PACT:

Programming ∧ Algorithms ⇒ Computational Thinking

Aidan Mooney, Joseph Duffin, Rosemary Monahan,
Thomas J. Naughton, and James F. Power

July 6, 2014

1 Introduction

The PACT programme is a partnership between researchers in the Department of Computer Science at NUI Maynooth and teachers at selected post primary schools around the country. 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.

Initially the focus of this partnership is 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.

2 Background and Motivation

NUI Maynooth 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 computing than the more traditional degrees, which often have a strong vocational orientation. The PACT initiative aims to build on the visibility of the *Computational Thinking* brand to bring the ideas at the heart of this degree to second-level students.

What is Computational Thinking? The phrase *Computational Thinking* was originally coined in the context of mathematics education by Seymour Papert [1], but came to prominence in Computer Science following an influential article by Jeannette M. Wing [2]. Wing's article described computational thinking as

> "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[1] (sponsored by Microsoft Research), and echoed in other programmes, including those by Google[2] and the *International Society for Technology in Education*[3].

Crucial elements of this perspective are that:

- Computational thinking is a way that humans, not computers, think.
- Computational thinking is a fundamental skill for everyone, not just for computer scientists (see Appendix A for examples).

Computational Thinking in Schools. There have been numerous efforts over the years to introduce computational concepts into Irish secondary schools but, at an official level, these never proceeded much beyond basic elements of information technology. The reform of the Junior Certificate cycle offers an opportunity to get this right: a course designed by Computer Scientists to display the depth and beauty of the field in a way that can challenge and engage second-level students. An emphasis on *Computational Thinking* allows us to explore the key concepts the underlie Computer Science, without necessarily having to achieve the full rigour of the professional scientific discipline.

We can identify three key levels of understanding in Computer Science:

- **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?
- **Computability** is the study of *problems* in computational terms: what can and cannot be computed, and why?

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 itself.

The PACT group at NUIM has collaborated with teachers across 9 secondary schools to develop a flexible module which will engage Transition Year students. The focus of the module is not on learning facts about computers but on developing creative ideas and new ways of thinking. Continuing feedback will be used to expand the module into a full Junior Cycle short course.

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

3 Related work

The NCCA has developed a specification for eight Junior Cycle short courses for use from 2014, and one of these is titled *Programming and Coding* [3]. This is a useful document as it provides an example of addressing the general skills requirements of a short course. The computing content seems to consists of basic programming skills along with some multimedia (web page) design, with a goal of developing students teamwork skills.

There is, however, no mention of a broader Computational Thinking agenda, nor any clear indication that the module is even intended to be scientific in nature. From a Computer Science perspective this curriculum is quite limited in scope (just as its title would suggest), and we would seek to develop a much broader understanding of the core principles of our discipline.

Other initiatives similar to PACT: There are at least two current initiatives in Ireland that are similar in some respects to PACT:

- The Lero group in the University of Limerick established an education and outreach programme starting in 2007, originally to support teaching using MIT's Scratch environment in Irish post-primary schools. They have developed a website[4] with support for primary and post-primary schools, including extensive lesson plans and teaching material, and run teacher training sessions as well as an annual competition.

  While the structure of the Lero initiative is similar to ours, we believe that a graphical language like Scratch is fundamentally limited in terms of teaching programming, even though it may serve as a useful introduction to some programming concepts. Further, the materials seem to be limited to teaching programming in Scratch, and do not seem to have a broader agenda in Computer Science or Computational Thinking.

- Bridge21[5] is an education programme based in Trinity College Dublin and supported by Google. It claims to be based on a new model of learning that can be adapted for use in secondary schools [4], and they provide professional development workshops for teachers to train in "21st Century Computer Science Teaching Skills".

  This initiative is directly targeting the post-primary (and Junior cycle) area, but appears to be more broadly based than our initiative, recently offering support for English, Maths and History classes. While the specific Computer Science deliverable is vague, it seem to be based at present on programming (and the Raspberry Pi), and it is not clear what their precise CS agenda is.

Even though both of these groups are established in the field, we believe that their focus in each case lacks a clear, distinctive Computer Science perspective. Our PACT initiative can fill this gap and, we believe, offers an alternative that provides a deeper insight into the fundamentals of our discipline.

[4] http://www.scratch.ie/
[5] http://www.bridge21.ie/

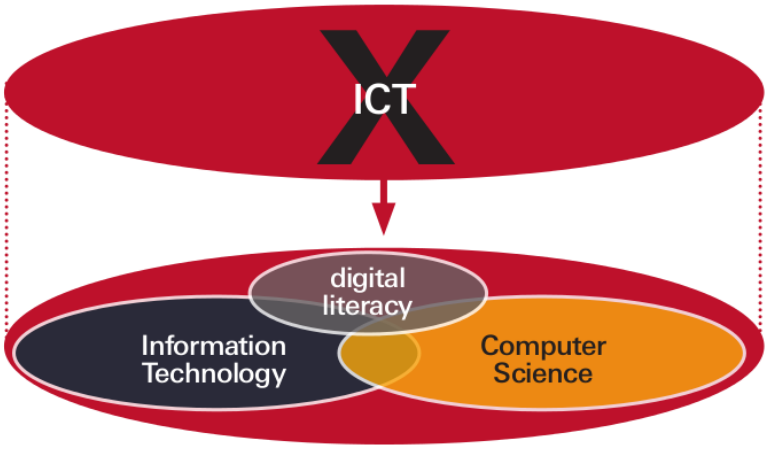

Figure 1: Replacing 'ICT': suggested terminological reform from the *Computing in Schools* initiative, emphasising the distinctive identity of Computer Science.

Recent initiatives in the UK: There has been a series of interesting related work in the UK culminating in a series of reports over the last two years.

The *Computing in Schools*[6] project was an initiative coordinated by the Royal Society in cooperation with 18 UK universities as well as several industry bodies culminating in a report published in January 2012 [5]. It notes the limitations of the previous National Curriculum in ICT, and emphasises the need to switch to 'creative, rigorous and challenging Computer Science'. It identifies that Computer Science is a rigorous academic discipline and that significant investment in continuing professional development for teachers will be needed.

Their suggested 'terminological reform' encapsulates the understood position of Computer Science from our perspective, but is nonetheless worth replicating here since it is not widely understood outside the discipline (Figure 1). While much of this document is discusses long-term funding needs and is thus directed at policy-makers, the emphasis on depth and rigour is similar in spirit to our PACT initiative.

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 an d published in March 2012 [6]. This curriculum develops a clear agenda for computing-related concepts, stratified in five stages that cover primary and post primary education. It 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. While this *Computing at School* is a much more ambitious agenda that the PACT initiative, it shares many of the themes and goals of PACT.

The summary of key points from a related *Computing at School* briefing note published in November 2011 [7] is worth quoting in full, since it covers many of the key points that have informed the PACT initiative (see Figure 2).

[6] https://royalsociety.org/education/policy/computing-in-schools/
[7] http://www.computingatschool.org.uk/

- It is vital to make a clear distinction between Computer Science as a rigorous subject discipline on the one hand, and IT applications and/or digital literacy on the other.
- Many countries, from the USA to India, routinely make Computer Science available to young people at school from an early age.
- There is increasing clarity that "Computer Science" means a lot more than "Learn to program in Java or C++". Programming is central to computing, but the underlying principles of algorithms, data structures, and computational thinking skills are both more fundamental and more durable.
- The confusion between Computer Science and ICT skills means that ICT is often delivered by non-specialists. Meanwhile, the continuing strong employer demand for IT professionals reduces the supply of well-qualified potential teachers. These factors conspire to mean that ICT/Computing teachers are under-valued and under-qualified. There is a desperate need for teacher training in Computer Science.

Figure 2: Key points identified by the *Computing at Schools* group based on a study of the international experience with teaching computing in schools (direct quotation from [7]).

Computer science education includes the following elements: design (both software and hardware), creation of digital artifacts, abstraction, logic, algorithm development and implementation, programming paradigms and languages, theoretical foundations, networks, graphics, databases and information retrieval, information security and privacy, artificial intelligence, the relationship between computing and mathematics, the limits of computation, applications in information technology and information systems, and social impacts of computing.

Figure 3: Elements of Computer Science education for K-12, as identified by the CSTA (direct quotation from [8]).

Recent initiatives in the US: In the US, much of the focus on K-12 CS edutaion is directed through the ACM-supported *Computer Science Teachers Association* (CSTA)[8]. In their 2010 report they identify gaps in state standards for CS education, as well as major terminological confusion between digital literacy, information technology and computer science [8]. They identify the need to establish Computer Science as an academic discipline within the STEM fields; the key topics they identify in CS education are shown in Figure 3. The CSTA has established a *Computational Thinking Task Force* to collect and disseminate teaching resources and projects in this area.

The CS10K community[9] 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

[8] http://csta.acm.org/
[9] http://cs10kcommunity.org/

solving, web design, programming (using *Scratch*) and robotics, while *Computer Science Principles* has a stronger programming and algorithms focus (and uses *App Inventor*, a Scratch-like environment for Android app development). The CS10K Project is supported by the National Science Foundation as part of its Computing Education for the 21st Century (CE21) programme.

As a further development, *Computer Science: Principles*[10] is a proposed AP course being developed by the College Board, descigned as an "introductory college-level course for everyone" [9]. It emphasises what it calls *computational thinking practices*, such as developing computational artefacts, creativity, abstraction and communication.

Each of these courses is considerably broader than that proposed by our PACT initiative, but many of the topics overlap, and it should be possible to reuse some of the rich collection of resources they have developed.

In summary, the principal initiatives from the UK, USA and other countries emphasise:

- the need to reform CS education at post-primary level,
- the need to distinguish clearly between digital literacy, information technology and CS,
- the need to teach CS concepts as a rigorous academic discipline within the STEM subjects.

4 Our work to date

During the Summer of 2013 a number of schools were approached to gauge if there was interest in running such a pilot programme. The schools which came on board were:

- Maynooth Post Primary School, Co. Kildare.
- Pobalscoil Neasain, Baldoyle, Dublin 13.
- Confey Community College, Leixlip, Co. Kildare.
- Castelcomer Community School, Co. Kilkenny.
- Scoil Mhuire Community School, Clane, Co. Kildare.
- Jesus and Mary College, Goatstown, Dublin 14.
- Salesian College, Celbridge, Co. Kildare.

At least one teacher from each of these schools agreed to participate in the pilot programme.

Teacher training sessions: The PACT team began to structure the content for the pilot programme and on the 25th of May and the 1st June 2013 two training sessions took place in NUIM. These sessions involved introducing the teachers to the content within the programme and also getting to know each other. It was hoped that a community of knowledge and learning would be created between all the teachers as they progressed with the pilot.

A Moodle system was hosted in the Department of Computer Science to store the content that was delivered in the training sessions. The content was divided in to five components, namely:

1. Introduction to Python I.

[10] csprinciples.org

2. Introduction to Python II.
3. Algorithms.
4. Graphics.
5. Recursion and self-reference.

The content was based on the *Python* programming language[11]. *Python* is a dynamically-typed programming language that is popular in the CoderDojos as a first programming language, but also is used in a number of professional and scientific applications. The choice of programming language is not essential for the syllabus, and we envisage providing options for teachers who wish to develop using other languages such as *Processing* or *Haskell*.

Components 1 and 2 of the training sessions covered the basics of installation of *Python* and also looked at how to create and run programs written in *Python*. They looked at the main features of the language and gave sample programs for the teachers to use. An exercise book was created for all lessons within these sections which the teacher could use in class to get their students working in *Python*. In total 19 PowerPoint presentations were prepared for these sections which the teacher could break up into whatever format they felt worked for them.

Component 3 focused on algorithms and the process of analysing problems to devise abstract computational solutions. It showed how to write (using pencil and paper) step-by-step procedures to solve some classic problems in Computer Science, framed as problems from domains such as board games, cryptography, and bioinformatics. Component 4 looked at generating simple 2-D graphics using the *Pygame* set of modules. This section focused on allowing the participants to create fully featured games in the *Python* language, and can provide a useful focus for project-based programming work. Component 5 covered topics related to recursion which has been defined as "the process of repeating items in a self-similar way" and is fundamental to the core theory of Computer Science.

5 Feedback

Towards the end of the 2013-14 academic year the PACT team organised for surveys to be made available for teachers who taught the PACT material [12] and also for the students who took the module [13].

To date there have been 62 student responses and 7 teacher responses. These 7 teachers represented 5 schools within these 5 schools 294 took the module across 13 class groups. There was one all-girls school, one all-boys school and three mixed schools. It is clear from the teacher responses that unless there is complete support and buy-in from the management of the school courses like this will not survive. One teacher commented that "The pupils were very keen to learn how programming was being used in the working world", something that the PACT team has tried to incorporate in to the module.

[11] https://www.python.org/

[12] https://docs.google.com/forms/d/1-17vL9ZxeTVIKfJcsNm0rMDiJsISttpcA2Cm5VQVLvk/viewform

[13] https://docs.google.com/forms/d/1_l0fmD1C9aLeD7pwdl5a9Im4Or3ZzuvHe6MfsD5KOs0/viewform

These figures will need to be updated when all submissions are received.

No streaming took place in any of the class groups which took this module. However, one teacher noted that there was "No streaming, but the students choose to take the course. 2/3 of them were ordinary level maths students and they could not get their head around the logical thinking required to program and they ended up moving over to Scratch instead". All of the classes covered Python 1 and Python 2 with most also covering Algorithms with some covering Self-Reference and Graphics.

Initial analysis of the feedback suggests that almost 65% of the students found that "taking part in this pilot programme has been beneficial to you"? 80% of the students who replied to the survey said that this module was very different compared to other subjects they have taken in school.

Asked as to what the students enjoyed in the module some of the positive responses were:

- "It was different to other things on the computer".
- "I enjoyed being able to tell the computer to do something and it would do it".
- "Learning the basics of coding".
- "I most enjoyed writing codes that had a visual result like the turtle code".

When the students were asked to provide some overall feedback on the programme and any suggestions on how this could be improved, some of the responses were:

- "I feel it should be continued as it seemed very beneficial and helpful for the other lads but 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.".
- "It was very enjoyable and different. Sometimes it was very hard but extremely good and beneficial".
- "It was good to learn how to programme/ make programmes (sic).".
- "I thought the whole program was good, and I did look forward to each class. I think that the questions that are posed to do should be a bit more practical and try give the user more of feeling of what the computer is actually doing. Example of where I previously learned code can be seen in Lua, from ComputerCraft. There is a ComputerCraft Edu that teaches people lua. This could be used to show people how code works and impacts better that through the IDLE.".

6 Future Work

As of the 15th May 2014 we have already received interest from the following schools:

- Tallaght Community School, Dublin.
- Sandford Park, Ranelagh, Dublin.
- St. Munchin's College, Limerick.
- Beneavin College, Finglas, Dublin.
- Coliste Cois Life, Lucan, Dublin.
- St. Wolstans Community School, Celbridge, Kildare.
- Colaiste Chiarain, Leixlip, Kildare.

- St. Mel's College, Longford.

The majority of these schools have approached us directly from seeing our webpage or from teachers who are involved in the pilot programme this year. We have also distributed letters to 10 schools based on their proximity to NUIM that are not currently enrolled in PACT. We hope to run this programme again for the academic year 2014-15 with a larger number of schools and take on board the feedback received from the teachers and students in the first year.

During the Summer of 2014 a student intern will be working with the PACT team to develop further resources for teachers, develop a wiki with resources for students and analyse possible future developments for PACT, amongst other things.

7 Future funding initiative

The Erasmus+ Programme which commenced on the 1st January 2014 offers the opportunity for Strategic Partnerships for organisations involved in school education. As outlined in Section 2, other Irish and international initiatives in teaching basic programming skills and multimedia design, aim to develop student teamwork and communication skills. While these initiatives are welcome, they are limited in that they do not address the broader principles of Computer Science. Initiatives to introduce Computer Science to schools in Europe and in the USA have concluded that there is a need to reform Computer Science education at post-primary level to promote both Computer Science and digital literacy.

The NUIM PACT programme provides the ground work for such a reform in Computer Science. It places NUIM in the ideal position of leading an Erasmus+ Strategic Partnership (under Key Activity 2) which would be funded for 3 years. These partnerships focus on activities designed to improve education provision across the participating countries, promoting cross-sectoral cooperation and addressing policy objectives, challenges and the needs of a specific field such as post-primary and third-level education. Through such a strategic partnership with schools, universities and industry, we will be empowered to implement joint initiatives on programme curriculum and teaching training, as well as promoting exchange of experience and know-how. In 2014, the allocation of funding in Ireland for such a partnership was €450,000. It is expected that the funding available in 2015 will be increased and the application deadline will be in April 2015.

8 Contact Us

The PACT team can be contacted at the following email address: pact@cs.nuim.ie.

The member of the Computer Science department currently involved in the PACT initiative are: Aidan Mooney (coordinator), Susan Bergin, Joe Duffin, Phil Maguire, Rosemary Monahan, Tom Naughton and James F. Power.

Our website is: http://www.cs.nuim.ie/pact.

References

[1] Seymour Papert. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1):95–123, 1996.

[2] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, March 2006.

[3] NCCA. Programming and coding: Draft specification for Junior Cycle short course. http://www.juniorcycle.ie/Curriculum/Consultation/Short-Courses/Programming-and-Coding, October 2013.

[4] C. Conneely, D. Murchan, B. Tangney, and K. Johnston. 21 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, March 2013.

[5] The Royal Society. Shut down or restart? The way forward for computing in UK schools. http://royalsociety.org/education/policy/computing-in-schools/, January 2012.

[6] Computing at School Working Group. Computer science: A curriculum for schools. http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf, March 2012.

[7] Computing at School Working Group. Computing at school: International comparisons. http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf, November 2011.

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

[9] Owen Astrachan and Amy Briggs. The CS principles project. *ACM Inroads*, 3(2):38–42, June 2012.

A Appendix: Computational thinking outside of computer science

We argue that examples of computational thinking problem instances can be found in everyday life, at home and in the workplace. Not only is it essential for computer scientists, and advantageous for those in senior management and senior technical positions, but it may indeed be useful for everyone to hone this innate high-order cognative skill. We present the following examples to illustrate.

A.1 Management

In the workplace, the job description of each level of management can be shown to require computational thinking. Management implies managing resources. One aspect of computational thinking is deciding how to most effectively and efficiently manage ones resources to solve a particular problem; a computer programmer applies this skill every day to most efficiently ultilise her computer machinery. While it might be a rare management skill to know the ability and temperament of the human resources at one's disposal, employing those resources in the most efficient way, too, is a classical computational thinking problem instance. Finding a way to quantify abstract resources, such as those related to human resources, is a particularly human skill that must be applied before computational thinking. However, measures such as time, money, speed, volume are already in a natural form for computational thinking. At its most mathematical, computational thinking would allow a manager to prove that given a particular set of resources, a solution to a particular task is impossible, possible, or optimal.

A.2 Route planning

Consider a day's list of errands that includes dropping off the family cat to the vet, collecting dry cleaning, buying fresh meat at the butcher's, buying frozen food at the supermarket, buying postage for a letter, and browsing for a new book of bedtime stories at the local library. Consider constraints such as the cat cannot be brought into the butcher's, you cannot carry both the cat and the dry cleaning at the same time, and as soon as you purchase the frozen food you must return home. Can you complete this list of errands in one trip? If one has ever planned a list of errands such as this into a single trip, one has practiced computational thinking. Further, given additional constraints such as knowledge of the travel time between each pair of destinations, and that the library closes at 13:00 for lunch, can you complete your list of errands in time for a 14:15 tennis match if you leave home at 12:45? Again, this is a classic computational thinking problem instance.

A.3 Sorting

Consider a person who arbitrarily picks two socks to wear each morning from a closet drawer containing individual socks. This drawer was originally full of indentical individual black socks, but after many weeks of wearing, washing, and bleaching in the sun, the socks exhibit a nonuniform range of different shades

of black. A sensible person would probably ignore these shades and continue to pick any two socks each morning. However, if one *did* try to match socks by colour as best one could (or 'good enough' according to some threshold of colour difference), no matter what one's system was, one would be practicing computational thinking. At its most mathematical, computational thinking requires one to analyse the result of one's computational thinking, to determine whether one's sock matching system was optimal or not.

A.4 Summing

Consider a schoolteacher who wishes to divide the class into two basketball teams of equal ability. The schoolteacher knows the skill level of each of their pupils. If the schoolteacher genuinely attempts to solve this problem, no matter what system they choose, they are practicing computational thinking.

A.5 $\mathcal{NP}$-complete problems

A University has n clubs and societies, the largest of which contains m members. Each students can be a member of multiple clubs and societies. The President of the University wishes to hold a dinner in honour of such student activities. Unfortunately, the Grand Hall can seat comfortably only k guests. The President's problem is as follows: can she construct a guest list of k students such that every club and society has at least one member in attendance? Solving this problem is an exercise in computational thinking. It is not known if it is possible to develop an efficient system that solves this problem for arbitrary values of n, m, and k. If one finds an efficient system, and makes it public, each large corporation worldwide would shave percentage points off its running costs. If one proves such an efficient system cannnot exist, the Clay Mathematics Institute will give a prize of US$ 1 million. Characterising the true difficulty of problems such as this clubs and societies problem (more formally, the hitting set problem) is one of the hardest tasks for computer science: the famous $\mathcal{P} = \mathcal{NP}$ question. Timetabling classes in a school, scheduling appointments in a hospital, optimising a fleet of delivery trucks, air traffic control, and so on, are mathematically identical problems to the hitting set problem; by which we mean that solving any one immediately solves each of the others. Therefore, we can say that instances of this one family of problems have to be solved in society every day, and we don't know if we could be doing it better.