A First Look at the Year in Computing

Sebastian Dziallas
School of Computing
University of Kent
Canterbury, CT2 7NF, England
+44 1227 827684
sd485@kent.ac.uk

Sally Fincher
School of Computing
University of Kent
Canterbury, CT2 7NF, England
+44 1227 824061
s.a.fincher@kent.ac.uk

Colin G. Johnson
School of Computing
University of Kent
Canterbury, CT2 7NF, England
+44 1227 762811
c.g.johnson@kent.ac.uk

Ian Utting
School of Computing
University of Kent
Canterbury, CT2 7NF, England
+44 1227 823811
i.a.utting@kent.ac.uk

ABSTRACT
In this paper, we discuss students' expectations and experiences in the first term of the *Year in Computing*, a new programme for non-computing majors at the University of Kent, a public research university in the UK. We focus on the effect of students' home discipline on their experiences in the programme and situate this work within the context of wider efforts to make the study of computing accessible to a broader range of students.

CCS Concepts
• Social and professional topics~Computer science education

Keywords
non-majors; curriculum; qualitative research; student experience

1. INTRODUCTION
In recent years, a push to broaden the base of students studying computing has taken place. These efforts are not merely limited to undergraduate students. Jeannette Wing argued in 2006 that "computational thinking" is a skill everyone should possess [21]. Furthermore, in 2016, President Barack Obama announced the *Computer Science for All* initiative in the United States with the goal of providing opportunities for all students from kindergarten through secondary education to learn computer science [5].

Computing has also moved to embrace a more inclusive view of what is part of the discipline, particularly in relation to other fields. The authors of the 2013 ACM/IEEE curriculum report called this the "big tent view" of computing. They wrote:

> *As CS expands to include more cross-disciplinary work and new programs of the form "Computational Biology," "Computational Engineering," and "Computational X" are developed, it is important to embrace an outward-looking view that sees CS as a discipline actively seeking to work with and integrate into other disciplines. [12]*

ITiCSE '17, July 03-05, 2017, Bologna, Italy

ACM 978-1-4503-4704-4/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3059009.3059049

Programmes for non-majors in computing can be roughly separated into three categories: those looking to broaden the student base in computing; to provide or increase the skills of students in their home discipline ("upskilling"); and to convert students into computing graduates.

An example for broadening the student base is the course for non-STEM students on media computation that Guzdial and Forte describe in their work. As part of their process of creating the course, they decided to maintain similar curricular goals as other introductory courses, while changing its context to motivate students [9, 10]. In the resulting course, students create and manipulate media, but still learn to program.

Other efforts are more concerned with providing students with a set of specific set of computing skills they can use in conjunction with their own subject area. For instance, *software carpentry* is a formal programme that stands outside of traditional institutional boundaries and brings together students from STEM backgrounds. It is designed to specifically address "small-scale and immediately practical issues" in software development using tools and techniques, such as version control and debugging [20]. Another example in this area is the work of DeJongh and LeBlanc, who intended to bring computer science concepts and tools to bioinformatics students [3].

At the university level, course offerings for non-majors have increasingly attracted interest, including from mainstream news organisations [18]. There is a wide range of courses: some are intended to introduce students to wide range of different concepts and applications (such as problem solving [13], algorithms and computational thinking [6], internet applications and web programming [8, 14]), others form the basis of a sequence of courses (and have adopted the have adopted the CS0 terminology [1, 11, 17]). At liberal arts institutions, mainly in the US, students may elect to complete a minor degree in computer science in addition to their major degree [4]. And in the UK, students can enrol in joint honours programmes that allow them to study two subjects during their time at university.

Finally, in terms of "converting" students who have already completed a degree in another subject, universities in the UK offer conversion MSc programmes which are open to all students with good results in their first degree.

In this paper, we examine the newly launched *Year in Computing* at the School of Computing at the University of Kent. The Year in Computing is currently in its first year and provides an opportunity for students from other disciplines to study computing. The programme is similar to MSc conversion programmes; however, those are taught at a postgraduate level and are intended to cover a large fraction of the core of Computer Science, whilst the Year in Computing is an undergraduate programme with a distinctive curriculum and a focus on web technologies. It is also an effort to both broaden the student base in computing at the University and to help students develop computing skills they can use in their home discipline or more generally in their future careers.

2. CONTEXT OF THE YEAR IN COMPUTING

Undergraduate degrees in the United Kingdom generally take three years to complete, although there are opportunities for an additional *Year in Industry* or a *Year Abroad*. Students declare their intended major upon application to university. Some students choose to enrol in so-called joint honours programmes that allow them to study two subjects (that do not necessarily have to be closely related) over the normal duration of an undergraduate degree. In a joint honours programme, each department is responsible for delivering their own respective courses, so students generally take existing courses together with undergraduates who are in single degree programmes. As a result, joint honours programmes often do not facilitate cohort formation among students. They also lack a coherent curriculum structure, while requiring that students complete the course requirements for both subject areas in a limited time, and often suffer from hidden pre-requisites in the courses they do take.

In contrast, the Year in Computing is a free-standing, self-contained additional year, offered to undergraduate students doing any degree in the University that does not contain computing. Students may take the Year in Computing in between the second and third year of their degree, or after year three. During the year, students work exclusively in the School of Computing. The programme operates as a stand-alone year of study in which students are taught as a single cohort; all students in the programme take the same courses at the same time as part of a coherent curriculum, instead of individual courses with other undergraduates. This allows for cohort identity formation and obviates issues of timetabling and pre-requisites that otherwise plague joint honours programmes.

The Year in Computing as a whole is "pass / fail". Successful students ultimately graduate with their degree title augmented with the designation "with a Year in Computing". If a student fails their Year in Computing, they return to their home discipline and graduate without the additional designation. So whilst the grades they receive in the courses appear on their transcript, they do not affect the classification of their degree which remains wholly dependent on their performance in their home discipline. Degree classification is a significant (although rather crude) measure of overall student achievement in the UK, used as a gateway to further study and employment, which does not reflect potentially wide disparities between performance when students are studying more than one subject, as in a traditional joint honours programme.

3. CURRICULAR CONTENT

The Year in Computing is aimed both at students who want to "convert" into computing for vocational reasons, and for students who want to integrate computing with their home degree studies. The latter could include students who plan to integrate computing into a scientific discipline (e.g. in bioinformatics), to use data science skills in a social science area (e.g. in analysis of data from social networks), or to use computing technologies as part of an artistic practice.

Students in the programme follow a curriculum specifically developed for this context. The courses were designed from the ground up (or, in the case of HCI and web technologies adapted versions of the modules that undergraduate students in computer science take). This allows us to focus on the aspects most relevant to the students in the programme.

Courses for non-majors traditionally focus either on a breadth of computing topics or depth in terms of programming [17]. Whilst the introductory courses in the BSc in Computer Science at the University of Kent rely on Java as a first programming language and include a large component of logic and discrete mathematics, we decided to focus instead on web technologies. This allows us to introduce a wide range of computing topics in the context of the web (providing breadth) while exposing students to an entire stack of software (addressing depth).

Table 1. Courses in the Year in Computing

Autumn Term	Spring Term
An Introduction to Computer Systems: From the desktop to the global Internet. (7.5 ECTS credits)	**Solving Problems with Data**: Collecting, analysing and portraying data from specific domains, businesses, and the world. (7.5 credits)
Human Computer Interaction and User Experience: Designing information and applications for their users and their purpose (7.5 credits)	**Web Development**: Building and managing large scale, dynamic, web applications. (7.5 credits)
An Introduction to Programming and Web Technologies (15 credits): • Presenting information (HTML and CSS) • A general introduction to programming, through coding in Javascript • Storing information (databases and SQL). • Dynamically generating content for web pages from stored data (PHP).	**Year in Computing Project** (15 credits): Putting learning into practice in a larger piece of work, perhaps related to a domain in the home discipline.

The curriculum for the Year in Computing (Table 1) then covers "back-end" topics such as understanding computer operating systems and networks, learning programming (through

JavaScript), storing and manipulating data, and integrating with a Web server. At the "front-end" it includes producing web pages using HTML, CSS and JavaScript that work well, look good and are easy to interact with.

We also wanted to include explicit opportunities for students to work in the context of their home discipline. Both the Solving Problems with Data course and (particularly) the Project component encourage students to use data and address problems from their own disciplines.

Performance in the programme is assessed by means of practical coursework and a small number of written examinations.

By the end of the programme, regardless of whether they intend to continue to work in computing or plan to return to their home discipline, we expect students to be able to:

- Understand the role of technology and how it is used in the contemporary world.
- Have a good foundational knowledge of coding that is focused on the ideas of programming, not just learning a specific language.
- Build dynamic, modern web-based systems.
- Understand how data can be used to tackle complex problems.
- Have a practical grasp of methods for presenting data and designing interactions with computer-based systems.

4. CHARACTERISTICS OF THE STUDENT BODY

The University of Kent is a medium-size university with approximately 15,000 undergraduate students. Within this student body, only those students in their second or third year who are not pursuing a degree in Computing are eligible to apply to the Year in Computing. There are currently 34 students enrolled in the programme.

One pre-requisite for admission to the Year in Computing is success as a student at the University to date, not any given subject knowledge about computing. (The Year in Computing is not for students who are unsatisfied with their performance in their home discipline and looking to switch into computing.) A side-effect of only recruiting students from the same university is that they are already familiar with the campus, the institutional processes and (to an extent) University systems, thus significantly reducing their initial familiarisation difficulties compared to incoming undergraduate students. This familiarity can also be a "false friend" however, highlighted in the differences in expectations (of how and where to study) between the Year in Computing and students' home subjects.

The application process for the programme is also deliberately light-weight: Students submit a formless application statement expressing why they are interested in the Year in Computing and take part in an interview designed to assess their enthusiasm about engaging with computing. The student body resulting from this process consists of students from a great variety of different backgrounds.

As well as disparity of academic background, they also have a wide variety of previous experience of computers and coding, either as part of their prior academic experience in secondary or tertiary education, or as "hobbyists". Although this was captured in the application process, and students who had (effectively) already covered the syllabus were excluded, judgements at the lower end of the experience range have been less robust.

Undergraduate students in Computing at most institutions are traditionally relatively homogenous: they are predominantly male and technology-focussed. For example, only 15% of the students in the BSc in Computer Science at the University of Kent are women. In contrast, 40% of the students in the Year in Computing are women. Furthermore, 47% of the 34 students in the programme are completing non-STEM degrees at the University.[1]

Studying this first Year in Computing will enrich our understanding and inform our approach to making computing attractive as a destination subject for students from a wider range of backgrounds and with a wider range of personal characteristics than those who typically choose this subject area.

5. THIS WORK

In this paper, we provide a 'first look' at the first term of the programme. We specifically focus on the role of students' home discipline, as well as their experiences in the programme to date.

We reviewed both students' application statements and their responses to an end-of-term survey conducted after the first semester in the course. (This was not the generic module evaluation form used for all courses, although some students took the opportunity to use it to provide feedback.) The survey asked students about their expectations for the course, their previous learning experiences at University, the amount of time they spent on the different modules, their own personal and professional goals, their assessment of the skills they developed, and the effect the Year in Computing on them to date. Out of 34 students in the programme, 12 responded to the end-of-term survey (35%).

6. THEMES

6.1 Why Students Chose to Apply

Students expressed different motivations in their application statements. Some of them were looking to enhance their employability within the context of their home discipline, particularly in STEM fields, such as chemistry and physics.

> *"Throughout my course, we have been taught that the ability to collect and analyse data is a central skill for forensic scientists and I believe that adding this additional qualification will be very useful for my future job prospects in this field*."

> *"I became interested in computing after my first year module in 'computing skills' where I learnt some basics in coding. This interest propelled me to my project in computational chemistry. [It] has made me want to learn more about computing … that would give me an edge when applying for jobs around chemistry and computational chemistry*."

> *"This is something I've done as an amateur for quite a while and having some proper grounding would allow me to do computing on a more professional basis. It would also give me skills that would apply to almost any area of physics work*."

But the home discipline did not play a decisive role for all students. One anthropology student was interested in the subject

[1] We are not providing a full list of disciplines as students might otherwise be identifiable due to the small number of participants.

matter and felt that facility with technical systems was important regardless of discipline.

> *"I am self-taught, and being formally taught computing would allow me to polish my skills and develop new ones. Although computing is less applicable to my degree than others, I think that technical knowledge and web-related development is important regardless of academic background*."

And for one student, who was unable to find a joint honours programme that offered their subjects of interest, the Year in Computing provided them a chance they never had:

> *"Unfortunately, while applying for University, I had to choose only one subject and could not find any proper combinations with architecture and computing. However, with this new 'Year in Computing' being launched by the University, I feel that I have an exceptional opportunity to study a combination that I truly want.*"

Not all students who were offered spaces in the Year in Computing ended up enrolling. In fact, the reasons why students did not choose to do a Year in Computing matched common reasons why students do not take part in a placement year: they could not arrange accommodation in time or did not want to miss graduating with their cohort in their home discipline [7].

6.2 Expectations & Reality

When we surveyed students at the end of their first term, 8 out of 12 (67%) expressed that the course had met their expectations, particularly with regards to both breadth and depth of the curriculum.

> *"I expected a general overview of aspects of computing, which for the most part I got."*

> *"I did expect to learn programming skills and the course is what I expected it to be."*

However, some students indicated that certain aspects were different than they had expected. Their comments focussed particularly on the individual modules—Introduction to Computer Systems, HCI/UX, and Programing and Web Technologies (which students commonly referred to as "the programming module"). This was not entirely surprising to us, as the curriculum structures material into the three distinct courses which each have their own a set of teaching staff.

> *"I expected to get an introduction into the world of computing, starting with some basic programming and understanding of computer systems. I don't really understand Computer Systems, but I'm getting better at programming, which is great!"*

> "*It's what I expected, apart from [the HCI module]. I was expecting some kind of design element to the course but it [the HCI module] went beyond what I had expected."*

> *"[The programming module] definitely makes it very clear when you've made progress and really makes you feel like you are learning and understanding and that the work you're putting in is actually producing something which is very motivating and encouraging especially in comparison to [the Computer Systems module] which can feel like swimming in quicksand at times."*

Of course, not all of the students' learning experiences were positive and for some students the course has been harder than expected.

> *"I believe some of the tasks have been too tricky for a beginner."*

> *"I expected the course to be for someone who is a complete novice […]."*

It is not immediately clear whether this student was expecting a course on computer literacy or computational thinking, rather than one with significant programming elements. Some of the students who expressed difficulty with the course included a different kind of reflection.

> *"It's made me feel intimidated to go into computing-based jobs, and question my ability to handle next term."*

> *"I was quite disheartened by finding JavaScript a particularly difficult topic to study - it made me feel like maybe going into bioinformatics wouldn't be the best idea […]."*

We take both these expressions – of the current challenge of the course and of an imagined future with regard to computing – as expressions of self-efficacy. According to Bandura, self-efficacy is defined as the belief in one's ability to accomplish a task [2]. These comments then appear to reflect the students' own perceived ability to succeed in the Year in Computing as a result of their experiences in the first term. Wiedenbeck analysed factors affecting non-majors' experiences when learning to program and found that self-efficacy, as well as knowledge organisation, played a central role in their experiences [19]. Lishinski et al. showed for students in a CS1 course that students' motivation affects their self-efficacy, which affects their performance in the course, which in turn affects their self-efficacy in a virtuous (or vicious) cycle [15].

For our students there are two, related, aspects to increased self-efficacy. One is mastery of the taught material, being able to complete the assessments, write required programs and design interfaces. In this they work by themselves and with others in the cohort, invoking three of Bandura's elements of self-efficacy—individual achievement, observation of achievement of peers and verbal encouragement of others. However, there is a second element, more fleeting in this data, which is one of general familiarity with computational environments and systems—not something that is explicitly included in the course. Thus we read hints that the less "literacy" experience a Year in Computing student has, the lower their belief in their own self-efficacy which becomes a barrier for their further learning. "I expected [it] to be for a complete novice" suggests the lack of an entirely separate set of skills.

This presents an opportunity for future work for us and we intend to follow up with these students in the future to explore their expectations and experiences further. Additionally, it has prompted us to add a more explicit exploration of an applicant's familiarity with technology at interview.

6.3 Contrast to previous experience

Because we are ultimately interested in how metacognitive skills of "being a good student" transfer between disciplines, students were also asked to identify contrasts to their previous learning experiences in their home discipline at University. One of the

students who found the course harder than expected was surprised to find that they were falling behind despite attending all of the scheduled class sessions.

> *"I did feel lost by the wayside for most of the term and easily left behind in terms of understanding the course content even when I was attending my lectures/classes."*

Thus suggesting that "attending all lectures/classes" was sufficient to this student's previous academic success, perhaps generally sufficient to academic success in their home discipline.

Conversely, one student arrived at the first Lab session for the programming course having already completed all the exercises, and being prepared to discuss their thinking (as, presumably, in seminars in their home discipline), but was surprised to find that they were expected to do it all (again) in the Lab.

Students were also surprised by the accessibility of staff in the programme.

> *"The coding [in the programming module] is exactly what I expected and I have really enjoyed that section the most. There are more complicated Computer Systems than I thought there would be, but there is far more support than I could have dreamt of."*

> *"The support offered far supersedes expectations and the approachability of the staff is fantastic."*

When we asked students what kind of advice they would offer to incoming students, they focussed on *practicing* both ahead of and during the term, rather than asking for help, in their responses: half of the students who responded to the survey said that they would tell their fellow students to "keep practicing". This advice was often focussed on the programming language used in the course, JavaScript:

> *"Mess around with JavaScript beforehand, really try and get to grips with it, ask for plenty of help when stuck."*

> "*Do a lot more outside work for programming than is given, i.e. on Codecademy / w3schools etc."*

Other differences in contrast to students' previous experiences at university concerned assessments, particularly in contrast how they are set in the humanities.

> *"In my previous degree, there were 4 big essays all due in the last two weeks of term. I really like the way there is smaller continuous assessments, though there are more exams than I thought there would be."*

> *"The spread of assignments is generally better than multiple essays due at the end of week 6 and end of week 12, and the variety is refreshing."*

> *"The assignments are much more involved and the coursework is more work-oriented rather than test-oriented."*

One student observed that the assessment criteria were also different:

> *"I would say the standard of assessment is different though however, in my home discipline of biochemistry, assessments given are expected to be as perfect as possible however for this course, mainly for programming, the code can not work and still be marked reasonably high."*

6.4 Goals & Changes

We were also interested in the effect of the Year in Computing. We asked students about their personal and professional goals and whether their experiences this term had changed what they intended to do in the future. Two students indicated that they were uncertain about their future goals. For others, the *Year in Computing* seemed intended to augment any kind of work they might do in the future.

> *"To be successful in any career I go into."*

> *"[To] secure a job before graduating, not having to move home after graduation situation permitting."*

But it also served as a way for students to expand on their home discipline.

> *"I would like to eventually go on and study a bioinformatics masters and then either stay in academia looking at protein structure and function prediction programs [...] or going into industry and using bioinformatics as a skill for investigating possible pharmaceutical drug targets."*

Again others had specific ideas about the work they planned to do after graduating, with a specific focus on computing.

> *"I want to focus on Web Development or Data Control as a profession within a technology focussed business."*

> *"Simulation, data handling, software development."*

> *"I hope to do the MSc conversion course and go into software development, data management or marketing."*

For these three students, the *Year in Computing* reinforced their confidence in choosing a career in a computing-related field.

> *"It has made me more certain that I want to go into computer based careers in the future and assured my passion for it that I was unsure of at the beginning."*

> *"I feel better equipped for creating applications and may go more down that route"*

> *"Rather than a specific focus on marketing my experiences have massively opened my eyes to other options and piqued my interest in computer science fields and showed how related marketing can be to design and other things we have done this term."*

For these students, achievement in course to date has increased their self-efficacy and confirmed their choice to study an additional year of Computing. The course also marked a significant change for other students, although not such a directly confirmatory one.

> *"I have chosen to go into teaching, which is a career I never thought of much before this course."*

> *"This course did make me consider becoming a web designer."*

Indeed, one student went so far as to indicate they *"cannot imagine having to go back to my home degree now"* and advised future students to take the *Year in Computing* after their third (and not the second) year as a result. In the future, we plan to follow up with students to conduct in-depth interviews to understand what experiences led to these changes for them.

7. CONCLUSION

The Year in Computing provides an opportunity for non-computing majors at the University of Kent to extend their degree programmes by studying computing. It also provides work-related skills to support students in their future study, research, or careers.

For us, as teachers and researchers, the Year in Computing provides a rare chance to teach and study a stable and coherent group of non-traditional students (who did not intend to become Computing majors when entering University) over an extended period, and affords us insights not only into their development in computing, but also into the hidden assumptions in our own discipline and practices.

On an institutional level, the Year in Computing broadens the School of Computing's student base and provides resilience against fluctuation in undergraduate or taught postgraduate numbers. It also provides other Schools in the University with a model to offer an intercalated year in their own discipline (e.g. the *Year in Business* or the *Year in Quantitative Methods* proposed as part of the UK national Q-Step programme [16]).

8. FUTURE WORK

This work provided a first look at the Year in Computing at the University of Kent, with a particular focus on the effect of students' home discipline on their experience studying computing. In the future, we also plan to use both narrative and traditional qualitative methods to examine the transfer of metacognitive skills (of being a "good student") across disciplinary contexts, the curricular and pedagogical adaptations made by staff in respect of students' diverse disciplinary backgrounds, and the longitudinal effect of the programme on students' experience after graduation.

We intend to follow up with this first cohort of students at the end of their second term to conduct in-depth interviews and to explore the effect of both their home discipline and self-efficacy further. And eventually, we would like to follow them back to their home disciplines and out to work.

9. REFERENCES

[1] Bailey, T. and Forbes, J. 2005. Just-in-time Teaching for CS0. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2005), 366–370.

[2] Bandura, A. 1977. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*. 84, 2 (1977), 191–215.

[3] Bioinformatics in the Computer Science Curriculum: *http://www.cs.hope.edu/~dejongh/bioinformatics/sigcse/*. Accessed: 2017-01-13.

[4] Cliburn, D.C. 2006. A CS0 Course for the Liberal Arts. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2006), 77–81.

[5] Computer Science For All: 2016. *https://www.whitehouse.gov/blog/2016/01/30/computer-science-all*. Accessed: 2017-01-09.

[6] Cortina, T.J. 2007. An Introduction to Computer Science for Non-majors Using Principles of Computation. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2007), 218–222.

[7] Fincher, S. and Finlay, J. 2016. *Computing Graduate Employability: Sharing Practice*. Council of Professors and Heads of Computing.

[8] Gousie, M.B. 2006. A Robust Web Programming and Graphics Course for Non-majors. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2006), 72–76.

[9] Guzdial, M. 2003. A Media Computation Course for Non-majors. *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2003), 104–108.

[10] Guzdial, M. and Forte, A. 2005. Design Process for a Non-majors Computing Course. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2005), 361–365.

[11] Hickey, T.J. 2004. Scheme-based Web Programming As a Basis for a CS0 Curriculum. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2004), 353–357.

[12] Joint Task Force on Computing Curricula 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM.

[13] Joyce, D. 1998. The Computer As a Problem Solving Tool: A Unifying View for a Non-majors Course. *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 1998), 63–67.

[14] Kurkovsky, S. 2007. Making Computing Attractive for Non-majors: A Course Design. *J. Comput. Sci. Coll.* 22, 3 (Jan. 2007), 90–97.

[15] Lishinski, A. et al. 2016. Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance. *Proceedings of the 2016 ACM Conference on International Computing Education Research* (New York, NY, USA, 2016), 211–220.

[16] Q-Step | Nuffield Foundation: *http://www.nuffieldfoundation.org/q-step*. Accessed: 2017-01-15.

[17] Reed, D. 2001. Rethinking CS0 with JavaScript. *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2001), 100–104.

[18] Stross, R. 2012. Computer Science for Non-Majors Takes Many Forms. *The New York Times*.

[19] Wiedenbeck, S. 2005. Factors Affecting the Success of Non-majors in Learning to Program. *Proceedings of the First International Workshop on Computing Education Research* (New York, NY, USA, 2005), 13–24.

[20] Wilson, G. 2006. Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive. *Computing in Science Engineering*. 8, 6 (Nov. 2006), 66–69.

[21] Wing, J.M. 2006. Computational Thinking. *Commun. ACM*. 49, 3 (Mar. 2006), 33–35.