AUTHOR(S): CARTER, J., BOUVIER, D., CARDELL-OLIVER, R., HAMILTON, M., KURKOVSKY, S., MARKHAM, S., MCCLUNG, O.W., MCDERMOTT, R., RIEDESEL, C., SHI, J. and WHITE, S.

TITLE: Motivating all our students?

YEAR: 2011

Publisher citation: CARTER, J., BOUVIER, D., CARDELL-OLIVER, R., HAMILTON, M., KURKOVSKY, S., MARKHAM, S., MCCLUNG, O.W., MCDERMOTT, R., RIEDESEL, C., SHI, J. and WHITE, S. 2011. Motivating all our students? In Proceedings of the 16th innovation and technology in computer science education annual conference: working group reports (ITiCSE-WGR '11), 27 - 29 June 2011, Darmstadt, Germany. New York: ACM [online], pages 1-18. Available from: https://doi.org/10.1145/2078856.2078858.

OpenAIR citation: CARTER, J., BOUVIER, D., CARDELL-OLIVER, R., HAMILTON, M., KURKOVSKY, S., MARKHAM, S., MCCLUNG, O.W., MCDERMOTT, R., RIEDESEL, C., SHI, J. and WHITE, S. 2011. Motivating all our students? In Proceedings of the 16th innovation and technology in computer science education annual conference: working group reports (ITiCSE-WGR '11), 27 - 29 June 2011, Darmstadt, Germany. New York: ACM, pages 1-18. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk.

ITiCSE 2011 Working Group Report
Motivating All our Students?

Janet Carter
School of Computing
University of Kent
Canterbury, Kent, UK
+44 1227 827978
J.E.Carter@kent.ac.uk

Dennis Bouvier
School of Engineering
Southern Illinois University
Edwardsville, IL, USA
+1 618 650 2369
djb@acm.org

Rachel Cardell-Oliver
School of CS & SE
University of Western Australia
Crawley, WA, Australia
+61 8 6488 2231
rachel.cardell-oliver@uwa.edu.au

Margaret Hamilton
School of Computer Science & IT
RMIT University
Melbourne, Australia
+61 3 9925 2939
margaret.hamilton@rmit.edu.au

Stanislav Kurkovsky
Computer Science Department
Central Connecticut State Uni
New Britain, CT, USA
+1 860 832 2720
kurkovskysta@mail.ccsu.edu

Stefanie Markham
Department of Computer Science
Georgia State University
Atlanta, GA, USA
+1 404 413 5700
smarkham@gsu.edu

O William McClung
Math & CS Department
Nebraska Wesleyan Uni
Lincoln, NE, USA
+1 402 465 2252
mcclung@newbrwesleyan.edu

Roger McDermott
School of Computing
Robert Gordon University
Aberdeen, Scotland, UK
+44 1224 26 2717
roger.mcdermott@rgu.ac.uk

Charles Riedesel
CS & Eng Department
University of Nebraska-Lincoln
Lincoln, NE, USA
+1 402 472 3486
riedesel@cse.unl.edu

Jian Shi
Electronics & Computer Science
University of Southampton
Southampton, Hants, UK
+44 23 8059 6000
js9g09@ecs.soton.ac.uk

Su White
Electronics & Computer Science
University of Southampton
Southampton, Hants, UK
+44 23 8059 4471
saw@ecs.soton.ac.uk

ABSTRACT

Academics expend a large amount of time and effort to sustain and enhance the motivation of undergraduate students. Typically based on a desire to ensure that all students achieve their full potential, approaches are based on an understanding that students who are highly motivated will learn more. Furthermore, institutional rewards accrue from effective use of academics' time, along with financial benefits associated with high levels of retention and progression. This working group report, based on practice in Europe, Australasia and North America, builds on previous work. It provides an updated and revised literature review, analyses a larger collection of survey data and has sought to triangulate earlier findings with qualitative data from practitioner interviews. The report covers established approaches in teaching, support and extra-curricular activities. It tracks emerging practice such as streamed and differentiated teaching, and research based and authentic learning. It also considers contemporary innovations in student activities. Finally it reports on a repository of tips and techniques which has been established to support faculty wishing to change or review current methods.

Categories and Subject Descriptors

K.3.2 [**Computers and education**]: Computer and information science education – *computer science education*

General Terms: Human Factors.

Keywordss: otivation, differentiation in the classroom, learning programming, Higher Education.

1. INTRODUCTION

This working group report reports on work which extends and develops the initial study undertaken by the 2010 ITiCSE working group "Motivating Our Top Students" [26]. A team drawn from ten different institutions across Europe, Australasia and North America has built on previous work specifically addressing some of the issues which arose from that initial analysis. The survey data has been extended and qualitative triangulation data gathered from practitioner interviews to augment our analysis.

Undergraduates in the computing disciplines are drawn from diverse backgrounds and student cohorts are often extremely heterogeneous as a consequence. Undergraduate CS (Computer Science) degree studies are often open to students who have no prior knowledge of programming or computing – it is not necessary to have studied IT or CS at pre-university level to gain entry to the course. We have designed courses to suit the majority of such students but we still have our 5% top performing students to teach, some of whom have previously studied the subject and may even be established and sometimes professional programmers who consider themselves already part qualified.

These students begin their first year at university looking forward to an opportunity to finally be challenged after possibly many mundane years at school, comfortably sitting at the top of the class. What a let down, after having possibly entered the hallowed doors of a much respected university, which has a stream of top academic scholars to its name, only to be asked to type in "hello world" after the first week of lectures.

Pre-university educators recognize that some students demonstrate characteristics such as the extreme need to learn at a much faster pace and process material to a much greater depth than others in their class. Some children may be so far ahead of their peer-group that they know more than half the curriculum before the school year starts, and the resulting boredom can lead to low achievement and poor grades; there are recognized mechanisms in place to help such students and their educators. Unfortunately, once these students reach higher education we are so busy providing extra help for our struggling students that their needs are often ignored. Concentrating upon catering to the lowest common denominator in the learning setting can lead us to fail to meet the needs of our more advanced learners in a student centric manner.

The 2010 report investigated ways in which instructors attempt to enthuse the entire cohort: and what special approaches are adopted to support strugglers and those capable of making very rapid progress. The working group also investigated the use of competitions as a motivational tool for engaging high achieving students [67]. The current working group has had the chance to clarify and expand understandings, addressing issues which were raised in the concluding paragraphs of the report.

Furthermore the initial intention of the 2010 working group was to create a repository of resources, but as issues emerged from the data collection it became apparent that the scope of the project was greater than originally anticipated. This year the working group has been able to establish a repository of tips and techniques which has been can be used to support faculty wishing to change or review current methods. The repository also includes a consolidated version of the references used across both years of investigation. Details of the repository content are covered in Section 9.

1.1 Why Worry About the Top Students?

Dropout rates are a major worry for many institutions. "On average, one-third of students "drop out" before they complete their first degree, regardless of whether they are following university level or advanced programs [83]. Some studies suggest that computing experiences a larger proportion of dropouts than other academic areas [47].

Whilst we lose students at the lower end of the achievement spectrum we do also lose those who might have been amongst our higher achieving students – often because they have become bored, then underachieve and ultimately either drop out or underperform. With increasing emphasis upon measuring the numbers of top grade final results it becomes increasingly important for each institution to ensure our high ability students perform to their maximum capabilities.

A number of changes in the context of higher education have worked to make this a particularly important objective. External desires for 'objective' measures of higher education provision have been driven by government or consumer demands. In many countries student experience and institutional achievements are being audited and published for analysis. In the UK, the introduction of a National Student Survey provides a portal of institutional data which can furnish league tables that can in turn have beneficial or detrimental effects upon student recruitment.

Students who have higher levels of satisfaction, and are well motivated are likely to provide better feedback. Profession pride will also motivate academics to ensure that their students have a more successful educational experience. In institutions where students can be recruited to study for higher degrees, working specifically to achieve the highest level of performance may in turn lead to greater numbers of postgraduate applications.

Every cohort of students contains a range of abilities and learning styles. No cohort is entirely uniform. The term "top students" is a relative one; the weaker students at one institution may outperform the stronger students at another.

1.2 Background

Enhancing student motivation leads to better and more effective learning for all our students, wherever their rankings in our cohort; it maximizes their potential. This is beneficial for the students and, as professionals wishing to do our best for our students, it is beneficial for us also.

The goal of this work has been to survey academics from around the world in order to explore the ways in which they enthuse their students; seeking things that work and things that don't. This has been supplemented by an in-depth literature review. In reviewing the literature and collecting evidence of current practice we identified and attempted to inter-relate a broad body of work which spans teaching methods, student motivations, curriculum design and some aspects of educational theory [17, 51, 118].

The synthesis of these formed the basis for the recommendations made by the group:

1.2.1 The Good

- **Challenge tasks** – setting graduated assessments emerged from both the literature and practical experience. Students are all presented with the same assessment, but they choose how much they wish to attempt. Students struggling to master the basics may opt to attempt only the baseline section of the work to obtain a pass, whilst students who find everything easy may opt to attempt everything in the hope of

attaining top grades.

- **Streaming** – Students who struggle can be demoralized by the students who don't. They are unlikely to ask questions about initial basic concepts in front of those asking questions aimed at testing the knowledge of the instructor. Some of these questions are a mechanism for top students to show the instructor that they understand, but also serve to reinforce their ranking within the class.

1.2.2 The Bad

- **Doing nothing** – it is easy to treat everybody in the class the same, but it benefits nobody.
- **Humiliating students** – some academics believe that humiliating students who know more than expected is a means to keeping them in line!

The aim for the 2011 working group was to further explore the educational arguments for and against some common themes that emerged previously. We identify the major themes and have conducted conduct in-depth interviews with practitioners who epitomize these themes. This has allowed us to identify and classify good practice and ideas, so that a repository can be created. It is important to further develop understanding and good practice in this important area.

1.3 Continuing the Work

The major stages involved in addressing our aims are outlined here. We detail what is actually current practice, locate it within educational theory, and create a dissemination mechanism. We based our work around the themes that emerged from the 2010 survey [67]:

- Streamed teaching [14, 27],
- Meeting student expectations [40],
- Research experiences [9, 61, 87] ,
- Maximizing individual potential [24],
- Interdisciplinary connections [115].

Motivating top students and strugglers is within the scope of this paper, but motivating non-majors is not – see [63, 41, 72] for this kind of work.

1.3.1 Analysis

We compared results seeking common issues, problems and themes; these form the basis of Section 3 of the report. It is important to be conscious of the fact that the evidence we have gathered is the product of individual or institutional compromises that must balance workload represented by staff student ratios and individual teaching commitments.

1.3.2 Artifact

The major output of our work to date is a repository of, and guidelines for the use of, materials to stretch and motivate the students who find the work easy, whilst not demotivating those who struggle with basic concepts.

Items for inclusion in the repository emerged from the data-gathering phase and we are in the early stages of formulating an appropriate categorization system and format for the repository at this stage. Appropriate links to currently available materials and sources are also provided. Visitors to the website will be encouraged to suggest examples of good or bad practice that they have encountered within the classroom to add to the current body of knowledge. It is envisaged that the website will eventually become a first port of call for any CS academic wishing to improve, alter or adopt practices aimed at motivating the students who do, or have the potential to perform at the top of their class.

2. WHAT WE DID

This paper draws together evidence based on practice in different countries, which in turn incorporate differing assumptions and process structures. In the US it is often normal for students' first experience of programming at university level to be as part of a broad program of studies. In contrast, in the UK, Australasia and Europe students typically embark on a specialized program of study from the outset. Thus this latter group of students will have already chosen Computer Science (CS) or its related disciplines as a future academic career path. Evidence drawn from different countries needs to be considered within this context, and, consequently, the findings of the research may be of differing value and relevance accordingly.

2.1 Where the Data Came From

The findings and recommendations presented in this report are based upon interviews conducted with academics across a range of institutions.

Several issues must be considered when assessing transferability. Not only are there differences between educational systems in the different countries that are represented, there are many differences between institutions within the same country. Issues such as entry qualifications and grading systems (e.g. GPA in USA) are more obvious than others. Do first year results count? Is it only final year results that determine degree classification? Do students choose a major once they are settled at college or did they apply for a particular subject? Is there an industrial placement as part of the program? All of these issues, and more, have been discussed and considered by the group so that we feel confident in presenting repository ideas that can be adapted to suit most situations.

2.1.1 Institutions

The ten institutions represented here range from a small liberal arts college with fewer than 2,000 students to a large international one with over 90,000 students enrolled upon both face-to-face and distance courses. Some institutions have existed as places of learning for over 100 years, many changing to university status at some point along the way, to purpose built modern universities that emerged from the 1960s expansion in Higher Education. All the institutions have their own ethos and atmosphere meaning that choosing to study there an issue of personal preference as well as league table listings.

2.1.2 National Issues

Here we provide a brief introduction to the educational systems of the countries involved to illustrate the contexts in which we use the responses.

2.1.2.1 USA

The United States has a mixed public/private system of higher education. In 1987 54% of colleges and universities were private and 46% were public. There is no centralized authority exercising control over colleges and universities. Regional accreditation agencies are responsible for ensuring that individual institutions meet corresponding standards. Additionally, professional and specialized accreditation agencies, such as the Computing Accreditation Commission (CAC) of ABET, establish the standards for individual accredited degree programs. Most colleges and universities use the results of two privately developed admissions examinations, the SAT and ACT, in their acceptance process.

2.1.2.2 UK

Students wishing to enter higher education in the UK for their first Bachelors (Honors) degree apply via a centralized system to a named degree program. They may make several choices of institution on the application form. Offers are made on the basis of a total number of points earned by different grades of entry qualifications. The majority of universities also offer postgraduate research degrees for which there is a different application process.

Degree programs usually last 3 years in England and 4 in Scotland. An extra year of Industrial placement may be added to degrees from some institutions in both countries; extending by one year the time taken to complete the degree program.

2.1.2.3 Australia and New Zealand

The Australian and New Zealand university application systems are similar to the UKs, although each state has a different Admissions Centre and students usually live at home during their studies going to university in their home state. Bachelor degree courses are offered by universities. Universities are self-governing with funding from a federal model. Degrees, diplomas and certificates are offered by TAFE (Technical and Further Education) Institutions funded federally. TAFE education is targeted to industry relevant, work-ready students. The selection process for both Universities and TAFEs is generally based on students' VCE marks, called their ENTER score. Most Universities offer Masters level courses, that enrol a large cohort of full-fee paying international students.

3. ACADEMICS' OPINIONS

We applied a qualitative methodology to the study, undertaking semi-structured interviews with academics teaching introductory programming in their institution. The questions we asked were based upon an agreed list of themes for consistency and comparability; the list was derived from the questionnaire administered last year [26].

The responses were recorded and transcribed to provide direct quotations. Previous work has shown that it is profitable to pool information gleaned from different institutions in this way [22, 23, 25]. Respondent validation is essential in order to ensure that the data collected by the interviewers was accurate.

The documentation of the academics' opinions, memories and experiences is more than a simple task of transcribing their comments from the interviews; it involves the subjective interpretation of what they say and do. We have, therefore, compared and contrasted their comments within the contexts of the different institutions and national educational systems.

The opinions documented here fall into nine distinct groups; eight sections contain tried and tested methods of enhancing motivation, and the last contains a summary of things that are less than helpful to try.

3.1 Differentiated Delivery

One outcome of the policy of increasing access to Higher Education is that many classes now contain students with a much wider range of abilities than was the case ten or fifteen years ago. Whereas, in the past, it was sometimes possible to teach to the median level of the class and be reasonably confident that the abilities of most students would fall in a narrow band either side, nowadays instructors often find that this strategy gives little support for those struggling with the course and, moreover, leaves high-achieving students unsatisfied because they feel insufficiently challenged by the material being presented. One way of addressing this issue is to incorporate some kind of mechanism in the delivery of courses which differentiates between the various parts of the spectrum of ability. While this may be an appealing response from the point of view of student-centered pedagogy, there are a number of operational issues which are problematic with regard to implementation. Among the most important of these is the mechanism by which one modifies teaching to respond to the needs of specific groups, e.g. the provision of extension material to top students, and forms of assessment which address the non-homogeneity of ability levels within a student cohort.

3.1.1 Providing Extension Material

One example of addressing differentiated teaching methods within a single class is to break the students into small groups of homogeneous ability level and ensure that they undertake problem-based learning. This allows the tutor to spend more time with the groups of less able students while allowing the groups of highly able students to progress at their own rate.

Extension material consisting of further activities can be provided on topics studied by all class members. Realistically, however, much of this will only be undertaken by students with a more advanced understanding of the subject. While this requires instructor effort, it serves to convince high-achieving students that they remain an educational priority. Such measures are well-received by this group of students who may often lose interest without such intervention.

Another strategy which addresses heterogeneity is examination of the contribution of students beyond the level required to minimally satisfy the learning objectives of the assessment. This can be done in several ways. Description of the assessment may be given in terms of a graduated set of learning objectives which correspond to grading criteria. This allows all students to engage with the assignment to achieve a passing grade, but allows more able students to receive a higher grade for satisfying more of the learning objectives. Alternatively, it is sometimes possible to specify the components that need to appear in a particular assessment but expect the students to inject elements of their own creativity into the assignment in order to attain a higher grade. This may involve additional criteria beyond mere correctness: for example, style or elegance in coding.

The disadvantage of this approach is the time it takes for the instructor to create the differentiated material. This may be significant and the number of students who undertake the extra exercises, and so receive benefit from it, may be small. Nevertheless, it contributes to a greater sense of satisfaction among all students and so better addresses the range of abilities which now appear in the classroom.

3.1.2 Tools for Differentiation

Several tools support computer science education. Though these tools support a variety of facets of the learning environment, differentiation is facilitated through either automating some facet of the course, or by facilitating peer activity. In either case, tool usage leads to increased student autonomy (students can proceed at independent rates) and allows the speedy provision of individualized feedback.

Studies have shown that using such tools increases students' learning. Hamer et al have produced an extensive overview of tools for use in both computing and general courses [44]. It

should, however, be noted that many existing tools are in a state of evolution and typically have low adoption rates.

3.1.2.1 Programming Evaluation Tools

Tools to aid program evaluation are particularly useful in courses that involve extensive amounts of programming. These tools can be classified as either instructor-directed (the instructor develops the testing criteria) or student-directed (students contribute to the testing criteria). In both cases, all students benefit from the availability of early and frequent feedback on their programming code. Ihantola et al [54] provide an overview of automatic program assessment tools, noting the strengths and weaknesses of each tool investigated.

3.2 Streaming

Some CS schools provide a variety of different entry streams for their students. Dependent upon the type of streaming offered, students may self-select their stream or may be allocated to a particular stream with regard to previous qualifications and experience; streams may be explicitly defined by the degree in which the student is enrolled, or they may be defined by the School, the University, funding from an outside organization for select students.

Self-selecting streams include different versions of a similar degree program and may be identifiable by a subtle difference in eventual degree title: e.g. BSc Computer Science (Database Systems), BSc Computer Science (Networks), etc. Such streams often share a common first year, and students can swap to a different stream (or degree program) if they wish once they arrive and have learnt more about the subject.

Qualification-based streams may be defined by entry pathways into the degree, such as the Renaissance Program [91], which provides multiple entries into a CS School in order to attract Humanities, Engineering, Science and Multimedia students. It delivers the main content of CS1 with four options: Java, C, Fortran/Matlab, python/Perl. The philosophy behind the scheme is to allow the same first year outcome regardless of the entry route; this can encourage the students to choose and continue with CS.

At some institutions registering for an honors degree is not automatic. One case study provided by our participants explains a scheme whereby High School students may win scholarships, financial benefits, the chance to undertake some specialized courses and an appropriate project that, subject to successful completion, allows them to graduate with Honors. Another pathway to an Honors degree in such an institution is for high achieving students who elect to enter the Honors program to identify the extra courses required and to attend extra labs and lectures as necessary.

It is occasionally the case that an outside body may fund the top stream of students. For example, the Raikes program [90] allows 30 talented students per year to enroll into Computer Science or specialized Business and Computing Engineering classes. Students benefit from a cooperative learning community living on campus and progressing as a group, taking special versions of classes and finishing with a two-year project in a design studio, offering them the opportunity to work on projects contracted in from external companies.

3.3 Tangible Results within a Context

Context is recognized as an important issue within CS education [37, 41]. Using a context allows educators to focus on a subject that is important and familiar to students while learning new concepts; it provides a motivational tool and offers a wider playing field for experimentation and engaging students in the educational process. Using motivational learning context has a strong potential to improve student success in introductory CS courses and, where applicable, increase student motivation to remain in the major. The ACM Computing Curriculum (CC) recognizes the role of mobile computing, games programming and robotics in today's world as well as in the CS discipline.

Academics employ a variety of techniques within a range of CS courses that are designed to allow their students to see the immediate, tangible results of their work; students attempt tasks that they find engaging and perceive as relevant to real-world applications.

The use of hands-on tools and techniques provides enough flexibility for academics to manage varying levels of student experience and background; the implementation of concepts is central to the learning process. To quote Sophocles, "One must learn by doing the thing; for though you think you know it, you have no certainty, until you try." It is very important to provide students with an opportunity to apply theoretical concepts in practice. For example, it is one thing to tell students about the computational complexity of an algorithm, but quite another to have the students apply the same algorithm to problems of differing sizes to see firsthand a significant difference in the execution time by a device with a relatively slow processor. It is one thing to tell a student about the properties of an algorithm exploring a game search space, but quite another to have students observe their implementation run out of the small amount of memory on a mobile device. Through such experiences, students gain a much deeper appreciation of the design issues at stake. Using hands-on experiences with real-world devices as a learning context in introductory CS courses aims to provide a simple and elegant means to motivate students and communicate the diversity and power of many advanced CS areas in a manner that engages students in experiential education.

Current research indicates that more participatory learning methods such as those used in graphics programming, robotics and mobile application development can level the playing field for different types of students. For example, CS and Engineering have historically been less accessible for female and underrepresented minority students, and as a result, these students are underrepresented in most CS and Engineering departments in this country. A shift to a learning environment which values interactivity, cooperation, and collaboration can result in female and minority students feeling more comfortable and, by extension, can lead to greater persistence and success. In fact, several studies have shown that these learning methods more closely match the learning styles and preferences of women and minorities [10, 13, 64, 100, 114].

Games development has been successfully used as a CS learning context at many levels. Young people have a special affinity to mobile gadgets; they would find it difficult to live a day without using their mobile phones to talk, check email, send text messages, and play games. Using mobile ubiquitous devices in the educational process adds a social dimension to the success of this learning context – students are able to see a connection between the technical material and their everyday lives.

3.4 Peer Mentoring

Peer mentoring refers to "students learning from students". This can take many forms: for example, a set time every week where first-year students can obtain help on homework from more

advanced students who are paid for their efforts. Another example is "Supplementary Instruction" in which the more advanced students who tutor get credits toward an extra-curricular certificate for volunteering rather than payment. A third example is the use of discussion boards such as those provided by learning management systems (e.g. Moodle). On these boards students ask and answer questions related to homework assignments. These questions can be anonymously posed. Another possible interpretation of peer mentoring is cooperative learning inside CS classes in which students work in groups on specific homework problems. This group discussion often clarifies an individual's understanding of the CS concepts exercised in these problems. Instructors can monitor students' progress on these problems and intervene with "mini-lectures" when it appears that peer learning has not been completely effective.

A risk in peer group learning is that individuals may not be able to accomplish individually what their group can. This is an assessment problem for the instructor. One way to minimize this risk [76, 112] is to conduct individual interviews of each member of the group after the group work has been submitted. A risk with first year students going to more advanced students for help is that the more advanced student will simply do the homework exercise of the first year student. This risk can be minimized by proper training of the advanced students. On discussion boards complete solutions to a homework exercise can be posted by competent students. In addition, anonymity allows students to post rude and inappropriate comments. As a consequence, posters may withdraw from using this resource. Top students can also "show off" by posting difficult questions on the board which tends to demotivate less able students. All these risks can be minimized by moderating the discussion board [7].

The mentors are often "top students". Top students tend to feel that helping less able students ensures that the course as a whole can progress more quickly and, hence, are also beneficial to them. They believe that explaining things in detail to others cements their understanding of the field and gives them a feeling of "helping others". Of course, not every top student is motivated to help others struggling with a topic they find easy. Also sometimes, students who have really struggled coming to terms with the material are better able and really want to explain it to other students [34].

A case study taken from one of our participants suggests that senior students can be encouraged to lead technical sessions and help prepare training resources and sample exercise questions that can be used to educate junior students. This form of peer tutoring has four major benefits:

1. Forcing senior students to solidify their understanding of the material: teaching others requires a concrete understanding of the material and explaining concepts reinforces ideas in one's own mind,
2. Creating interest and a sense of contribution for the senior students (a controlled avenue to show off their understanding), creating a sense of community among the cohort of students,
3. Providing senior students with valuable teaching experience, and lastly,
4. Reducing faculty workload with regard to developing and delivering technical material, thus freeing time for renewing the program and setting high-level direction.

Peer learning, in all its forms, reduces the amount of instructor time spent both in lecturing and answering questions. This counteracts some of the time taken to moderate discussion boards and interview individuals whose work is submitted [45]. An obvious cost of providing paid student helpers is their wages. But if only one student who would have failed passes as a consequence of this help, this has more than recouped the investment.

3.5 Self and Peer Assessment

Students assessing the work of peers has been shown to support learner autonomy and self-regulation [12, 43, 97]. It provides opportunities to develop the skills and abilities required to critically appraise one's own work in the context of providing assessment of work done by others. This feedback complements the feedback given by instructors.

Possibly the strongest argument for the use of peer (and self) assessment is that it demands significant student engagement with the both course content and the assessment criteria. As noted by Nicol [81], students cannot be passive when giving feedback whereas they can be passive in using the feedback they receive. This type of activity can help students to more fully understand academic expectations of 'good performance'; lack of such understanding is a significant factor in student under-performance. Students engaged in peer review typically come into contact with several examples of different solutions to the same problem they are solving. This facilitates an understanding of different approaches to assignments and the different ways that high-quality work can be produced.

There has also been research that suggests that the use of such activity can promote the social or collaborative aspects of learning [64, 108] such as interaction between students, between students and teachers, and the development of learning communities. It can also lead to an enhanced motivation to learn and improved time commitment to study.

3.5.1 Practices

Peer and self-assessment has a role in promoting higher cognitive skills and enhancing student learning, as well as employability skills. This has often been within the context of collaborative group activities, although some recent work (e.g. Nicol, [80]) has suggested routinely incorporating peer review into individual assignments by students. In a collaborative context, students engaging in team programming exercises are asked to appraise both the technical and non-technical performance of their peers.

Another obvious benefit of peer assessment is that, with appropriate management, much of the workload associated with the provision of immediate feedback is transferred from academic staff, thus allowing them to engage more profitably with specific groups of students. While manual administration of such a process for large numbers of students will be time-consuming, and therefore impractical, there are a number of software tools (e.g. [45]), which can be used to automate the various sub-tasks will facilitate scalability of the activity to arbitrary class-sizes.

3.5.2 Problematic Issues

Clearly, appraisal of student's work by their peers could be fraught with difficulty. Students often feel unable to engage in the activity without clear guidance on both the nature of the assessment criteria and their applicability to the exercise in question. This itself requires that the activity be scaffolded by auxiliary instruction, which may be time-consuming for the tutor. Students themselves often articulate the fear that affective reactions may color an impartial assessment of their work by other students. Some studies show a tendency for high achieving

students to be more self-critical and so award proportionately lower marks. These issues are not superficial and may well impact significantly on the student's appreciation of the reviewers' responses which, in turn, affects the assimilation of learning associated with the feedback. Student perceptions concerning the validity of criticism can, to some extent, be ameliorated by careful analysis of the feedback itself and the software tools that facilitate large-scale peer-review are often able to identify statistical outliers in grades that can then be moderated by tutors. However, a different response to this issue is often to use the reviews as formative feedback on a first draft of the assessed material, rather than as a summative grade. This allows the reviewee to incorporate any criticism back into a subsequent submission of the assignment. An alternative approach is to also allow the reviewee some role in assessing the quality of feedback.

3.6 Competitions

National and international programming competitions such as the ACM International Collegiate Programming Contest (ICPC) are widely used to motivate students. Respondents to our survey (mostly from Australia and the USA) cite the ACM International Collegiate Programming Contest (ICPC) as the most popular programming competition. Competition teams practice advanced programming, data structures, algorithm analysis, mathematical modeling, discrete mathematics and computational geometry.

Competitions can, however, inspire students at all levels. Within a programming course they provide an opportunity for students to test their ability in designing, understanding and implementing code. Competition can also be the spur that pushes ordinary students to achieve much more than in classroom situations and participation in competitions enhances a graduates' CV.

Key reasons why university teachers use competitions when they are teaching students to program include: testing students' ability, design, understanding and implementation [24]; students strengthening their basic programming skills and insight into practical problems [60, 86]; learning effective teamwork and communication [18, 24, 60]; and stimulating student enthusiasm.

Not only can competitions and training programs inspire students to learn, they can also yield rewards for the university when teams do well in the regional, national, or international level. This publicity can attract high school students and graduate students to study at the host University.

Material from competitions, such as the ACM ICPC, can be integrated into degree courses such as Algorithms and Data Structures, using past problems to highlight concepts and for assessment purposes. The training program can also seed capstone group student projects to develop support software.

Competition teams are usually mentored and trained by one or more Faculty member. There is a high time factor cost associated with this, which needs to be recognized as a service activity that is part of the department's overall strategy.

Other programming competitions include Microsoft Imagine Cup [77], IEEEExtreme [53] and TopCoder [102]. Rosenbloom reported that establishing an in-class competition was "a great, motivational, educational and engaging break from their usual routine" [96]. Moving away from coding competitions O'Leary [84] presents a poster competition as a way to motivate students.

Participating in competitions is not motivating for all students. Some will not enjoy that type of pressured experience and some do not like finding out that they are not the best.

3.6.1 ACM ICPC

Institutions that enter teams in the ACM ICPC competition [1] support their teams with technical sessions and practice competitions that extend the material taught in the standard program. Sessions increase in difficulty over the course of the year: initial training sessions focus on preliminary skills, competition strategies, and team-work essentials; later sessions focus on advanced material. Academics are invited to contribute by providing expert advice on technical topics in their areas of specialty.

One Faculty mentor for this program notes that:

The success of the training program is due to two core principles: challenge and fun. The material presented as part of the technical components of the training program is advanced in nature, exposing students to some difficult algorithmic concepts not presented in their regular undergraduate computing courses. This advanced material provides a challenge for even the very best students; students who may have become bored or disillusioned by the slower pace of the traditional undergraduate programs and are seeking something to challenge or drive them further.

3.6.2 TOPS UK

The Teaching Over-Performing Students competition also complements the usual scenario of helping strugglers and focuses on the other extreme of the cohort – high performing students [23]. The competition involves teams of six students from each participating institution and is split into two sections: designing a challenge for other student teams to attempt in pairs; attempting the challenges designed by students from the other institutions. Four students from each team pair up to attempt the coding challenges; this allows students with commitments or who are reticent about competing in the programming stage of the competition to join in, as well as allowing for drop-outs.

The teams are given the brief to design a challenge that can be undertaken by a pair of students sharing a laptop within the timeframe of 1-hour. The challenges must relate to a specific scenario such as "something useful for a group of students attending an event in London". Even the process of choosing teams is worthy of note. Some students push themselves forward because they want to achieve for themselves, others will nominate the strongest students in their group in order for their own institution to have the best chance of winning.

3.6.3 CSE Day Nebraska

CSE day is a high school competition that has run for more than 10 years and is modeled after the ICPC. It is used for recruitment, and the winning teams are awarded scholarships into the Nebraska degree program.

3.7 Research Experiences

The integration of research into undergraduate teaching is a theme in many university CS departments. The definition of "research" activities is rather broad, covering both software development and traditional research projects. A defining characteristic that links both strands is that the learning is student driven: "students should be seen as producers, not just consumers of knowledge" [48]. The Scottish Higher Education Enhancement Themes has reported on research-teaching linkages in Computer Science [52]. Their report identifies many different types of suitable research project; documents student and academic attitude to research; and presents case studies of successful schemes in Information and Mathematical Sciences.

Jenkins and Healy [48] classify four ways in which undergraduates may engage with research and inquiry across many disciplines:

1. research-led: learning about current research in the discipline,
2. research-oriented: developing research skills and techniques,
3. research-based: undertaking research and inquiry,
4. research-tutored: engaging in research discussions.

University-based research enrichment programs are typically offered to top-performing students. Such research programs are often outside the formal curriculum, e.g. summer enrichment programs [62]. In a number of countries these summer internships are sponsored (e.g. NSF REU, DAAD RISE) to provide scholarships and living expenses for the students undertaking them [9]. Students are supervised and mentored by academic staff. The one-to-one supervision of research activities is deemed to be an important benefit of the experience. Many of these research projects are interdisciplinary, which can be particularly motivating for students [48, 62].

Academics and universities maintain active partnerships with industry and projects with industrial (real world tangible contexts – see section 3.3) links can be initiated by a Faculty member or by an industry partner and may be undertaken by individual students or groups.

Research experiences can begin as early as the first year [43] and may be integrated throughout the degree program [97]. In the UK, Sweden and Australia, undergraduate honors degree programs include a research project assessed by dissertation [12].

There are many advantages and benefits to supporting undergraduate research. For example, academic departments benefit from the reflected glory of successful projects which can be reported on web sites and in news articles. Student successes in research projects help encourage and motivate other students as well.

There are two major disadvantages to offering research experiences to undergraduates: one-to-one supervision takes up a significant amount of academic time; students need to be paid some sort of living expenses either at an hourly rate, by an honorarium or scholarship. For these reasons, most departments offering such experiences do so only to selected students.

3.8 Extra Curricular Activities

Extra curricular activities can provide a welcome diversion for students with an interest in aspects of the course that are not met by the curriculum. An example of this is cited by one of our respondents: a programming club.

The programming club is open to all students within the university, as well as High School Students. It is run by students and an academic who maintain an external website [94] and the only requirement for membership is a keen interest in computer programming. The club meets once per week in a lab that is specially reserved for them. The Programming Club provides a stimulating, cooperative environment where students can interact and learn from each other by solving various programming problems.

Some objectives of the club are to:

- Gain and develop good programming skills and general problem solving skills,
- Learn how to work in a team
- Develop programming solutions using various data structures, sorting algorithms, arithmetic, algebra, graph algorithms, and number theory,
- Enhance students' career prospects (it looks really good on a CV),
- Start a potential research project which may lead to postgrad studies,
- Learn from senior and more experienced students,
- Have fun with other fellow students!

3.9 Motivation Anti Patterns

As well as identifying interventions and approaches that help to improve motivation, we have identified several motivation *anti-patterns*. An anti-pattern is an approach that demotivates students and so should never be used. In the context of motivating all our students, anti-patterns include approaches that motivate one group of students whilst strongly de-motivating another.

3.9.1 Doing Nothing

The easiest approach is to treat everyone in the class in the same way. But this isn't fair to anybody – even the instructor suffers when students fail and drop out.

3.9.2 Sink or Swim

In this anti-pattern it is argued that students are adults and so it is up to them to find out what is required of them, and then how to achieve that and whether they have done so. The working group argues that teachers should take responsibility for ensuring that expectations are made clear, and that students are taught how to learn as well as what to learn.

3.9.3 Teach to Only One Section of the Class

Some instructors elect to teach to one section of the class only: the best students, the middle students or to focus on the weakest students. This approach may be justified as teaching only the students "we want to have" or focusing on those "we know we have". Alternatively, instructors may focus on only one learning style, ignoring the needs of other types of learners. In each case, the students who are not being catered for are likely to lose interest and either drop out or under-achieve in the course. We argue that the diversity of student abilities and preferences needs to be addressed by a mix of different teaching approaches.

3.9.4 Humiliating Students

Telling students they aren't as good as they think they are, or constantly reminding them about a silly error they once made (when they were having a bad day / ill / hung-over / suffering a bereavement) is not the way to motivate anybody.

3.9.5 Scare Tactics

Many Computer Science courses have high failure rates and meaningful learning requires significant effort on the part of the learner. However, since anxiety inhibits learning, negative motivation techniques are not an effective strategy. Negative motivators include unreasonable workload, exam questions that are badly matched to students' level of development, and taking high failure rates as a given and telling the students that will happen (whatever they do).

4. WHAT THE STUDENTS SAY

There are a number of significant challenges when attempting to motivate all students: the diversity of students' prior experience; preferred activities; and their attitude towards problems they encounter. Another major hurdle is that every student is motivated by something different. In Section 3 we presented

academics' views on techniques they have used to motivate their students. In this section we present the views of the students. The quotations presented here are taken from course evaluations from the UK, Australia and the USA. They represent the voices of cross sectional mix of students (including female, mature, international) taking CS degree programs.

Differing student motivations are evidenced by the contradictory statements made by different students taking the same course. To put these differences into context we need to consider the diversity of students' backgrounds and their different reasons for taking Computer Science courses.

[My first choice unit] got cancelled and I needed a unit so I chose this one. This is by far the best unit I have ever done, and it was all an accident.

I liked things about programming since I was young, so basically no opinion. The lectures and labs are just fine for me.

Honestly, I just do NOT like programming at all! It's my parents force me study this boring subject, because they thought I would have a great job with high income after graduation. – Female student, whose parents work in IT and used to be programmers.

Motivating all Computer Science students is challenging because those entering Computer Science degrees have a wide range of prior experience in the subject area. For example, entrants to a first year programming unit included:

No programming background

Picked up programming as a hobby about...11 months before I started this course

This is my 5th semester at uni ... did some C programming in other units, learning c# at same time ... playing around with the xna dev environment for games

Had 2 years off in which I worked in web development, programmed interactive websites (PHP, JS, SQL) and did work creating a relatively simple PHP project tracking system. During that time I also taught myself a (very) little bit from many programming areas (Python, C++, VMC, Programming paradigms etc.). This is my first semester (Bachelor of CS).

4.1 Differentiated Delivery

The need for differentiated delivery can be appreciated from the wide range of prior experience of students in Computer Science. Differentiated assessment tasks appeal to top students, who can demonstrate their skills by attempting the most complex challenges. Weaker students, however, may still struggle with the basic tasks that are set.

I enjoyed implementing features that were more advanced than what the assignment required (e.g. drawing the "Don't Bother" flags) and optimizing important methods.

What did I like most about this unit? Absolutely the labs. The discrete marking style was excellent motivation to aim for the harder tasks.

This course nearly drove me to insanity. I put so much time into this subject – time that was so desperately needed in my other subjects – with almost always fruitless results

I can sing a song, but I can't compose one. The same deal with Java

4.2 Streaming

Students quickly become aware of the wide range of abilities in their classes, and their ranking within the group. From this they may then identify a need for streamed classes or differentiated delivery.

I know it is more work and a pain for you, but the unit needs to be split, possibly into 2 units: One for total beginners and one for people with knowledge but no formal education. The second unit could run faster, requiring more work to keep up

The pace was a bit slow for me ... I still felt the course was worth my time, and the assessments gave me a bit of room to think for myself (in the extensions) and construct something I was proud of.

Students who do not plan on majoring in computer programming or engineering should not be required to take the same class as those students who do plan on majoring in programming orE.

The problem with this class is it is a very hard subject to grasp. Either you get it or you don't. It's very hard to understand some of the things taught. I always said this class was like someone handing you tennis shoes. The teachers and book told me all about this shoe they told me what the laces were and the tongue and the sole. Then I would get to lab and they would say tie the shoe. I could tell you everything about the shoe, but I have no idea how to tie it. It is the same as C programming. I can tell you what a struct and an array is but I have no idea how to properly implement them.

4.3 Tangible Results Within a Context

Hands-on projects that allow students to see immediate tangible results for their work have been developed in many institutions and are widely, although not exclusively, appreciated by their students. A major aspect of projects with tangible results is that students are motivated by activities that they see to be relevant to their future careers.

The assignments [were the most enjoyable part of the course] because it was good to see the coding I wrote actually do something.

I found the last assignments, ... very interesting, as the programs were 'real' and required some careful thought and problem solving.

I just really hope there could be more opportunities for me to create some real applications.

This unit was boring ... If this course was the first programming I'd ever experienced, I would drop computer science ... doesn't get the student interested because we never end up with a standalone application that we can run

Nothing else, more practical work please! I wanna build up something that people may actually use. I'm tired of doing some work only for the purposes of checking if we could master the theories or not.

Sometimes I just feel that I'm wasting my time, learning stuff which would never be used in my future career. It would be great if I knew what the real work looks like in advance.

4.4 Peer Mentoring

Students usually appreciate mentoring from their peers and from tools. However, working with peers can also be frustrating.

I was also encouraged ... to help train other students. This provided me, and other strong students involved, with invaluable teaching experience.

Every time ... I do teamwork, there will always be some strugglers hidden behind, doing nothing. Sometimes I just coded for a teamwork all by myself, which I do NOT think I could practice my

cooperation ability. So I'm thinking, maybe the real managers from IT companies could lead us, showing what the REAL TEAMWORK is.

4.5 Self and Peer Assessment

Many students benefit from using tools for self and peer assessment.

[The best aspect of the unit were] the test cases and the help [forum] meant you didn't have to stop when you hit a wall. It just kept the right amount of difficulty in the unit.

I really liked the codelab within this course. I thought it did a good job of applying what we learned in the lecture. Also most of the labs did a good job as well.

[The lecturer was] really helpful outside of the class. The drill exam [a second opportunity to retake an examination] really help for understand what is wrong of the exam, it is good that the drill exam as a bonus credits.

4.6 Competitions

Most students find programming competitions motivating; they value the professional and team work skills that they gain from training and taking part in competitions.

Overall, participating in [the ACM ICPC] training program was one of the highlights of my time at [university] and I've recommended it to every CS undergrad I've met.

A very inviting and rewarding experience. Truth be told, I doubt I would still be doing my CS degree if it weren't for taking part in the ACM.

The ACM training offered a highly engaging environment for reinforcing both my own studies and extending less experienced students with material they would not otherwise encounter.

I believe that the program provides students with experience that is not normally available to students through coursework.

I joined the ACM-ICPC [after joining the] challenging but extremely entertaining training course. It was such an important and thrilling learning experience that I attended every year after that.

The training program is structured so well that it is no longer simply a matter of learning how to win, but enjoying the learning while we do it, which to my mind makes the information stick a whole lot easier.

I liked that we were supposed to work at our natural pace and that we had to think.

It was really intense, but great fun.

Training taught me to identify cases where knowledge learned in my degree could be applied to real problems and inspired me to learn above and beyond my usual undergraduate studies.

The structure of the competitions and discussions encourages students to learn from each other. These activities are particularly rewarding, combining a socially-supportive environment with the acquisition of widely-applicable problem-solving techniques.

Before I participated in the ICPC I was a student who was rather isolated and, at times, quite uncomfortable working with my peers. As a result of the training, I became far more effective at working with others and formed mutually beneficial relationships with other motivated students.

Working together was great, everyone worked amazingly well in teams…I felt I learnt an enormous amount from the activity, and I thank you sincerely for making it available to us.

5. DISCUSSION

The intrinsic motivation for motivating all students is obvious: educators aim for students to experience "higher learning" in their courses [5], and high student motivation is an important pre-requisite for higher learning. There are also many extrinsic motivations for motivating all students. Most institutions are aiming to reduce both their failure rates and their withdrawal rates for students. Furthermore, even if they do not leave, demotivated students may simply switch off and so achieve a low final degree grade instead of the high grade they were capable of. There are hidden curriculum benefits to motivating students in that the advertised achievements of top students (e.g. in competitions and research projects) are motivating the other students. They introduce an organizational culture of excellence. Successes can also be used for attracting and retaining students. Top performing undergraduate students are an asset to their departments on graduation, either as employees in industry or as PhD students.

Solving the problem of motivating all our students is particularly challenging for two main reasons. First, Computer Science cohorts are extremely diverse: students have different backgrounds and expectations, learn at different rates, and are motivated by different types of activities. Second, most departments are under pressure to reduce their course delivery costs, but techniques to motivate all students usually increase costs because they require the development of new material, running multiple level classes, and offering differentiated material.

Three major categories of concern emerged from the analysis:

1. Program Level – providing mechanisms at the degree program level, such as a series of honors courses, etc.,
2. Course Level – techniques and tools used within a single course (Table 1),
3. Extra-Curricular Issues – strategies outside of the degree program.

Alongside issues relating to the level at which an intervention strategy should be employed or considered were issues relating to different sectors of the cohort – not split by ability or aptitude for the subject.

5.1 Issues Transcending Teaching

There are two further, overarching, issues that require consideration, whichever level of intervention or change is being considered. Dealing with all our students does not simply mean all ability levels, but also students of differing social backgrounds. The other issue that cannot be ignored is that of plagiarism.

	Lecture	Laboratory / Assignment	Other
Classroom Approach	• Studio Lecture in Lab	n/a	n/a
Material	• Engaging/Relevant	• Challenge	n/a
Tools	• Classroom interaction	• Automated program assessment	• Peer study tools • Customized problems / quizzes
Mentoring	• Peer	• Pair Programming (Peer) • Tutors	• Tutor

Table 1: course level techniques

5.1.1 Dealing with Under-represented Groups

According to the United States National Science Foundation, the major underrepresented U.S. groups CS and Engineering include women and these three minorities: Blacks, Hispanics, and American Indians [82]. Underrepresented groups in Europe and Australia also include women [2, 49].

Generally, students from underrepresented groups have lower self-confidence, and women in computing particularly experience this challenge [69]. Current research indicates that women tend to perform better in collaborative environments [10, 13, 64]; pair programming has been shown to be especially beneficial [110]. Students belonging to underrepresented groups tend to perform better when paired with the team members from the same group as well as when they have a mentor or a role model from the same underrepresented group [14, 19, 50]. While role models and teammates from the same group can be helpful, it is important to acknowledge that a mixed group of men and women is best for overall team performance for all students [83]. Therefore an ideal team would include more than one female along with other males. However, considering actual enrollment percentages, achieving the ideal team may not feasible.

Helping female students overcome their lack of self-confidence can reveal their natural abilities and raise their achievement level. The same is true of interest. The curriculum has been shown to bias toward the majority demographic in the student population [47]. Once the interests are discovered and taken into consideration in the curriculum without excluding any particular group, then the students can realize their capabilities and rise to their natural level of achievement.

5.1.2 Plagiarism

We tend to think of plagiarism and collusion in Computer Science as one student copying another student's code and submitting it as their own, or two or more students working on the same assignment together as a group project, and then each submitting the same work as their own individual assignment. In the context of large classes, and with many international students, some students expect that this will not be discovered [117]. However, when a student posts their assignment to a website, such as Rent-a-coder, and offers to pay the least amount of money for their completed assignment [57], or offers to pay a private tutor to complete their assignment [116] this also falls under the same banner of plagiarism and collusion. It can be argued that this is a more serious form of plagiarism, and should be considered differently.

Some universities distinguish between "accidental" or "minor" plagiarism, which occurs when a student may not know or understand the problems associated with submitting the work of another as their own. The remedy for this kind of plagiarism is often a warning, or process of education such as a workshop or direction to read material on how to reference correctly, in particular how to reference code appropriately [46]. However, when students deliberately set out to cheat on their assessments, and steal or buy or coerce other people into carrying out their work, this is the kind of case that requires further investigation as to the cause.

In our interviews, we have identified various approaches to investigating this problem. One approach is to interview the student and find out why they have taken this path. One scenario that is relevant here occurs when students are under time pressure from outside work commitments, where the student has to undertake paid work to be able to afford their studies; in countries such as Australia, many international students find they must work in order to support themselves during their expensive study abroad. Some students, however, choose to undertake the paid work because they find their study easy and even boring, and think they will benefit from real world experiences, and paid employment. These students can easily undertake too many hours of work per week, in the mistaken belief that their university work is easy now, and will continue to be so. One risk of making initial work easy to aid the struggling novice, is that average to middling programming students are encouraged to believe that they can undertake more outside activities, such as paid employment, without affecting their grades. When they realize they cannot, often the day before the assignment is due, they can take desperate measures to submit their assignment.

Other triggers that have been cited for students submitting assignments to an external website or newsgroup or forum, are the isolation of an individual, either through physical location (distance education) and students for whom the language of instruction is not their mother tongue; students who struggle through lack of personal support from the university.

The risk of aiming delivery of education at one particular cohort, is the isolation of another cohort. The level of plagiarism may be an indicator of students being disengaged, and not recognizing the relevance of their assignment work to learning, or indeed the necessity of why they are undertaking the learning in the first place.

5.2 Mechanisms

A number of pedagogical interventions have been made in response to the issues articulated in previous sections. Some of these, such as peer mentoring, and peer and self-assessment, try to circumvent many of the difficulties inherent in teaching large classes by using the students themselves to provide some of the functions usually associated with tutors (e.g. reviewing work, giving feedback and providing continuing academic support). While this may provide a range of benefits to the instructor in terms of reduced workload, it does not specifically address the problem of the alienation of various parts of the ability spectrum when confronted with a lecturer who teaches to the level of the average student. These include those students who struggle to assimilate the work and also those high achievers who do not find the material enough of a challenge to

sustain interest. A second approach may be to divide the target audience into several groups with a narrower ability range by explicit streaming of classes, or by implicit differentiation using voluntary or extra-credit extension material and divergent assessment practices.

5.2.1 Tangible Results Within a Context

Sometimes it may appear as if educators and students live in two different universes and have completely different ideas of what represents the "real world" of computing. A smart phone or a gaming console, not a desktop PC, is what exemplifies a real-world computing device for many undergraduate students of today. Using a relevant context allows educators to focus on a subject that is important and familiar to students while learning new concepts; it provides a motivational tool and offers a wider playing field for experimentation and engaging students in the educational process. Application of hands-on tools and techniques provides enough flexibility for varying levels of student experience and background, with implementation of concepts being central to the learning process. All the student participants that had experienced contextualized learning thought it a valuable experience and many who had not expressed the wish to do so.

5.2.2 Competitions

Competitions can be good motivators for students. They also provide good publicity for the institutions with victorious contestants. While many students do enjoy competition and have a desire to compete some do not.

5.2.3 Peer Learning

Such mentoring reduces instructor time and, hence, may scale to large classes. Top students, who take on "teaching" roles, often increase their engagement with the class as a result. Any financial costs incurred by such measures are often balanced by student retention. When student mentors are trained appropriately it is often the case that weaker students would rather approach them than faculty with issues relating to early material.

6. WHAT NEXT?

The outcomes from this work do begin to address all the goals identified at the outset and they also continue previous work and form a solid basis for future work in the area. We have identified recurrent themes and linked current practice with current literature. The beginnings of a repository have emerged, but this needs to be expanded.

It is also the case that what works at one institution with a one set of circumstances and a particular instructor with particular ideals and personality may not translate elsewhere without serious tweaking; learners and groups of learners will experience initiatives differently. For this reason an evaluation of the effectiveness of the transferability of the suggestions is now required.

7. ACKNOWLEDGEMNETS

Thanks to all participants, both academics and students, who agreed to be interviewed or complete the questionnaire.

8. REFERENCES

[1] ACM international collegiate programming contest, http://www.csse.uwa.edu.au/icpc/UWATrainingSessions

[2] ACS-W Board, *ACS Survey of Women in IT Report*, Australian Computer Society, 20 September 2010

[3] Anderson N, *Why aren't Australian girls getting into IT?*, http://www.schools.ash.org.au/litweb/gender.html, 2000

[4] Anderson R, Anderson R, Davis P, Linnell N, Prince C, Razmov V and Videon F, *Classroom Presenter: Enhancing Interactive Education with Digital Ink*, IEEE Computer, September 2007

[5] Angelo TA, *A teacher's dozen: Fourteen general, research-based principles for improving higher learning in our classrooms*, AAHE Bulletin 45(8), 1993

[6] Baldwin J, Crupi E and Estrellado T, *WeBWorK for programming fundamentals*, SIGCSE Bulletin 38(3), June 2006

[7] Barnes D, *Public Forum Help Seeking: the impact of providing anonymity on student help seeking behavior*, proceedings of Computer Based Learning in Science (CBLIS '99), Czech Republic, 1999

[8] Barnes T, Richter H, Powell E, Chaffin A and Godwin A, *Game2Learn: building CS1 learning games for retention*, SIGCSE Bulletin 39(3), June 2007

[9] Barker L, *Student and Faculty Perceptions of Undergraduate Research Experiences* in Computing, Transactions on Computing Education 9(1), March 2009

[10] Baxter-Magolda MD, *Knowing and Reasoning in College: Gender-Related Patterns in Students*, Intellectual Development. San Francisco: Jossey-Bass, 1992

[11] Bayliss J and Strout S, *Games as a "flavor" of CS1*, SIGCSE Bulletin 38(1), March 2006

[12] Beck J, Buckner B and Nikolova O, *Using interdisciplinary bioinformatics undergraduate research to recruit and retain computer science students*, proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07), Covington, 2007

[13] Belenky MF, Clinchy BM, Goldberger N and Tarule JM, *Women's Ways of Knowing: The Development of Self, Voice, and Mind*, New York: Basic Books, 1986

[14] Biggers M, Yilmaz T and Sweat M, *Using collaborative, modified peer led team learning to improve student success and retention in intro cs*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[15] Birch M, McCormick F and Haddow J, *Improving Student Progression by a combination of Streaming, Close Attendance and Target Setting*, proceedings of 6th Annual HEA-ICS conference, York, August 2005

[16] Bornat R, *Programming from First Principles* , Prentice Hall International, 1987

[17] Bower M, *A Taxonomy of Task Types in Computing*, proceedings of ITiCSE'08, Madrid, 2008

[18] Bowring JF, *A new paradigm for programming competitions*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[19] Boyer KE, Thomas EN, Rorrer AS, Cooper D and Vouk MA, *Increasing technical excellence, leadership and commitment of computing students through identity-based mentoring*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[20] Bransgrove E, *Teachers understanding of gender implications for learning with computers*, Australian Educational Computing, May 1994

[21] Buckley M, Nordlinger J and Subramanian D, *Socially relevant computing*, proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08), Portland, 2008

[22] Carter J and Boyle R, *Teaching Delivery issues: Lessons from Computer Science*, Journal of Information Technology Education, Volume 1, pp77-90, 2002

[23] Carter J, Efford N, Jameison S, Jenkins T and White S, *The TOPS Project – Teaching our Over-Performing Students*, proceedings of 8th Annual HEA-ICS conference, Southampton, August 2007

[24] Carter J, Efford N, Jamieson S, Jenkins T, and White S, *Taxing our best students*, ITALICS, 7(1):120-127, June 2008

[25] Carter J and Jenkins T, *The Problems of Teaching Programming: Do They Change with Time?*, proceedings of 11th annual HEA ICS conference, Durham, 2010

[26] Carter J, White S, Fraser K, Kurkovsky S, McCreesh C and Wieck M, *ITiCSE 2010 Working Group Report: Motivating our Top Students*, proceedings of the 2010 ITiCSE working group reports, ACM digital library, 2010

[27] Chan CK and Lee EY, *Fostering knowledge building using concurrent, embedded and transformative assessment for high-and low-achieving students*, proceedings of the 8th International Conference on Computer Supported Collaborative Learning, New Brunswick, 2007

[28] Chao C, *An Investigation of Learning Style Differences and Attitudes toward Digital Game-based Learning among Mobile Users*, proceedings of the 4th IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education, Washington, 2006

[29] Clua E, Feijó B, Rocca J, Schwartz J, das Graças M, Perlin K, Tori R and Barnes T, *Game and interactivity in computer science education*, ACM SIGGRAPH 2006 Educators Program SIGGRAPH '06, Boston, 2006

[30] Codelab, http://turingscraft.com/

[31] Denny P, Luxton-Reilly A and Hamer J, *Student use of the PeerWise system*, SIGCSE Bulletin 40(3), June 2008

[32] Die Initiative Bundesweit Informatiknachwuchs fördern, http://www.bwinf.de/competition-workshop/Submissions/11_Cormack.pdf

[33] D'Souza D, Hamilton M and Harris M, *Software Development Marketplaces - Implications for Plagiarism*, Computing Education 2007, CRPIT (66), pp27-33, 2007

[34] D'Souza D, Hamilton M, Harland J, Muir P, Thevathayan C and Walker C, *Transforming Learning of Programming: A Mentoring Project*, Tenth Australasian Computing Education Conference (ACE2008), Wollongong, Australia, January 2008

[35] Edwards S, Using *software testing to move students from trial-and-error to reflection-in-action*, proceedings of the 35th SIGCSE technical symposium on Computer science education (SIGCSE '04), Norfolk, VA, 2004

[36] Farrell JJ, Moog RS and Spencer JN, *A Guided Inquiry Chemistry Course,* Journal of Chemistry Education 76(4), 1997

[37] Forte A and Guzdial M, *Computers for Communication, Not Calculation: Media as a Motivation and Context for Learning*, proceedings of 37th Hawaiian International Conference of Systems Sciences, Big Island, HI, 2004

[38] Fu X, Peltsverger B, Qian K, Tao L and Liu J, *APOGEE: automated project grading and instant feedback system for web based computing*, SIGCSE Bulletin 40(1), March 2008

[39] Google Docs, http://preview.tinyurl.com/5t4wvcg

[40] Guerreiro P and Georgouli K, *Combating Annonymousness in Populous CS1 and CS2 Courses*, proceedings of ITiCSE'06, Bologna, 2006

[41] Guzdial M, *Teaching computing to everyone*, Communications of the ACM 52(5), May 2009

[42] Guzdial M, *Technology for Teaching the Rest of Us*, Keynote talk ITiCSE '11, Darmstadt, June 2011

[43] Hadfield s and Schweitzer D, *Building an undergraduate computer science research experience*, proceedings of the 39th IEEE international conference on Frontiers in education conference (FIE'09), San Antonio, Texas, 2009

[44] Hamer J, Purchase H, Luxton-Reilly A and Sheard J, *Tools for "contributing student learning"*, proceedings of the 2010 ITiCSE working group reports, ACM Digital Library, 2010

[45] Hamer J, Kell C and Spence F, *Peer Assessment Using Arop a*, proceedings of Ninth Australasian Computing Education Conference (ACE2007), Ballarat, Australia, 2007

[46] Hamilton M, Tahaghoghi S and Walker C, *Educating Students About Plagiarism Avoidance - A Computer Science Perspective*, proceedings of ICCE2004: International Conference on Computers in Education, Lisobon, 2004

[47] HEA, *A Study of Progression in Irish Higher Education*, Higher Education Authority, 2010

[48] Healey M and Jenkins A, *Developing undergraduate research and inquiry*, HE Academy, York, 2009

[49] Higher Education Statistics Agency (HESA) United Kingdom, *Student Introduction and Population reports 2009/10*, http://www.hesa.ac.uk/, 2010

[50] Horwitz S, Rodger SH, Biggers M, Binkley D, Frantz CK, Gundermann D, Hambrusch S, Huss-Lederman S, Munson E, Ryder B and Sweat M, *Using peer-led team learning to increase participation and success of under-represented groups in introductory computer science*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[51] Huang T and Briggs A, *A Unified Approach to Introductory Computer Science: Can One Size Fit All?*, proceedings of ITiCSE'09, Paris, 2009

[52] Hughes J, Gregor P, Chaplain M, Coleman G and McIver L, *Research-Teaching Linkages: enhancing graduate attributes Information and Mathematical Sciences*, The Quality Assurance Agency for Higher Education, http://www.enhancementThemes.ac.uk/documents/ResearchTeaching/Info_Math_Science.pdf, 2009

[53] IEEEExtreme, http://www.ieee.org/xtreme

[54] Ihantola P, Ahoniemi T, Karavirta V and Sepp O, *Review of recent systems for automatic assessment of programming assignments*, proceedings of the 10th Koli Calling, International Conference on Computing Education Research (Koli Calling '10), Berlin, 2010

[55] Institute for personal robots in education http://www.roboteducation.org/

[56] International Contest on Informatics and Computer Fluency http://www.bebras.org/en/welcome

[57] Jenkins T and Helmore S, *Coursework for Cash: The Threat from On-Line Plagiarism*, proceedings of 7th Annual HEA ICS conference, Dublin, 2006

[58] Johnson, Johnson, & Smith, *Active Learning: Cooperation in the College Classroom*, Interaction Book Company, 1991

[59] Jplag, https://www.ipd.uni-karlsruhe.de/jplag/

[60] Kearse IB and Hardnett CR, *Computer science olympiad: exploring computer science through competition*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[61] Kölling M, and Barnes DJ, *Enhancing Apprentice-Based Learning of Java*, presented at 35th SIGCSE technical symposium on computer science education, 2004

[62] Knox DL, DePasquale PJ and Pulimood SM, *A model for summer undergraduate research experiences in emerging technologies*, proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, 2006

[63] Lahtinen E, Ala-Mutka K and Jarvinen HM, *A Study of the Difficulties of Novice Programmers*, proceedings of ITiCSE'05, Lisbon, 2005

[64] Liu N and Carless D, *Peer feedback: the learning element of peer assessment*, Teaching in Higher Education, 11(3), 279-290, 2006

[65] Lundeberg MA and Diemert S, *Influence of Social Interaction on Cognition: Connected Learning in Science*, Journal of Higher Education, 66(3), pp312-335, 1995

[66] Luxton-Reilly A, Plimmer B and Sheehan R, *StudySieve: a tool that supports constructive evaluation for free-response questions*, proceedings of the 11th International Conference of the NZ Chapter of the ACM Special Interest Group on Human-Computer Interaction (CHINZ '10), Massy, NZ, 2010

[67] Machado R, Guerreiro P, Johnston E, Delimar M and Brito M, *IEEEXtreme: From a student competition to the promotion of real-world programming education*, proceedings of 39th Frontiers in Education Conference, San Antonio, 2009

[68] Mahmoud Q and Dyer A, *Mobile Devices in an Introductory Programming Course*, Computer 41(6), June 2008

[69] Margolis J and Fisher A, *Unlocking the clubhouse: Women in computing*, MIT Press, Cambridge MA, 2002

[70] Markham S, *Georgia State University Department of Computer Science: a Positive Place for Women and Minorities*, The Young Scientist: A Career Guide for Underrepresented Science Graduates, Spectrum Publishers, 2011

[71] Markham S and King K, *Using personal robots in CS1: experiences, outcomes, and attitudinal influences*, proceedings of the fifteenth annual conference on Innovation and technology in computer science education (ITiCSE '10), Ankara, 2010

[72] Matthiasdottir A, *What Student find Difficult in learning Programming*, proceedings of 5th Annual HEA-ICS conference, Ulster, August 2004

[73] Mazur E, *Peer Instruction: A User's Manual*, Prentice Hall, 1997

[74] McWhorter W and O'Connor B, *Do LEGO® Mindstorms® motivate students in CS1?*, proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09), Chattanooga, 2009

[75] Media computation resources, http://www.mediacomputation.org

[76] Mendes E, Al-Fakhri LB and Luxton-Reilly A, *Investigating pair-programming in a 2nd-year software development and design computer science course*, proceedings of ITiCSE'05, Lisbon, 2005

[77] Microsoft Imagine Cup, http://www.imaginecup.com/

[78] Mobile games, https://sites.google.com/site/csmobilegames/

[79] Moritz S, Wei F, Parvez S and Blank G, *From objects-first to design-first with multimedia and intelligent tutoring*, SIGCSE Bulletin 37(3), June 2005

[80] Morrison B and Preston J, *Engagement: gaming throughout the curriculum*, proceedings of 40th ACM technical symposium on Computer science education (SIGCSE '09), Chattanooga, 2009

[81] Nicol, D, http://www.enhancementthemes.ac.uk/documents/G21C/Assessment_230210.pdf, 2001

[82] NSF, *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2011*, http://www.nsf.gov/statistics/wmpd/pdf/nsf11309.pdf

[83] OECD, *Higher Education: Quality, Equity and Efficiency*, 2008

[84] O'Leary E, *Fancy A Prize? Motivating Students Using Competitions in Formative Assessment*, proceedings of the 3rd annual ICEP conference, NUI Maynooth, 2010

[85] Parlante N, Murtagh T, Sahami M, Astrachan O, Reed D, Stone C, Heeringa B and Reid K, *Nifty Assignments*, proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09), Chattanooga, 2009

[86] Pastor J, Gonzalez I and Rodrigues FJ, *Participating in an International Robot Contest as a Way to Develop Professional Skills In Engineering Students*, proceedings of the 38th Annual Frontiers in Education Conference, New York, 2008

[87] Peckham J, Stephenson P, Hervé J, Hutt R and Encarnação M, *Increasing student retention in computer science through research programs for undergraduates*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[88] Programming challenges, http://www.programming-challenges.com/pg.php?page=index

[89] Project Euler, http://projecteuler.net/

[90] Raikes program http://raikes.unl.edu/

[91] Renaissance Program http://cse.unl.edu/~riedesel/pub/Advise/CSbull11.pdf

[92] Rentacoder, http://www.rent-acoder.com/

[93] Riedesel C, http://cse.unl.edu/~riedesel/pub/cse155N/HW1.docx

[94] RMIT programming club https://sites.google.com/site/rmitprogrammingclub

[95] Rocktest, http://cs.sru.edu/~contest/rocktest

[96] Rosenbloom A, Running a Programming Contest in an Introductory Computer Science Course, proceedings of ITiCSE'09, Paris, 2009

[97] Schneider GM, *A new model for a required senior research experience*, SIGCSE Bulletin 34(4), December 2002

[98] Skiena S and Revilla M, *Programming Challenges*, Springer-Verlag, USA, 2003

[99] Skupas B, Dagiene V and Revilla M, *Developing classification criteria for programming tasks*, SIGCSE Bulletin 41(3), July 2009

[100] Starr C, Bergman D and Zaubi P, *The development and implementation of a context-based curricular framework for computer science education in high schools*, proceedings of the 14th annual ITiCSE conference, Paris 2009

[101] Tew A, Fowler C and Guzdial M, *Tracking an innovation in introductory CS education from a research university to a two-year college*, SIGCSE Bulletin 37(1) February 2005

[102] TopCoder, http://www.topcoder.com/

[103] TOPS http://www.cs.kent.ac.uk/~jec/TOPS

[104] Tremblay G, Laforest L and Salah A, *Extending a marking tool with simple support for testing,* proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '07), Dundee 2007

[105] TurnItIn http://turnitin.com/static/index.php

[106] Universidad de Valladolid, http://acm.uva.es/

[107] University of Nebraska-Lincoln CSE day, http://www.cse.unl.edu/cseday/

[108] Van den Berg I, Admiraal W and Pilot A, *Peer assessment in university teaching: evaluating seven course designs*, Assessment and Evaluation in Higher Education, 31(1), 19-36, 2006

[109] Vincent Chu, *Programming contest management project overview*, http://www.sfu.ca/~vwchu/projects/programmingcontestenvironment.pdf

[110] Weinberg J, Pettibone J, Thomas S, Stephen M and Stein C, *The impact of robot projects on girls' attitudes toward science and engineering*, Robotics science and systems (RSS) workshop on research in robots for education, Georgia Institute of Technology, Atlanta, GA, 2009

[111] Werner L, Hanks B and McDowell C, *Pair-programming helps female computer science students*, Journal of Educational Resources in Computing 4(1), March 2004

[112] Wilkerson M, Griswold W and Simon B, *Ubiquitous presenter: increasing student access and control in a digital lecturing environment*, proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05), St Loius, 2005

[113] Williams L, *Lessons learned from seven years of pair programming* at North Carolina State University, SIGCSE Bulletin 39(4), December 2007

[114] Wirth A and Bertolacci M, N*ew algorithms research for first year students*, SIGCSE Bulletin 38(3), June 2006

[115] Wolz U, Barnes T, Bayliss J and Cromack J, *Girls do like playing and creating games*, proceedings the 40th SIGCSE Technical Symposium on Computer Science Education, Chattanooga, 2009

[116] Zhang M, Lundak E, Lin C, Gegg-Harrison T and Francioni J, *Interdisciplinary application tracks in an undergraduate computer science curriculum*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[117] Zobel J, *"Uni cheats racket": a case study in plagiarism investigation*, proceedings of ACE'04, Dunedin, NZ, 2004

[118] Zobel J and Hamilton M, *Managing Student Plagiarism in Large Academic Departments*, Australian Universities Review 45(2), pp23-30, 2002

[119] Zohar A and Peled B, *The effects of explicit teaching of metastrategic knowledge on low- and high-achieving students*, Learning and Instruction 18(4), August 2008

9. APPENDIX – REPOSITORY

This section contains detailed instructions about how some of the ideas presented in this report have been implemented. They should, however, be treated with caution. A method which works at institution X in country Y with student intake Z may need adaptation to be applicable for student body A at institution B within country C.

The sections provided here are indicative of content created to date. Differences between the styles and lengths of the sections are determined by the nature of the resources we refer to. Some sections include links to on-line documents and support materials. Where these do not exist we describe the approach and provide references to paper documentation.

The repository can be found at:

www.cs.kent.ac.uk/~jec/ITiCSE2011wg

All references in the Appendix that relate to web documentation are links within the repository.

9.1 Study Support Tools

Support tools can be an aid to differentiated teaching (Section 3.1). The tools can be classified into three broad groups: outside the classroom; inside the classroom; evaluation.

9.1.1 Learning Outside the Classroom

- WebWork – automatic generation of program based problems for quizzes or homework [6],
- PeerWise – student collaboration in creating and assessing multiple choice questions [31],
- StudySieve – student collaboration in creating and assessing free response questions [66].

9.1.2 Classroom Interaction Tools

Tools can facilitate differentiation of instruction even in a large classroom setting.

- Clickers – provides a way to engage students in lecture providing feedback through integrated testing-in-lecture, also can be used to facilitate peer instruction,
- Classroom / Ubiquitous Presenter – Ubiquitous Presenter is an extension project from Classroom Presenter [4, 112].

9.1.3 Programming Evaluation Tools

- Online Judge – originally used in programming contests; a recent version is available as a Moodle plug-in and allows for assignment assessment [99],
- Oto – "Oto is a customizable and extensible marking tool which aims at providing timely feedback to students. Based on simple test cases description formats, Oto also includes operations that help students easily test, even "mark", their own programs" [104],
- APOGEE – (Prototype of Automated PrOject Grading and instant fEEdback system for web computing) - used to evaluate many facets of web-development projects [38],
- WebCAT – Tests not only the students' submitted code, but also provides mechanisms for students to provide test cases [35].

9.2 Group Work

9.2.1 Collaborative Book Review

A common criticism of graduates from technical disciplines such as computing is that they emerge from their course of study with poor communication skills in general, and academic writing skills in particular. Attempts to remedy this problem have been made in many institutions and usually take the form of a module on technical communications, which is included somewhere in the curriculum, often with questionable success. Embedding academic writing exercises within a computer science context is sometimes more successful but instructors usually find this type of activity very time-consuming to mark as many students have a limited grasp of the need to include basic punctuation and elementary grammar, and this presented an initial barrier to communication of technical ideas and concepts.

One case study that seeks to help alleviate this problem uses a Collaborative Book Review, in which students are asked to compile a chapter-by-chapter review of an accessible introduction to a computer programming. This particular activity is useful because it allowed groups of students to do some pre-processing on the text of their assignments prior to submission. Iteration of student feedback resulted in significant improvement in the quality of the academic writing: spelling and punctuation errors were reduced through peer review and the dialectic structure of the arguments was generally enhanced [39].

9.2.2 Tools for Peer Assessment

Peer assessment is very useful pedagogical tool (Section 3.5), which students do seem to appreciate, but is notoriously difficult to scale up for large numbers of students. John Hamer, from the University of Auckland, has developed a system called Aropa, which can be used to automate many of the processes that are needed to make peer review and assessment feasible in large classes. These include addition of courses, submission management, setting rubrics and deadlines, etc. [45].

9.2.3 Matlab Programming Project

Assessments that include extension material, which can be studied by more able students, are examples of differentiated teaching mechanism. There are numerous examples but one that has a group element and includes extra-credit material is a Matlab programming project [93].

Rather than completely specifying the project and the required external knowledge base, it is left to the students to collaboratively determine what is required. The scenario for the students is a workplace environment where they are to pool specialized knowledge. They do this by forming small teams and producing a variety of prototype systems from which management will choose the best ones. For evaluation purposes, all collaboration is to be logged. Otherwise, this is a friendly competition among cooperative teams. The resulting prototypes are presented as full reports detailing their features and limitations, testing, design details, operating instructions, references to external sources and research, and collaborations as well as the fully documented code.

9.3 Tangible Results within a Context

9.3.1 Mobile Game Development

Incorporating mobile game development projects into introductory CS courses provides instructors with a relevant learning context to reinforce a number of fundamental topics (e.g. loops or inheritance) while exposing students to a number of advanced topics (e.g. networking or databases). A collaborative project between Central Connecticut State University and Rose Hulman Institute of Technology funded by NSF developed several curricular modules aimed at programing games for mobile devices using Java ME. An overarching goal of this project is to improve student success and satisfaction, and, as a result, decrease student attrition in introductory CS courses at high school, college, and university levels. This project uses casual games, which do not require any special skills to play and can be enjoyed over short bursts of time. Each project module is designed to be completed within a week and result in a playable game that students can upload to their mobile phones [78].

9.3.2 Personal Robots

Students at various levels, K12 through college, find robots very interesting. Educators realize using robots as an earning context offers a new way to motivate and engage their students. The Institute for Personal Robots in Education (IPRE) promotes using robots as a context for computer science education. IPRE is a joint venture between Georgia Institute of Technology and Bryn Mawr College sponsored by Microsoft Research [55]. Georgia State University uses IPRE robots in CSc 2010 (Introduction to Computer Science). Each student is loaned (or is required to purchase for about US$200) a small robot designed by IPRE for the duration of the semester. Each robot is equipped with three wheels, two motors, a variety of sensors including a video camera, and a speaker. In this course students learn how to control the robots by writing programs in the Python language.

9.3.3 Media Computation

Media Computation is an approach to teaching introductory CS courses using the context of manipulating media, such as still images, video, and audio. Mark Guzdial pioneered this approach, which has been popularized by a number of scholarly publications and textbooks. Specific approaches to incorporating media computation principles vary depending on the programming language (Java or Python) and the level of the target course [75].

Some examples of learning contexts offering opportunities for student engagement and motivation:

- robotics [74, 100, 110],
- game development [8, 11, 29, 80],

- mobile devices / computing [28, 79],
- multimedia computing [79, 101],
- social applications (Facebook, Twitter) [21, 71, 85].

9.4 Peer Mentoring

A number of approaches to Peer Mentoring have received considerable academic attention. Peer Instruction, first proposed by Erik Mazur [73], involves the students engaging in classroom activities that require them to apply the main concepts by explaining them to their peers. A related approach is that of Process Oriented Guided Inquiry Learning (POGIL) which was initially used in the context of Chemistry teaching to teach communication and collaboration skills using an inquiry based learning approach.

In addition to these two large-scale pedagogical programmes, many instructors have used peer-mentoring methods in a less formal context.

9.4.1 Groups with Pre-assigned Roles

An example of this approach relates to a CS 1 course in Java in which all programs are graphic Java applets. The class meets each week for two 90-minute sessions in an ordinary classroom (no computers) and one 120-minute period in a computer lab.

Before the class starts, students are broken up into groups of size 2-4 (ideally 3) based on their grades in university math courses or, if that information is not available, on math ACT and self-described prior programming experience. Women are placed into groups in which they are not the minority [58].

Before each class students read a section of the textbook and answer questions on the reading posed by the instructor. These are graded 0 (not submitted) or 1 (submitted) since their purpose is to ensure that students have read the text before coming to class.

In the first ten minutes of class, the instructor describes and demonstrates the Java applet(s) of the day. For some relatively complex CS 1 topics, e.g. recursion, the instructor might also provide examples to supplement the texts. In the remainder of the class period, each group writes on the board the code for the required applet(s).

There are three roles in a group: the driver, who writes the code on the board with advice from the group; the simulator, who draws the applet on the board and synchronizes the picture with the evolving code; and the navigator, who checks syntax, book in hand, and ensures that driver and simulator are truly synchronized. Roles are permuted each class.

The instructor carefully monitors each group's progress by inspecting the evolving code on the board and listening carefully to oral interchanges among group members. Evidence of conceptual misunderstanding prompts a "mini-lecture" by the instructor for a particular group. The instructor also provides hints for those groups whose progress is sufficiently slow. Because of homogeneity, the instructor spends most of the time with less competent groups. It is not unusual for highly competent groups to create working applets without any help from the instructor. Students learn the programming concepts through peer discussion and the group problem-solving effort.

After the class, the applets are tested and debugged in the lab with assistance from the instructor or completed as homework (without assistance from the instructor). The students' programs are graded by the instructor for both correctness and adherence to a predetermined programming style.

With this approach students learn somewhat more that a normal lecture presentation [16]. While students are initially sceptical that a peer learning, student-centric approach can be more effective than an instructor-centric approach, at the end most prefer this pedagogy. Women, in particular, feel more comfortable with a cooperative learning approach and, hence, are more successful with this approach as compared to lecture [90].

9.5 Programming Competitions

Programming competitions can be used to motivate students, although not all students – including the best in the class – are happy to participate (Section 3.6).

Contests may be set up within a single class, between classes at one site or between several schools (excellent for promoting rivalries!) or distributed across many. They can be single round or multiple round, ad hoc or as part of a wider contest such as the International Collegiate Programming Contest (ICPC) [1]. Other types of contests can be found online, for example:

- Bebras [56],
- International Olympiad in Informatics [32],
- Microsoft Imagine Cup [77],
- IEEEExtreme [53].

9.5.1 An ICPC Style Contest

Contests can be quite varied, meeting the constraints and educational goals of the administrators. An ICPC [1] style contest is outlined here:

9.5.1.1 Rules

Here is an overview of the rules for an ICPC style competition. Teams of three students attempt a set number of previously unseen problems within a session of maximum duration 5 hours. They score points for each problem solved. The time taken to solve the problems is noted. An incorrect solution will incur a penalty time. In the event of a tie the team with the shortest time for the same number of points is awarded victory. Undergraduate students may choose to program in C, C++ or Java, whilst high school students are permitted virtually any available language.

9.5.1.2 Personnel

There are several crucial roles that must be filled, but they can be combined if the contest is of a small scale.

- Site Director – overall responsibility
- Problem Writers/Editors – create, develop solutions, verify non-ambiguity and completeness of descriptions
- Chief Judge – overall responsibility for judging, decision is final
- Judges – coordinate with chief judge
- Communications Coordinator – team/coach contact person
- Sys Admin Team – set up and maintain hardware/software for contest
- On-site Volunteers – general assistance

9.5.1.3 Problem Sets

The characteristics of a good problem set include this classic line: "All can solve at least one, no one can solve all." The number of problems is usually between 6 and 12, dependent upon the ability level of the students and duration of the contest. There should be a variety of easy, medium, and hard problems placed in random order. Problems should be "interesting" with a story line that draws attention. Make sure there is a well-constrained solution that is easy to score (electronically, for example using diff, or eyeballing). Problems should not favour a solution in one language over another and they should be thoroughly tested by more than one Problem Writer or Judge prior to the contest.

9.5.1.4 Contest Management Software

- Rocktest [95]
- Programming Contest Environment [109]

9.5.1.5 Training Teams

Training may involve just a presentation of rules and orientation to the programming environment; instruction in the official rules is essential. In addition, you may wish your students to work on content and teamwork using practices, summer camps, special classes, mock contests using online sources [106] can be very valuable.

9.5.2 TOPS

TOPS (Teaching Over-Performing Students) is an on-going UK project which has been led by the Universities of Kent and Southampton in conjunction with a number of other universities in the UK. It is researching how to teach students who arrive with already established programming skills and experience. Issues include maximizing the educational effects and sustaining motivation for each participant [103].

The most popular aspect of TOPS is the competition. This competition involves two major components: designing a challenge for the other students to attempt in pairs; and attempting the challenges designed by the teams from the other institutions. Students find it intense but enjoyable because of the challenging and collaborative atmosphere. The academics gain a deeper insight into both the educational achievements of students from other universities and the aspects which interest their students as evidenced by the challenges the teams create.

9.6 Extra-Curricular Activities

Extracurricular activities may involve the formation of clubs or groups, either to encourage or include students of under-represented groups, or higher-achieving, extension-seeking individuals. Sometimes these groups may require materials to work on to unify or direct their activities.

A programming club is an example of such a group, and the website recommend by one such club for extra programming challenge idea is aptly titled "programming challenges" [88]. Skiena and Revilla also have a useful book of the same title [98].

Project Euler [89] provides another example of a series of challenging mathematical/computer-programming problems that require more than just mathematical insights to solve. As their website states: "The motivation for starting Project Euler, and its continuation, is to provide a platform for the inquiring mind to delve into unfamiliar areas and learn new concepts in a fun and recreational context." Their intended audience includes "students for whom the basic curriculum is not feeding their hunger to learn, adults whose background was not primarily mathematics but had an interest in things mathematical, and professionals who want to keep their problem solving and mathematics on the edge." The project includes problems of varying degrees of difficulty which may be tackled by individuals or groups.

9.7 Engaging Underrepresented Groups

It has been established that the traditional style for teaching computer science has done little to attract and retain women in technology. By making simple changes in the educational techniques and the presentation of the course material, female and minority students can engage and be successful in this field. How to attract and retain women to this field remains an on-going subject of research. One solution is to focus on open-ended projects as well as offer assignment choices. This approach can help lessen gender or racial bias in the coursework. Ideally, educators track, identify, and share interest areas of women and minorities in order to make the study more attractive for these underrepresented groups. For example, "the male students in the robot-based introduction to computing course chose projects that included gaming and sports while the female students created applications that focused on multimedia, relationships and fashion. Both groups learned how to program and enjoyed the class because they were able to apply the technology to areas of interest and relevance in their lives" [70].

9.8 Plagiarism

When an assignment is set, it is important to be clear from the outset about whether students are allowed to use code from elsewhere (including the text book, lecture notes, tutee classes) and if they can, *how* they should reference it [46]. It is also important to explain upfront whether you intend to use plagiarism detection software to identify copied work.

Many universities around the world subscribe to Turnitin [105] which is a software package designed to compare assignment submissions to their repository. This ensures original work "by checking submitted papers against 14 billion web pages, 150 million student papers and leading library databases and publications." Turnitin [105] is useful for text and reports, however, for computer code, there are many other open source solutions.

jPlag [59] is a useful system for detecting plagiarised java and C code. jPlag, however, does not compare student work with code on the internet, merely with other assignment submissions in the same batch. Hence it can detect if two or more students have copied code from each other, or if two students have copied from the same website, but not if an individual has paid a private tutor.

It is possible to search around popular websites, such as Rentacoder [92] and many others to check whether any assignment submissions are posted to these software marketplaces [33].

Finally, it is useful to be explicit about how many hours you expect a student to spend on the work for this assignment, why it is relevant to their learning, placing it in a context and motivating the kind of submissions which are expected.

9.9 Reference List

The reference list that has been created over the last 2 years will also form a section within the repository. We have spent a considerable amount of time compiling a list of over 250 suitable references, both papers and on-line resources. If visitors find references to resources or work that we have not listed they are welcome to send us details and we will add them to the list.