Overcoming Obstacles to CS Education by Using Non-programming Outreach Programmes

Tim Bell[1], Paul Curzon[2], Quintin Cutts[3],
Valentina Dagienė[4], and Bruria Haberman[5]

[1] University of Canterbury, Christchurch 8041, NZ
tim.bell@canterbury.ac.nz
[2] Queen Mary University of London, London, E1 4NS, UK
paul.curzon@eecs.qmul.ac.uk
[3] University of Glasgow, Glasgow, G12 8RZ, Scotland
quintin.cutts@glasgow.ac.uk
[4] Vilnius University, Faculty of Mathematics and Informatics,
Naugarduko str. 24, Vilnius LT-03223, Lithuania
valentina.dagiene@mif.vu.lt
[5] Holon Institute of Technology, Holon,
Israel, and Davidson Institute of Science Education,
Weizmann Institute of Science, Rehovot 76100, Israel
bruria.haberman@weizmann.ac.il

Abstract. Formal Computer Science curricula in schools are currently in a state of flux, yet there is an urgency to have school students exposed to CS concepts so that they can make informed decisions about career paths. An effective way to address this is through outreach programmes that can operate outside or in conjunction with the formal education system. We compare 5 successful programmes. Each downplays programming as a pre-requisite skill for engaging with Computer Science ideas. This makes them accessible in short bursts without formal curriculum support. The formats used include contests, shows, magazine articles, and resources for teachers. We compare the 5 approaches to draw out key ideas for successfully addressing a school student audience. This can be used as the basis for designing new outreach programs.

Keywords: CS Education, Informatics Education, K-12 outreach, Information Technology, Computational Thinking.

1 Introduction

While formal school curricula around the world are gradually introducing Computer Science (CS) as a subject, there is an urgency to get school-age students interested in the topic, and consequently many outreach programmes have emerged that either work outside the school system, or supplement what is available in schools. Here we compare five such approaches to introducing Computer Science to high school age students. An important common feature of these is that none assumes programming as a preliminary or pre-requisite topic – they enable students to engage with concepts

I. Kalaš and R.T. Mittermeir (Eds.): ISSEP 2011, LNCS 7013, pp. 71–81, 2011.

from Computer Science without having to first learn how to program. In a formal school programme there would be time to develop programming skills and provide good computing resources, but outreach programmes must necessarily make do with limited time with students and whatever facilities happen to be available.

Teaching CS without programming is achieved in a variety of ways including off-line kinesthetic activities, problem solving challenges that involve computational thinking, engaging magazine articles that present ideas from Computer Science, and mentoring by experts. All the approaches described here started as extracurricular outreach initiatives independent of formal school curriculum and constraints, thus there was a lot of freedom and space for imagination in their design. They also provide "grass-roots" trials of approaches for introducing students to CS, and the successful elements of these "trials" can later be absorbed into formal curricula. The approaches discussed here have all had widespread adoption and influence (typically tens of thousands of students) so there is potentially much to be learned from exploring their commonalities and differences. Our aim for this comparison is to compare and contrast existing successful approaches, drawing out common features and themes to give guidelines for the design of future initiatives.

There are three main motivations for downplaying the role of programming when first introducing students to CS. (1) In the context of outreach where a relatively short time is available to interact with students (often just a single short lesson), there isn't time to teach programming. (2) By engaging students without requiring them to learn to program first, a potential barrier is removed that could deter students from pursuing Computer Science. Some students will enjoy CS concepts such as problem-solving more than programming, and by engaging with those concepts first they will have more motivation to learn programming, which they are likely to encounter as a prerequisite for studying Computer Science formally. (3) A non-programming approach is a practical way to engage students in Computational Thinking [13]. Students thus gain benefits beyond a computing career.

In all the approaches described here, the key is that programming isn't the central element. Eventually students will likely need to learn programming, but it can be *after* students become engaged.

In the following sections we first describe each programme. In Section 7 we then provide a classification tool to highlight their similarities and differences which can be used to provide guidance on building a successful programme.

2 Bebras

Many competitions in computing and IT are intended for very talented students and focus on areas such as developing algorithms and programming. The Bebras competition instead has students solve problems from a broad range of areas without programming [6, 8, 9]. Because many students enjoy competition, such contests at school can be used to attract students to the domain covered by the contest.

The idea of a competition based around informatics and computer fluency for a wide population of high-school students started in Lithuania in 2003. It was named "Bebras" ("Beaver" in English) after the hard-working, persistent, intelligent, and lively animal. The main goals of the project are to promote students' interest in

informatics (i.e. Computer Science) and Information and Communication Technology (ICT) from the start of their school career, to motivate students to learn and master computers, and to engage in computational thinking [6, 7, 13]. The contest is for all lower and upper secondary school pupils, divided into four age groups. Students have to solve 18 to 27 tasks on different levels within 45-60 minutes, entering answers via computer. They do not require prior topic knowledge, but do require students to be able to reason with common structures in the CS/informatics canon.

The tasks involve concepts such as algorithms (sequential and concurrent); data structures (heaps, stacks and queues, trees, and graphs); modeling of states, control flow and data flow; human-computer interaction; and graphics. Students do not study these topics formally, instead, the topics are introduced implicitly by having the students attempt imaginative tasks. A "narrative cover story" is used to relate the tasks to an underlying topic.

More than 10 countries now participate in Bebras. Since the contest is now international, one specific challenge is to find a balance between national and global standards for the contest. Hence, discussion on common standards and tasks suitable for all countries takes place at annual international workshops. A shared collection of tasks is developed including mandatory tasks to be included by all countries in their contests; additional tasks can be added to this to adapt the competition to the educational framework of each country. Surveys and informal feedback reported by the organizers in different countries suggest that the contest motivated students to get to know computer science and information technology better.

3 CS Unplugged

CS Unplugged [2] provides a variety of resources that engage students in Computer Science activities without using a computer. Instead of programming ideas from Computer Science, students interact with them through magic tricks, games and puzzles. Most activities have a strong kinaesthetic component and take a constructivist approach: students are given enough clues so they can work out principles themselves. A constructivist approach, where students are guided with leading questions so that they can discover CS principles for themselves, is important for outreach as it demonstrates to students that they could have invented much of the knowledge themselves; and of course, this is a lot more engaging than simply being told impressive facts.

The CS Unplugged resources are available for free download; as well as activities with specific guides for the presenter, there are videos demonstrating the activities or providing challenges to students, and links to extensive related material for follow-up. The activities have been translated into over a dozen languages. Another format for the material is a one-hour show [1], designed for a broad coverage of CS topics in a short time, rather than in-depth work by the students.

CS Unplugged is used in many situations around the world, generally relating to one-off events and visits, but increasingly as components of teaching programmes. It started around 1992 as a collection of ideas for outreach from universities to K-12 schools. Through its inclusion in the ACM K-12 curriculum in 2003 it started to be seen as the basis of a new approach for teaching CS in schools, either as the main

material, or a supplement providing a break from being in a computer lab. It is now used widely in a variety of situations, including outreach, clubs, summer camps, and regular classrooms.

4 cs4fn

cs4fn [4] uses research topics to spark enthusiasm in students about Computer Science, providing them with ways to learn more about the subject so that the initial spark develops into a more sustained interest. It consists of four elements: a 20-page free magazine sent twice yearly to UK schools and subscribers worldwide; a website with up to 10 new articles per month; shows, such as the hour long cs4fn magic show [3], and booklets that allow audiences to explore topics more deeply.

The core target audience for cs4fn resources are school students aged 14+ though some are also presented to younger (9+) children and family audiences at science festivals. The magazines and website are read by people of all ages and professions including students, teachers, professionals and interested members of the general public.

The time commitment for cs4fn participants need not be high; school talks last only one school period, and at science festivals contact with individuals may be as short as 10 minutes. Likewise, students may only read a few articles at a time. The programme focuses on telling engaging stories about research, illustrating CS concepts using examples that resonate with students' lives. It also examines thought-provoking topics with a philosophical dimension, such as artificial intelligence. cs4fn talks incorporate lots of interaction. The project uses a concept of a 'sticky web' of approaches, the idea being that whether individuals find it via web, magazine or a talk, they are then drawn further in to the other strands of the project.

Across all strands cs4fn aims to show that computing is a fun, enjoyable subject that students should take further for intrinsic motivational reasons. Feedback from teachers and participants is overwhelmingly positive for all four elements of the project.

5 CS Inside

The CS Inside approach [5] draws much from CS Unplugged: predominantly kinaesthetic activities, undertaken without the need for computers, designed to be used by presenters with varying levels of computing experience. Whereas CS Unplugged was originally designed for students up to about the age of 12, the CS Inside activities were written for high school students from the start, recognising the different motivations that these two student groups will have for taking part in kinaesthetic activities. "Inside" is used in the approach's name because each activity brings out some of the Computer Science to be found inside the technology that is part of students' everyday lives, such as mobile phones, web browsers and game consoles. The aim is to engage students in issues of relevance to them about computing technology, and open up those issues by exploring the computer science inside.

The areas of Computer Science covered by the activities are not explicitly chosen to match with any particular school curriculum. The original context, however, was Scottish schools, and links are identified from the CS Inside activities to precise parts of the Scottish schools' computing curriculum where they may be useful. Some teachers use only those activities that help support the curriculum while others are happy to use them all (and are hungry for more!), recognising that materials of this nature are genuinely inspiring for students.

The activities are typically structured in four parts: the *Grab* captures the students' attention by asking questions about technology that are relevant to students; the *Intro* shifts attention from the students' context to the technology context to be addressed in this activity – this should be as small as possible to avoid losing the students' interest; the *Activity* is the main task; and the *Sustain* carries the learning from this activity out into their everyday lives, so that everyday events concerning technology will remind them of what is going on inside.

6 CS, Academia and Industry

This programme aims to expose students directly to state-of-the-art research, advanced technologies, software engineering methodologies, and professional norms by having the students interact with leading experts [14]. It is extracurricular, designed especially for talented high-school students in Israel who major in CS. Its main goal is to bridge the gap between school education and the "real world" of computing, especially relating to content, learning culture, and professional norms.

There are three main motivations for the approach: students can (1) add state-of-the-art computing research and development to the fundamentals taught at school, (2) be encouraged to become self-learners by experiencing a "taste-based", breadth-oriented learning approach, and (3) participate in "real-world" software development through a comprehensive project.

The two-year programme blends formal and informal learning and includes enrichment meetings, field trips and software development projects under the supervision of experts. Talented students are recommended to attend by their teachers.

The first stage is designed for 11th grade students and consists of a 7-month enrichment workshop looking at contemporary issues in computing. In the second phase, the 12th grade students (chosen from the first stage attendees) develop comprehensive software projects under the apprenticeship-based supervision of professional mentors (scientists and engineers from academia and the hi-tech industry).

An underlying principle of the first phase, which avoids using programming, is that students should be taught to employ a breadth-oriented learning style, in which their initial exposure to an unfamiliar topic will be accomplished by exposing them only to its essence (i.e., the main high-level abstract ideas). In other words, a complete understanding, including knowing the concrete details and mastering procedural aspects, should not be considered as the immediate aim of an initial exposure to a new topic. To achieve this, monthly enrichment meetings are conducted in which a variety of advanced topics are introduced in plenary sessions by leading representatives of CS/SE academia and industry. The sessions cover topics such as computer sciences and biology, artificial intelligence, computing in space, and professional norms.

In addition, the following non-programming learning activities were conducted, challenging algorithmic problems, role-playing simulation games, creative thinking in computer science, model-based-development, and a competition in testing software.

Feedback so far indicates that the programme contributes to developing a culture of learning befitting the dynamic world of industrial computing, thus providing the students with an entry point into the computing community of practice [10,14].

7 Comparison of the Programmes

Tables 1 and 2 compare the five approaches using criteria that highlight the similarities and differences between them. These criteria have emerged as a result of the previous discussion. By drawing out the commonalities of such large-scale successful programmes, we can draw lessons about important ingredients for future initiatives.

Table 1 focuses on the design of each approach. Comparing the five approaches, we see that in terms of the range of topics covered, all offer a breadth across Computer Science, exposing students to the range of topics that they might choose from to specialise in if they undertake the discipline. All operate on a large scale, with thousands of participants, and dozens (if not hundreds) of resources for presenters to draw on. All have a high level of visibility, some more locally, while others have a large international following. All rely on a pool of contributors and have some form of quality control. Most of the approaches engage students for around an hour at a time, though magazines and videos may only require minutes to engage with.

Table 2 analyses how the participants (students and teachers) interact with each programme. Finding a motivation for students to participate is key. Because the work is largely outside the curriculum, and therefore does not count towards formal grades, there need to be other motivations, either intrinsic or extrinsic. From the table, we can see that intrinsic motivation in the form of satisfying curiosity and enjoying problem solving are common to all the programmes. However, specific motivations used depend on the approach. A magazine article needs to capture enough interest at the start to keep the students reading for a few minutes. Humour and story-telling are used in the shows to keep a larger and possibly reluctant audience engaged for longer.

Programmes that require more commitment from students use prizes or certificates, and experts can serve as role-models to keep students engaged. A programme over several years needs either to use a deep intrinsic motivation of enjoyment in the subject or help students achieve their long term goals using, e.g., long-term career prospects as motivation.

All of the approaches considered avoid having programming as a primary focus, at least initially. The high level of uptake of all 5 indicates that this doesn't prevent students from being interested. These contrast with other popular approaches to outreach that are largely based around programming, such as robotics competitions, or introductory languages such as Scratch, Alice and Greenfoot [12]. The two-stage "CS, Academia & Industry" programme combines both approaches; it doesn't require programming in the preliminary stage, but the advanced stage is based around it.

We note that programming can be integrated in a variety of ways with these predominantly non-programming approaches, either as a follow-up where students implement ideas they have been exploring, or conversely where the non-programming

Table 1. A comparison of the five outreach approaches: design

	Bebras	CS Unplugged	cs4fn	CS Inside	CS, academia & industry
Web site	www.bebras.org; www.bebras.lt	csunplugged.org	cs4fn.org	csi.dcs.gla.ac.uk	davidson.weizmann.ac.il/eng/projects.php?cat=488
Topics covered	Broad coverage of technical and soft topics	Broad coverage of technical topics	Research based, technical, soft and interdisciplinary topics	Broad coverage of technical topics	Broad coverage of state-of-the-art R&D in various CS and software engineering topics
Scale of resources	100-200 tasks each year (since 2004)	25 main activities, 49 videos, more added each year	600− articles, 6-10 new per month, 10 talks per month	11 main activities. Further activities developed each year	A collection of projects developed by the students
Scale of activity	200,000 participants in annual competition	850 website page views daily, 100 video views daily	Annually 2-3 magazines, up to 23,000 copies,70 shows to 9000 students. 2000 website visitors daily.	1200 registered users. 250 Scottish schools. 700 activity downloads per year.	Over 7 years, >1,000 total students attended first stage, 200 of whom developed projects in the advanced stage
Adoption	20 countries are involved (2010), tasks translated into 15 languages	International, translated into 14 languages, in ACM K-12 curriculum	International website and magazine readers. Some material translated into 4 languages	Adopted principally in UK, with individual users from over 20 countries.	Conducted at one site, attended by students from 30 schools across Israel.
Contributors to resources	Annual workshop from multiple countries. Teachers and CS students contribute.	Suggestions from around the world, collected and quality controlled by project leader	Academics and team at Queen Mary, University of London with article contributions mainly from UK	Academics and students at University of Glasgow	Academics at Weizmann Institute of Science and other universities in Israel plus industry mentors
Typical time per activity	Contest: 45-60 mins. Some countries have two-round contests.	Classroom: 1 to 2 hours per activity; Show: 1 hour; Videos: a few mins	Shows: 45-120 mins; Festivals: typically 10 minutes contact; Articles: 10 mins	30-70 minutes per activity	Preliminary stage: 50 min lecture <> 2 + 90 min activity monthly; Advanced stage: over 9-10 months.

Table 2. A comparison of the five outreach approaches: analysis

	Bebras	CS Unplugged	cs4fn	CS Inside	CS, academia & industry
Student motivation	Interesting, attractive tasks, problem solving, certificates, prizes, shows	Kinaesthetic activity, problem solving,curiosity, humour, small prizes	Curiosity, offbeat links, humour, engaging stories, kinaesthetic activities	Relevance to technology in everyday lives, kinaesthetic, problem solving, curiosity, school testing	Intrinsic interest, career prospects
Prerequisites	Basic mathematical and reasoning skills	Basic mathematical and reasoning skills	None for most, some require basic reasoning skills	Basic mathematical and reasoning skills	Studying computer science according to a formal national program
Relationship to programming	No programming skills required. Logo-based commands used in some problems.	No programming, but can use programming as follow-up, or use activities as break from programming.	No programming, though some articles explain programming concepts.	Principally no programming. Ideas can be programmed as a follow-up.	First phase: no programming Second phase: programming project
Relationship to curriculum	Intends to influence common understanding of CS curriculum for high schools. A subset of problems may relate to national curricula.	Usually extra-curricula, but links to curriculum provided and can be used to supplement curricula material.	Does not address curricula directly, but goes beyond it. Teachers use it as support to the curriculum with some guidance.	Some activities designed to support Scottish curriculum. Links to Scottish curricula included.	Supplement to school curriculum.
Teachers' involvement	Teachers motivate students to attend run preparatory activities, and organise local online contest. At end link the tasks to informatics.	The material can be presented by teachers, or they organise visits from/to universities	The material can be presented by teachers, or they organise visits from/to universities	The activities are primarily run by teachers, but students and academics present them too	Teachers accompany students through the whole 2-year period; attend enrichment meetings and mediate between the students and their mentors throughout project

activities provide a physical break from programming at a computer. In fact, there is evidence that including programming as a follow-up is useful, to help students see more deeply how the material relates to computers [11].

The material discussed in this paper is not generally taught as part of a formal curriculum, typically being used in outreach and interest programmes such as one-off visits, out-of-school clubs, science centres, and special events. This is particularly useful if CS isn't compulsory in schools, as is the case in many countries and states, as it helps to expose students to a subject that they may have little idea about. It also helps teachers and careers advisers to understand the topic, supporting them to help their students make appropriate career decisions.

Looking at the impact on teachers, we note that the programmes support teachers by providing resources for them rather than expecting them to do the preparation and planning. In fact, an unexpected focus of the CS Inside project was the development and support of teacher communities. Simply providing materials to teachers isn't sufficient; high school CS teachers are often a forgotten community, and making personal contact and building trust has been a key component of CS Inside's success. The other approaches also report spin-offs from an improvement in relationships with teachers, partly by providing a reason for contact between schools and universities, and also in helping teachers gain a significantly better understanding of CS concepts.

Each of these approaches either already shares ideas with the others, or can benefit from sharing them. For example, some of the CS Inside and cs4fn lessons are based on CS Unplugged activities; conversely, the CS Unplugged website provides links to the relevant CS Inside and cs4fn activities as extension or follow-up ideas. New puzzles in the Bebras project can be constructed by looking at the challenges in the other projects and creating a story/context to make them accessible to contestants, and phase 1 of the CS, academia and industry program can use activities and lesson plans from the other approaches to provide a means to engage with students. The non-programming approaches here also work well as supplements to more traditional programming-based approaches, providing an active break from sedentary work at the computer.

All the approaches are flexible – the resources themselves generally provide creative ideas for teaching that can be adapted to the audience and by the audience (teachers or students) for their own needs. The programmes are presented in a way that is accessible for teachers and other organisers; resources are provided at no direct cost through well-resourced websites, and the details that can be time-consuming for teachers (such as preparing slides and handouts) are taken care of. Notably, all the programmes include significant commitment and resources over a long period of time from their organisers.

8 Conclusions

There are a variety of ways that students can be exposed to Computer Science without the barrier of requiring programming as a pre-requisite. The large number of students who have participated in the approaches described here indicates that the avoidance of programming can indeed generate ongoing interest and so can be considered successful. This contrasts with approaches where programming is taught in-depth

first; in schools where time is limited the programming-first approach can lead to the misconception that CS is *only* about programming, and thus only students whose main interest in computing is programming are motivated to continue in CS.

Our analysis here shows that despite the five approaches having quite different formats, there are considerable commonalities between them that can guide the development of new initiatives. One valuable motivation for students to engage with these activities is a deeper understanding of the computing devices all around us. However, because the focus in all of these approaches is Computer Science, not computers, alternative motivators for students can be important: contest prizes, the challenge of solving a problem, curiosity, humour, and ideally, appealing to the intrinsic interest of the student in this kind of thinking and reasoning. This means that the material needs to be carefully crafted to attract and retain student interest (e.g. direct relevance to their life, engaging story telling, well planned magic tricks, questions to stimulate their curiosity), or it needs to create a culture that attracts students (e.g. past participants recommend it, high status or rewards for competition winners, a good reputation for the event).

The non-curriculum approach provides the opportunity for grass-roots influence on formal curricula when a top-down state-led approach often struggles to efficiently deliver CS education in schools. The success of several of the approaches here has led to them being recommended for curricula, so the authors then have an influence on formal CS education. An important issue that will need to be addressed then is how materials largely designed for outreach can be adapted for settings where a certain level of assessment will inevitably be required.

We have presented several creative approaches with the common goal of attracting students to study Computer Science. As a result of the comparison between them we developed the criteria presented in this paper. These criteria will enable those designing outreach and teaching programmes to evaluate approaches, to choose the most suitable approach for their students, and to adapt the approaches to the target population and context.

Acknowledgments. We are grateful to Cecile Yehezkel, Peter McOwan and Jonathan Black for their work with some of these programmes, and EPSRC, Google and our own institutions who have supported them.

References

1. Bell, T.: A low-cost high-impact computer science show for family audiences. In: Australasian Computer Science Conference, Canberra, Australia, pp. 10–16 (2000)
2. Bell, T., Alexander, J., Freeman, I., Grimley, M.: Computer Science Unplugged: School Students Doing Real Computing Without Computers. The New Zealand Journal of Applied Computing and Information Technology 13(1), 20–29 (2009)
3. Curzon, P., McOwan, P.W.: Engaging with Computer Science through Magic Shows. ACM SIGCSE Bulletin 40(3), 179–183 (2008)
4. Curzon, P., Black, J., Meagher, L.R., McOwan, P.W.: cs4fn.org: Enthusing Students about Computer Science. In: Proceedings of Informatics Education Europe IV, Freiburg, Germany, November 5-6, pp. 73–80 (2009)

5. Cutts, Q., Brown, M., Kemp, L., Matheson, C.: Enthusing and informing potential computer science students and their teachers. ACM SIGCSE Bulletin 39(3), 196–200 (2007)
6. Dagienė, V.: Information technology contests – introduction to computer science in an attractive way. Informatics in Education 5(1), 37–46 (2006)
7. Dagienė, V., Futschek, G.: Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 19–30. Springer, Heidelberg (2008)
8. Dagiene, V.: Supporting computer science education through competitions. In: Proc. 9th WCCE 2009, Bento Goncalves, Paper-Nr. 76, 10 pages (2009)
9. Dagiene, V., Futschek, G.: Bebras International Contest on Informatics and Computer Literacy: A contest for all secondary school students to be more interested in Informatics and ICT concepts. In: Proc. 9th WCCE 2009, Bento Goncalves, Paper-Nr. 161, 2 pages (2009)
10. Haberman, B., Yehezkel, C.: A computer science educational program for establishing an entry point to the computing community of practice. J. of Information Technology Education (JIRE) 7, 81–100 (2008)
11. Taub, R., Ben-Ari, M., Armoni, M.: The effect of CS unplugged on middle-school students' views of CS. SIGCSE Bull. 41(3), 99–103 (2009)
12. Utting, I., Cooper, S., Kölling, M., Maloney, J., Resnick, M.: Alice, Greenfoot, and Scratch - A Discussion. Trans. Comput. Educ. 10(4), Article 17, 11 (2010)
13. Wing, J.M.: Computational thinking. Communications of the ACM 49(3), 33–35 (2006)
14. Yehezkel, C., Haberman, B.: Bridging the gap between school computing and the "real world". In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 38–47. Springer, Heidelberg (2006)