The Programming Studio: Supporting Teachers' Readiness to Program in the North and South of Ireland

Final Report

Dr Pamela Cowan, Queen's University Belfast
Elizabeth Oldham, Trinity College Dublin, University of Dublin
Ann FitzGibbon, Trinity College Dublin, University of Dublin

July 2015

Table of contents

Introduction

Rationale and context

The paucity of programming experience in the current school curricula is claimed to be a contributory factor to the small number of Computer Science students graduating from third level institutions in both the Northern Ireland and the Republic of Ireland (henceforth, NI and RI respectively) and internationally. The current situation as regards the teaching of programming, and more generally Computer Science, is well captured by the following description:

> "*[Digital technology] in school curricula is often diluted because it has to cover three quite different directions: (1) using computers as a tool for teaching (e.g.* ***e-learning****), (2) using computers as a tool for general purpose applications (sometimes called* ***ICT****), and (3) computing as a discipline in its own right (including programming and CS [****Computer Science****]). Sometimes administrators and leaders confuse these roles, and this can make it difficult for Computer Science to be visible as a discipline in its own right*" (Bell, Andreae, and Lambert, 2010, p.17, emphasis added).

With regard to *e-learning*, the potential of digital technology for teaching and learning may have distracted attention from the role of *Computer Science* in appealing to at least some students. For *ICT*, Bell, Andreae, and Lambert (2010) raise an additional and relevant issue of concern. While ICT courses may contain some programming elements – say, for game development – overall they typically misrepresent the nature of Computer Science, leading students to opt for the latter at university or college for inappropriate reasons. However, in several countries [see for example Thompson and Bell (2013)], interest in Computer Science or programming in schools has recently grown, or has revived after a period in which the topics had become unfashionable. The latter is true of both Irish jurisdictions. The detailed situations in the two jurisdictions are different; relevant features are outlined in turn, with links to further information provided.

NI schools mainly offer ICT courses in Key Stage 3 (the junior cycle) of second-level education (http://www.ccea.org.uk/) with strong cross-curricular links to other subjects. Dedicated ICT lessons typically do involve some work with Scratch (www.scratch.mit.edu), a low-floor programming tool developed from LOGO (Crook, 2009), and GameMaker (www.yoyogames.com/studio), intended for game development. However, pressure for the introduction of Computer Science reflects the UK-wide drive for 'computing' in schools (see http://www.computingatschool.org.uk/). Pre-service teacher education for ICT courses currently includes some programming to address national qualifications in the area; there are limited opportunities for student teachers to teach programming beyond Key Stage 3 in schools. GCSE and Advanced level (equivalent to senior cycle) Computing classes tend to be the preserve of a limited number of grammar schools with relatively small class sizes in comparison to ICT or Applied ICT classes. These schools register with UK examination boards offering Computer Science for assessment purposes.

There has been less formal provision in RI. However, extra-curricular activities are provided in some schools, and informal provision has burgeoned in recent years via CoderDojos (www.coderdojo.com). The Coderdojo movement – which started in Ireland – allows children to learn coding voluntarily, often via Scratch. As regards teacher education, short professional development courses on Scratch for teachers have been offered, notably by or in conjunction with the Computers in Education Society of Ireland (CESI), a voluntary body interested in all aspects of digital literacy. Since the Programming Studio project was initiated in 2013, an optional 'short course' on Coding was approved for the junior cycle of the second-level curriculum, together with one on Digital Literacy (www.ncca.ie); their introduction is now being supported by a major professional development initiative on the teaching of Computer Science topics, run at Trinity College Dublin (www.scss.tcd.ie/disciplines/information_systems/crite/crite_web/).

Outline of the study

Thompson and Bell (2013, p.87) state: "*Teachers play a pivotal role in the adoption of a new computer science curriculum, and therefore it is important to understand teachers' needs and attitudes to new material.*" In neither Irish jurisdiction was there a current and clear picture of the extent to which teachers were ready and willing to teach programming. It was against this background that the Programming Studio project was started. The key question being addressed in this study is:

> *Are there currently teachers in schools with the capacity to teach programming to examination classes?*

The first phase of the study aimed to evaluate teachers' motivation, willingness, competency and self-efficacy towards programming through using a form of the Technology Acceptance Model (TAM) (Davis, 1989; Venkatesh *et al.*, 2003; Teo, 2010) adapted to focus on programming. An online survey was offered to a purposive sample of potential teachers of programming, via existing contacts who either currently teach ICT or are actively embedding new and innovative technologies into their subject area, making them potential candidates for teaching programming if needed.

The second phase of the study focused on the Programming Studio, a series of online courses hosted in virtual learning environment, available to all schools in NI with guest access available for collaborative partners outside the jurisdiction. These online courses were designed to offer a limited group of post-primary school ICT teachers (from NI) and a mix of primary and secondary teachers (from RI) an opportunity to experience programming as a 'digital literacy' by mastering the programming language(s) identified above through a form of games-based learning. The wider skill of 'Computational Thinking' at the heart of computer programming was embedded in the Programming Studio game plan and evaluated by the teacher participants.

Outline of the report

In this report, first, the research questions for the study are listed; then a short literature review is offered. The methodology for the two phases of the study is described, and selected findings are presented. (Fuller accounts of the literature, methodology and findings are included in papers already given or due to be given after the submission of the report.) The body of the report ends with discussion, conclusions and recommendations.

Research questions

The dual nature of the study is reflected in the research questions posed from the outset.

RQ1 To what extent are prospective teachers of programming, in Northern Ireland and in the Republic of Ireland, ready and willing to teach programming in schools?

RQ2 To what extent do the teachers see programming as an essential element of school curricula?

RQ3 How competent are teachers to teach programming? What programming languages are they most familiar with?

RQ4 To what extent does gamification support the learning experience?

RQ5 How does the literature define 'Computational Thinking' (CT) and how do teachers view CT as a 21st century skill?

Literature review

Literature from five key areas informed the study.

The Technology Acceptance Model (TAM)

The theoretical framework underpinning the first phase of the study is the Technology Acceptance Model (TAM). It was introduced by Davis (1989) and subsequently refined and developed in particular by Venkatesh and Davis (2000) and Liu, Li, and Carlsson (2010). In the simplest form of the model, the constructs *Perceived Usefulness* (PU) and *Perceived Ease of Use* (PEU) jointly affect a user's intention to use a system (*Behavioural Intention*, BI), with this in turn impinging on *Usage Behaviour*. Moreover, as a system that is easy to use is more useful than one that is not, PEU impinges on PU.

The simple model has been expanded to include determinants of the constructs, in particular of PU and PEU. Among the additional elements, those perceived to be of relevance to this study are:

- *Subjective Norm* (SN): a person's perception that most people who are important to the person think he/she should or should not perform the behaviour in question;
- *Image*: the degree to which use of an innovation is perceived to enhance one's status in one's social system;
- *Voluntariness*: the extent to which potential adopters perceive the adoption decision to be non-mandatory;
- *Result Demonstrability* (RD): the tangibility of the results of using the innovation.

The Computer Science, programming and Computational Thinking conundrum

The current economic climate has seen a renewed interest in Computer Science as a discipline, not just nationally but internationally. Henderson *et al*. (2007) blame the equating of 'programming' with Computer Science as having a detrimental effect on the public perception of careers in computing, resulting in a lack of talented students entering the profession. Computer Science consists of mechanics, design principles and practices (Denning, 2003) so is neither programming nor computer literacy. Computational Thinking is often viewed as the precursor to programming but it too does not equate to Computer Science (Lu and Fletcher, 2009). Programming, itself, is only one of the four core practices in the Great Principles Framework for Computer Science (Denning, 2009, p.30).

Digital literacy in the 21st Century

Digital literacy is more than being technically competent to use ICT; it requires users to be creative in their choice of forms of expression, to access existing information and to author new material. Digital literacy goes beyond accessing, retrieving and processing information, although these are key components of being digitally literate. It encapsulates the ability to design, create and express oneself with new technologies (Resnick, 2012), making digital literacy a skill that evolves over time and is context-dependent in line with emerging technologies.

Defining Computational Thinking

Computational Thinking (CT) is a relatively new term coined by Wing (2006) to "*describe a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applicable in the information society.*" (Lee *et al*., 2011, p.32). For some, CT is **the** skill of the 21st Century ranking alongside reading, writing and arithmetic as a core skill; for others CT is

the use of abstraction, automation and analysis in problem-solving (Cuny *et al*., 2010). Terms such as mathematical thinking, or algorithmic thinking, the thinking skills associated with engineering science or design, are already in existence, and many people view these as encapsulating the concepts inherent in Computational Thinking (Cooper, Perez, and Rainey, 2010).

Games-based learning and gamification

Games-based learning (GBL) is defined as having interactivity (Thornton *et al*., 1990), rules and goals (Johnston and de Felix, 1993) challenges and risks (Baranauskas *et al*., 1999) and elements of fantasy, curiosity, challenge and control (Malone, 1981). It has defined learning outcomes and aims to balance subject matter with gameplay. Ideally, GBL should develop skills and knowledge relevant to the real world. With the new capabilities of human-computer interactions (Pivec *et al.*, 2003), increased levels of active engagement can be achieved in both the non-immersive contextual material as well as in immersive virtual worlds.

GBL uses actual games to teach concepts (GCO, 2012) while gamification uses "*game mechanics, dynamics, and frameworks to promote desired behaviours*" (MacMillan, 2011), thus replicating the gaming experience within a defined learning context. Gamification aims to "*inspire and motivate people to perform specific activities, to increase engagement … by creating enjoyable experiences in playful interactive environments*" (Deterding *et al*., 2011 as cited in Tzouvara *et al.*, 2013, p.2). Possibly the most notable feature of gamification is the use of "*rewards, feedback and reputation using elements like points, badges, progress bars, customized messages and leaderboards*" (Tzouvara *et al*., 2013, p.1) to motivate and encourage the learners to successfully complete the tasks set by the teacher.

The design of gamification systems should be challenging on three levels: cognitive, emotional and social, while also addressing the four elements of games design: Aesthetics, Story, Mechanics and Technology (Schell, 2008). It should be presented in a compelling way which envelops the player, increasing their desire to engage with the game. The mechanics of the game define how it works, "*be that points, levels, cash, badges…*" (Betts, 2011). Points or rewards within the game/course motivate the learners and inspire them to be active seekers of knowledge (Raymer, 2011).

Methodology

Research Ethics

Ethics approval for the research was granted by the School of Education, Queen's University Belfast, and by Trinity College Dublin.

Population and sample

Given that the emphasis on introducing programming or Computer Science courses in both jurisdictions was primarily on second-level education, the focus of the Programming Studio is chiefly on teachers in second-level schools. However, using a nationally representative sample of teachers in such schools would have been inappropriate; most questions would be meaningful only to people with some interest or competency in the area. Consequently, purposive samples aimed at targeting prospective teachers of programming were identified. In NI, the obvious candidates were teachers of the subject ICT, so schools' ICT co-ordinators and/or Heads of Department were contacted to disseminate the survey to their colleagues who taught ICT. In RI, because of the lack of an established ICT course in the curriculum, a different approach had to be taken. Sources for locating suitable teachers were via the CESI (Computers in Education Society of Ireland) membership list and lists of participants in appropriate professional development courses.

Survey instrument

The questionnaire for Phase 1 of the study was designed to explore relevant knowledge and dispositions key to willingness to teach programming. As well as seeking background information (gender, age, number of years in teaching, and so forth), it addressed teachers' qualifications, their view of the role of programming in the school curriculum, and their knowledge of and attitudes to programming – the latter part being based on TAM. The final sections of the survey presented more open-ended questions on Computational Thinking and further support needs for programming in other languages than those offered in the Programming Studio.

Piloting and administration

The instrument was piloted with Master's students studying technology and learning in Trinity College Dublin, and minor amendments were made in the light of feedback. The final version was localised where necessary to reflect differences in the NI and RI education systems and contexts. For example, the two categories used with regard to programming in the junior cycle at second level are not identical; in Northern Ireland they represent the Key Stage 3 and Key Stage 4 aspects of compulsory schooling, whereas in RI they represent short courses and full courses for national certification at the end of the cycle. However, the section on knowledge of programming and the TAM section were identical in both versions of the questionnaire.

The questionnaire was shared online using the free survey tool, SurveyGizmo. Approaches were made to prospective teachers of programming via heads of ICT departments in schools and personal contacts (NI) and the CESI mailing list and organisers of some courses for teachers (RI). The survey was open for six weeks in Spring 2014, and reminders were sent regularly before closing to ensure all respondents had an opportunity to complete their response.

Data processing

The data were downloaded and entered into SPSS, with responses 'strongly disagree' to 'strongly agree' being coded from 1 to 5. Very incomplete records that would have contributed nothing to the analysis were eliminated. Altogether, 161 records were retained; details of this achieved sample are given in the Findings section. Descriptive statistics were computed, and correlations obtained between variables of interest, in particular the TAM items.

As the TAM section of the questionnaire was designed specifically for the present study, investigation was needed to see if it measured the intended constructs described above. Analysis for TAM was based on 137 of the 161 cases, reflecting some pattern of non-response at the end of the questionnaire. The computation of correlation coefficients immediately pointed to deviations from the intended structure. Exploratory factor analyses were undertaken, several models being used in an attempt to maximise *accounting for variance* together with obtaining factors that *were reliable* and *could be interpreted*. The version deemed most successful is described and discussed in the Findings section below.

For each of the factors, item scores were added and the total divided by the number of items to produce a scale score for each respondent. Possible scores range from 1 ('strongly disagree' with all component items) to 5 ('strongly agree' with all). Scale scores were computed also for a variable designated as CURRICULUM, formed by finding the mean of the four 'programming in schools' scores for each respondent. Again, the range is from 1 to 5. For these scale scores, means and standard deviations were calculated for each sample. Comparisons between jurisdictions were made for relevant variables, using t-tests.

The online courses

Phase 2 of this study was planned to adopt a social constructivist view in introducing programming to teachers who had expressed an interest in learning how to build a simple computer game either in Scratch, GameMaker or Greenfoot. To address differing levels of expertise in programming, and to ensure there was a package which was new and therefore suitable for all participating teachers, three parallel courses with associated activities were developed, one for each programming language.

As the teachers were located across a broad geographical area in Ireland, an online course hosted in the Fronter Virtual Learning Environment (VLE) platform was used to connect the teachers 'virtually' and facilitate any collaborative discussions around the series of tasks designed to scaffold the learning of the new programming language. Key questions were posted in the discussion forums, for example to encourage these teachers to reflect on the processes of learning to program, to identify any CT skills being developed through the series of tasks to be completed online, and to discuss if designing a simple game was a suitable context in which to learn the programming language.

Eight sessions were created in the programming courses in the Fronter VLE, which is available to all schools in NI. Guest access was given to the participants from RI for the duration of the project. The online games design courses that mimicked the 'use-modify-create' cycle (Lee *et al*., 2011), commencing with an opportunity to 'play' some simple games (such as PacMan) and reflect on the importance of clear instructions, rules and an end point in any game.

Typical course content is shown in Table I. Week numbers can be replaced by session numbers if users are working continuously on their game.

Week number	Tasks
1	Types of games and playing some games (Use)
2	Storyboarding and planning the characters (Modify)
3	Create the room, sprites and objects (Create)
4	Get the sprites moving! Looking at speed and collisions with walls.
5	Scores and Lives – initializing variables
6	Sound effects – reasons: hitting a wall, losing a life, gaining points (eating)
7	Increasing the difficulty – speed or numbers of objects in new room
8	Certificates of Completion – overall reward.

Table I. Illustration of the content of a games design course.

In two of these online courses, the gamification process remained the same, namely the award of air miles for the successful completion of the tasks in each session. A different 'reward' system, using the board game Monopoly, was used in the third game to establish the extent to which the type of gamification impacts on the motivation of the learners.

It was intended that a selection would be made from the teachers who, in the survey, declared their interest in taking the online courses. Groups addressing each programming language would be formed, with members in each group from each of the two jurisdictions, with the aim of enriching the discussion as participants shared their experiences of different systems and curricula.

A collage of screenshots of the online courses

Selected findings

Technology Acceptance Model (TAM) findings

The structure for TAM resulted from a Principal Components Analysis retaining eigenvalues greater than 1 and using the Varimax rotation. This gave five factors with reliability (as measured by Cronbach's alpha) greater than 0.7, and accounted for 58% of the variance. From inspection of the items retained – those with loading greater than or equal to 0.5, together with two that had loadings greater than 0.4 and retention of which improved reliability – the five factors were tentatively described as follows:

- PUT: Perceived Usefulness for Teaching
- PEU: Perceived Ease of Use
- PIBI: Personal Innovative Behaviour and Intentions
- IMAGE: Image
- SNS: Subjective Norm in School.

Details of the factors, together with indicative items for each one, are shown in Table II. It should be noted that the factors do not correspond exactly to those in standard versions of TAM and to some of the intended constructs for the study.

Scale	No. of items	Specimen item(s)	Alpha
PUT	7	Being able to teach a programming language enhances my effectiveness as a teacher I would be keen to introduce programming in my school	.889
PEU	4	Teaching programming is easy Programming is easy	.838
PIBI	6	If I heard about a new technology, I would look for ways to experiment with it in my teaching	.785
IMAGE	3	People in my school who can teach programming will have a high profile	.817
SNS	2	Senior staff who influence my behaviour think that I should be able to teach programming	.726

Table II. Factor details (number of items, reliability) with selected items.

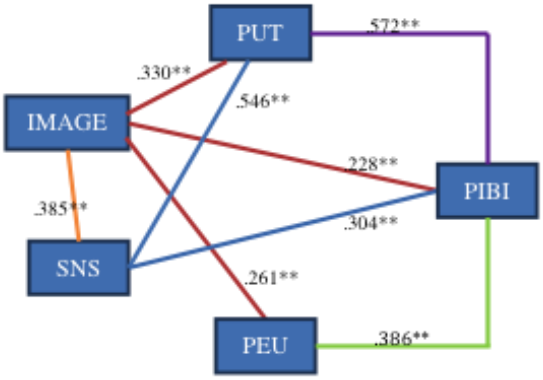

Figure 1. Relationship between constructed TAM scales, showing Pearson correlation coefficients.

Note: **Correlation is significant at the 0.01 level

The relationships – highly significant Pearson correlation coefficients – between the scales are shown in Figure 1 in graphical form. Nodes are placed in positions suggested by the standard TAM model, but the data do not provide evidence of the direction of any causal relationships. Unlike the case for comparable scales in the standard model, there is no significant correlation between PUT and PEU.

Findings for respondents in Northern Ireland and in the Republic of Ireland

Selected findings for the respondents from the two jurisdictions are reported in this section. (As indicated above, further findings are reported in papers already presented or to be delivered after the submission of this report.) While comparisons are made, the different nature of the samples means that these should be interpreted as highlighting two different 'story lines,' rather than in terms of one group being regarded as superior or inferior to the other.

Biographical data

As indicated above, after data cleaning and the removal of very incomplete responses, 161 cases remained: 62 responses for NI and 99 responses for RI. The NI sample was approximately one-third male and two-thirds female, with an average age of 38.6 years old and on average 13.7 years of teaching experience, indicating that the average teachers entered their career in their mid-20s, shortly after graduating. In contrast, the RI sample was 80% male and 20% female, but also with an average age of 38.2 years old and on average 13.9 years of teaching experience.

Over 90% of the respondents (57 teachers) from NI taught 11-16 year olds, while 83.9% (52 teachers) taught post-16 students. Only 8 teachers were not specialist teachers of ICT/Computing in NI; 3 were Business Studies teachers, 2 specialised in Geography, and one each in Music, Languages and Technology & Design. 24.2% of respondents (15 teachers) from NI declared they had no formal qualification in computing nor had they attended informal/CPD courses for the subject. For the RI sample, almost 90% of the respondents (89 teachers) taught Junior Cycle (12-15 year olds), nearly 80% (77.8%, 77 teachers) taught Transition Year students (age 15-16) and 86.9% (86 teachers) taught Senior Cycle students (age 16-18 years). Only 9 teachers had specialised in ICT as one of their subjects. The majority of teachers were either Engineering/Design Communication Graphics or Construction/Materials Technology experts; there were 10 primary school teachers in the sample.

Opportunities for students to learn to program

Of the 62 NI teachers who responded, 37.1% (23 teachers) were currently teaching programming and 16.1% (10 teachers) were planning to soon. Nineteen teachers (30.6% of NI respondents) had embedded programming as a subject or formal course within the normal school timetable, 14.5% (9 teachers) were informally teaching computer programming in a computer club typically at lunchtimes or after-school, while 3.2% (2 teachers) were engaged in other Coder Dojo-type activities. Almost

one-third (32.3%) (20 teachers) of respondents in NI said they would be keen to start an after-school computer programming club while an additional third (33.9%) (21 teachers) were giving the idea some consideration, ticking 'maybe'.

The pattern in the RI revealed less school-based activity in teaching programming. Of the 99 teachers who responded, 12.1% (12 teachers) were currently teaching programming. Eight teachers (8.1%) indicated they taught programming as a subject or formal course in school; 4.0% (4) offer it as an extra-curricular subject at lunchtime or after-school. Only 1 teacher was engaged in other CoderDojo type activities. In terms of starting an after-school programming club, 24.2% (24) teachers, in RI were willing to try this, while another 42.4% (42) said they would think about it.

Knowledge and use of programming languages / tools

The NI cohort reported considerably more awareness and competency of programming languages / tools than the cohort from RI. In particular, except for Scratch, JavaScript, Java and HTML, for which the figures in both jurisdictions were high (over 75%), the percentages in the former who reported having 'heard of' the languages are of the order of twice as large as those for the latter. NI respondents were generally familiar with Scratch and GameMaker as well as with languages associated with web use (HTML, JavaScript, Java), though not necessarily ready to teach these latter two in school. Those from RI reported some familiarity with Scratch; apart from that and awareness of the web-related languages, reported familiarity was very low.

In responses to a question about other courses or programming languages which they would like to be offered for professional development purposes, the web-based ones such as Java / Javascript and HTML emerged as popular as Python both in RI and NI. Teachers in NI also requested C#, reflecting the current curriculum requirements at A level and the need for pedagogical support in delivering any programming language to pupils.

Place of programming in schools

The mean scores for respondents' levels of agreement with the statements that the inclusion of programming in school curricula is 'essential' at different stages of the school system were calculated, and are shown in Figure 2. All mean scores are above 3, representing views that on average are at least neutral and in general somewhat positive towards programming being deemed essential. The CURRICULUM scale, with reliability (Cronbach alpha) 0.876, provides a useful summary. The NI mean scores are significantly higher except for the 'minor' course in the junior cycle/Key Stage 3 – the level at which the Coding short course has just been introduced in RI – and for primary level.

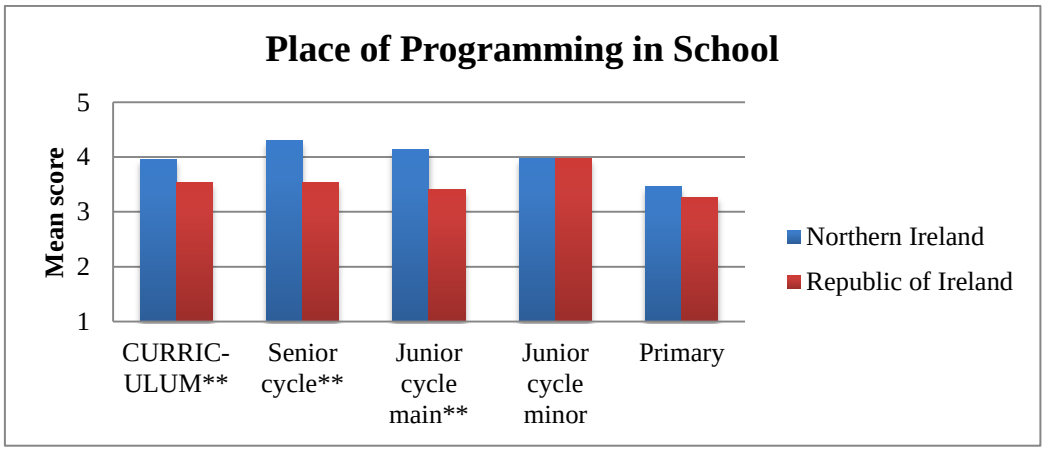

Figure 2. Extent of agreement with programming being essential in school curricula at different levels of the education system – mean scores, by jurisdiction.

Note: ** Northern Ireland scores significantly higher ($p < .01$).

Attitudes to programming using the TAM scales

Attitudes to programming as reflected in mean TAM scale scores are shown in Figure 3. For PUT and PIBI, means are on the positive side of neutral, perhaps indicating recognition among participants of a role for teachers of programming and also some willingness to undertake that role. For PEU and IMAGE, however, means are on the negative side, suggesting that programming and teaching it are challenging but that the respondents do not see the role as ascribing high status in their school systems. The results for SNS point to the NI teachers as being under more pressure to undertake the role.

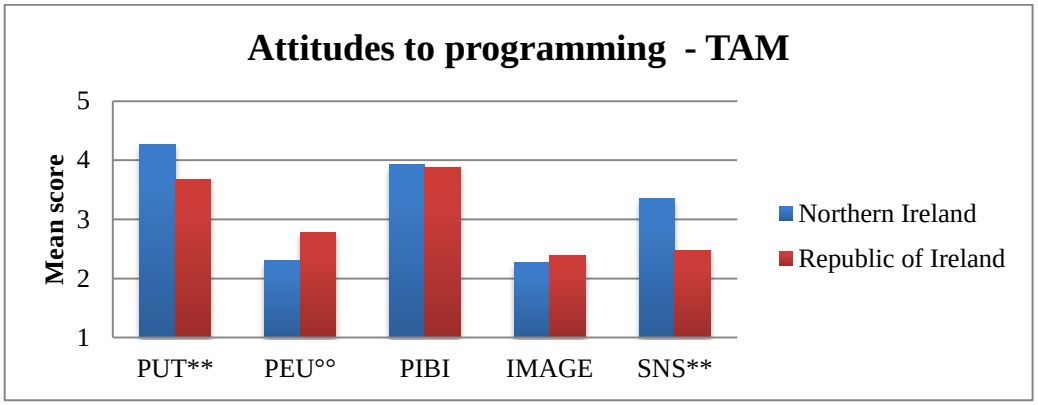

Figure 3. Mean scores on constructed TAM scales, by jurisdiction

Note: **NI scores significantly higher ($p < .01$)

°°RI scores significantly higher ($p < .01$)

Teachers' perceptions of Computational Thinking (CT)

Of the 62 respondents from NI who completed this question, 19.4% said they were familiar with the term 'Computational Thinking' and 17.7% felt 'somewhat' familiar. The remaining 41.9% had no awareness of this term. In RI a very similar pattern emerged with 18.2% saying they were familiar with the term 'Computational Thinking' and 19.2% felt 'somewhat' familiar. The remaining 52.5% had no awareness of this term.

Of those who had indicated some knowledge of CT, they were asked to put four statements considered to be components of CT into order of importance. The outcome was as follows (Table III):

Computational Thinking is …	NI – rank order	RI – rank order
Solving problems	1	1
Understanding human behaviour	2	3
Designing systems	3	4
Understanding of the concepts fundamental to Computer Science	4	2

Table III. Comparison of NI and RI teachers' definitions of CT

It is clear from the table that there is consensus on Computational Thinking being primarily about solving problems, however the remaining three components were ranked in different orders with the 'understanding of the concepts fundamental to Computer Science' creating the most divergent opinion.

Findings from the online courses

Overall analysis of the data indicated learners had a positive experience when working online; the CT skills were embedded well in the tasks but may need to be explicitly highlighted in various activities, and choice of programming languages (due to the nature of the games design task) may have limited the ability to develop the CT skill of abstraction. Most learners were positively disposed to the inclusion of gamification in the learning process as a tool for motivation and to sustain engagement for the duration of the online course.

Considering each of these areas in turn, the following findings can be reported.

The online course

All participants found the courses accessible, engaging and easy to navigate, and enjoyable to complete. They thought the course was suitable for the age and ability of their pupils, was topical and interesting – "*Sounds are fun, and we live in a world full of noise and sounds so a game without sound would be like an old black and white movie.*" Participants liked the 'new take' on an old game (such as PacMan) and the cross-curricular nature of the themes in the courses – "*Science would definitely be discussed – we are, in effect, building an antibody reaction here!*" Most participants felt they had a strong grasp of the programming language at the end of the course and would be able to extend their expertise further to stretch and challenge more able learners in their class. Structure and scaffolding in the course design assisted in the learning of the programming language, but some learners indicated it diminished the potential for creativity due to the linear nature of the course structure: "*At the end of the course, participants would produce pretty similar games with little to differentiate between them, I think."*

Computational Thinking skills

Through analysis of the comments posted on the discussion forums, the three aspects of CT, namely the use of abstraction, automation and analysis in problem-solving, were identified. Participants agreed that many of the CT skills were addressed during the games design courses – "*any student who has reached this stage of the game will definitely have been introduced to CT.*" Parallels were drawn between storyboarding and algorithmic thinking - "*Storyboarding helps pupils to create the direction, the structure and sequence of the game*." Abstraction was mentioned by some through identification of patterns in the code – *"In this lesson, students are establishing patterns, and unconsciously trying to abstract into something from which they can build a smooth algorithm*." Others recognised the lack of abstraction could be a limitation of the programming context - "*Not so much abstraction but I can imagine this is something that GameMaker can do but not at introductory level. Abstraction of any kind is a difficult concept particularly for pupils*." Although the term 'automation' was not specifically mentioned in the discussion forum, the storyboards and finished games included illustrations of code that is run repeatedly, such as the calculation of scores or lives, in which conditional loops were utilized and Boolean variables coded.

Evidence of different definitions of CT emerged through the games design process with one participant declaring "*Digital literacy to me is basic hardware IT and word processing skills. CT on the other hand can be achieved by using a computer or with a pen and paper and doing mathematical puzzles*." The model of Use-Modify-Create was supported, and even extended, by some participants – "*Use-Modify-Create for me is the process of CT. I would also include problem-solving in this list. … Using logic and understanding, the thought processes behind logic and developing deductive arguments also plays a role here*." Indeed other models of CT were also proposed, indicating an existing interest in the area – "*I tend towards the 4-pronged DPAA - decomposition, patterning, abstraction and algorithm (mainly because I can use http://games.thinkingmyself.com/ to explain it to*

my students and peers). Each was touched on here. The most lacking probably was the analysis in problem-solving, methinks in this course the debugging may have been done on the fly without much deep thinking about why it did or it didn't work. I am referring to both teachers and students in that sweeping assumption!"

When asked if they felt they were developing CT skills or digital literacy skills during the games design tasks, the former skills dominated. "*Yes, most definitely GameMaker is developing CT. Each session is low entry however it contained enough challenge to keep me engaged and sometimes to make me stop and re-think, how do I? What am I expected to do?…Students will probably experience the same*."

Analysis relating to debugging any errors in the code was inherent in all the tasks and 'rewards' were given only for successful completion of the task. Where self-evaluation and/or peer assessment were used in the online course, there was clear evidence that the learners understood the problem-solving process associated with CT and could analyze their problem-solving approach well noting areas for future improvement in their game design.

As noted already, the linear and structured nature of the online course reduced the opportunities for creativity and experimentation. As one participant noted, "*Students could complete the tasks by closely following the task sheets without necessarily thinking too hard about what they were doing or perhaps even understanding all that much about what they were doing*." The addition of a follow-up lesson in which learners designed and created a game to their own specification, with original sprites, novel backgrounds and self-recorded sounds may reveal their true capacity to think computationally. In addition, it was suggested that the use of discussion forums to share and collaborate during the games design process would assist in developing the metacognitive skills associated with CT and may promote transferable skills between programming languages, especially Scratch and GameMaker.

Features and techniques in programming

The underlying concepts of procedures, recursion and types were inherent in the event-driven code; however, the drag-and-drop nature of the environment (for GameMaker and Scratch) may require additional comments in the future to pinpoint the presence of these skills for the learner. Indeed, the drag-and-drop nature of both Scratch and GameMaker meant the syntax and semantics of the code was not an issue, but less experienced learners struggled with the syntax in Greenfoot and therefore became distracted from the CT skills being promoted.

Data types, although not identified explicitly, are more prominent in the blocks of the code in Scratch and GameMaker especially where variables have been created for scores or lives. Greenfoot, on the other hand, required an understanding of data types (for values), class types (for objects) and kinds (types of type). From a programming perspective, Greenfoot offered a greater insight into the semantics and syntax of coding, whereas Scratch and GameMaker assisted with the concepts of CT. This outcome may have an impact on teachers' choice of programming language with respect to their prior knowledge and experience of programming.

Exceptions and monitors were prevalent in the Greenfoot code, but were not highlighted using these terms in the task description. In most cases, the presence of a run-time error was the signal that debugging was required, regardless of the programming language being used. There are two approaches to programming language implementation: interpreter and compiler. Depending on the programming language chosen, the learner would have experienced one of these (for example, compiler in Greenfoot). Program analysis and program transformation were not considered in the games design courses.

Gamification

Like any game, the purpose needs to be clear to the player. In two of the online games, "*the boarding pass is a recognition of achievement at each level, each new boarding pass means you have already achieved so much and you are ready for the new challenge. The boarding card holds the key information about the challenge so students may even race each other to get them.*"

There was a general consensus that younger students are more likely to enjoy the gamification of the learning experience more than the older age group; however it was acknowledged that all ages like recognition for achievement, and a reward system, whether badges or air miles, will offer this personalized form of praise. One participant admitted that "*it was way more engaging and entertaining"* while another said *"it adds a frisson of fun to class."* However, not everyone was so enthusiastic, as one participant revealed: *"For me the gamification was more of a distraction to the task at hand … each week/lesson I had two tasks, one the gamification part and the other working in [package] though of course success at one required engagement/completion of the other".*

In terms of the types of 'rewards', suggestions that "*Making the award/reward age appropriate would help*" and it should also be directly related to the achievement, that is *"a badge system that recognises the work they have done and what they have achieved along the way to the finished project*." This approach would avoid distracting the learners from the learning objectives and might even act as positive reinforcement of the skills gained. Gender and personality were also noted as being influential in the acceptance or otherwise of the 'reward' system. "*Personality types dictate the value of the air miles as with most reward based learning. Some love a 'badge' of success at any age, others are take-it-or-leave it from a young age. I strongly believe the reward should always be offered though*." In addition, gender differences were raised, with the observation that once a game is introduced into any classroom scenario, boys become very competitive. Gamification "*provides a competitive environment, which boys especially love to work in and it helps to engage everyone in the learning process,*" but it was also noted that logistics have a role to play as "*the organisation and monitoring of rewards may prove difficult in a large school. This might be easier to manage in a primary school setting*."

The motivational role of games and hence gamification was also mentioned by the participants – "*The Boarding Pass has its uses for those who need a check list or reminder to motivate their progression.*" A distinction was made between the continuous nature of rewards and the finality of an award for successful completion – "*I would view the points as a reward and motivation factor and the certification at the end as an award*." On the latter point, it was suggested that the award/certificate be earned "*by programming a game that reached 80 points*." However, it was acknowledged that the gamification process provides the benefit of extrinsic motivation for reluctant learners and it "*gives a shared language of direction and achievement to teachers and students*." A drawback was also highlighted in terms of the underlying message the reward system is promoting – "*The gamification process reinforced this [a lack of motivation to experiment or take ownership of the learning] to a certain extent by awarding points for sticking to and completing the tasks in the task sheet*."

When asked midway through the game design process how they would feel if the air miles or points were withdrawn, the participants were in agreement that it would "*encourage students to give up and that is not the end goal of the task,*" some equating this move with traditional assessment – "*No negative marking for me - has no effect apart from demoralization*." Indeed, suggestions were even made that "*I would even allocate attempt points to those who get so far as if you can keep the pupils trying this is a skill in itself*" and, taking it further, "*the reward system needs to be extended for stretch and challenge activities, maybe an upgrade to business class or an in-flight meal / treat in keeping with the theme.*"

Discussion, conclusions and recommendations

Discussion and conclusions

The key question being addressed in this study was:

> *Are there currently teachers in schools with the capacity to teach programming to examination classes?*

In the first phase of the project, a survey was undertaken of a purposive sample of teachers who might be likely to teach programming; they are described here as 'prospective teachers of programming.'

The research aimed to evaluate these teachers' motivation, willingness, competency and self-efficacy towards programming through using a form of the Technology Acceptance Model (TAM) (Davis, 1989; Venkatesh *et al.*, 2003; Teo, 2010) adapted to focus on programming. In the second phase, online courses were designed to offer a limited group of post-primary school ICT teachers (from NI) and a mix of primary and secondary teachers (from RI) an opportunity to experience programming as a 'digital literacy' by mastering a programming language(s) through a form of games-based learning. The wider skill of 'Computational Thinking' at the heart of computer programming was embedded in the Programming Studio game plan and evaluated by the teacher participants. It should be noted that the over-emphasis on delivering the technical, product-related items without establishing the core determinant of uptake, namely user-acceptance, has led to numerous failed innovations in the field of technology (Cuban, 2001; Verdegem and De Marez, 2011). For this reason, this study focused on teachers, not the school resources.

In response to the above question and the overarching goal of supporting potential teachers of programming to make the transition from users to creators of programs, the following research questions were addressed.

RQ1 To what extent are prospective teachers of programming ready and willing to teach programming in schools?

The TAM scale scores indicate that respondents overall were moderately willing, or perhaps not unwilling, to teach programming; however, they recognised the challenges involved and did not expect to benefit in terms of enhanced status in their schools. A higher proportion of the NI than the RI cohort reported that they were already teaching programming in some form, either within the timetable or outside it; also, while about two-thirds of each cohort expressed interest in starting an after-school computer club, the NI group was more positive. This probably reflects the contrast in the existing and likely forthcoming curricular situations in the two jurisdictions, as well as the difference between the two samples.

RQ2 To what extent do the teachers see programming as an essential element of school curricula?

Responses to the questionnaire indicate that the teachers in the purposive samples – likely to be well disposed towards digital technology – were, on average, slightly positive towards programming being an *essential* element of the curriculum, giving a rather undifferentiated endorsement to its inclusion at all curricular levels. The word 'essential' was chosen to provoke strong reactions; it would have been interesting to triangulate by using a less forceful set of statements, but the length of the questionnaire was an issue without introducing additional items.

RQ3 How competent are teachers to teach programming? What programming languages are they most familiar with?

There are differing responses from the two jurisdictions in relation to their knowledge of and experience in teaching a variety of programming languages, again reflecting the differences in background of the purposive samples. For example, NI respondents – teachers of ICT – were generally familiar with Scratch and GameMaker as well as with languages associated with web use (HTML, JavaScript, Java), though at the time of responding they were not necessarily ready to teach these in school and were looking for professional development programmes in this area. Those from RI reported some familiarity with Scratch, currently the focus of attention via CoderDojo and informal professional development courses, and now addressed also through the professional development initiative to support the new Coding short course; apart from that and awareness of the web-related languages, reported familiarity was very low.

It should be recalled that the landscape is changing rapidly, especially in NI. As discussed above, the educational and economic climate in NI has forced the rapid inclusion of programming across schools; teachers have been under considerable pressure to introduce programming, notably at A level, while ICT teachers have expressed a need to embed the fundamentals of Computer Science, including programming, at GCSE and across Key Stage 3. This would account for their increased levels of awareness of the breadth of languages applicable to school-aged pupils. There has been less

development so far in RI, but the introduction of short courses into the junior cycle and the pressure for further curricular change provide a somewhat similar context. As a result, in both jurisdictions there has been a recent upturn in the availability of programming courses, offered or supported by industry and university departments alike, for upskilling experienced teachers. The survey was undertaken in Spring 2014; while there is still a concern about teachers' competence to help students learn to a standard required for high-level national certification, it is likely that further surveys would illustrate a noticeable improvement.

RQ4 To what extent does gamification support the learning experience?

In general, the use of gamification to motivate and engage the learners of the programming language fulfilled its goal with comments like "*I enjoyed the fact that each new session my Pacman game became more powerful and could do more things. If I was a student it means that I have something to show my achievement that is instant and visual*." Terms of endearment such as "*more engaging*", "*entertainin*g" and "*a frisson of fun*" were used to describe the experience. Most of the participants were positively disposed to the use of 'rewards' – whether points, air miles or badges – and some even advocated extending the system to address the gifted and talented students or to reward effort applied by the less able learners. Despite initial concerns over the 'childish' nature of the approach, the removal of the rewards at the midpoint of the course was met with disapproval. Gender differences were noted, due to the competitive nature of boys, it was anticipated that the gamification process would positively impact most on this subgroup. The aesthetic appeal of the online course was noted as attracting (and sustaining) the interest of the girls.

The navigation of the online environment was intuitive with signposts and instructions clear within the courses however feedback did indicate that the Fronter VLE and MS Office interfaces slowed the learning process and incompatibilities arose with some Mac users.

RQ5 How does the literature define 'Computational Thinking' and how do teachers view CT as a 21st century skill?

Wing (2006 in Lee *et al.*, 2011, p.32) defines CT as "*describ[ing] a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applicable in the information society.*" The cross-curricular nature of CT is noted in this definition and in the wider literature in the field. Cuny *et al.*'s (2010) definition that CT is "*the use of abstraction, automation and analysis in problem-solving*" reflects the more computer-based and technical/programming perspective of many of the NI teachers who equated CT to the requirements of the A level coursework in terms of Design, Analysis, Testing and Evaluating a solution to a real-life problem. In contrast, the teachers from RI, being from a variety of disciplines, had a wider understanding of CT more aligned to Wing's (2006) definition and gave examples from mathematics and physics, control systems and robotics, mapping and timelines, proofs and portfolios.

When asked to rank four key components of CT, in order of importance there was agreement that 'solving problems' was paramount, but 'understanding the concepts fundamental to Computer Science' caused greatest divergence of opinion, being ranked 4th by NI teachers and 2nd by RI teachers. This difference may reflect the NI teachers' insight into Computer Science based on having to introduce it as a subject in the curriculum.

In conclusion, as Lu and Fletcher (2009, p.264) declare "*To truly integrate computational thinking into current primary and secondary curricula undoubtedly presents significant challenges. It will necessarily be a gradual and evolutionary process, and requires concerted efforts and co-ordination among many constituents of the wider education community*." Nonetheless it is an achievable goal.

Recommendations

Based on the findings above, it is apparent that there is an increased level of willingness and readiness by teachers to introduce programming into schools both north and south. This is currently reflected in the rise of professional development (PD) opportunities coming from universities and industry to support potential teachers of programming.

- *Whilst such PD opportunities are making an impact on educational change currently, it should be noted that long-term commitment is required to sustain pace with the ever-changing computing industry.*
- *This commitment will require both funding and also the facility to release teachers to participate in these PD activities.*

It is not sufficient to have pupils' only programming experience in the ad hoc provision of CoderDojos, good though these can be. There should be systematic curricular provision, supported by appropriate pedagogies and assessment methods.

- *A more integrated and inclusive opportunity to learn to program is required for all students regardless of age.*
- *Programming opportunities need to be contextualized in a non-threatening and supportive environment in which learning is fun, collaborative and creative.*
- *Pedagogically creative approaches to learning programming languages are required, such as games design or app creation to motivate and engage young students to think computationally and enjoy learning to program.*
- *Greater connections may be required between Creative Learning Centres / Education Centres and informal learning spaces such as Bridge21 to facilitate these pupil experiences.*
- *There should be recognition of the practical element in programming, with suitable assessment methods being used. This has implications for examining bodies in (or affecting) the two jurisdictions, and indeed also for employers.*

In view of the drive in both jurisdictions to improve uptake of Computer Science courses, it is important that programming should be presented in a context that does not misrepresent the subject or alienate people from it.

- *Greater acceptance of the partnership between schools, university and industry is required to make students aware of the range of employment opportunities available to computer scientists.*
- *The gamification approach used in the Programming Studio courses should be piloted with students to establish if there are gender issues with regard to competitiveness, and in particular to find if such an approach might discourage girls from opting for Computer Science and related subjects.*

Computational Thinking has emerged as a cross-curricular skill with intrinsic value as well as a component of Computer Science and a key aspect in programming.

- *Teachers addressing problem solving should be encouraged to use the vocabulary associated with Computational Thinking in their own contexts. This may be relevant especially, but not only, for teachers of subjects such as Mathematics, Science, and Technology / Design Technology / Design and Communication Graphics.*

Acknowledgements

The authors would like to thank the following agency and people for facilitating the research:

- SCoTENS for funding the project;
- Students in the Master's course on Technology and Learning, Trinity College Dublin, who piloted the survey and gave valuable feedback;
- Survey participants North and South for their time, commitment and feedback;
- PGCE ICT/Computing students in Queen's University Belfast for their involvement in the online games design;
- Dedicated 'gamers' who completed the online course and engaged in the reflections on CT and gamification.

References

Baranauskas, M., Neto, N., & Borges, M. (1999). Learning at work through a multi-user synchronous simulation game. *Proceedings of the PEG '99 conference, Exeter, UK*.

Bell, T., Andreae, P., & Lambert, L. (2010). Computer Science in New Zealand High Schools. *Proceedings of the 12th Australasian Computing Education Conference (ACE 2010), Brisbane, Australia*, 15–22.

Betts, B. (2011). The 4 Pillars of Gamification. [http://www.ht2.co.uk/ben/?p=330] Last accessed July 2015.

Cooper, S., Perez, L.C., & Rainey, D. (2010). K-12 Computational Learning. *Communications of the ACM, Viewpoints* 53(11), 27–29.

Crook, S. (2009). Embedding Scratch in the classroom. *International Journal of Learning and Media* 1(4), 17–21.

Cuban, L. (2001). *Oversold and Underused.* Cambridge: MA, Harvard University Press.

Cuny, J., Snyder, L. and Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript referenced http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13 (3), 319–340.

Denning, P. J. (2003). Great Principles of Computing. *Communications of the ACM* 46(11), 15-20.

Denning, P. J. (2009). The Profession of IT: Beyond Computational Thinking. *Communications of the ACM* 52(8), 28–30.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining Gamification. *Proceedings of MindTrek 2011, Tampere, Finland.*

GCO (2012). Gamification vs. Game-Based Learning in Education. http://www.gamification.co.uk/2012/01/13/gamification-vs-game-based-learning-in-education/

Henderson, P. B., Cortina, T. J. & Wing, J. M. (2007). Computational Thinking. Proceedings of the 38th SIGCSE technical symposium on Computer Science education, 195-196.

Johnston, R. T., & de Felix, W. (1993). Learning from video games. *Computers in the Schools* 9 (2–3), 199–33.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads* 2(1), 32–37.

Liu, Y., Li, H., & Carlsson. C. (2010). Factors driving the adoption of m-learning: An empirical study. *Computers & Education* 55(3), 1211–1219.

Lu, J. J., & Fletcher, G. H. L. (2009). *Thinking about Computational Thinking*. *SIGCSE* Chattanooga, Tennessee.

MacMillan, D. (2011). *Bloomberg BusinessWeek Magazine: Gamification: A growing business to invigorate stale websites*. Viewed: 11 May 2014. Available:http://www.businessweek.com/magazine/content/11_05/b4213035403146.htm

Malone, T. W. (1981). What makes computer games fun? *Byte* 6(12), 258-–77.

Pivec, M., Dziabenko, O., & Schinnerl, I. (2003). Aspects of Games-based Learning. http://www.unigame.net/html/I-Know_GBL-2704.pdf

Raymer, R. (2011). Gamification: Using Game Mechanics to Enhance eLearning, eLearn ACM.

Resnick, M. (2012). Reviving Papert's Dream. *Educational Technology* 52(4), 42–46.

Schell, J. (2008). *The Art of Games Design.* Boca Raton, FL: CRC Press.

Teo, T. (2010). A path analysis of pre-service teachers' attitudes to computer use: applying and extending the technology acceptance model in an educational context. *Interactive Learning Environments*18 (1), 128–143.

Thompson, D., & Bell, T. (2013). Adoption of new Computer Science high school standards by New Zealand teachers. *WiPSCE (8th Workshop in Primary and Secondary Computing Education) '13, November 11–13, 2013, Aarhus, Denmark*, 87–90.

Thornton, G. C. & Cleveland, J. N. (1990). Developing managerial talent through simulation. *American Psychologist*, 45, 190-199.

Tzouvara, K., & Zaharias, P. (2013). *Towards a Framework for Applying Gamification in Education.* Cyprus: Open University of Cyprus.

Venkatesh, V., & Davis, F.D. (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science* 46 (2), 186–204.

Venkatesh, V., Morris, M.G., Davis, G.B., & Davis, F.D. (2003). User Acceptance of Information Technology: Toward a unified view. *MIS Quarterly* 27 (3), 25–478.

Verdegem, P. & De Marez, L. (2011). Rethinking determinants of ICT acceptance: Towards an integrated and comprehensive overview. *Technovation*, 31(8), 411-423.

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

Research outputs

Publication (peer reviewed)

Cowan, P., Oldham, E., & FitzGibbon, A. (2015). The Programming Studio: investigating teachers' readiness for teaching programming in the island of Ireland. In L. Leite, M. A. Flores, L. Dourado, M. T. Vilaca, & S. Morgado (Eds.), *ATEE Annual Conference 2014: Transitions in teacher education and professional identities* (pp. 299–309). Brussels: ATEE (Association for Teacher Education in Europe).

Conference presentations given

Cowan, P., Oldham, E., & FitzGibbon, A. (2014). The Programming Studio. Presented at the AUCEi (Association of Ubiquitous and Collaborative Educators International) 3rd Annual Meeting, Dublin, 20-22 July 2014.

Cowan, P., Oldham, E., & FitzGibbon, A. (2015). Breaking the code: teachers' readiness to teach programming in Northern Ireland and the Republic of Ireland. Presented at the AUCEi (Association of Ubiquitous and Collaborative Educators International) 4th Annual Meeting, Dublin, Ireland, 23-25 June 2015. [**Note**: we have been invited to submit our paper for review for the inaugural issue of the Association's journal, JUCEi.]

Conference presentations forthcoming (abstracts accepted)

Cowan, P., Oldham, E., & FitzGibbon, A. (2015). Think, Code, Succeed: the Programming Studio as a games-based collaboration. To be presented at the Annual Conference of the Association for Teacher Education in Europe, Glasgow, 24-26 August 2015.

Cowan, P., Oldham, E., & FitzGibbon, A. (2015). Factors influencing the re-introduction of Computer Programming in Schools in Ireland: a North-South collaboration using Games-Based Learning. To be presented at the Annual Conference of the British Educational Research Association, Belfast, 15-17 September 2015.

Cowan, P., Oldham, E., & FitzGibbon, A. (2015). What do practitioners know about computational thinking? A study in the island of Ireland. A contribution to the symposium "Computational thinking across the lifecourse." To be presented at the Annual Conference of the British Educational Research Association, Belfast, 15-17 September 2015.

Conference paper submitted (outcome awaited)

Cowan, P., Oldham, E., & FitzGibbon, A. Investigating Secondary Teachers' Perceptions of Learning Computational Thinking through Games Design: A case study in Ireland. Submitted for the 2016 Annual Conference of the American Educational Research Association.

Note

This report to SCoTENS draws substantially on the published paper and the abstracts for the conference presentations.