Differences in Learning Gains When Programming a Tangible Object or a Simulation?**	Amruth Kumar **Helping Students Solve Parsons Puzzles Better**	Louis Nisiotis and Styliani Kleanthous **The Relationship between Student's Engagement and the Development of Transactive Memory Systems in MUVE: An Experience Report**

Session 1A: Identity (Room: Regent)

Ella Taylor-Smith, Sally Smith and Colin Smith

Identity and Belonging for Graduate Apprenticeships in Computing

In September 2017, our university's first graduate apprentices began degrees in Software Development, Cybersecurity, and Information Technology Management for Business. This study explores how apprentices experience their association with the university and identities as students, but also employees. In Scotland, Graduate Apprenticeships (GAs) are undergraduate honours degrees in which the students are in full-time employment, while completing degree modules over four years, as for a traditional full-time degree. The curriculum follows a skills framework designed by employers so that graduates have the professional and technical attributes required by the industry. The degrees parallel Degree Apprenticeships in England, though there are national differences in implementation. Themes of identity and belonging are central to current investigations of the experience of STEM students, especially computing students, as fewer students choose STEM courses, and many transfer out of their subjects or do not complete their degrees. The research hypothesis is that the apprentices' employment will provide a strong IT professional identity supports their progress at university.

Semi-structured interviews with apprentices in the first computing cohorts explored their situated perspectives. Responses were identified which concerned the apprentices' identity as students or employees, including themes around belonging. Thematic analysis of these responses revealed that apprentices defined themselves in opposition to traditional student identities and did draw strength from their identity as employees. They experienced belonging specifically within their GA cohort—the first of its kind in the university. A better understanding of identity and belonging can be used by universities to address the challenges of retention.

Amanpreet Kapoor and Christina Gardner-Mccune

Understanding CS Undergraduate Students' Professional Identity through the lens of their Professional Development

Academic institutions play a crucial role in development of students' professional identities. However, we have limited knowledge of how computing professional identity develops. This paper aims to understand how CS undergraduate students develop their professional identity through analyzing students' reflection on their career goals, experiences in CS degree programs, and engagement in professional development. We present empirical findings from qualitative analysis of 14 semi-structured interviews with CS undergraduate students in the United States. We analyzed (1) the extent to which students feel committed to a specific computing profession, (2) the reasons why they are interested in these professions, and (3) the factors that shape the development of their professional identity. We found that a majority of CS freshman and sophomores are not committed but exploring CS professions while juniors and seniors are committed to a computing profession and engage in relevant professional development activities. We identified several reasons students are committed to a computing profession: intrinsic factors (e.g., interest, perception of ability, and personality), and discipline specific factors (e.g. utility and growth). We also found several factors that shape their professional identity: coursework, informal activities like hackathons, and professional development

activities including internships and conferences. These findings suggest that the development of computing professional identity is not limited to students involvement in the academic degree programs but the engagement they have with the broader computing community.

Mikko-Ville Apiola and Mikko-Jussi Laakso

The Impact of Self-theories to Academic Achievement and Soft Skills in Undergraduate CS Studies: First Findings

There is strong evidence of the impact of self-theories to students' academic achievement and behavior in many domains of education, including CS. However, the research on self-theories in CSE are far from conclusive. In this research, we studied 1st year CS and CE students' self-theories and looked at associations to academic achievement in their first courses, as well as the students' conceptualisations of intelligence. Contradictory to previous research, students with a fixed view on intelligence received the best exam scores. Students' conceptualisations of intelligence were found to lean towards cognitive g-theories of intelligence. These initial findings show a number of crucially important future research directions in relation to self-theories, soft skills development, and academic achievement in CS studies.

Session 1B: Novice Programmers (Room: NK6)

Ricardo Caceffo, Pablo Frank-Bolton, Renan Souza and Rodolfo Azevedo

Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory

A misconception is a common misunderstanding that students may have about a specific topic. The identification, documentation, and validation of misconceptions is a long and time-consuming work, usually carried out using iterative cycles of students answering open-ended questionnaires, interviews with instructors and students, exam analysis, and discussion with experts. A comprehensive list of validated misconceptions in some subject can be used to build formal evaluation methods like the Concept Inventory (CI), a multiple-choice questionnaire that is usually performed as pre-post tests in order to assess any change in student understanding. In CS1, validated misconceptions were identified and documented in C and Python programming languages. Although there are studies related to misconceptions in the Java language, these misconceptions lack the formality, comprehensiveness, and robustness of their C and Python counterparts. On this work, we propose a methodology to adapt the validated misconceptions in C and Python to Java. Initially, through the analysis of an initial list of 33 misconceptions in C and 28 in Python, we identified and documented in an antipattern format 31 possible misconceptions in Java. We then developed a final term exam, composed of 7 open-ended questions, in which each question was designed to address some of the misconceptions covered in the course (N = 27). Through the analysis of the exams answers (N = 69 students), it was possible to validate 22 of the misconceptions (81%). Also, 6 new misconceptions were identified, leading to a total of 28 valid misconceptions in Java.

Tony Lowe

Explaining novice programmer's struggles, in two parts

The ITiCSE 2004 working group conducted a follow-up study to the McCracken group investigating why students struggled with presumedly easy coding tasks. Instead of coding, they chose multiple-choice questions to ascertain if students lacked fundamental knowledge of coding constructs instead of simply weak problem-solving skills. I am revisiting their analysis of two of their twelve questions to reinterpreting the results using dual process theory as a framework. Dual process theory breaks cognition into two parts: one slow, focused, rational thinking and the other quick, automatic and effortless responses. Dual process thinking has made its mark in economics, among other domains, and may provide insight into programmers and programming education. In reviewing how one student traces code and the group of participants responded to these two questions, dual process theory provides a simpler and more detailed explanation for novice tendencies and might better explains some of Lister et al.'s counter-intuitive findings. This piece is an introduction to dual process theory in the context of programming as first slice of a broader theoretical framework under construction which hopes to chart the mind of the programmer and novice. The first step is realizing how dual process theory changes the epistemological meaning of what is to 'know' how to program. The ability to code exceeds knowledge of concepts, also demanding speed and intuition.

Kathryn Cunningham, Mark Guzdial, Shannon Ke and Barbara Ericson

Patterns, structure, and persistence: Implications for interactive program visualization from novice sketching and tracing

We know that sketching out a code trace on paper is associated with higher scores for code reading problems. Why do students sometimes choose to not draw out a code trace, or choose a different sketching technique than their instructor has demonstrated? In this study, we interview 13 CS1 students retrospectively about their decisions to sketch and draw on a recent programming exam. We find that student sketching choices do not always align with a strict execution of the notional machine, and that a search for a program's goals, an attempt to create organizational structure among intermediate values, and tracking of prior steps and results drives many sketching choices. These results suggest new approaches for visualizing the action of the notional machine that better align with student problem-solving.

Session 1C: Curriculum (Room: NK)

Henrik Nygren, Juho Leinonen and Arto Hellas

Non-restricted Access to Model Solutions: A Good Idea?

In this article, we report an experiment where students in an introductory programming course were given the opportunity to view model solutions to programming assignments whenever they wished, without the need to complete the assignments beforehand or to wait for the deadline to pass. Our experiment was motivated by the observation that some students may spend hours stuck with an exercise, leading to non-productive study time. At the same time, we considered the possibility of students using the sample solutions as worked examples, which could help students to improve the design of their own programs. Our experiment suggests that many of the students use the model solutions

sensibly, indicating that they can control their own work. At the same time, a minority of students used the model solutions as a way to proceed in the course without learning, leading to poor exam performance. We discuss both the benefits and downsides of our experiment and provide guidelines for researchers and teachers planning to organize their course in a similar fashion.

Simon, Brett Becker, Sally Hamouda, Robert McCartney, Kate Sanders and Judy Sheard

Visual Portrayals of Data and Results at ITICSE

We present an analysis of the visual portrayals of data (including results) in the full papers and working group reports of ITiCSE from 2013 to 2018. We find that tables are the most common visual portrayal of data in these publications, but that there are a number of graphical forms that are widely used. We examine the quality of the data portrayals for tables and graphs using sets of visual quality indicators devised from guidelines sourced in the literature.

Overall, our findings are not positive. We find large numbers of papers that present data in a way that cannot be readily interpreted. The most common problem is captions that do not adequately describe the table or figure. The main issue affecting readability of numeric data is poor alignment of numbers within a column and the use of unnecessary notations and unwarranted precision. On the other hand, we identify a number of papers whose visual presentation of data is exemplary.

We conclude with guidelines for future authors to ITiCSE and other computing education venues, in the hope that we can contribute to an improvement in the quality of computing education publications.

Brett A. Becker

A Survey of Introductory Programming Courses in Ireland

Between January and April of 2018, a comprehensive survey of introductory programming courses was undertaken across all sectors of Irish third-level institutions (Universities, Institutes of Technology, and Private Colleges). The survey instrument was based on, and nearly identical to, recent surveys in the UK and Australasia. In total we report on 39 introductory programming courses at 25 third-level institutions. This includes 6 of 7 Universities and 13 of 14 Institutes of Technology, representing 90% of all publicly funded institutions. We also report on 4 private colleges representing 80% of colleges in the Irish Higher Education Colleges Association that offer computing degrees.

In addition to general course structure and composition, we explore programming language use, the reasons for using them, and their impact on teaching and learning. We also gain first-hand insight from instructors through an analysis of their own course aims. We also explore their viewpoints on teaching introductory programming. The results of this survey provide a unique insight into the CS1 courses of a single country that is currently introducing Computer Science at the K-12 level.

Session 1D: Pedagogy (Room: NK 1)

Grégoire Fessard, Patrick Wang and Ilaria Renna

Are There Differences in Learning Gains When Programming a Tangible Object or a Simulation?

Physical computing is about programming and interacting with a tangible object to learn fundamental concepts of Computer Science (CS). This approach presents several benefits regarding motivation, creativity, and learning gains. Yet, these learning gains hardly are compared with those resulting from the programming of a simulation of a tangible object. In this article we present the results of a comparative study that has been conducted to explore this issue. With this study, we aimed to determine whether the programming of a tangible object or its digital simulation yields significantly different learning gains. In the experiment we conducted, participants (aged 14-17 with little or no prior programming knowledge) were divided into two groups: one programmed a tangible electronic board (the BBC micro:bit) while the other programmed a simulation of it. The results of this experiment suggest that, while each group significantly improved their understandings of fundamental CS concepts (i.e., variables, conditions, and loops), no significant difference was found when comparing the learning gains between the two groups.

Amruth Kumar

Helping Students Solve Parsons Puzzles Better

In a Parsons puzzle, the student must re-assemble the lines of a program that are provided in scrambled order, and eliminate any distracters included within the scrambled lines of code. We investigated two issues in terms of whether they helped students solve the puzzles better: 1) Would it be better to present distracters and the lines of code of which they are a variant paired together or randomly separated apart? 2) Would telling students that they would not be penalized for any mistakes they make before submitting a complete solution for the first time enable them to get closer to the correct solution when they first submit it? We conducted a controlled study over five semesters and used ANOVA to analyze the data collected from introductory programming students who solved Parsons puzzles on if-else statements. We found that when students were told that their puzzle-solving actions before the first submission would not be penalized, they took significantly more exploratory actions. But, their solution was not significantly closer to being correct. So, trial-and-error exploration was no better than deliberate approach at solving Parsons puzzles. Presenting distracters paired together with the original line of code was found to be beneficial only on the longer puzzle.

Louis Nisiotis and Styliani Kleanthous

The Relationship between Student's Engagement and the Development of Transactive Memory Systems in MUVE: An Experience Report

Student engagement is a very important topic in higher education hence, it drew a lot of research interest over the years. The use of educational Multi-User Virtual Environments (MUVEs) that provide synchronous interaction, dynamic, interactive and social learning experiences have the potential to increase student engagement and contribute to their learning experience. Due to increased social and cognitive presence, the use of such environments can result in greater student engagement when compared to traditional

asynchronous learning environments. In this work, we hypothesized that students' engagement in collaborative learning activities will increase if TMS constructs are present. Thus, we employed the theory of Transactive Memory System (TMS) that emphasizes the importance of Specialization, Coordination and Credibility between members in a team. The results show that there is a significant correlation between the development of TMS and students' engagement. In addition, further quantitative and qualitative analysis reveals some interesting facts about students' engagement with respect to their collaboration in group activities.

Afternoon Sessions

12:00-13:00 **Elphinstone Hall**	**Lunch**
12:00-13:00 **Elphinstone Hall**	**WG1: World Café on Climate Change and Sustainability in CS Education**
13:00-14:00 Chair: Bruce Scharlau **Room: ALT**	Mine Cetinkaya-Rundel and Shane Wilson **GitHub Education Presentations**

Sessions	14:00	14:30	15:00
2A: Expectation and Experience **Chair: Judy Sheard** **Room: Regent**	Jyoti Bhardwaj **New Entrants' Expectations of the First Year Computer Science Experience in the Context of a New National High School Curriculum**	Remin Kasahara, Kazunori Sakamoto, Hironori Washizaki and Yoshiaki Fukazawa **Applying Gamification to Motivate Students to Write High-Quality Code in Programming Assignments**	Yingjun Cao, Leo Porter, Soohyun Nam Liao and Rick Ord **Paper or Online? A Comparison of Exam Grading Techniques**
2B: Programming **Chair: Andrew Luxton-Reilley** **Room: NK 6**	Frank Höppner **Measuring Instruction Comprehension by Mining Memory Traces for Early Formative Feedback in Java Courses**	James Power and John Waldron **Quantifying Activity and Collaboration Levels in Programming Assignments**	Hieke Keuning, Bastiaan Heeren and Johan Jeuring **How Teachers Would Help Students to Improve Their Code**
2C: Curriculum **Chair: Mats Daniels** **Room: NK10**	Ella Taylor-Smith, Sally Smith, Khristin Fabian, Tessa Berg, Debbie Meharg and Alison Varey **Bridging the Digital Skills Gap: Are Computing Degree Apprenticeships the Answer?**	Emma Riese, Olle Bälter, Björn Hedin and Viggo Kann **Programme integrating courses fighting to get engineers to reflect on non-technical topics**	Ari Korhonen and Marianna Vivitsou **Digital Storytelling and Group Work: Integrating the Narrative Approach into a Higher Education Computer Science Course**

2D: Pedagogy **Chair: Robert McCartney** **Room: NK 1**	Michael Lodi **Does Studying CS Automatically Foster a Growth Mindset?**	Seth Copen Goldstein, Hongyi Zhang, Majd Sakr, Haokang An and Cameron Dashti **Understanding How Work Habits Influence Student Performance**	Amber Solomon, Vanessa Oguamanam, Mark Guzdial and Betsy DiSalvo **Making CS Learning Visible: Case Studies on How Visibility of Student Work Supports a Community of Learners in CS Classrooms**

World Cafe During Monday Lunch

During the latter part of the lunch (after you've had a chance to eat something) Working Group 1 would like to find out your views on a range of topics using the World Cafe format. There will be a series of tables hosted by WG members to facilitate your answering of questions about their topic so that your views can be incorporated into their report. We'll let you know where this will happen on the day.
WG 1 is exploring "1.5 Degrees of Separation: Computer Science Education in the Age of the Anthropocene".

GitHub Education Session (Room: ALT)

Mine Cetinkaya-Rundel and Shane Wilson

GitHub Education Presentations

GitHub as Platinum Sponsors are offering a session by two academics, who use the platform in their classrooms.
Dr. Mine Çetinkaya-Rundel, University of Edinburgh. Çetinkaya-Rundel, who has used GitHub Classroom in her courses at Duke University, has taken that knowledge to build ghclass—a package for managing her Classroom on GitHub using R, and to enable peer evaluation. She is an advocate for using open source tools in data science and statistics. Shane Wilson, University of Ulster. All teachers struggle with how to deliver feedback when students need it, and at scale. Wilson will share their story of how they increased passing rates and decreased grading time using GitHub Classroom and Travis CI. Wilson is a certified GitHub Campus Advisor, and can field questions about integrating Git and GitHub.

Session 2A: Expectation and Experience (Room: Regent)

Jyoti Bhardwaj

New Entrants' Expectations of the First Year Computer Science Experience in the Context of a New National High School Curriculum

Computer science (CS) has the highest first year dropout rate amongst UK undergraduates. In this context, understanding the expectations of new students is a first step to designing a CS curriculum that will improve their first year experience. The aim of this study was to present new entrants' reflections on their school CS, once they have started university and can compare teaching and learning in the different institutions.
The study is based on interviews with 84 Scottish-educated students in two cohorts which bridge the introduction of the Curriculum for Excellence (CfE) and new public examinations in Computing Science, intended to bring about significant change in students' attributes and attitudes to learning. The participants vary in level of computing qualifications, reflecting the heterogeneous background of CS entrants.
Both cohorts were asked if they felt their learning in the senior years of high school had prepared them for university, and whether they felt they were taught or learned for themselves during school. The students were further asked whether they expected that the proportion of teaching to learning would differ at university.
The findings indicate that many students speak positively about high school CS, but many criticise the dominance of teacher-led instruction that ill-prepares them for self-regulated learning at university. Despite the intentions of CfE, the findings indicate little difference between the cohorts. However, students provide valuable perspective on the Scottish Computing Science experience and the role of Advanced Higher computing, which should prove interesting to those responsible for retention and curriculum design.

Remin Kasahara, Kazunori Sakamoto, Hironori Washizaki and Yoshiaki Fukazawa

Applying Gamification to Motivate Students to Write High-Quality Code in Programming Assignments

Background: Traditional programming education focuses on training students' ability to write correct code that meets the specifications in programming assignments. In addition to correctness, software engineering studies argue that code quality is important.
Problem: It is difficult to nurture students' ability to write high-quality code in programming assignments due to two main reasons. (1) Considering code quality while grading is undesirable because there are no objective and fair measurement metrics. (2) Grading assignments from multiple viewpoints (correctness and quality) is difficult and time-consuming. Approach: We thus propose applying gamification with code metrics to measure code quality in programming assignments. Our approach can motivate students to write code with good metric scores independent of grading. We implemented our approach and conducted a control experiment in a programming course at a university.
Result: We found that two key results: (1) our approach did not interfere in students' submissions and (2) our approach statically significantly improved metric scores. We thus conclude that our approach can engage students to write high-quality code.

Yingjun Cao, Leo Porter, Soohyun Nam Liao and Rick Ord

Paper or Online? A Comparison of Exam Grading Techniques

As computer science enrollments continue to surge, exam grading requires significant instructional resources. Online grading platforms have been developed in recent years and have been adopted at a number of institutions; however, their effectiveness compared with traditional grading on paper is not fully known. This study is the first in CS to compare online and paper grading. Comparing overall time to grade, including all factors, online grading doesn't show a consistent advantage compared with paper grading. We observed that online grading is much faster during the actual grading phase, but some of this benefit is offset by the additional overhead prior to grading (e.g., scanning) for online grading. Examining student and grader preferences based on feedback, both groups show a strong preference for the online format, predominately due to the convenience of the online platform. Graders report being able to grade more accurately online with the ability to modify rubrics, but that grading on paper tends to result in more social interactions among graders.

Session 2B: Programming (Room: NK 6)

Frank Höppner

Measuring Instruction Comprehension by Mining Memory Traces for Early Formative Feedback in Java Courses

To support students (in a computer science programme) in their first steps with programming (in Java), we present a tool that provides early formative feedback regarding the comprehension of the instruction types. We consider instruction comprehension as an important milestone that must be reached before students actually use the instructions in their first own programs. The tool asks the students to provide memory traces for ordinary Java programs, which simplifies the provision of exercises, but renders the subsequent identification of frequently occurring problems more difficult. We therefore present an approach to automatically detect code fragments that are likely to cause difficulties for the group of students and discuss first results. Error statistics and problematic code fragments provide valuable insights how well the instruction set has been understood by the group and/or individual students.

James Power and John Waldron

Quantifying Activity and Collaboration Levels in Programming Assignments

This paper presents an experience report from a third-year undergraduate compilers course. We implement metrics to measure the degrees of similarity between submissions for programming assignments, as well as measuring the level of activity. We analyse data from a study of practical assignments, evaluated in the context of take-home formative assignments and a supervised summative examination. We present the results of our study, and discuss the utility of these metrics for our teaching practice.

Hieke Keuning, Bastiaan Heeren and Johan Jeuring

How Teachers Would Help Students to Improve Their Code

Code quality has been receiving less attention than program correctness in both the practice of and research into programming education. Writing poor quality code might be

a sign of carelessness, or not fully understanding programming concepts and language constructs. Teachers play an important role in addressing quality issues, and encouraging students to write better code as early as possible.
In this paper we explore to what extent teachers address code quality in their teaching, which code quality issues they observe and how they would help novices to improve their code. We presented student code of low quality to 30 experienced teachers and asked them which hints they would give and how the student should improve the code step by step. We compare these hints to the output of professional code quality tools.
Although most teachers gave similar hints on reducing the algorithmic complexity and removing clutter, they gave varying subsets of hints on other topics. We found a large variety in how they would solve issues in code. We noticed that professional code quality tools do not point out the algorithmic complexity topics that teachers mention. Finally, we give some general guidelines on how to approach code improvement.

Session 2C: Curriculum (Room: NK10)

Ella Taylor-Smith, Sally Smith, Khristin Fabian, Tessa Berg, Debbie Meharg and Alison Varey

Bridging the Digital Skills Gap: Are Computing Degree Apprenticeships the Answer?

This paper describes a study investigating whether apprenticeship computing degrees in Scotland are attracting additional entrants who will become IT professionals and fill a skills need. Government policy reports, from around the world, set out plans to address computing skills shortages by introducing additional education or training programmes. In the UK, this is put forward as a key goal of the new higher level apprenticeships, including Graduate Apprenticeships in Scotland and Degree Apprenticeships in the rest of the UK. These are promoted as bridging the skills gap by involving employers in curriculum design and widening access to people who would not want to study towards a traditional degree, because of the financial burden or time out of employment. New graduate apprentices at three Scottish universities were encouraged to complete a short survey, asking about their route into the degree, including their aspirations, motivations, and previous experience. Most respondents had begun an IT career before they started the apprenticeship and were upskilling and gaining an internationally recognised qualification. A third could be considered new to IT, including those coming straight from school and those moving into IT mid-career. The apprentices' primary motivation was to gain skills. They chose the apprenticeship, rather than a traditional degree, because of the integration of work experience, followed by financial reasons: graduate apprentices earn salaries and can avoid student debt. These apprenticeships create a new route to a computing degree-level qualification. We consider their potential in addressing digital skills shortages.

Emma Riese, Olle Bälter, Björn Hedin and Viggo Kann

Programme integrating courses fighting to get engineers to reflect on non-technical topics

Programme integrating courses aim to tie students, teachers and courses in an education programme closer together. In this study, we investigate three programme integrating courses, as part of engineering programmes in computer science and media technology. The purpose of this study was to gain a deeper understanding of how students and mentors experience the programme integrating courses, with a focus on the assessment

and the relationship between students, and students and mentors. We used a mixed method approach, interviewed 22 students and 6 mentors, and sent out questionnaires to all 25 mentors and all students from two of the three courses (630+470 students). The results showed that the students and mentors appreciated the social aspects of the course, getting to know each other and being able to share experiences. However, some were uncomfortable discussing personal opinions and reflect upon the given non-technical topics. On a general level, the students agreed with the statement that their mentors assessed their reflections correctly. The interviewed students did, however, experienced the assessment to be a bit unclear. The students were in general sceptical towards being graded on a scale other than pass/fail. We propose some actions build on the results.

Ari Korhonen and Marianna Vivitsou

Digital Storytelling and Group Work: Integrating the Narrative Approach into a Higher Education Computer Science Course

This study aims at discussing the integration of digital storytelling and the narrative approach into a University level Computer Science course. The pedagogical intervention took place on a project basis from October to December 2018. The pedagogical plan involved student work in groups for the production of digital stories in three phases, including an abstract, a manuscript and a final story.

The overall instructional design included planned workshops and lectures, online instructions and other teaching material, and group work. The students were assigned to explore the topic of recursion. Face-to-face meetings for the coordination of group work were emphasized during lectures, workshops and project instructions.

The study uses qualitative research methods and the findings indicate two main patterns of group work. The first pattern follows from loose coordination and division of tasks among group members at the initial stages of the project. This results in documentary-like and program-based video stories. The second pattern involves tighter collaboration with face-to-face meetings for common task completion, video recording and editing, and manuscript improvement. This mode of work results in short-film style stories where recursion is well-represented. In both patterns, however, the videos present external rather than internal examples of recursion. As a result, the digital stories represent what the code does instead of how it does it.

Session 2D: Pedagogy (Room: NK 1)

Michael Lodi

Does Studying CS Automatically Foster a Growth Mindset?

Many arguments are used to advocate the introduction of Computer Science (CS) / Computational Thinking / "coding" in K-12 education. Growth mindset theory (GM) is becoming as well very popular among educators and researchers. Some claims stating that studying CS can foster a GM emerged. However, education research shows that transfer of competences is hard. Very little research has been conducted on the relationship between GM and CS learning, with conflicting results. We measured some indicators (e.g. GM, "CS GM") at the beginning and the end of a High School year in five different classes: three CS oriented, one Chemistry oriented, and one Transportation&Logistics oriented. In one of the CS oriented classes, we did a very brief

GM intervention. At the end of the school year, none of the classes showed a statistically significant change in their GM. Interestingly, non-CS oriented classes showed a significant decrease in their CS GM. In the intervention class, students suggested, to stimulate a GM, activities that are more creative, engaging and related to the real world and their interests.

Seth Copen Goldstein, Hongyi Zhang, Majd Sakr, Haokang An and Cameron Dashti

Understanding How Work Habits Influence Student Performance

Understanding the relationship between a student's broader work habits and their performance, particularly on open-ended programming assignments, is key towards being able to guide students towards success. In spite of this, most evidence of student behavior and its relationship to performance is anecdotal. The advent of large-scale courses which use online tools for delivering course content, monitoring the programming environment, and providing automatic feedback as well as grading now makes it possible to dive into the data and develop data-driven methods for understanding how a student's approach to an assignment---from their first exposure to the description of the problem to their final submission of their completed assignment---influences their final performance. This study is a first look at a subset of the data collected from a project-based, online, upper-level course on cloud computing. We examine three raw data streams which include information on the time students spend on reading the project write-up, the timing, grades, and number of submissions they make, and the cloud resources used (both time and cost) in solving the assignment. Using these and several synthetic metrics we were surprised to find that there are few behaviors that are highly correlated to student success. Instead, we find that there is a strong correlation between students who continually apply consistent behaviors over the course of the semester to good final performance in the course. From this we use LASSO to create a predictor of final performance based on two kinds of consistency measures across 15 metrics.

Amber Solomon, Vanessa Oguamanam, Mark Guzdial and Betsy DiSalvo

Making CS Learning Visible: Case Studies on How Visibility of Student Work Supports a Community of Learners in CS Classrooms

Modern learning theories emphasize the critical social aspect of learning. Computer science classrooms often have "defensive climates" that inhibit social learning and prevent the development of a community of learners. We believe that we can improve the social context of computer science learning by expanding CS learning beyond the single person in front of the display. Our theory is that the single person and single display inhibits collaboration and collaborative awareness of student work. In this paper, we present two case studies where we explored ways to make student work visible to peers. The first case study involved using a studio model for learning enabled by projection-based Augmented Reality (AR), and the second case study involves using a maker-oriented curriculum to make student work visible. Findings suggest the visibility of student work in CS classrooms helped support a community of learners: students collaborated, used each other as sources of inspiration, and felt more comfortable asking for help.

15:30-16:00 **Elphinstone Hall**	**Coffee/Posters** Raghav V. Sampangi, Angela A. Siegel **Designing Engaging Learning Experiences** Gloria Townsend, Khadija Stewart **Computing Opportunities for Students of Color: Towards a best representative tech world** Mark Zarb, Tiffany Young, William Ballew **Integrating Real-World Clients in a Project Management** Tiffany Young, Mark Zarb **Challenges of Delivering a Graduate Apprenticeship** Tiffany Young, Mark Zarb **Incorporating On-Campus Days in a Graduate Apprenticeship** Melissa Stange, Cara Tang, Christian Servin, Cindy Tucker, Markus Geissler **Shaping Curricular Guidelines for Associate-Degree Cybersecurity Programs** Barbara Sabitzer, Iris Groher, Johannes Sametinger **COOL: Cooperative Open Learning for Beginning Programmers** Lubna Alharbi, Floriana Grasso, Phil Jimmieson **A conceptual framework for identifying emotional factors leading to student disengagement in Virtual Learning Environments** Gaetano Geck, Artur Ljulin, Jonas Haldimann, Johannes May, Jonas Schmidt, Marko Schmellenkamp, Daniel Sonnabend, Felix Tschirbs, Fabian Vehlken, Thomas Zeume **Teaching Logic with Iltis: an Interactive, Web-Based System**

Sessions	16:00	16:30	17:00
3A: Panel 1 **Chair: Bedour Alshaigy** **Room: Regent**	Angela Siegel, Mark Zarb, Richard Glassey and Janet Hughes **Perspectives on the Student Transition into CS1**		
3B: Pair Programming **Chairs: Kate Saunders** **Room: NK 6**	Nicholas Bowman, Lindsay Jarratt, KC Culver and Alberto Segre **How Prior Programming Experience Affects Students: Pair Programming Experiences and Outcomes**	Lindsay Jarratt, Nicholas Bowman, KC Culver and Alberto Segre **A Large-Scale Experimental Study of Gender and Pair Composition in Pair Programming**	Briana Bettin, Linda Ott and Leo Ureel **More Effective Contextualization of CS Education Research: A Pair-Programming Example**
3C: CS1 **Chair: Barbara Ericson** **Room: NK 10**	Rita Garcia, Katrina Falkner and Rebecca Vivian **Instructional Framework for CS1 Question Activities**	Soohyun Nam Liao, Sander Valstar, Kevin Thai, Christine Alvarado, Daniel Zingaro, William G. Griswold and Leo Porter **Behaviors of Higher and Lower Performing Students in CS1**	Quintin Cutts, Matthew Barr, Mireilla Bikanga Ada, Peter Donaldson, Steve Draper, Jack Parkinson, Jeremy Singer and Lovisa Sundin **Experience Report: Thinkathon - Countering an "I Got My Program Working" Mentality with Pencil-and-paper Exercises**
3D: Student Reasoning **Chair: Violetta Loneti** **Room: NK 1**	Aubrey Lawson, Eileen Kraemer, Megan Che and Cazembe Kennedy **A Multi-level Study of Undergraduate Computer Science Student Reasoning About Concurrency**	Claudia Szabo and Michael Pointon **Final Year Students' Approaches to Implementing Complex Distributed Systems**	Cazembe Kennedy and Eileen Kraemer **Qualitative Observations of Student Reasoning: Coding in the Wild**

Session 3A: Panel 1 (Room: Regent)

Angela Siegel, Mark Zarb, Richard Glassey and Janet Hughes

Perspectives on the Student Transition into CS1

As students transition into higher education, their experience can be quite new and foreign to them. This experience, while an individual one, consists of many concerns that are shared amongst these transitioning students. Partly as a result of these concerns, retention rates of first year Computer Science students suffer. Members of this panel have been involved in multi-year studies across Scotland as well as an international study looking at the student experience as they transition into undergraduate Computer Science. This panel hopes to discuss the transition into CS1 and implications for Computer Science retention rates from varied international perspectives as well as through the lens of online learning. It further hopes to discuss new ways that we might be able to better support first year CS student retention in light of collected transition data, as well as the current state of any implemented recommendations in previous work (you may want to rephrase this).

Session 3B: K-12 Outreach (Room CY007)

Nicholas Bowman, Lindsay Jarratt, KC Culver and Alberto Segre

How Prior Programming Experience Affects Students: Pair Programming Experiences and Outcomes

Pair programming is a collaborative learning approach in computer science in which students (or employees) work closely with a partner on the same programming task. A long-standing question within pair programming is whether certain combinations of students lead to greater learning, effort, and/or performance. Earlier studies have explored the role of prior programming experience, including the discrepancy between partners' experience, as a potentially important factor in shaping these outcomes. However, the previous findings are highly inconsistent, which may result from divergent (and often suboptimal) ways of defining previous experience or skill, problems with self-selection into pairs, and small sample sizes that often yield nonsignificant results. The present study sought to improve on all of these limitations through an examination of 587 undergraduates who each participated in three different randomly assigned pairings. Not surprisingly, students' own programming experience was positively related to understanding concepts from lab, confidence in the finished product, and overall interest in computer science. However, students who worked with a more experienced partner actually had poorer outcomes, including lower effort exerted on the assignment, perceptions that their partner gave more effort than they did, less time in the driving role (i.e., typing out the assignment), lower understanding of concepts from lab, and less interest in computer science overall. The partner's experience is unrelated to other outcomes, including confidence in the finished assignment, feeling productive during lab session, and average grades received during the pairing. These results provide important considerations for the assignment of students to pairs.

Lindsay Jarratt, Nicholas Bowman, KC Culver and Alberto Segre

A Large-Scale Experimental Study of Gender and Pair Composition in Pair Programming

The proportion of women in computer science majors is currently lower than in any other STEM major. Various studies have sought to explain—and ultimately find ways to reduce—gender disparities in computer science participation and persistence. Pair programming has been proposed as a practice that may not only promote outcomes overall within college and workplace environments, but also help diminish isolation and boost the confidence of women in computer science. Some promising results have been obtained for women in pair programming, but the findings are not consistent across studies, and the limitations of previous research make it difficult to draw strong conclusions. The present study examined 969 undergraduates in several introductory computer science courses who engaged in three different pairings throughout the semester. All pairings were randomly assigned, so the findings reflect the causal influence of the gender pair characteristics on a variety of student outcomes. Overall, having a female partner led to several positive outcomes relative to having a male partner; these included greater lab section attendance as well as greater confidence in the finished product and confidence in the solution for the pair programming assignment. The advantages of having a female partner were occasionally greater for female students than

for male students. Overall, the significant findings were most pronounced in the course intended for computer science majors. These results offer evidence for the educational benefits of pair programming for promoting women's participation in computer science, as well as the need for careful consideration of pair composition.

Briana Bettin, Linda Ott and Leo Ureel

More Effective Contextualization of CS Education Research: A Pair-Programming Example

This paper discusses the need for greater inclusion of context in the literature describing computer science education research. Our work arose from efforts to compare our experiences using pair programming in an introductory computer science course with experiences described in the literature. Pair programming has proven effective in both academia and industry for the value it can provide to code quality and programmer success. However, the settings and behaviors that encompass the term "pair programming" can differ greatly between universities. Still, descriptions of computer science education research often reference pair programming as if it were universally implemented in an identical way.

A brief literature survey is used to demonstrate omissions and differences in pair programming interpretation and the settings in which it has been used. Attributes that may be valuable to provide context are identified and discussed. Many attributes are likely appropriate for much CS education research. Pair programming, however, is specifically considered in the literature reviewed and in the development of the attributes considered. This anchors our paper and demonstrates specific attributes that require consideration beyond the general computer science classroom.

The goal of this paper is to begin conversations on providing greater context in computer science education research. By doing this, the research conducted can be better understood, studies can be more strongly replicated or distinguished, a greater understanding of attributes influencing the research can be gained, and other educators can more easily determine the relevance of the research to their classroom environment.

Session 3C: CS1 (Room: NK 10)

Rita Garcia, Katrina Falkner and Rebecca Vivian

Instructional Framework for CS1 Question Activities

Questioning is a learning activity that can promote the use of critical thinking skills, where thinking processes are required to answer a posed question. In this paper, we map instructional question types to Bloom's Taxonomy-a classification of critical thinking skills required for cognition-to form a framework for educators to construct learning activities through questioning. The preliminary instructional framework is applied to a question activity within a blended CS1 learning environment to support students in better understanding on how to solve a programming assignment. Our results show students meeting the desired Bloom's cognitive level when answering the question activity. Future research opportunities are presented to test the framework for upper-division CS courses and further explore the framework as an intervention for programming assignments.

Soohyun Nam Liao, Sander Valstar, Kevin Thai, Christine Alvarado, Daniel Zingaro, William G. Griswold and Leo Porter

Behaviors of Higher and Lower Performing Students in CS1

Although recent work in computing has discovered multiple techniques to identify low-performing students in a course, it is unclear what factors contribute to those students' difficulties. If we were able to better understand the characteristics of such students, we may be better able to help those students. This work examines the characteristics of low- and high-performing students by performing interviews with students from an introductory computing class. We identify a number of relevant areas of student behavior including how they approach their exam studies, how they approach completing programming assignments, how they reflected on assignments after submitting them, whether they sought help after identifying misunderstandings, and how and from whom they sought help. Particular behaviors within each area are coded and differences between groups of students are identified.

Quintin Cutts, Matthew Barr, Mireilla Bikanga Ada, Peter Donaldson, Steve Draper, Jack Parkinson, Jeremy Singer and Lovisa Sundin

Experience Report: Thinkathon - Countering an "I Got My Program Working" Mentality with Pencil-and-paper Exercises

Goal-directed problem-solving labs can lead a student to believe that the most important achievement in a first programming course is to get programs working. This is counter to research indicating that code comprehension is an important developmental step for novice programmers. We observed this in our own CS-0 introductory programming course, and furthermore, that students weren't making the connection between code comprehension in labs and a final examination that required solutions to pencil-and-paper comprehension and writing exercises, where sound understanding of programming concepts is essential. Realising these deficiencies late in our course, we put on three 3-hour optional revision evenings just days before the exam. Based on a mastery learning philosophy, students were expected to work through a bank of around 200 pencil-and-paper exercises. By comparison with a machine-based hackathon, we called this a Thinkathon. Students completed a pre and post questionnaire about their experience of the Thinkathon. While we find that Thinkathon attendance positively influences final grades, we believe our reflection on the overall experience is of greater value. We report that: respected methods for developing code comprehension may not be enough on their own; novices must exercise their developing skills away from machines; and there are social learning outcomes in programming courses, currently implicit, that we should make explicit.

Session 3D: Student Reasoning (Room: NK 1)

Aubrey Lawson, Eileen Kraemer, Megan Che and Cazembe Kennedy

A Multi-level Study of Undergraduate Computer Science Student Reasoning About Concurrency

In computing, concurrency relates to the situation in which different processes or threads may execute at the same time or in an interleaved manner. Many real-world computing systems utilize concurrency, but concurrency-related concepts have proven difficult for students. With an informed understanding of student reasoning, instructors will be better able to create appropriate pedagogical materials and aid student learning. The research questions we seek to address are: What is the nature of undergraduate computer science students' conceptions of concurrency? and How does student reasoning about concurrency differ across the levels of the undergraduate computing program? To study student conceptions of concurrency, we used Lewandowski et al.'s "concert tickets" problem, in which multiple vendors sell tickets to a concert and students are asked to provide a natural language response that identifies problems that may occur and to provide possible solutions. We present an analysis of sixty student responses drawn from all four years of the undergraduate CS program. The analysis utilizes a qualitative open coding approach to uncover common features in student solutions. Most students identified the key problem of double-booking and did so uniformly across student year. We performed a thematic analysis of student responses about double-booking and found emerging clusters around access control, speed, visualization, human protocols, and added states. Knowledge of these student conceptions can help instructors to construct pedagogical materials that build on the strengths and prior knowledge that students bring with them at each level and to address troublesome aspects of these concepts.

Claudia Szabo and Michael Pointon

Final Year Students' Approaches to Implementing Complex Distributed Systems

Understanding how final year students build complex software systems is critical for determining whether desired graduate outcomes have been met, for identifying curriculum gaps, and for designing scaffolding and support structures. A large body of work focuses on the programming strategies employed by novice programmers, with few existing research in understanding programming strategies and development focus of final year students, in particular with respect to non-functional requirements. In this paper, we analyse consecutive revisions of 77 students across two cohorts that implemented a large and complex Distributed Systems assignment with several non-functional requirements. To obtain a qualitative overview of the students' approach to software development, we manually read and tagged all sourcefiles in all assignment revisions with specific development focus categories. Our analysis identifies how the students' development focus evolves throughout the assignment timeline. We visualise the software development process and identify several areas that require further support.

Cazembe Kennedy and Eileen Kraemer

Qualitative Observations of Student Reasoning: Coding in the Wild

Understanding student thinking and identifying student misconceptions are important precursors to developing high quality pedagogical materials and approaches. Prior work

has used conceptual assessment surveys, in some cases paired with follow-up one-on-one task-based interviews, to pursue this knowledge. In these methods, students typically evaluate existing code and are asked to predictor explain the code's behavior. However, features of student thinking captured under these conditions of evaluating existing code intended to probe at a particular concept may differ from features of student thinking that occurs "in the wild." We present the results of a study conducted with 10 students at a large, engineering-focused US university who were about to complete or had just completed CS1. In this work, students were asked to "Please, think aloud" as they interacted in a development environment to implement a solution to a defined task. We captured and analyzed video recordings of the screen and audio recordings of their utterances to characterize their actions, current task, the relevant CS concept involved, current problem-solving phase, and expressed level of certainty. We also took note of the nature of their uncertainties and if and how these uncertainties were resolved. We report preliminary findings showing that students showed uncertainty regardless of success at task completion, but successful students managed to resolve these uncertainties through live experimentation to identify and fix flaws in their thinking and coding.

Morning Sessions

08:00-09:00 **Elphinstone Hall**	**Lean Coffee**

Sessions	9:00	9:30	10:00
4A: Mendix Presentation **Chair: Roger McDermott** **Room: Regent**	**Teaching Software Development using high-productivity tools like Mendix**		Networking time
4B: Panel 2 **Chair: Amber Settle** **Room: NK 1**	Monica McGill, Miles Berry, Leigh Ann DeLyser and Briana Morrison **Current Resources for Researching and Teaching Computing Education in Primary and Secondary Schools: What Exists and What is Still Needed**		
4C: Tips, Techniques and Courseware 1 **Chair: Jane Sinclair** **Room: NK 10**	Michael Black, Joseph Matta **Teaching AVR Assembly by Building an Arduino Arcade Machine** Ali Erkan, Adam Lee **Connecting Majors/Non-Majors Courses Via Tabular Data** Stephan Euler **Automatic Evaluation of the Quality of Solutions for an Open Programming Task** Amir Rubinstein, Noam Parzanchevski and Yossi Tamarov **In-Depth Feedback on Programming Assignments Using Pattern Recognition and Real-Time Hints**		Networking time

Session 4A: Mendix Presentation (Room: Regent)

Accelerate student learning of both the fundamentals of Programming and advanced concepts like Security and APIs using high-productivity tools like Mendix, the industry leading low-code application development platform. Prepare students with industry-standard skills sets such as understanding the agile development process and being able to use a full stack platform to build a solution from back end to front end. Mendix offers completely free use of their low-code app development technology, certifications and curriculum for Professors and students at 100+ universities worldwide.

Session 4B: Panel 2 (Room: NK 1)

Monica McGill, Miles Berry, Leigh Ann DeLyser and Briana Morrison

Current Resources for Researching and Teaching Computing Education in Primary and Secondary Schools: What Exists and What is Still Needed

In this panel presentation, we will present an overview and discussion of several recent and complementary resources geared towards primary and secondary computing education. The moderator will present a brief introduction to the purpose of this panel and provide the format for the session. Each panelist will have 5 minutes to present the purpose of their resource followed by a demonstration of its content. Each panelist will share with the audience the resource, its content, its intended target demographic, and how it can be and has been used. Following the initial presentations, several minutes will be allotted for immediate questions and answers pertaining to the resource just presented.

Immediately after the presentation of all of the resources, the moderator will ask the panelists questions about future plans for each site in an effort to address resources still needed for the primary and secondary computing education community (including researchers). We will also engage with the audience to discuss these resources and discuss what types of resources are still needed to improve the quality of primary and secondary education. Attendees will be encouraged to ask questions and provide suggestions for improving and enhancing the resources if they are currently using one or more of them.

Session 4C: Tips, Techniques and Courseware 1 (Room: NK 10)

Michael Black, Joseph Matta

Teaching AVR Assembly by Building an Arduino Arcade Machine

This paper describes a series of homework assignments, labs, and a project developed to teach AVR assembly language as part of a sophomore-level Computer Organization course. At the culmination of this module, students built a handheld arcade machine out of an Arduino Uno and programmed it, in assembly language, to play the classic game of Breakout on an LED matrix display. While completing this project students learned most of the fundamentals of low-level programming, including registers, bit operators, control flow, stack handling, subroutine calls, hardware interrupts, delay loops and instruction timing, and communicating with I/O. This project was assigned to three classes of students in the Fall 2018 and Spring 2019.

Ali Erkan, Adam Lee

Connecting Majors/Non-Majors Courses Via Tabular Data

As is the case with many Computer Science departments, we offer one set of courses for our majors and another (smaller) set for non-majors. Historically, the divide between these sets has been fairly visible, an academic reality that may be typical for many institutions. Initially triggered by a desire to improve and advance our most heavily enrolled non-majors course, we have been working for the past several years to dovetail courses across this "great divide" by focusing on tabular data. This report is an outline of our progress (in general) and this one course that experienced most change (in particular); we believe the

direction we are headed and the gains we hope to see might be valuable for other institutions as well.

Stephan Euler

Automatic Evaluation of the Quality of Solutions for an Open Programming Task
This paper describes a tool for analysis of solutions to programming tasks. The analysis addresses both the code and the generated graphical output. First results on the correlation between quality of code and complexity of the output are reported.

Monica McGill, Adrienne Decker

csedresearch.org: Resources for Primary and Secondary Computer Science Education Research
In this Tips, Techniques, and Courseware talk, we will provide an overview of csedresearch.org, a site designed to provide resources to those in engaged in primary and secondary computer science (CS) education research. This site features a repository of manually-curated data on over 500 articles focusing on primary and secondary CS education research across ten targeted venues, 2012-2018, and over 150 evaluation instruments, many of which are focused on computing. Users can search for articles by filtering various curriculum and activity components, including student demographics, instructor demographics, and program components. Users can also search for evaluation instruments on criteria such as type, year published, constructs measured in the instrument, and whether the instrument has been validated or tested for reliability. We will provide a brief summary of the rationale for the site, the current state of the site, and how educators and researchers are using the site to date. We will also discuss future plans for the site and enable the audience to provide feedback that may help us shape the site to better meet the needs of the community.

10:30-11:00 **Elphinstone Hall**	**Coffee/Posters** Doctoral Consortium poster presentations: see pp. 9-10

Sessions	11:00	11:30	12:00
5A: Working Groups Presentations 1 **Chair: Guido Rößling** **Room: Regent**	**WG 1: 1.5 Degrees of Separation: Computer Science Education in the Age of the Anthropocene** **WG 2: Program Comprehension: Identifying Learning Trajectories for Novice Programmers** **WG 3: Pass Rates in STEM Disciplines Including Computing** **WG 4: Data Science Education: Global Perspectives and Convergence** **WG 5: A Periodic Table of Computing Education Learning Theories** **WG 6: An International Benchmark Study of K-12 Computer Science Education in Schools**		
5B: Professional Development **Chair: Becky Grasser** **Room: NK 1**	Alexandra Milliken, Christa Cody, Veronica Catete and Tiffany Barnes **Effective Computer Science Teacher Professional Development: Beauty and Joy of Computing 2018**	Yihuan Dong, Veronica Catete, Nicholas Lytle, Amy Isvik, Tiffany Barnes, Jennifer Albert, Robin Jocius, Richard Robinson, Deepti Joshi and Ashley Andrews **Infusing Computing: Analyzing Teacher Programming Products in K12 Computational Thinking Professional Development**	Lori Postner, Heidi Ellis and Gregory Hislop **Impact of HFOSS Education on Instructors**

5C: Tips, Techniques and Courseware 2 **Chair: Angela Siegel** **Room: NK 10**	Monica McGill and Ian Pollock **If Memory Serves: A Game to Supplement the Instruction of Concepts Related to Pointers** Monica McGill, Adrienne Decker **csedresearch.org: Resources for Primary and Secondary Computer Science Education Research** Samah Senbel **Teaching Self-Balancing Trees Using a Beauty Contest** Shaveen Singh **Exploring the Potential of Social Annotations for Predictive and Descriptive Analytics** Mark Zarb and Michael Scott **Laughter over Dread: Early Collaborative Problem Solving through an Extended Induction using Robots**

Session 5A: Working Groups Presentations 1 (Room: Regent)

WG 1: 1.5 Degrees of Separation: Computer Science Education in the Age of the Anthropocene
WG 2: Program Comprehension: Identifying Learning Trajectories for Novice Programmers
WG 3: Pass Rates in STEM Disciplines Including Computing
WG 4: Data Science Education: Global Perspectives and Convergence
WG 5: A Periodic Table of Computing Education Learning Theories
WG 6: An International Benchmark Study of K-12 Computer Science Education in Schools

Session 5B: Professional Development (Room: NK 1)

Alexandra Milliken, Christa Cody, Veronica Catete and Tiffany Barnes

Effective Computer Science Teacher Professional Development: Beauty and Joy of Computing 2018

The Advanced Placement Computer Science Principles (AP CSP) course has now been active for 2 years, garnering a large group of diverse students, thus confirming the need for highly trained CSP teachers, especially in effective practices for diversity and equity. We have conducted summer professional development (PD) workshops from 2012-2018 which have prepared 748 teachers to teach the AP CSP course using our CSP curriculum. In Summer 2018 alone, we reached 135 teachers. To improve equity and readiness for teaching, we have refined our PD by: shortening the PD from 6 weeks to 1 week; developing new, highly scaffolded pre-PD work; and modifying the in-person schedule to incorporate more pedagogy and teaching experiences, while continuing to provide in-depth, hands-on support for teachers to learn the basics of programming. The most recent revisions to our PD schedule resulted in improved post-PD survey results, with 2018 teachers planning to adopt more of the CSP curriculum than in past years. From 2017 to 2018, planned adoption rates increased by 13%, resulting in 73% of the 2018 PD participants planning to adopt more than 60% of the CSP curriculum and 58% planning to adopt 80-100% of the CSP curriculum in their classrooms in 2018-2019. In

this paper, we discuss the most recent CSP PD schedule and provide evidence of increased teacher self-efficacy in areas including fostering interest in computing for underrepresented populations.

Yihuan Dong, Veronica Catete, Nicholas Lytle, Amy Isvik, Tiffany Barnes, Jennifer Albert, Robin Jocius, Richard Robinson, Deepti Joshi and Ashley Andrews

Infusing Computing: Analyzing Teacher Programming Products in K12 Computational Thinking Professional Development

In summer 2018, we conducted two week-long professional development workshops for 116 middle and high school teachers interested in infusing computational thinking (CT) into their classrooms. Teachers learned to program in Snap!, connect CT to their disciplines, and create infused CT learning segments for their classes. This paper investigates the extent to which teachers were able to successfully infuse CT skills of pattern recognition, abstraction, decomposition, and algorithms into their learning products.

In this work, we analyzed 58 teacher-designed programming products to look for common characteristics, such as project type, intended coding requirements for their students, and code features/functionality. Teacher-created products were classified into five types: animation, interactive story, quiz, intended game, and simulation/exploration tools. Coding requirements varied from using and/or explaining provided code, modifying existing code, programming with starter code, to building entire programs. Products were classified according to the extent to which they involved sprite manipulation, questions/answers, event handling, drawing, and control blocks. We found that teachers from different disciplines created products that vary in type, coding requirements, and features to suit their specific needs. Moreover, we found relationships between discipline, project type, and the required coding teachers expected students to do.

Our results inform future Infusing Computing Professional Development (PD) to provide more targeted training to support different teacher needs.

Lori Postner, Heidi Ellis and Gregory Hislop

Impact of HFOSS Education on Instructors

Student involvement in Humanitarian Free and Open Source Software (HFOSS) increases student self-reported learning while providing the motivation of "doing good". Supporting this type of student participation also appears to have an impact on instructors that may include modifying teaching approach and changing how instructors perceive their role. So far, little investigation has been done into instructor experiences of supporting student involvement in HFOSS. This paper explores the impact on instructors resulting from supporting student participation in HFOSS, including consideration of pedagogical practices, self-perceptions of teaching, and exposure to open source culture and technologies. The analysis presented here is based upon a qualitative analysis of semi-structured interviews with 24 faculty members. Results indicate that exploring HFOSS education changes instructors' pedagogical approaches, provides significant technical learning opportunities for instructors as well as students, and has impact on instructors' attitudes and motivations in teaching.

Session 5C: Tips, Techniques and Courseware 2 (Room: NK 10)

Monica McGill and Ian Pollock

If Memory Serves: A Game to Supplement the Instruction of Concepts Related to Pointers

In this Tips, Techniques, and Courseware presentation, we will provide an overview of If Memory Serves, a game designed to complement the instruction of pointers in computer science courses. The game was first designed during a working group in ITiCSE 2017 as a tool to enable deeper understanding of pointers due to the difficulties students often face when they are learning about pointers. Over the last two years, the game has continued to evolve and has moved from a concept of the game to near-alpha. During this talk, we will demonstrate the game, provide insight into why it was developed, and discuss how it can be used to teach pointers. We will also briefly discuss our concept behind this open-source game and invite feedback from the audience as well as invite others to remix, test, and engage in research on testing the game's effectiveness.

Amir Rubinstein, Noam Parzanchevski and Yossi Tamarov

In-Depth Feedback on Programming Assignments Using Pattern Recognition and Real-Time Hints

Automatic grading of students' programs is a challenge in computer science (CS) education. In the last two years we have been using a machine learning tool called 'Sense Education' in our 'Intro to CS' course. The tool provides real-time pedagogical hints while students work on solutions, and in-depth feedback on submissions. It also provides a "bird's eye" view of the class's current capabilities, misconceptions, and biases in approaches to solutions. We describe this effort and several test cases. Our goal is to provide teachers with tips on how to effectively use such tools.

Samah Senbel

Teaching Self-Balancing Trees Using a Beauty Contest

Trees data structures and their performance is one of the main topics to teach in a data structures course. Appreciating the importance of tree structure and tree height in software performance is an important concept to teach. In this paper, a simple and amusing activity is presented. It demonstrates to students the importance of a well-balanced tree by comparing the height of a binary search tree to a balanced (AVL) tree build upon some personal data to find the "prettiest" tree (minimum height). The activity highlights the fact that, irrelevant of your data sequence, a balanced tree guarantees a height of O(log n) and everyone "wins" the beauty contest.

Shaveen Singh

Exploring the Potential of Social Annotations for Predictive and Descriptive Analytics

Academic performance is typically measured through assessments on standardised tests. However, considerably less is known about how students learn from reading materials. This paper presents a preliminary analysis of data gathered during a blended course offering, with respect to students self-reports on learning material as possible predictors of their academic outcomes. The results point to the predictive potential of such self-reports,

to the importance of students exercising their self-understanding during learning, and to the potentially critical role of incorporating such students self-reports in learner modelling and driving teaching interventions.

Mark Zarb and Michael Scott

Laughter over Dread: Early Collaborative Problem Solving through an Extended Induction using Robots

Employers in the software development industry want new hires to possess strong interpersonal and problem solving skills. To ensure the development of such skills, they must be embedded throughout the curriculum. However, many students struggle to engage with their peers and in self-regulated practice during the early stages of their course. Explicit scaffolding can, however, motivate such engagement. This tips and techniques session illustrates how an ice-breaker using LEGO EV3 robots at two UK institutions enhanced peer interaction and engaged students in greater self-regulated practice over the first four weeks of 2016-17 and 2017-18.

Afternoon Events

12:30-13:00 **Elphinstone Hall**	**Collect packed lunches**
13:00-18:00	**Excursions**
19:00-23:00	**Conference Banquet**

Morning Sessions

08:00-09:00 **Elphinstone Hall**	**Lean Coffee**
9:00-10:00 Chair: Arnold Pears **Room: ALT**	**Keynote** Marian Petre **Lessons from Experts: Software Design Dialogues**
10:00-10:30 **Elphinstone Hall**	**Coffee/Posters** Patricia Morreale, Nohelia Diplan, Dustin York **A Gamification Pathway for Computer Science Student Success** Lillian Cassel, Darina Dicheva, Christo Dichev, Keith Irwin **Student Motivation and Engagement in STEM Courses: The Potential Impact of Gamification** Debzani Deb, Russell Smith, Muztaba Fuad **Infusing Data Science Across Disciplines** Huda Alrashidi, Mike Joy, Thomas Ullmann **A Reflective Writing Framework in Computing Education** Svetlana Peltsverger, Sourav Debnath **Instructional Pseudocode Guide to Teach Problem-Solving** S. Zahra Atiq, Michael Loui **A Qualitative Investigation of First-year Engineering Students while Learning Loops** Sheikh Ghafoor, Mike Rogers, David Brown, Thomas Hines **Unplugged Activities to Introduce Parallel Computing in Introductory Programming Classes: An Experience Report** Nina Bresnihan, Glenn Strong, Lorraine Fisher,Richard Millwood, Áine Lynch **OurKidsCode: a national programme to get families involved in CS education** Greg Gagne **We Need to Talk!! - A Chatroom Application using a Student-Designed Protocol** Joseph Maguire, Quintin Cutts, Jack Parkinson, Matthew Barr, Derek Somerville **Devising Work-based Learning Curricula with Apprentice Research Software Engineers**

Sessions	**10:30**	**11:00**	**11:30**
6A: Panel 3 **Chair: Janet Hughes** **Room: Regent**	Francesco Maiorana, Gretchen Richards, Chery Lucarelli, Miles Berry and Barbara Ericson **Interdisciplinary Computer Science Pre-service Teacher Preparation**		

6B: Working Groups Presentation 2 and Doctoral Consortium Presentations Chair: Michalis Giannakos and Mark Zarb **Room: NK 6**	**WG 7: Toward Developing a Cloud Computing Model Curriculum** **WG 8: Securing the Human: Broadening Diversity in Cybersecurity** **WG 9: Towards an Ability to Direct College Students to an Appropriately Paced Introductory Computer Science Course** **WG 10: Unexpected Tokens: A Review of Programming Error Messages and Design Guidelines for the Future** **Doctoral Consortium**		
6C: Programming Michael Kölling **Room: NK 10**	Michel Adam, Moncef Daoud and Patrice Frison **Direct Manipulation versus Text-based Programming**	Rebecca Smith, Terry Tang, Joe Warren and Scott Rixner **Auto-Generating Visual Exercises for Learning Program Semantics**	Matthias Kramer, Mike Barkmin and Torsten Brinda **Identifying Predictors for Code Highlighting Skills**
6D: Experience Reports Chair: James Paterson **Room: NK 1**	Bjørn Fjukstad, Nina Angelvik, Morten Grønnesby, Maria Wulff Hauglann, Hedinn Gunhildrud, Fredrik Høisæther Rasch, Julianne Iversen, Margaret Dalseng and Lars Ailo Bongo **Teaching Electronics and Programming in Norwegian Schools Using the air:bit Sensor Kit**	Saturnino Garcia and Caitlin Fanning **Below C Level: A Student-Centered x86-64 Simulator**	Kiev Gama **Developing course projects in a Hack day: An experience report**

Session 6A: Panel 3 (Room: Regent)

Francesco Maiorana, Gretchen Richards, Chery Lucarelli, Miles Berry and Barbara Ericson

Interdisciplinary Computer Science Pre-service Teacher Preparation

Many researchers recognize that great Computer Science (CS) teaching demands great pedagogy, technology, and subject knowledge. For subject knowledge and pedagogy, a global movement is promoting the study of CS in the lifelong learning experience with a focus on teacher preparation. It is believed that without this focus the gap in CS knowledge will be difficult to close and the support to curricular indications as the national

curriculum for computing in England, Computer Science Principles, CS 10K, CS for All, CSTA, Scientix will not sustainable. World-spread projects as Computing At School, Mobile Computer Science Principles or Initial Teacher Education Lab emphasize the role of teachers, professors, and educators in reaching students, sustaining their daily practice and research effort. Teachers are considered the critical connection between schools, universities, and work experiences. International initiatives as code.org, CoderDojo, Code Club are great examples of well propagated curricular, pedagogical and technological innovations in education. In relation to technology the research effort has produced many block-based programming languages, tools, and technologies "Low floor and high ceiling". These tools support the education community in climbing the abstraction pyramid of acquiring abilities and competencies in many domains such as mobile, parallel and distributed programming, data structures, media computation, data manipulation, supporting all programming paradigm with applications from the technological sectors (IoT, 3DPrinting) to humanities (Latin). Whit this deluge of content, pedagogical and technological innovations there is the necessity to involve and sustain teachers in getting confidence with CS. Ways to accomplish this will be discussed.

Session 6B: Working Groups Presentation 2 and Doctoral Consortium Presentations (Room: NK 6)

WG 7: Toward Developing a Cloud Computing Model Curriculum
WG 8: Securing the Human: Broadening Diversity in Cybersecurity
WG 9: Towards an Ability to Direct College Students to an Appropriately Paced Introductory Computer Science Course
WG 10: Unexpected Tokens: A Review of Programming Error Messages and Design Guidelines for the Future
Doctoral Consortium

Session 6C: Programming (Room: NK 10)

Michel Adam, Moncef Daoud and Patrice Frison

Direct Manipulation versus Text-based Programming

The standard approach to programming is to learn a programming language, write a program in a development environment, compile it, and run it to check how it works. Another approach is possible using direct manipulation programming. This approach makes it possible to directly manipulate the programming objects (variables, arrays, indices) to implement a given algorithm and to automatically produce the corresponding program. In this paper, we report on the results of an experiment that we performed to compare the standard approach with direct manipulation programming. The experiment was conducted with an audience of 54 beginner students divided into 2 groups. The first group programmed with Python Tutor and the second with AlgoTouch, a direct manipulation programming tool. In this article, we present the experience and detail the result obtained.

Rebecca Smith, Terry Tang, Joe Warren and Scott Rixner

Auto-Generating Visual Exercises for Learning Program Semantics

Understanding program execution is a challenging task for novice programmers. The semantic rules which determine how execution affects the program state are numerous and complex, and students frequently hold fundamental misconceptions about these rules. If students do not build a correct mental model of program execution early on, they will face substantial hurdles as they try to develop and debug their code. This paper presents VizQuiz, a tool for auto-generating multiple choice quizzes designed to help students gain insight into the semantic rules which govern program execution. VizQuiz provides students with an initial state and a piece of code, and tasks them with mentally tracing the execution of that code and selecting the correct final state. Reference diagrams are used to depict the initial and final states, and as feedback to help students visualize the correct behavior if they select the wrong answer. This feedback is auto-generated, so students can immediately correct their misconceptions and re-attempt. VizQuiz was piloted in an introductory class on computational thinking, where it was evaluated alongside a more traditional quiz model. In this study, students that interacted with VizQuiz reported significantly greater improvements in their computational thinking abilities ($p=0.023$) and critical thinking skills ($p=0.031$) as compared to the control group.

Matthias Kramer, Mike Barkmin and Torsten Brinda

Identifying Predictors for Code Highlighting Skills

Programming comprises a multitude of involved skills and abilities. To assess these, an even larger multitude of tasks exists, ranging from very complex to basal levels. Recent demands call for the assessment of these basal skills, with the aim to figure out possible problems of learners more precisely. In order to read and interpret source code sufficiently, students must be able to recognize given concepts in a source code, e.g. to identify the objects and methods in an object-oriented program. Whether this skill relies solely on conceptual knowledge and the abilities in handling syntax is still unclear. This study gives evidence that isolated skills in single areas are outperformed by the interaction of both areas. We developed a test consisting of conceptual questions, fill-in-the-Java-keyword items and highlighting items. We investigated whether the knowledge about object orientation and the skills in handling Java syntax sufficiently predict the ability to recognize concepts in Java source code via a multiple linear regression. We found that the model with interaction term explains the data better compared to the null model, the isolated predictors or combined predictors without interaction term. This leads to the conclusion that even though programming consists of skills in certainly different areas, it is more important to interconnect these areas with each other than to teach them in an isolated manner.

Session 6D: Experience Reports (Room: NK 1)

Bjørn Fjukstad, Nina Angelvik, Morten Grønnesby, Maria Wulff Hauglann, Hedinn Gunhildrud, Fredrik Høisæther Rasch, Julianne Iversen, Margaret Dalseng and Lars Ailo Bongo

Teaching Electronics and Programming in Norwegian Schools Using the air:bit Sensor Kit

We describe lessons learned from using the air:bit project to introduce more than 150 students in the Norwegian upper secondary school to computer programming, engineering and environmental sciences. In the air:bit project, students build and code a portable air quality sensor kits, and use their air:bit to collect data to investigate patterns in air quality in their local environment. When the project ended students had collected more than 400,000 measurements with their air:bit kits, and could describe local patterns in air quality. Students participate in all parts of the project, from soldering components and programming the sensors, to analyzing the air quality measurements. We conducted a survey after the project and describe our lessons learned from the project. The results show that the project successfully taught the students fundamental concepts in computer programming, electronics, and the scientific method. In addition, all the participating teachers reported that their students had showed good learning outcomes.

Saturnino Garcia and Caitlin Fanning

Below C Level: A Student-Centered x86-64 Simulator

Learning an assembly language introduces students to important computing concepts such as the program stack and lays the conceptual groundwork for topics such as caching. Assembly languages are therefore a critical part of computing curricula, often showing up in introductory computer organization or systems courses. Unfortunately, students often struggle with this topic as it requires them to learn new, lower-level computing abstractions and because assembly languages lack the structure of higher-level languages.

While there are many assembly languages, x86-64 still dominates the desktop and server environments and is therefore widely used in introductory computer systems courses. Unfortunately, x86-64 is a complex language and as a result students often have difficulty understanding and visualizing the execution of x86-64 programs. Control flow, the difference between registers and memory, and memory organization (e.g. the stack) are all concepts that students struggle to understand.

While students can use debuggers to help them step through the execution of a program and examine the program state, these tools are challenging to learn and are therefore not an ideal fit for introductory level students. This paper introduces Below C Level (BCL), an x86-64 simulator aimed at helping novices overcome the barriers to learning this challenging language. BCL offers an intuitive interface and many features that allow students to easily simulate x86-64 code snippets or programs. During simulation BCL helps students decipher individual instructions and visualizes the program stack and register file, allowing students to quickly trace through their program and gain a deeper understanding of x86-64 code.

Kiev Gama

Developing course projects in a Hack day: An experience report

Some researchers are starting to point out the importance of hackathons as an alternative venue for college students practicing and learning. Many positive aspects were highlighted such as peer learning and improvements in problem-solving, project management and task priority analysis. However, this hackathon phenomena is something that is happening outside the curriculum, with scarce reports of teachers taking advantage of similar approaches in undergraduate courses. In this article we present our experience of an undergraduate introductory course on Distributed Systems where students have to participate in a "hack day" in the end of the semester. The goal is to take advantage of the hackathon model to bring a more realistic and engaging scenario where students can develop their course project and put into practice what they have learned in the semester. With this model, they have to do face-to-face group work and also have the opportunity to share knowledge with other groups. Our experience indicates positive results with this approach. Students reported a high level of engagement and enthusiasm. There were different situations of peer learning being reported and according to their perception, students felt they learned more in the hack day than in lectures or laboratory practice.

Afternoon Sessions

12:00-13:00 **Elphinstone Hall**	**Lunch**

Sessions	13:00	13:30	14:00
7A: Programming and Computational Thinking Issues **Chair: Quintin Cutts** **Room: Regent**	Nicholas Lytle, Veronica Catete, Danielle Boulden, Yihuan Dong, Jennifer Houchins, Alexandra Milliken, Amy Isvik, Dolly Bounajim, Eric Wiebe and Tiffany Barnes **Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes**	Marcos Javier Gomez, Marco Moresi and Luciana Benotti **Text-based Programming in Elementary School: A Comparative Study of Programming Abilities in Children with and without Block-based Experience**	Michael Lee **Exploring Differences in Minority Students' Attitudes towards Computing after a One-Day Coding Workshop**
7B: CS for All **Chair: Michal Armoni** **Room: NK 6**	Peter J Rich, Garrett Egan and Jordan Ellsworth **A Framework for Decomposition in Computational Thinking**	Alexander Repenning, Anna Lamprou, Serge Petralito and Ashok Basawapatna **Making Computer Science Education Mandatory: Exploring a Demographic Shift in Switzerland**	Yorah Bosse, David Redmiles and Marco Gerosa **Pedagogical Content for Professors of Introductory Programming Courses**
7C: Programming **Chair: Claudia Szabo** **Room: NK 10**	Claudio Mirolo and Cruz Izu **An Exploration of Novice Programmers' Comprehension of Conditionals in Imperative and Functional Programming**	Rodrigo Duran, Jan-Mikael Rybicki, Sanna Suoranta and Arto Hellas **Towards a Common Instrument for Measuring Prior Programming Knowledge**	Carol Zander, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Kate Sanders and Lynda Thomas **Copying Can Be Good: How Instructors Use Imitation in Teaching Programming**

7D: Software Architecture and Algorithms Chair: Stan Kurkovsky **Room: NK 1**	Eng Lieh Ouh and Yunghans Irawan **Applying Case-Based Learning for a Postgraduate Software Architecture Course**	J. Ángel Velázquez-Iturbide **Students' Misconceptions of Optimization Algorithms**	Dinesh Mehta and Jack Rosenthal **AlgoBOWL: A Competition-Based Group Project for Algorithms Courses**

Session 7A: Programming and Computational Thinking Issues (Room: Regent)

Nicholas Lytle, Veronica Catete, Danielle Boulden, Yihuan Dong, Jennifer Houchins, Alexandra Milliken, Amy Isvik, Dolly Bounajim, Eric Wiebe and Tiffany Barnes

Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes

Computational Thinking (CT) is being infused into curricula in a variety of core K-12 STEM courses. As these topics are being introduced to students without prior programming experience and are potentially taught by instructors unfamiliar with programming and CT, appropriate lesson design might help support both students and teachers. "Use-Modify-Create" (UMC), a CT lesson progression, has students ease into CT topics by first "Using" a given artifact, "Modifying" an existing one, and then eventually "Creating" new ones. While studies have presented lessons adopting and adapting this progression and advocating for its use, few have focused on evaluating UMC's pedagogical effectiveness and claims. We present a comparison study between two CT lesson progressions for middle school science classes. Students participated in a 4-day activity focused on developing an agent-based simulation in a block-based programming environment. While some classrooms had students develop code on days 2-4, others used a scaffolded lesson plan modeled after the UMC framework. Through analyzing student's end of activity reflections, classroom observations, and teacher interviews, we illustrate differences in perception of assignment difficulty from both the students and teachers, as well as student perception of artifact "ownership" between conditions.

Marcos Javier Gomez, Marco Moresi and Luciana Benotti

Text-based Programming in Elementary School: A Comparative Study of Programming Abilities in Children with and without Block-based Experience

This paper describes an elementary school intervention to teach a text-based programming language to 10-11 year old students. We compare students with no previous programming experience with students with 3 semesters of experience with a block-based programming language. We analyze students' performance and learning based on detailed logs in an online programming platform and on multiple choice tests. Although both groups have a similar percentage of syntactical errors, the experienced group showed a better performance on exam scores and a lower number of test case errors. These findings suggest that, 10-11 year old students benefit from block-based experience when learning a new text-based programming language.

Michael Lee

Exploring Differences in Minority Students' Attitudes towards Computing after a One-Day Coding Workshop

As programming continues to be an essential 21st century skill, it is critical to focus on diversity and increasing participation of underrepresented groups in computing. To address this need, we must better understand minorities' views and attitudes towards programming, especially in their youth, as literature shows that children form ideas about their interests and careers in middle school or earlier. To explore this, we provided middle school students in the U.S. with a full day (7 hours) of programming activities to learn about their initial attitudes towards computing and how a short intervention might change these attitudes. We ran two separate one-day events, serving a total of 34 minority students (21 males and 13 females; grades 6 and 7) from a low-income, urban area. We found that students' initial attitudes towards computing were high, and that one day of learning programming increased their reported attitudes in computing even more. We also found differences in attitudes by gender and ethnicity. These findings highlight the positive attitudes minority students have towards computing, and the importance of providing resources and support to help maintain their interests in computing while recognizing demographic differences.

Session 7B: CS for All (Room: NK 6)

Peter J Rich, Garrett Egan and Jordan Ellsworth

A Framework for Decomposition in Computational Thinking

Computational Thinking has become an important cognitive skill for computing education. Despite its increasing popularity, the construct itself is only partially understood. There are few measures currently in place that advance our understanding of computational thinking and its sub-constructs. In this study, we analyze existing measures of computational thinking (CT), looking specifically at measures of decomposition. Decomposition is defined as the process of breaking down a problem into its sub-components. Even though most definitions of computational thinking include decomposition, few break down the decompositional process beyond a basic definition. As one of the first steps in the computational thinking process, it is important to better understand the various manners in which decomposition occurs, which methods are most effective, and under what conditions. To better understand the decompositional process, we analyze evidence of decompositional process in a variety of disciplines. We then present a framework for decomposition in computational thinking. We demonstrate how this framework may help computer science educators to better prepare students to break down complex problems, as well as provide guidance for how decompositional ability might be measured.

Alexander Repenning, Anna Lamprou, Serge Petralito and Ashok Basawapatna

Making Computer Science Education Mandatory: Exploring a Demographic Shift in Switzerland

A promising approach to make K-12 Computer Science education more systemic could arise from a strategy focusing mostly on pre-service teachers educated through

mandatory courses instead of self-selected in-service teachers. When employing mandatory courses, schools of education can reach all future teachers, but what are potential consequences resulting from this demographic shift? Pre-service teachers may not expect to acquire programming skills and may not be convinced of the relevance of Computer Science. In 2017, one of the first mandatory Computer Science education courses for pre-service K-12 teachers was introduced at the School of Education of northwestern Switzerland (PH FHNW). The mandatory nature of the course was possible because of the introduction of Computer Science as a subject in a new national curriculum. The course, based on Scalable Game Design, was taken by over 600, mostly female (75%), pre-service elementary school teachers. This paper explores the characteristics of this new audience and investigates the consequences of mandatory pre-service teacher Computer Science education. While our research shows that the course was successful, with regards to improving the students' skills and self-efficacy, it reveals significant gender effects concerning attitudes towards Computer Science and video games.

Yorah Bosse, David Redmiles and Marco Gerosa

Pedagogical Content for Professors of Introductory Programming Courses

Teaching introductory programming requires knowledge on both content and pedagogy. Pedagogy, in particular, includes knowing the typical difficulties faced by students as they try to learn about a set of topics as well as knowing the potential strategies that help students overcome these difficulties. Our research aims to offer material to improve the pedagogical knowledge instructors have to teach introductory programming courses, especially those new in this area. This material presents difficulties that students face in order to learn the basic contents necessary to program, as well as a set of didactic strategies to help these professors in the transmission of knowledge as well as mitigating the difficulties. With the above motivation, we conducted 16 semi-structured interviews with instructors who teach introductory programming courses and we also collected diaries filled by 110 students during their studies. The qualitative analysis of this data revealed a set of difficulties faced by students and a set of didactic strategies. The results were reviewed by senior instructors in order to confirm the results achieved and also by junior instructors to verify the importance of this material in their view. The main contribution of our paper is a set of difficulties faced by students in the learning of programming, the classification of the most harmful for the learning of programming, and the didactic strategies usually used to teach and avoid these difficulties. Thus, we provide the basis for the pedagogical content necessary to junior and senior professors to plan their classes of introductory programming courses.

Session 7C: Programming (Room: NK 10)

Claudio Mirolo and Cruz Izu

An Exploration of Novice Programmers' Comprehension of Conditionals in Imperative and Functional Programming

Students of introductory programming courses are expected to develop higher-order thinking skills to inspect, understand and modify code. However, although novices can correctly write small programs, they appear to lack a more abstract, comprehensive grasp

of basic constructs, such as conceiving the overall effect of alternative conditional flows. This work takes a little-explored perspective on the comprehension of tiny programs by asking students to reason about reversing conditionals in either an imperative or a functional context. More specifically, besides deciding if the given constructs can be reversed, students had to justify their choice by writing a reversing program or by providing suitable counterexamples. The students' answers to four reversibility questions have been analysed through the lens of the SOLO taxonomy. 45% of students correctly identified the reversibility for the four code items; furthermore, more than 50% of each cohort were able to provide correct justifications for at least three of their four answers. Most incorrect answers were due to failures to consider border cases or to edit the conditional expressions appropriately to reverse the construct. Differences in comprehension between functional and imperative languages are explored indicating the explicit else paths of the functional examples facilitate comprehension compared with the implicit else (no action) of its imperative counterpart.

Rodrigo Duran, Jan-Mikael Rybicki, Sanna Suoranta and Arto Hellas

Towards a Common Instrument for Measuring Prior Programming Knowledge

Computing education researchers and educators use a wide range of approaches for measuring students' previous knowledge in programming. A sufficient understanding previous programming knowledge can offer many benefits for instructional designers. Such knowledge can help adapt the learning goals and assessment tools for groups of learners at different skills levels and backgrounds.

There seems to be no consensus on if and how previous programming knowledge should be measured. Traditional background surveys are often ad-hoc or non-standard, which do not allow comparison of results between different course contexts, levels, and learner groups. Moreover, surveys may yield inaccurate information and may not be useful due to lack of detail. In contrast, tests can provide much higher detail and accuracy than surveys about student knowledge or skills, but large-scale tests are typically very time-consuming or impractical to arrange.

To bridge the gap between ad-hoc surveys and standardized tests, we propose and evaluate a novel self-evaluation instrument measuring previous programming knowledge for introductory programming courses. This instrument investigates in higher detail typical course concepts in programming education considering the different levels of proficiency. Based on a sample of two thousand introductory programming course students, our analysis shows that the instrument is internally consistent, correlates with traditional background information metrics and identifies students of varying programming backgrounds.

Carol Zander, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Kate Sanders and Lynda Thomas

Copying Can Be Good: How Instructors Use Imitation in Teaching Programming

Students' "copying" is often considered negatively. In this paper, we explore the ways in which copying and imitation are used positively by computing instructors in their teaching. We found that instructors expect their students to use these strategies in many different contexts and at many different levels.

Session 7D: Software Architecture and Algorithms (Room: NK 1)

Eng Lieh Ouh and Yunghans Irawan

Applying Case-Based Learning for a Postgraduate Software Architecture Course

Software architecture given its level of abstraction remains a difficult subject for learners to grasp and for educators to teach. On the other hand, case-based learning (CBL) is a popular teaching approach used across disciplines especially in business, medicine and law where students work in groups apply their knowledge to solve real-world case studies, or scenarios using their reasoning skills and existing theoretical knowledge. In this paper, we provide how we apply case-based learning to address the challenge in teaching a postgraduate software architecture course. Our learners are postgraduate students taking a master's program in software engineering. We first describe our design of the case-based learning for our software architecture course. We then analyze the survey ratings and learners' profile to evaluate the effectiveness of the proposed case-based design. These data are gathered from 9 class runs over a period of 8 years. Our analysis results show the effectiveness of our case-based design and significant relationships between this effectiveness to the learners' years of working experiences and the number of learners. Key contributions in this paper are our proposed case-based design for software architecture and the analysis findings.

J. Ángel Velázquez-Iturbide

Students' Misconceptions of Optimization Algorithms

Interest in identifying misconceptions of different computing topics, including algorithms, has grown in recent years. The paper addresses students' misconceptions of optimization problems and their algorithms. Optimization problems are a very important class of problems in computer science, thus their associated mis-conceptions may form a relevant subset of misconceptions on algorithms. The findings presented are based on a number of evaluations conducted in past years. In the paper, we describe the assignment statements used as primary materials, the analysis methodology, and the results of evaluating each assignment. We present 9 misunderstandings, structured into 5 general themes. As we could have expected, three themes refer to three specific algorithm design techniques, but we also identified several misconceptions which fit two more general themes, basic concepts of optimization and basic concepts of specification and algorithms. We also discuss the implications of these findings for instruction and curricula.

Dinesh Mehta and Jack Rosenthal

AlgoBOWL: A Competition-Based Group Project for Algorithms Courses

We describe a competitive group project that has been in use in the Algorithms course at our institution each semester for about seven years. The class is given an NP-hard optimization problem; each group is asked to create a heuristic algorithm for the problem. The heuristic is run on a set of inputs (with each group supplying one input) and groups are ranked based on their aggregate performance over a specified time period. This paper describes our experience with this approach, including some of its pitfalls, and our recent efforts to address these through a web application used to facilitate the competition.

14:30-15:00 **Elphinstone Hall**	**Coffee/Posters** Kiev Gama, Andrew Diniz Da Costa, Ricardo Almeida Venieris, Hendi Lemos Coelho **The Peanut Butter and Jelly Sandwich Challenge as an Approach to Improve Students Abilities in Test Case Writing** Luiz Ricardo Begosso, Luis Franco, Douglas Cunha, Luiz Carlos Begosso **The use of Gamification to support the process of teaching Scrum** Markus Geissler, Cara Tang, Cindy Tucker, Christian Servin, Melissa Stange **Adapting the IT2017 Curricula for 2-year Transfer Programs: Determining Appropriate Competencies for the First Two Years of a Baccalaureate IT Program** Tony Lowe **Findings from a multi-year study of CT in K-2 students in formal and informal settings** Ananda Gunawardena **From Data to Insights in CS1** Pedro Guillermo Feijóo-García, Christina Gardner-McCune **Improving Functional Programming Understanding through Student-Created Instructional Videos** Ashish Aggarwal, Christina Gardner-Mccune, David S. Touretzky **Evaluating the Effectiveness of Explicit Instruction in Reducing Program Reasoning Fallacies in Elementary Level Students** Mirela Gutica **Motivating High School Girls to Study Computer Science** Nouf Almujally, Mike Joy **A Framework for Improving the Sharing of Teaching Practices among Computer Science Instructors** Marissa Walther, Leo Ureel, Charles Wallace **A Prototype MATLAB Code Critiquer**

Sessions	15:00	15:30	16:00
8A: School Education Issues **Chair: Monica McGill** **Room: Regent**	Huda Gedawy, Saquib Razak and Hanan Alshikhabobakr **The Effectiveness of Creating Localized Content for Middle School Computing Curriculum**	Ofra Brandes and Michal Armoni **Using Action Research to Distill Research-Based Segments of Pedagogical Content Knowledge of K-12 Computer Science**	Leigh Ann Delyser and Lauren Wright **A Systems Change Approach to CS Education: Creating Rubrics for School System Implementation**
8B: Operating Systems and Security **Chair: Mark Zarb** **Room: NK 6**	Adalbert Gerald Soosai Raj, Eda Zhang, Saswati Mukherjee, Jim Williams, Richard Halverson and Jignesh M. Patel **Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course**	Hui Chen, Agnieszka Ciborowska and Kostadin Damevski **Using Automated Prompts for Student Reflection on Computer Security Concepts**	James Walker, Man Wang, Steven Carr, Jean Mayo and Ching-Kuang Shene **Teaching Integer Security Using Simple Visualizations**
8C: Learning Environments **Chair: Åsa Cajander** **Room: NK 10**	Samiha Marwan, Nicholas Lytle, Joseph Jay Williams and Thomas Price **The Impact of Adding Textual Explanations to Next-step Hints in a Novice Programming Environment**	Shaveen Singh and Bernd Meyer **Using Social Annotations to Augment the Learning Space and Learner Experience**	Simon Larsén and Richard Glassey **RepoBee: Developing Tool Support for Courses using Git/Github**

8D: Evaluation and Analytics **Chair: Simon** **Room: NK 1**	Susana Masapanta-Carrión and J. Ángel Velázquez-Iturbide **Evaluating Instructors' Classification of Programming Exercises Using the Revised Bloom's Taxonomy**	Lauren Margulieux, Briana Morrison and Adrienne Decker **Design and Pilot Testing of Subgoal Labeled Worked Examples for Five Core Concepts in CS1**	Lawton Nichols, Kyle Dewey, Mehmet Emre, Sitao Chen and Ben Hardekopf **Syntax-based Improvements to Plagiarism Detectors and their Evaluations**

16:30-17:00 **Room: ALT**	**Closing Session**

Session 8A: School Education Issues (Room: Regent)

Huda Gedawy, Saquib Razak and Hanan Alshikhabobakr

The Effectiveness of Creating Localized Content for Middle School Computing Curriculum

The need for computing education in K-12 is growing all over the world. Several countries have created national curriculum standards to meet this growing need. In most cases, the actual implementation of the curriculum, including the technical content, is left to individual schools or teachers. In this paper, we outline our work in implementing a computing curriculum based on Alice software in the State of Qatar. We present our experience in creating the content and evaluating its effectiveness. The curriculum was developed in cooperation with schoolteachers, education professionals, and the Qatari Ministry of Education. Materials developed included lesson plans, academic calendar, assessment tools, and student textbooks. We address the issues encountered in creating this computing curriculum content.

Ofra Brandes and Michal Armoni

Using Action Research to Distill Research-Based Segments of Pedagogical Content Knowledge of K-12 Computer Science

Teachers' pedagogical content knowledge (PCK) is an important factor concerning teaching and learning processes. As such, it plays an important role in training pre-service teachers and in the professional development of in-service teachers. In line with this, the pedagogical content knowledge of K-12 computer science teachers (CS-PCK) has received considerable attention in current computing education research. However, very little is known regarding effective ways for extracting valid and reliable CS-PCK segments from the practical work of CS in-service teachers; hence, only a limited inventory of such segments is available for CS educators. In this paper we report on research in which we developed and investigated a new strategy for extracting research-based CS-PCK segments from the practical work of experienced teachers who participated in the research. This strategy incorporated action research, conducted by the CS teachers in their classes, as part of a long and extensive workshop for professional development of CS teachers. Our findings show that the use of action research within the unique platform provided by the workshop yielded research-based CS-PCK segments. Furthermore, our findings emphasize the important role of the workshop, and in particular, its social context, in teachers' success in conducting reliable and valid action research. In addition, according to our findings, teachers' attitudes regarding the use of action research as a tool for improving their practice were positive, as well as their tendency to adopt and use it in their practice.

Leigh Ann Delyser and Lauren Wright

A Systems Change Approach to CS Education: Creating Rubrics for School System Implementation

Computer science education is moving from an elective subject in K-12 schools to a more compulsory topic with a focus on all students. As a new discipline is added to education, we need to explore not only the implications and theory behind student and teacher learning and development, but also the systems change that will happen in organizations

in order to produce sustainable CS education. In this paper, I explore theories of systems change, especially in education, and describe the creation of a series of rubrics to help districts self-assess their CS education supports. Early data is presented from a pilot year of workshops with 67 school districts showing a partial landscape of how school districts rate their own strengths and weaknesses and provide a discussion of initial evidence of validity for the rubrics.

Session 8B: Operating Systems and Security (Room: NK 6)

Adalbert Gerald Soosai Raj, Eda Zhang, Saswati Mukherjee, Jim Williams, Richard Halverson and Jignesh M. Patel

Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course

Understanding an operating systems (OS) code base is a difficult task since it involves understanding a huge amount of low-level C and assembly code. The inherent level of difficulty associated with OS topics is high because of the high element interactivity (i.e., material consists of elements that heavily interact). The mental effort associated with learning a complex subject like OS may be higher for non-native English speakers, when the subject is taught in a natural language (i.e., English) that is not the students' native language. We were interested in finding the effect of an instructional design that combines the students' native language along with English on students' understanding of select topics in OS. We designed an experiment to teach CPU virtualization using xv6 to two groups of undergraduate students in Tamil Nadu, India. We taught the experimental group using English and Tamil (native language of students in Tamil Nadu) and the control group using only English. We conducted a pre-test and a post-test to test students' understanding of the OS topics taught, before and after our intervention respectively. We also collected data on the questions that students asked in lectures during our intervention. We found that teaching OS using native language and English is as good as teaching OS using only English. We also found that the native language had an impact on the student engagement and classroom interaction by creating more dialogue within the Tamil+English (experimental) classroom when compared to the English-only (control) classroom.

Hui Chen, Agnieszka Ciborowska and Kostadin Damevski

Using Automated Prompts for Student Reflection on Computer Security Concepts

Reflection is known to be an effective means to improve students' learning. In this paper, we aim to foster meaningful reflection via prompts in computer science courses with a significant practical, software development component. To this end we develop an instructional strategy and system that automatically delivers prompts to students based on their commits a in source code repository. The system allows for prompts that instigate reflection in students to be timely with respect to student work, and delivered automatically, thus easily scaling up the strategy.

In this paper, we describe the design of a rule-based prompt delivery system, including a list of security related reflection prompts. We collect preliminary evidence for the reflection strategy in a course targeting mobile development. The evaluation provides evidence that

such a system can help realize a reflection-in-action instructional strategy at scale and improve students' learning.

James Walker, Man Wang, Steven Carr, Jean Mayo and Ching-Kuang Shene

Teaching Integer Security Using Simple Visualizations

Integer errors can introduce significant vulnerabilities into C programs. We have developed a program analysis and visualization tool to help students understand integer representation and type conversions with the goal to help students avoid introducing these errors into the code they develop. The visualization is through the Integer Representation (IR) window within a larger system for analysis and visualization of security issues in C programs. The system is called the Visualization and Analysis for C Code Security (VACCS) system. In this paper, we describe our experience with teaching fundamental aspects of integer security in a junior-level systems programming course, the IR window, and an evaluation of the tool. Our results indicate that students found the tool to be useful and that it enhanced the course in which it was used.

Session 8C: Learning Environments (Room: NK 10)

Samiha Marwan, Nicholas Lytle, Joseph Jay Williams and Thomas Price

The Impact of Adding Textual Explanations to Next-step Hints in a Novice Programming Environment

Automated hints are a powerful feature of many programming environments that have been shown to improve students' performance and learning. New methods for generating these hints use historical data, allowing them to scale easily to new classrooms and contexts. These methods often generate next-step, code hints that suggest a single edit for the student to make to their code. However, while these code hints tell the student what to do, they do not explain why, which can make these hints hard to interpret and decrease students' trust in their helpfulness. In this work, we augmented code hints by adding adaptive, textual explanations in a block-based, novice programming environment. We evaluated their impact in two controlled studies with novice learners to investigate how our results generalize to different populations. We measured the impact of textual explanations on novices' programming performance. We also used quantitative analysis of log data, self-explanation prompts and frequent feedback surveys to evaluate novices' understanding and perception of the hints throughout the learning process. Our results showed that novices perceived hints with explanations as significantly more relevant and interpretable than those without explanations, and were also better able to connect these hints to their code and the assignment. However, we found little difference in novices' performance. Our results suggest that explanations have the potential to make code hints more useful, but it is unclear whether this translates into better performance and learning.

Shaveen Singh and Bernd Meyer

Using Social Annotations to Augment the Learning Space and Learner Experience

Learning has always been considered a social and collaborative activity. From the social constructivism perspective, students learn through the process of sharing experiences and building knowledge and understanding through discussion. This pilot study demonstrates the successful implementation of a social annotation tool in a content

authoring platform, that allows students to discuss learning material with their fellow classmates. We investigate how the students leveraged the tool for effective learning and to log their sentiments while reading the material and attempting programming activities. We find that, those students who had access to the annotation tool spent - an above average amount of time on the material. They also had a higher completion rate and performance compared to the control group who did not have the annotation feature enabled for them. Feedback from the students who used the tool was also very positive. Finally, we delved deeper into analysing the annotation patterns and how by referring to annotations as feedback of students' learning, instructors can make use of this fine-grained analytics to inform and adapt the design of teaching and learning material and approaches in the online or blended context.

Simon Larsén and Richard Glasse

RepoBee: Developing Tool Support for Courses using Git/Github

The use of version control systems within computing education is growing in popularity. However, this is challenging because such systems are not particularly designed to support educational situations, nor are they easy to use with confidence in teaching, as specialist knowledge and experience is required. This experience paper reports the development of a tool to assist in the use of the git version control system in an educational context. The tool provides a straightforward interface for managing batch tasks such as repository generation and cloning for setting and gathering assignments, opening and closing of issues to communicate with students, as well as facilitating peer reviews. Parts of the tool are open to integration with third party tools for additional tasks, such as running unit tests or static analysis on student repositories. We also include the perspectives of both teachers and teaching assistants who have been using the tool as part of a first year course for computer scientists.

Session 8D: Evaluation and Analytics (Room: NK 1)

Susana Masapanta-Carrión and J. Ángel Velázquez-Iturbide

Evaluating Instructors' Classification of Programming Exercises Using the Revised Bloom's Taxonomy

It is a well-known fact that different instructors understand Bloom's taxonomy differently, thus they classify given test exercises into different levels of the taxonomy. The article reports on the impact of training instructors on the use of the taxonomy for programming courses with two innovations: encouraging instructors to classify each exercise into several cognitive processes necessary to solve it, and providing them with a wide range of examples of classified programming exercises. We conducted an evaluation composed of two training sessions. In the first session, the revised Bloom's taxonomy was introduced to the participants, and in the second session, the two innovations were presented. In both sessions, participants were asked to classify ten exercises. The results show that classification into several cognitive processes was natural to instructors. However, other results were mixed. On the one hand, accuracy was increased, and confidence was increased for experienced participants. On the other hand, variation was increased, and confidence was decreased for non-experienced participants. Results show that our

approach assists instructors in classifying exercises, but higher increases of expert instructors suggest that longer training is necessary for non-expert instructors.

Lauren Margulieux, Briana Morrison and Adrienne Decker

Design and Pilot Testing of Subgoal Labeled Worked Examples for Five Core Concepts in CS1

Subgoal learning has improved student problem-solving performance in programming, but it has been tested for only one to two hours of instruction at a time. Our work pioneers implementing subgoal learning throughout an entire introductory programming course. In this paper we discuss the protocol that we used to identify subgoals for core programming procedures, present the subgoal labels created for the course, and outline the subgoal-labeled instructional materials that were designed for a Java-based course. To examine the effect of subgoal labeled materials on student performance in the course, we compared quiz and exam grades between students who learned using subgoal labels and those who learned using conventional materials. Initial results indicate that learning with subgoals improves performance on early applications of concepts. Moreover, variance in performance was lower and persistence in the course was higher for students who learned with subgoals compared to those who learned with conventional materials, suggesting that learning with subgoal labels may uniquely benefit students who would normally receive low grades or dropout of the course.

Lawton Nichols, Kyle Dewey, Mehmet Emre, Sitao Chen and Ben Hardekopf

Syntax-based Improvements to Plagiarism Detectors and their Evaluations

Software plagiarism cheats students out of their own education and leads to unfair grading, making software plagiarism detection an important problem. However, many popular plagiarism detection tools are inaccurate, language-specific, or closed source, limiting their applicability. In this work, we seek to address these problems via a novel approach. We adapt the optimal Smith-Waterman sequence alignment algorithm to precisely measure the similarity between programs, greatly improving detection accuracy relative to competitors. Our approach is applicable to any language describable by an ANTLR grammar, which includes most programming languages. We also provide a new type of evaluation based on random program generation and obfuscation. Finally, we make our approach freely available, allowing for customizations and transparent reasoning about detection behavior.

Conference Banquet

Tuesday, night conference banquet, July 16, 19:00 for 19:30

Beach Ballroom, Beach Promenade, Aberdeen AB24 5NR

Welcome
Dinner
Whisky/Gin Tasting
Ceilidh

End 23:00

WiFi Access and Emergency Contacts

WiFi Access

Eduroam

Delegates associated with an institution belonging to Eduroam, can use local Eduroam WiFi service using their own institutional credentials username@institution and password). To use the service, there are no special settings, the protocols used are WPA2/AES (WPA2 Enterprise). The network name is 'eduroam' and users will be authenticated by their own university server.

Aberdeen City Connect

Internet is also available for delegates to use free of charge by accessing the Aberdeen City Connect network. This might restrict access to some protocols if you're trying to do some development work while here.

University WiFi Account

Should this fail, please contact the helpdesk who can issue you with an account to connect to the UoA secure networks.

Emergency Contacts

Anywhere in Scotland: Dial 999 and ask for police, fire service, ambulance.
Campus security +44 (0)1224 273327 for issues on campus

Contact Numbers

Bruce Scharlau (conference co-chair) +44 (0)7852 132714
University of Aberdeen reception +44 (0)1224 270000

Local Matters in Aberdeen

Aberdeen

Aberdeen is generally walkable. You can walk to most places you want to visit within 30-40 minutes from the university. You either go one of three ways:

a. Via the Spital (the cobblestone street through campus heading south), or
b. Via King Street (the main road with buses) alongside campus, or
c. Via Bedford Road to George Street from just beyond the library (the glass cube), and head up the hill towards the left, and then left at the light at the top of the road).

Most destinations are along Union Street, or one of its side streets.

Buses

On arrival to Aberdeen International Airport, it is recommended to purchase a Grasshopper bus pass from Stagecoach who operate the Jet 727 bus to Aberdeen City Centre. This bus pass will then allow you to travel on the number 1, 2 or 40 FirstGroup buses which go to the University of Aberdeen King's College campus along King Street. Buses in Aberdeen City Centre are mainly operated by FirstGroup - http://www.firstgroup.com/ukbus/aberdeen/.

Stagecoach Bluebird buses mainly operate to destinations outside the city starting from the Bus Station at Union Square. However, they also operate the Jet 727 bus from the airport to the city centre.

Weekly bus passes can be obtained for either bus company. Please note that First Buses require exact change whereas Stagecoach buses will give change if available. All buses now accept contactless payment.

Taxis

Taxis can be ordered from: Rainbow City Taxis - 01224 878787, Comcabs - 01224 353535, Central Taxis - 01224 898989.

Should you require a taxi at any time, please contact the event organisers who can arrange a taxi for you – please ensure you provide plenty of time to book your taxi. Please confirm if you would be willing to taxi share on your journey.

Rail

Whilst there is no city rail network or underground there is a mainline railway service to near the international airport and some adjacent towns to the coast south of Aberdeen (e.g. Stonehaven c10 mins).

Currency and Banks

The official currency of the UK is £ pound sterling. Commonly accepted credit cards are Visa and Mastercard. Display signs are usually visible in all restaurants and shops indicating which cards they accept.

There is a range of banks in Aberdeen City Centre with ATMs. Most banks are closed on Saturdays and all day Sunday. A bank is located at the St Machar roundabout on King Street where there are ATMs also. There is an ATM outside the Student Association (formerly the Hub) on Elphinstone Walk, or you can obtain cash back from the shops on the High Street (open Mon-Fri during working hours).

Notes

Notes

Notes

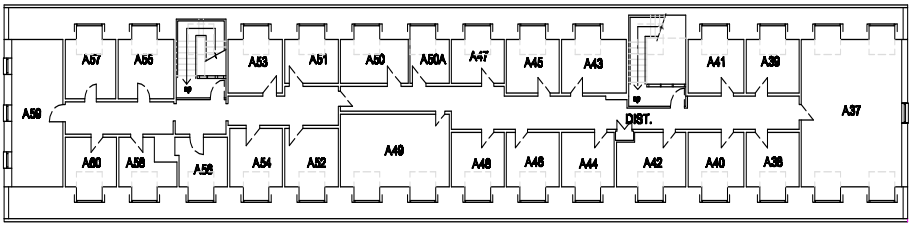
Notes

Notes

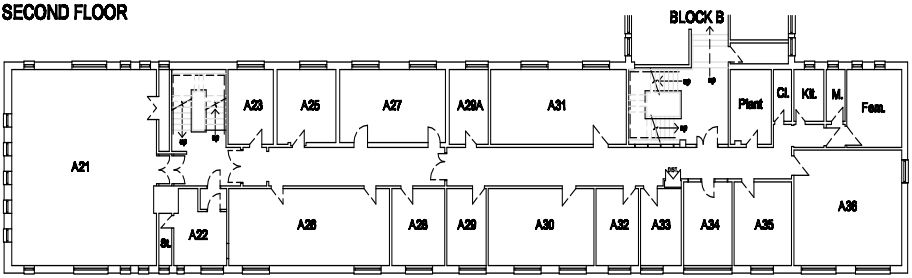
Notes

Conference Room Plans

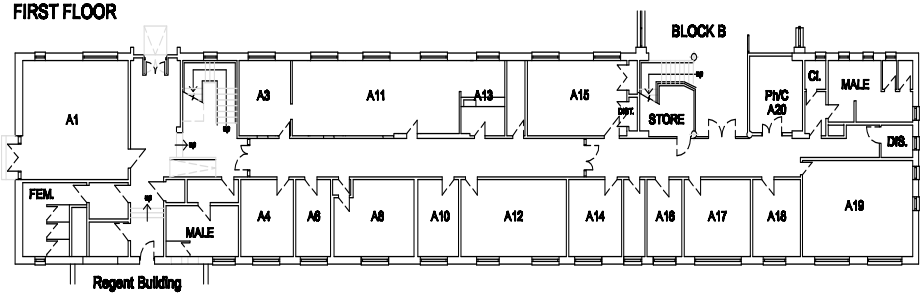
Taylor



SECOND FLOOR



FIRST FLOOR



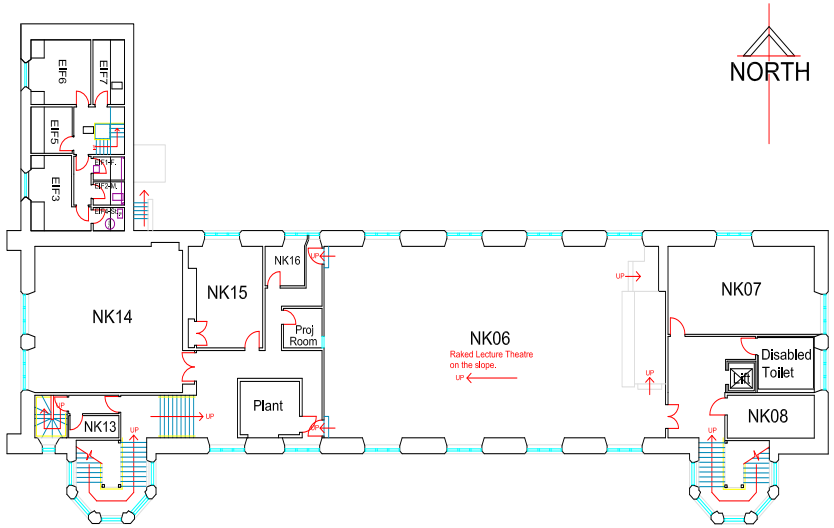
Regent Building

GROUND FLOOR

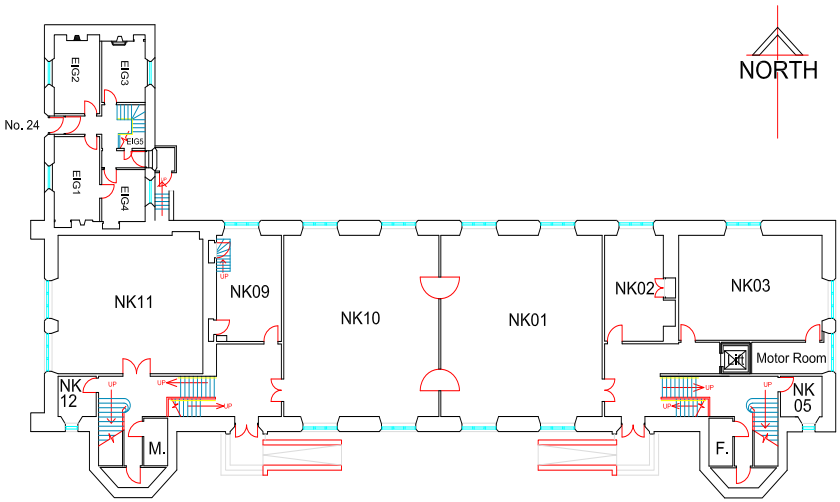
Conference Room Plans

New Kings

First Floor

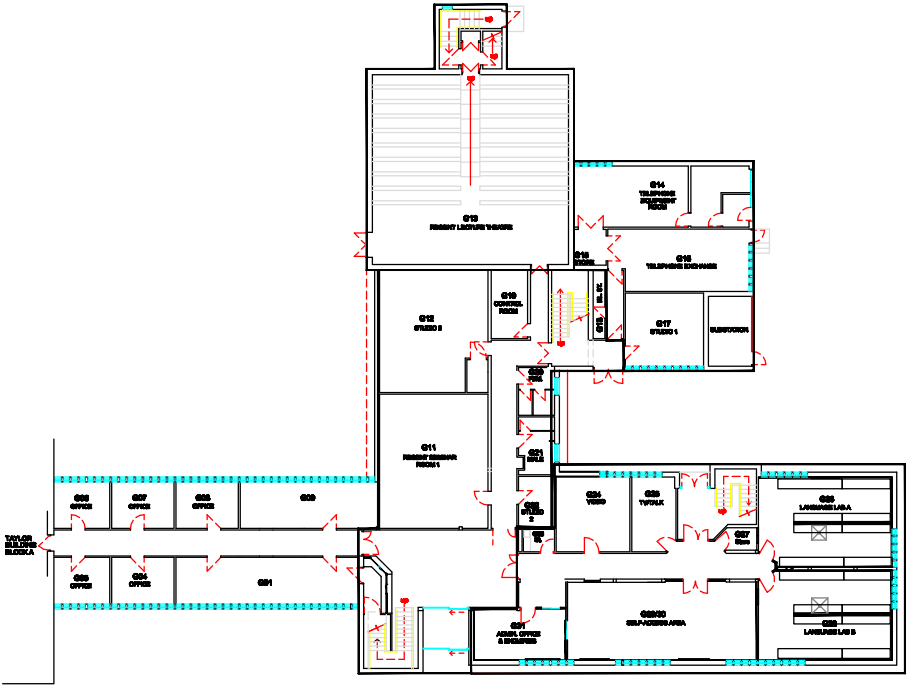


Ground Floor



Conference Room Plans

Regent



UNIVERSITY OF ABERDEEN CAMPUS MAP

1 Zoology Building
2 Cruickshank Building
3 23 St Machar Drive
4 King's Museum (Old Town House)
5 Students' Union Building
6 St Mary's
7 Fraser Noble Building
8 Elphinstone Road Halls
9 The Sir Duncan Rice Library
10 Meston Building
11 Multi-Faith Chaplaincy
12 Confucius Institute
13 Security Office/Mailroom
14 Counselling Service
15 Edward Wright Building
16 Edward Wright Annexe
17 MacRobert Building
18 William Guild Building
19 Arts Lecture Theatre
20 Taylor Building
21 Old Brewery
22 New King's
23 Regent Building
24 University Office
25 Elphinstone Hall
26 Linklater Rooms
27 King's College Chapel
28 King's College Centre
29 King's College
30 King's Pavilion
31 50-52 College Bounds
32 Butchart Centre
33 Crombie Annexe
34 Crombie Halls
35 Rocking Horse Nursery
36 King's Hall
37 Powis Gate/Muslim Prayer Room
38 Johnston Hall
39 Johnston Annexe
40 Humanity Manse
41 Bedford Road Workshops/CHP
42 Johnston Central
43 International Centre
44 Infohub

Ⓟ Regulated Parking

Conference Venue Map