PACT:

An initiative to introduce computational thinking to second-level education in Ireland

Aidan Mooney, Joe Duffin, Thomas Naughton,
Rosemary Monahan, James Power and Phil Maguire.

{amooney, jduffin, tomn, rosemary, jpower, pmaguire}@cs.nuim.ie

Department of Computer Science,

Maynooth University, Maynooth,

Co. Kildare, Ireland.

Abstract

PACT (Programming ∧ Algorithms ⇒ Computational Thinking) is a partnership between researchers in the Department of Computer Science at Maynooth University and teachers at selected post-primary schools around Ireland. Starting in September 2013, seven Irish secondary schools took part in a pilot study, delivering material prepared by the PACT team to Transition Year students. Three areas of Computer Science were identified as being key to delivering a successful course in computational thinking, namely, programming, algorithms and computability. An overview of the PACT module is provided, as well as analysis of the feedback obtained from students and teachers involved in delivering the initial pilot.

Keywords

Computational Thinking, Transition Year, Computer Programming, Algorithms, Computability.

1. Introduction and Motivation

Maynooth University has been delivering programmes in Computer Science at undergraduate and postgraduate level for 25 years and, since 2012, has been delivering a unique *BSc in Computational Thinking*. This degree is a blend of Computer Science, Mathematics and Philosophy, and aims to provide a deeper education in the foundations of computation than the more traditional degrees, which often have a stronger emphasis on technology. The PACT initiative aims to build on the energy of this Computational Thinking initiative to bring the ideas at the heart of our degree to second-level students.

Although there have been numerous efforts over the years to introduce computational concepts into Irish secondary schools, at an official level, these never proceeded much beyond basic elements of information technology. The reform of the Junior Certificate cycle, introducing greater diversity, offers an opportunity to offer our perspective: a course designed by computer scientists to display the depth and significance of the field in a way that can challenge and engage second-level students. An emphasis on computational thinking, as opposed to information technology, allows us to explore the key concepts that we feel underlie Computer Science, without necessarily having to achieve the full rigour of the professional scientific discipline.

The NCCA has developed a specification for eight Junior Cycle short courses for use from 2014, one of which is titled *Coding* [1](NCCA, 2013). The computing content consists of basic programming skills along with some multimedia (web page) design, with a goal of developing students' teamwork skills. While the module specification provides an extensive list of learning outcomes, it will still be quite a challenge for non-expert teachers to deliver quality instruction and assessment. Furthermore, the potential is there to develop a broader understanding of the core principles of computation.

A secondary school student who is considering their choice of third-level degree options faces two problems. First, if they have never done a computing-related subject before, how can they commit to doing a degree in the area? Second, even if they have decided to try computing, how should they choose from the many degree courses available? For example, for the 2013-14 academic year, the CAO *Points Required for Entry to Courses*[2] listing shows 941 level-8 degree courses offered by a total of 41 different third-level institutions. Of these

---

1 http://www.curriculumonline.ie/Junior-cycle/Short-Courses/Coding

2 The data were assembled from information available at www.cao.ie in November 2014.

we can classify 81 degrees as being generally in the computing domain based on their name containing terms such as "computing", "information" or "digital media", or close variants. These are offered by just 23 different institutions, suggesting that, even within many third-level institutions, a student must choose which "flavour" of computing they wish to study right from the beginning. These 81 courses can be further subdivided into three (non-exclusive) sub-categories: 53 computer science or computing, 17 information technology or informatics and 18 digital media or multimedia. In fact, there are 65 different degree titles in use between these 81 courses. To someone working in the sector these degree titles may suggest quite different approaches to the subject, but it is not obvious how a second-level student might acquire or analyse this information, even supposing they had a clear preference.

The trend towards more specific degree titles has been criticised by the Irish Universities Association, which has responded by taking steps to reduce the number of separate entry routes to their courses (Nolan, 2014). What is clear, however, is that computing-related subjects suffer particularly from this problem. Furthermore, this takes place in a context where second-level students are particularly ill-equipped to distinguish between options in computing. Our approach to resolving this issue is to develop a programme focusing specifically on the fundamental principles underlying Computer Science, which can be delivered to second level students.

The PACT programme is a partnership between researchers in the Department of Computer Science at Maynooth University and teachers at selected post-primary schools around the country, with the aim of introducing students to core concepts in the discipline of Computer Science. Through PACT we have identified three key levels of understanding:

- **Programming** is a threshold concept in Computer Science, as it introduces some of the basic challenges of the discipline.
- **Algorithms** involves studying solutions in computational terms: which solutions are better, in what circumstances, and why?
- **Computational Thinking,** in scientific terms, is the study of *problems* in computational terms: what can and cannot be computed, and why?

Starting in September 2013 a number of Irish secondary schools took part in a pilot study, delivering material prepared by the PACT team to transition year students. The goal of the PACT programme is to guide students through the key topics in programming and algorithms

towards the ultimate goal of studying the process of computation via Computational Thinking.

2. Background and Related Work

In an appendix to a major report on computing, Peter Denning offered the definition of the discipline of computing as "the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, 'What can be (efficiently) automated?' " (Denning, 1989). Our PACT initiative is based on exploring some of the ways in which this fundamental question can be addressed.

The phrase *Computational Thinking* was originally coined in the context of mathematics education by Seymour Papert (1996), but came to prominence in Computer Science following an influential article by Jeannette Wing (2006). According to Wing: "Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine". Computational thinking confronts the riddle of machine intelligence: 'What can humans do better than computers?' and 'What can computers do better than humans?' Most fundamentally it addresses the question: 'What is computable?'" This perspective has been developed by the *Center for Computational Thinking* at Carnegie Mellon University[3] (sponsored by Microsoft Research), and echoed in other programmes, including those developed by Google[4] and the *International Society for Technology in Education*[5].

2.1 Previous Irish initiatives

The Lero group in the University of Limerick pioneered an education and outreach programme beginning in 2007, originally to support teaching using MIT's Scratch environment in Irish post-primary schools. While the structure of the Lero initiative is similar to ours, we believe that a restricted graphical language like Scratch is fundamentally limited in terms of teaching programming. Another alternative is the Bridge21 programme based in Trinity College Dublin and supported by Google (Conneely, 2013). The Computer

[3] http://www.cs.cmu.edu/~CompThink/
[4] http://www.google.com/edu/computational-thinking/
[5] http://www.iste.org/learn/computational-thinking

Science deliverable at present has a technological focus on programming and the Raspberry Pi, but also provides professional development workshops for teachers to train in "21st Century Computer Science Teaching Skills".

Our PACT initiative seeks to broaden the choice available to second-level students by offering a syllabus with a greater emphasis on the fundamentals of Computer Science.

2.2 Recent initiatives in the UK and USA

Related work in the UK has culminated in the replacement of ICT as a national curriculum subject at all key stages, with three distinct strands identified. The Computing in Schools[6] project, coordinated by the Royal Society, in cooperation with 18 universities and several industry bodies, emphasises the need to switch to "creative, rigorous and challenging Computer Science" (Royal Society, 2012). The Computing at School[7] initiative by the British Computer Society has produced a curriculum for schools that has been endorsed by Microsoft, Google and Intellect (Computing, 2011; Computing, 2012). Again, this curriculum clearly identifies Computer Science as a STEM discipline, distinguishes it from Information Technology, and emphasises core skills such as programming as well as more abstract skills such as designing algorithms and computational thinking.

In the USA, much of the focus on K-12 CS education is directed through the ACM-supported *Computer Science Teachers Association* (CSTA)[8] who established a *Computational Thinking Task Force* and identify the need to establish Computer Science as an academic discipline within the STEM fields (Computer, 2010; Astrachan 2012). The CS10K community[9] is a separate initiative supported by the National Science Foundation as part of its Computing Education for the 21st Century (CE21) programme. It aims to place 10,000 qualified computer science teachers into high schools to broaden access to computing education. They have developed resources in two streams: *Exploring Computer Science* covers problem solving, web design, programming and robotics, while *Computer Science Principles* has a stronger programming and algorithms focus.

In summary, the principal initiatives from the UK and USA distinguish Computer Science from digital literacy, information technology and programming, viewing it instead as the

6 https://royalsociety.org/education/policy/computing-in-schools/
7 http://www.computingatschool.org.uk/
8 http://csta.acm.org/
9 http://cs10kcommunity.org/

scientific and practical study of computation: what can be computed, how to compute it, and how computation may be applied to the solution of problems. These insights formed the foundation of our PACT programme.

3. What Parts to Teach?

If one has made the decision to teach the fundamentals of computer science at second level, then what should this cover? Many of the most salient technological advancements of our era are not fundamental, even though they may have become ubiquitous and necessary in modern society. Technologies such as word processors, hard disk drive technology, email, web apps, the internet, the cloud, and so on, will be unrecognisable after 100 years. Our approach does not view these topics as fundamental to computer science.

Computer programming is being increasingly recognised as an important skill to master in the modern world (Wing, 2006), and clearly forms a fundamental component of computer science. However, one lacuna of some computer programming courses, such as those aimed at children, is that they can proceed for weeks without students being exposed to the study of algorithms. The students can become competent programmers, creating intricate and complicated games, for example, but may not have a concept of the separation between an algorithm and its implementation. Even at third level, introductory programming can be taught independently of algorithms when the focus is on the syntax of the programming language, programming paradigms, use of standard libraries, and refining pseudocode into code. In such a scenario, students may never be challenged to compare multiple equivalent algorithms in terms of elegance, running time, or memory usage.

Conversely, it can be argued that the study of algorithms can be taught independently of computer programming, through pencil and paper exercises, thought experiments and class discussions (see Computer Science Unplugged, http://csunplugged.org/, for dozens of examples). Furthermore, computational analyses of, and mathematical comparisons between, different solutions to a problem can only be performed when the solutions are expressed as an algorithm at this abstract level, and are infeasible to perform if dealing with solutions expressed as fully implemented computer programs.

Our view is that a subject should communicate a flavour of the core of the discipline, where the core is defined as the set of topics that are unique to it and that set it apart from other scientific disciplines. Aside from programming and algorithms, two distinct topics that

set computer science apart from other disciplines are *undecidability* and *intractability*. With undecidability we can prove that there are conceptually simple computational problems that are impossible to solve with a computer. The concept of intractability highlights simple problems that can be solved with a relatively short computer program, but that so far have been impossible to solve in practice because for any practical input sizes the computer program requires millions of years to run. These topics are uniquely studied by computer scientists, and will not lose their place of importance in computer science in the future. As such we argue they should be at the core of a syllabus in Computer Science.

In sum, while programming is central to computing, the PACT programme aims to highlight how the underlying principles of algorithms and computational thinking are both more fundamental and more durable. PACT defines a process by which one can combine the study of programming and algorithms to achieve learning outcomes in computational thinking. The expanded acronym "P $\wedge$ A $\Rightarrow$ CT" is designed to illustrate our view that both programming and algorithms are distinct and necessary for the process, and that studying their combination in a particular way leads to computational thinking.

4. Pilot Study

During the summer of 2013 a number of schools were approached to gauge if there would be interest in running a pilot programme in Computational Thinking. Seven schools ultimately came on board, with at least one teacher from each agreeing to participate. Initially, the focus of the PACT partnership has been on teaching programming to Transition Year students, but ultimately the goal is to develop a framework for delivering a short course on computational thinking as part of the new Junior Certificate cycle. As a start, a flexible module was prepared which would engage Transition Year students by focusing, not on learning facts about computers, but on developing creative ideas and new ways of thinking.

The module developed consisted of five separate components (Programming 1, Programming 2, Algorithms, Graphics and Recursion). Teachers were invited into to participate in a two-day training programme. The first day involved a lecture style overview of the aims of the course, with teachers introduced to a Virtual Learning Environment containing the resources developed. These resources included lecture notes in presentation style format, practical exercises to work on with students, online resources which the teachers would find helpful, and an interactive website allowing students to step through programming

material in an interactive fashion, with computer programs generated and run online (see http://pact.cs.nuim.ie/). The second day was spent carrying out programming tasks and practical exercises in all of the components of the module. In addition, an interactive website has been created since the training.

Approximately 320 students participated in the PACT programme starting in September 2013. In line with the new decentralized model for the Junior Cycle, teachers had full responsibility for delivering the material in their schools as they saw fit. Upon completion of the module all teachers and students were asked to complete questionnaires. Eight teachers representing six schools completed the teacher feedback questionnaire, while 61 students representing four schools completed the student questionnaire.

4.1 Teacher Survey

The feedback revealed that, on average, 1.4 hours were spent per week delivering the PACT programme. All classes studied the Programming 1 section, while four of the schools represented responded that they had also delivered Programming 2 and Algorithms. Only a single class group completed all five sections of the programme. The teachers were asked to select the sections of the programme that they felt engaged the students the most. All but one teacher identified that Programming 1 as engaging for the students, with Graphics being the next most popular and Algorithms in third place. Teachers identified that they were able to get further with the material after they had already taught the module to multiple groups. One teacher found that grounding the concepts through visual programming was most effective: *"I taught 3 classes for 12 consecutive weeks each so I learned as I went along. I got furthest with the third group. I also learned to sequence topics differently. I found that teaching the basics through turtle graphics programs was the best approach"*.

Under the new Junior Cycle initiative, it is intended that teachers will grade their own students. However, we found that the majority of teachers requested a formal accreditation or qualification to motivate their students: *"TY students are difficult to motivate. The idea that they would get a formal qualification would definitely give them more incentive during TY. It would also ensure that classes were not missed due to other TY events taking priority"*.

The familiar problem of significant differences in student programming ability (see Maguire et al., 2014; Mayer, 2013) also arose, with teachers suggesting stratification of material: *"Varied the lessons for mixed ability classes. Some students have a high level of*

computer knowledge while others can barely turn a PC on!" In contrast to the PACT module's focus on core theoretical concepts, several teachers requested a more grounded approach: *"Give more examples of how Python can be used in real life. The pupils were very keen to learn how programming was being used in the working world"; "When the course is being presented to the general cohort, it will need to be as interactive as possible and the outcomes need to be visual and stimulating to the general group"*

4.2 Student Survey

Approximately two-thirds of the 61 students that completed questionnaire viewed the programme as having been beneficial. Positive feedback focused on the broadening of career options: *"I learned that I really enjoy coding and hope to be able to use it in the future and pursue a career in it"*; *"I believe that by having a background in computer programming is not only brilliant to have on a C.V., it is also a really good skill to have as workplaces are very much becoming more and more technologically based"*. The other third that did not view the experience positively cited the difficulty of the material: *"I don't feel I've gained from this experience, it's confused me quite a lot"*; *"I found it very hard to keep up with my classmates"*. Several students reported that the lessons were too rushed: *"It seemed very rushed and I had problems understanding how to proceed"*. The gap in ability between students also caused problems: *"Since not everyone in the class was that interested it got very hard for the people that were interested to actually learn"*.

There was a near universal consensus regarding the divergence between the PACT module and other subjects taken in school, with 60 out of 61 viewing it as different. One aspect noted was how learning was more collaborative, with teachers depending more on peer learning than in traditional subjects: *"I enjoyed the way we could consult and help each other"*. In line with the teachers, some students felt that a more grounded approach would have been better: "*I think that the questions that are posed to do should be a bit more practical and try to give the user more of a feeling of what the computer is actually doing"; "It just wasn't for me really. One way I think it could be improved would be if there was some kind of game involved rather than just doing exercises all the time"*

5. Conclusions and Future Work

In conclusion, we found that there was considerable demand for an expert led course in Computer Science, with additional schools requesting to join the PACT program after its first year. Both schools and teachers were positive about learning these skills, with teachers willing to travel to Maynooth University on a Saturday for tuition, despite the lack of associated CPD credits. Those without a formal Computer Science background found it a challenging subject to teach, but they grew in confidence with repeated deliveries. The materials and support provided to teachers by the PACT team were critical in allowing them to explore more advanced, fundamental topics than are traditionally covered by second level computing courses.

Students found the material challenging, and both students and teachers reported that a less theoretical and more practical approach might have been more helpful for introducing key concepts. Computer Science presents a completely different way of thinking, which can help students strengthen their creativity, precision, and reasoning, and contribute towards a well-rounded education. However, the difficulties experienced by some raises the question of whether deep concepts such as computational thinking are best presented theoretically up front, or whether they should be ingrained more gradually through extensive applied practice. Although Sternberg (2001) advocates exposing students to deep theoretical concepts, he highlights that students and teachers can be reluctant to recognise the value of knowledge that is less amenable to explicit testing. Further exploration is required for us to answer this question. In particular, assessing the depth and quality of the learning stemming from the PACT module, perhaps through the establishment of an objective certification of competence as requested by teachers, is a critical future goal of the project. The problematic divergence in aptitudes between class members suggests that Computer Science education might be best introduced at a much earlier stage of the school curriculum, before such differences have the chance to emerge.

Over the next few years we intend to develop the material into a full Junior Cycle short course, building on our experience of running the world's only undergraduate degree in Computational Thinking. The network of participating schools will be expanded and teachers will be encouraged to share the material they develop with each other. We hope that this collaborative initiative will ultimately result in an agile course focused on the theoretical principles of Computer Science, which incorporates the valuable insights of teachers and their students.

Acknowledgement:

The authors of this paper would like to acknowledge the work and commitment of the teachers involved in piloting this programme, as well as the students that took part.

The authors would also like to acknowledge the work of Keith Nolan in compiling the survey results.

References

Astrachan, O., & Briggs, A. (2012). The CS principles project. *ACM Inroads*, *3*(2), 38-42.

Computing at School Working Group (2011). Computing at school: International comparisons. http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf

Computing at School Working Group (2012). Computer science: A curriculum for schools. http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf

Computer Science Teachers Association (2010). Running on empty: The failure to teach K12 computer science in the digital age. http://www.acm.org/runningonempty/

Conneely, C., Murchan, D., Tangney, B., and Johnston, K. (2013). 21st century learning - teachers and students experiences and views of the Bridge21 approach within mainstream education. In *Society for Information Technology & Teacher Education International Conference*, pages 5125-5132.

Denning, P. (ed.) (1989). Computing as a discipline, *Communications of the ACM*, 32(1), 9-23.

Maguire, P., Maguire, R., Hyland, P., & Marshall, P. (2014). Enhancing collaborative learning using pair programming: Who benefits?. *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education*, *6*(2).

Mayer, R. E. (Ed.). (2013). *Teaching and learning computer programming: Multiple research perspectives*. Routledge.

NCCA (2013). Programming and coding: Draft specification for Junior Cycle short course. http://www.juniorcycle.ie/Curriculum/Consultation/ Short-Courses/Programming-and-Coding

Nolan, P. (2014). The CAO system must change, but reform must be careful. *Irish Independent*, 11/08/2014.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Royal Society (2012). Shut down or restart? The way forward for computing in UK schools. http://royalsociety.org/education/policy/ computing-in-schools/

Sternberg, R. J. (2001). Why schools should teach for wisdom: The balance theory of wisdom in educational settings. *Educational psychologist*, *36*(4), 227-245.

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