# ITiCSE 2010 Working Group Report
# Motivating our Top Students


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

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

Karen Fraser
School of Computing and Maths
University of Ulster
Newtownabbey, N. Ireland, UK
+44 28 9036 8938
k.fraser@ulster.ac.uk

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

Colette McCreesh
School of Elec, Elec Eng & CS
Queen's University Belfast
Belfast, N. Ireland, UK
+44 28 9097 4816
c.mccreesh@qub.ac.uk

Malcolm Wieck
School of Computing
Christchurch Polytechnic Inst
of Technology, Christchurch, NZ
+64 3 940 8319
wieckm@cpit.ac.nz



## ABSTRACT

It would be unlikely for any first year programming class to be solely composed of novices with the same aptitude for learning. We all have students who arrive with a range of abilities and backgrounds. We have students who barely know their way around a keyboard and those who have programmed professionally; this starting knowledge is also no indicator of learning ability. We need to support struggling students with little knowledge whilst maintaining the enthusiasm of those who are quick to learn, and trying not to demotivate the ones in the middle.

The aim of this working group was to explore the ways in which academics around the world enthuse their high achieving students; seeking things that work and things that don't. This has been achieved by a mixture of literature review and survey of current practice. The synthesis of these forms the basis for the recommendations we make.



## Categories and Subject Descriptors

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

## General Terms

Human Factors.

## Keywords

Motivation, differentiation in the classroom, learning programming.


## 1. INTRODUCTION

This paper draws together evidence based on practice in different countries, which are in turn incorporate differing assumptions and process structures. In the US it is 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 usually have already definitely fixed upon computer science or its related disciplines as a future academic career path. Evidence drawn from different countries needs to be considered in this context, and findings of the research may be of differing value and relevance accordingly.

In the UK in the 1960s only 5% of 17 to 30 year olds received a university education. These were the top performing high school students. Since then and some governments later, the UK government's aim has up to now been that there be a 50% participation in Higher Education by 2010. Currently it stands at 43%. Universities have opened their doors to quite a sizable proportion of the school leaver population. It is for this reason that undergraduate courses are very much pitched at the majority of the group; a similar story can be told in many countries.

Undergraduate Computer Science courses are also often aimed at students who have no prior knowledge of programming or computing, because it is not necessary to have done IT or CS at pre-university level to gain entry to the course. Having designed courses to suit the main body of such students we still have our 5% top performing students to teach, some of whom have previously studied the subject.

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 entered the hallowed doors

of possibly 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-age friends 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.

Anecdotally there is an oft cited typical scenario involving undergraduate students who are ahead of the curriculum before they enroll. It involves them taking an entry-level course and realizing that they already know much, if not all, of the material covered. For some students their belief in both their own ability and the level of the course they are taking will be reinforced by scoring highly on the first few assignments, labs and tests without much, or indeed any, studying. These students then stop attending classes because they assume that all the material will be a repetition of what they already know.

Educators are aware that for the majority of students in this category it is common that shortly after they stop attending, the course moves on to new material, with which they are not familiar at all. The students finally receive a huge shock when they discover, on a midterm or final test, that they cannot answer most of the questions; often such students end up with failing grades.

A number of students arriving with the same level of previous knowledge as the group cited here do attend everything and revel in the fact that the work is easy and have their self-belief confirmed. But how should we deal with the students we have disappointed and disillusioned? We 'lost' them because we did not impress them and failed to challenge them right from the beginning. Once they have stopped attending it may be too late for any kind of intervention.

# 2. BACKGROUND

A significant challenge that faces any teacher of introductory programming is undoubtedly the diversity of the class. At one extreme there will be students who have never programmed before, while at the other there will be students who have many years experience of programming. This diversity does not mean that some of the students cannot benefit from a programming course. Rather, it is simply that their needs are different. A novice needs to understand the mysteries of loops, conditional statements and all the usual programming minutiae. But the experienced programmer can still learn; there is a chance for consolidation, to pick up a new language, or to explore more advanced topics.

The challenge for the instructor is how to provide content suitable for both these groups and, of course, all those who fall between. Handling this diversity is difficult. The temptation for the instructor is often to focus on the novice group and to assume that the others will get by with minimal supervision. This is understandable, but it can be risky. There is a very real risk that the neglected group of experienced programmers becomes bored and disengages from the course. At the worst, they can lose motivation and fail or drop out altogether.

If students don't engage with the material they are not going to expend the effort required to learn [64, 65]; motivation is the key. In 2001 Davis et al [38] observed that the majority of university level courses offer a similar experience to all students. They argued that in the teaching of introductory programming this practice had become increasingly difficult to justify. There have been several initiatives aimed at increasing student motivation and engagement. The use of toys, such as robots [124, 125] has successfully been tried on several occasions. Some less effective strategies, such as musical composition [126] and Judo grading [127], have experienced partial success but may be categorized as gimmicks rather than something that seamlessly blends with the topic being taught.

## 2.1 Helping Strugglers to Success

Much has been written about helping the students struggling with learning to program [6, 18, 19, 36, 48, 54, 68, 72, 90, 91, 95, 96]. Ragonis and Ben-Ari [121] identified areas where students struggle and suggested ways of helping them. Ala-Mutka [1] identified ways in which automation can help.

## 2.2 Boring the Experienced?

Whilst much has been written about helping the students struggling to learn the basics, there is much less literature available about motivating / catering for the students who have little difficulty mastering the techniques and can become bored waiting for the others to catch up. Whilst it may be tempting to state that these students should help their less able peers it hardly seems fair; they haven't enrolled on our courses to become unpaid tutors. It is also not helpful to suggest that these students ware not as good as they think they are. Jenkins and Davey [135] describe an approach to segregating the class and allowing the students who can to attempt more challenging tasks that fulfill the learning outcomes whilst maintaining interest.

## 2.3 Changing approaches over time

Methods of instruction and classroom interaction have been influenced by external expectations and changes across the broader educational community. There has been a gradual shift since the 1990s away from instructionalism towards creating constructivist learning environments [128] and promoting active learning [8, 10, 18, 32, 56, 59, 61, 62, 74, 94, 109].

At the same time many institutions have purposefully revised curricular activities in a manner which they have described as being 'student centered' as opposed to teacher centered (which by association privileges covering curriculum content over pacing the experience of the learner. This move away from teacher focused to student focused has been made explicit through Biggs' work on the SOLO taxonomy [11] an educational framework which has been applied in a number of contexts to the area of Computer Science Education, see for example [22] .

### *2.3.1 External Initiatives and Accreditation*

The work of external organizations such as the National Science Foundation, national funding councils and professional and statutory bodies reflect wider intellectual understandings and in turn influence behaviors in university practice which is recorded in the literature.

NSF funding on the research experience in universities has stimulated initiatives which have provided opportunities for high achieving students to engage in authentic activities where they can

apply their programming skills in a research context [4, 37, 71, 83, 85, 88, 89, 104].

Changes in the way courses are taught and assessed and in the wider structure of degrees have been influenced by the work of accreditation authorities on specifications such as the ACM curriculum, ABET and the UK's QAA subject benchmark statements. These specifications tend to be expressed in terms of expected outcomes in students' understandings / behaviors / competencies or knowledge, skills and understandings.

The Computer Science Education literature reflects the impact of such external changes in what we do and how we do it demonstrating that approaches to teaching and assessing programming have evolved to incorporate response to demands and expectations of external stakeholders such as accrediting bodies and employers [50, 141].

## 2.4 Communities of Practice

The educational literature has given us the theory and vocabulary to identify and discuss communities of practice [140]. Associations such as SIGCSE, ITiCSE and the HEA-ICS act as foci for such communities for Computer Science Education. The debate on approaches to the teaching of programming is well documented with much effort being invested in addressing the needs of learners who struggle to master programming and overcome its associated conceptual challenges [140, 141]. In the UK additional concern for the needs of the more able students has emerged as a component of this discourse.

The HEA-ICS 1-day teaching of programming conferences began in Leeds in 2001. The idea was to bring together like-minded people who were struggling to come to terms with the fact that, despite our best efforts, our attempts at teaching programming were still leading to final year students graduating without being able to write code. For example, in the UK the HEA-ICS [57] runs between 25 and 30 workshops/conferences a year around the UK with Computing Science academics. One of these is specific to programming and has been running for 10 years.

This has created an environment which fosters collaboration and innovation [139]. One example of an initiative that was born via this mechanism is the Teaching Over-Performing Students (TOPS) project.

### *2.4.1 What is TOPS?*

There is no argument about students having a variety of learning styles and requiring different approaches to teaching but we all too often focus on the students who are less able. Discussion will almost exclusively focus on ensuring students keep up, it is only the TOPS research (Teaching Over-Performing Students) that veers away from this scenario and focuses on over performing students.

TOPS looks at motivating and engaging students through competition and through enticing them to achieve.

Independent competitions, such as those run by the BCS, Microsoft and IBM can be used to motivate students – winning looks good on the CV – but the challenge may not fit well with the local syllabus and student knowledge. Activities which are designed in the specific context of an existing curriculum can therefore have greater educational strengths. Furthermore, students can be encouraged to learn new skills and extend those that we consider important for their future educational career.

The competition had to be fun, but it also had to address our departmental learning objectives. Peer observation facilitates the sharing of existing practice within its natural context and also enables a comparison of student cohorts.

The competition component is split into two sections: designing a challenge for the other student teams to attempt in pairs; attempting the challenges designed by students from the other institutions.

Teams comprise five students, four of whom 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 could 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 investigation. 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. RESEARCH APPROACH

The working group set out to identify the broad set of literature which underpins our existing understanding of the factors which influence the design and delivery of the curriculum associated with initial programming, and interventions which have been crafted to address any specific additional needs of learners within that curriculum. The review covered conference and journal publications across the area of computer science education and associated cognate areas which extend across the science, technology, engineering and mathematics (STEM) disciplines.

A brief survey was designed to specifically elicit evidence of current practice with respect to the initial teaching of programming. The survey supported a mixed methods approach by gathering quantitative and qualitative data. The qualitative data was sought to illuminate detail of current practice and to provide an opportunity to investigate attitudes and beliefs held by faculty associated with their motivations and experience of classroom practice.

The survey questions are included in this paper as an appendix. The survey data has been analyzed across three broad sub sections

1. Demographic data
2. Current practice in introductory programming courses
3. Investigating key practices identified in the literature

The findings from the survey and the literature review are presented together in the results and discussion sections below.

# 4. RESULTS

An online survey was created and responses were solicited. A copy of the survey questionnaire forms the Appendix of this paper. Over 80 people provided some kind of response, but only 41 were complete. The complete responses have been analyzed here to provide an indication of current practice and issues.

## 4.1 Demographic Data

The majority of responses were from the USA, but other countries were represented (question 1). Figure 1 shows the locations of the respondents' institutions.

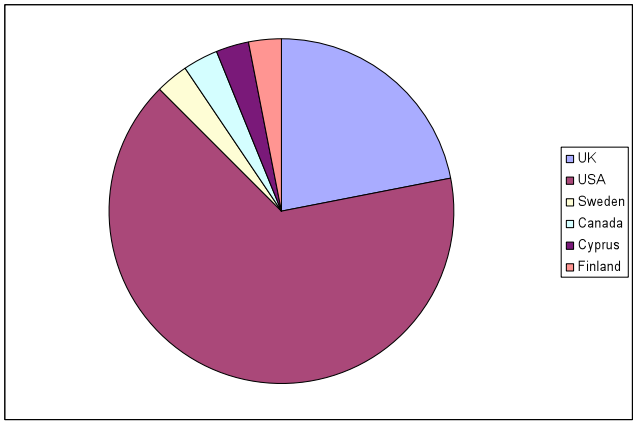


**Figure 1. Respondent location**

Respondents were asked to identify the type of institution and the level (undergraduate / postgraduate / etc) of the students being taught their first programming course (question 2). The institution type was predominantly university (32 out of 41) with seven liberal arts colleges and 2 community colleges represented. 34 of the respondents stated that the first programming course was taught to undergraduate students, with 4 stating postgraduate and a further 3 stating sub-degree level. There were no qualitative differences between the responses from the different categories of institution or course level so the data has been treated as a whole for the remainder of the analysis.

## 4.2 Initial Teaching Patterns

Respondents were asked to indicate how they initially treated the three major discernible categories of students; struggling from the outset; new to programming but coping; experienced programmers (question 3). Each of the textual responses was classified and six discernibly different approaches emerged:

1. peer support
2. differentiated teaching
3. slow pace
4. novelty
5. external motivation
6. nothing (ie the usual mix of lectures, labs and TA support)

The frequency with which different intervention methods were indicated was analyzed, as shown in table 1 (A = strugglers, B = copers, C = experienced), and a number of common patterns emerged.

The prevalence of different approaches was considered from a number of different perspectives. Figure 2 shows the relative distribution of the different methods which were adopted by instructors.

**Table 1. Initial intervention by student category**

| Intervention method | A | B | C |
|---|---|---|---|
| 1) Peer support | 5 | 4 | 0 |
| 2) Differentiated teaching | 7 | 10 | 17 |
| 3) Slow pace | 6 | 1 | 0 |
| 4) Novelty | 4 | 4 | 5 |
| 5) External motivation | 5 | 5 | 5 |
| 6) Nothing | 14 | 17 | 14 |

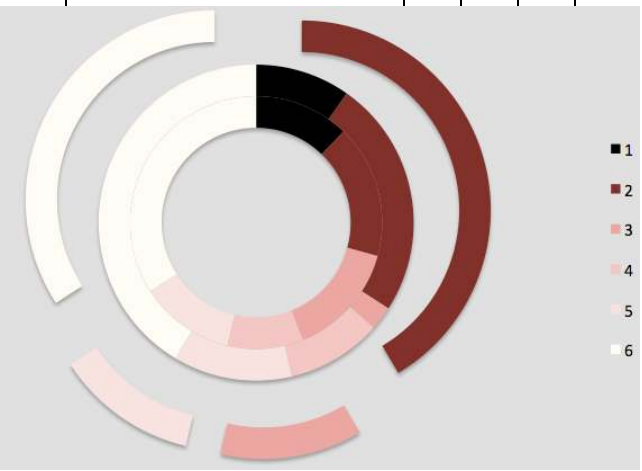


**Figure 2. Distribution of Instruction Methods**

this visualization was derived from the data in table 1, where the innermost ring (a) represents struggling from the outset, the next ring (b) represents new to programming but coping and the final ring (c) represents experienced programmers.

The data was also visualized as a network diagram, figure 3, where the different approaches adopted by instructors were mapped as a vector. This analysis of this data is further recounted below.

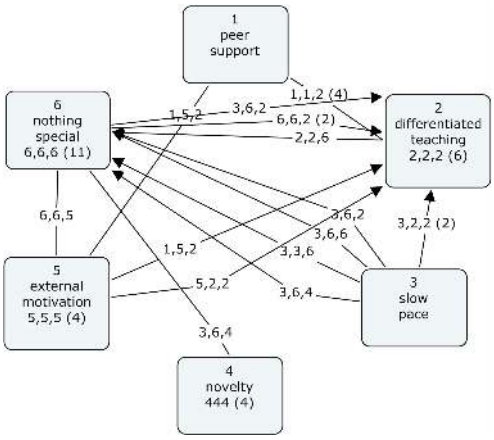


**Figure 3. Network Diagram Mapping Approaches**

### *4.2.1 Business as usual – 6,6,6 (11)*

The most frequent response suggested a familiar mix of lectures, labs and support from teaching assistants. In these responses there was no indication of any special interventions for either the strugglers or the highest achieving groups of students (total 11 out of 41 responses).

*"We do not have a formal support program. We have a few TAs, and the professors provide what help they can, but students who do not take the initiative to seek help slip through the cracks."*

#### *4.2.1.1 Business as usual – plus something for the best 6,6,5 (1) 6,6,2 (2)*

A further three respondents indicated whilst initially that they made no special concessions for the strugglers, they did suggest that they made additional effort for the higher achieving students

Further reflection - it might be worth looking at this in a different way – Twenty seven of the respondents took an approach which did not (at least) begin not business as usual.

*"lots of hands-on exercises, some "challenge" or "bonus" questions"*

### *4.2.2 Active interventions*

There are a number of active intervention patterns identified by the responses. Altogether sixteen respondents indicated that they began with an active intervention, and 13 of those respondents took a predominantly active approach

#### *4.2.2.1 Differentiated Teaching – 2,2,2 (6)*

The proponents of differentiated teaching were mostly committed to it throughout (six out of 41 respondents).

*"We have three tracks through the intro sequence: one for those interested in bio, one for those with some previous experience, and one for those with no previous experience."*

One respondent appeared to apply it for the strugglers (2,2,6) while addressing the needs of the most advanced students with no special interventions.

Lesser clusters – in a number of response patterns four out of 41 responded in the same ways

#### *4.2.2.2 Peer support – 1,1,2 (4)*

A number or respondents chose peer support initially and for the strugglers, with differentiated teaching for the higher achievers.

*[strugglers] "1. in-class activities and time to work on homework with instructor assistance 2. encourage students to work with a partner of similar ability 3. evening workshop run by experienced student trained as a tutor 4. "fun" assignments, typically involving graphics or simple games"*

*[high achievers] "open ended assignments where they can go above and beyond core requirements"*

An additional response which initially privileged peer support - 1,5,2 (1) also incorporated

#### *4.2.2.3 Purposeful beginnings 3,2,2 (2) 3,3,6 (1) 3,6,2 (1) 3,6,4 (1) 3,6,6 (1)*

Another active intervention of note was those who started out with slow pace. They opted for a range of different strategies for the strugglers and top students – 3,2,2 (2) 3,3,6 (1) 3,6,2 (1) 3,6,4 (1) 3,6,6 (1)

*[strugglers] "I work with them slowly, repeating sections they do not understand, and try to give assignments that are easy enough to get them to understand those basics."*

*[everyone else] "I assign extra credit work that is beyond the nail-the-basics assignments, to push students that are experienced, and challenge those who find programming easy with just the basics."*

### *4.2.3 Teacher sets context*

A smaller group of respondents took approaches which might be conceived as being dependent upon the individual teaching leader. These colleagues emphasised approaches which tend towards enhancing motivation, either through personal leadership/guidance in the teaching style, or through the use of novelty and technical challenge.

#### *4.2.3.1 External motivation rules 5,5,5 (4)*

Those who believed that external motivation is powerful, were consistent in their use of this intervention across the piece. One additional respondent used external motivation for openers, but indicated that the most able students and the most able students were later supported by differentiated teaching - 522 (1)

*"job opportunities, make money, let them know practice is the key (not talent)"*

#### *4.2.3.2 Novelty rules 4,4,4(4)*

Those who believed that novelty is powerful, were consistent in their use of this intervention. It may be that these interventions are, in effect, led by the teacher's personal beliefs (or perhaps experience).

*"I use an assignment with Origami to teach about proper syntax and clarity in writing"*

## 4.3 As the Course Progresses

Respondents were asked to describe how they deal with the extremes of the class once the course has been running for a while (question 4). It is possible that instructors wait until they know the students well before tailoring their teaching to suit individuals, so a snapshot of practice and a comparison of initial and later responses was made.

The responses relating to how instructors adapt to teaching strugglers and over-achievers were categorized and tabulated (Table 2, A= strugglers, B = over-achievers) as previously and 4 distinct categories emerged:

1. peer tutoring
2. differentiated teaching
3. providing extra help
4. nothing

**Table 2. Later interventions by student category**

| Intervention method | A | B |
|---|---|---|
| 1) Peer support | 5 | 0 |
| 2) Differentiated teaching | 9 | 22 |
| 3) Extra help | 3 | 0 |
| 4) Nothing | 24 | 19 |

### *4.3.1 Changes with Time*

Analysis of changes in approach throughout the course proved to be interesting.

Many who initially practiced differentiated teaching continued to do so. Whilst absolute numbers for doing nothing special remain constant it is the case that the respondents' answers varied with time. Many who do nothing at the outset do adopt differentiated teaching practices as time progresses.

#### *4.3.1.1 Strugglers*

One third of the respondents who do nothing special at the outset do progress to differentiated teaching. On the negative side however there are twice as many doing nothing special once the course is well underway. It appears that many respondents begin with good intentions but, whatever their initial stance, these interactions/initiatives tail off with time.

#### *4.3.1.2 Over achievers*

Although actual numbers indicating no special interventions remain constant it is the case that approx one third start to differentiate teaching as time progresses, although many beginning the course with external motivation to maintain enthusiasm stop this after a while.

## 4.4 Rating Importance of Help

As well as being asked what teaching and support strategies are used respondents were asked to rate the importance, on a scale of 1 (unimportant) to 10 (very important), such interventions and strategies for the 3 groups of students (question 5). Table 3 summarizes the responses:

**Table 3. Importance of Helping Students at Different Levels**

| | mean | s.d. | Mode (n) |
|---|---|---|---|
| Strugglers | 8.6 | 1.7 | 10 (19) |
| Over-achievers | 7.0 | 2.4 | 10 (11) |
| Rest of class | 8.0 | 1.7 | 10 (13) |

Correlating numerical ranking with actual interventions shows that even instructors who provide no extra help or motivation for students at the extremes of the cohort think it important to do so. 11 of the 19 responses rating help for strugglers as very important (score of 10) admit to doing nothing extra to actually help them.

## 4.5 Pair Programming

Experience of collaborative work that students can achieve through pair programming has been shown to benefit student satisfaction and retention [7], [109], [115], [117], [118].

Pairing students together can be an important factor in improving student success, particularly in introductory CS courses. In the context of CS education, pair programming involves two students using a single computer to work on an assignment or a project. One student is the "driver" in charge of designing and typing up the code, while the other, "navigator", is responsible for monitoring the driver's work to detect errors and suggest ideas how to solve the problem. The students would periodically switch these roles [120].

Instructors using pair programming in CS courses expect students to learn from one another as they cooperate to complete an assignment. One of the problems here is that a single student can often solve the assigned exercises or problems with no meaningful or structured cooperation among students [8].

In our survey (question 7) 13 instructors indicated that they always use pairing of students to work on assignments and projects, while 22 other instructors said that they do so sometimes. Only six respondents explicitly stated that then never use student pairing.

When selecting which students to put in a pair, many factors must be considered in order to ensure that the pair produces a meaningful outcome, and most importantly, that both students benefit from such an experience. Closely matching class schedule is one of the most obvious factors in pairing students.

*"Pairs are drawn from the same supporting class so that students are timetabled together." "Students have such busy schedules that they may have actually found the ONLY other student in the class with overlapping study time."*

In addition to the schedule of classes and, possibly, a work schedule, survey respondents indicated that a number of other factors related to convenience and/or preference of students has to be taken into account:

*"I have students pair up, based on living location, work habits (weekend vs evening), aggressiveness (like to work immediately and finish early)."*

*"I try to put people with others they probably live near (based on their "college" -- there are 6 undergrad colleges that make up our campus). A woman student has reported to me that she was so happy that she got paired with someone who lives in her dorm."*

Pairing students who can easily find a mutually convenient time and place to work together is an important prerequisite for a successful learning outcome. This is especially important because from the student's viewpoint, the single most important problem with pair programming is finding time to work on the program together with their partner [76].

Social interaction plays a major role in pair programming, therefore, choosing a suitable partner is an important factor in ensuring the success of a pair of students working on a programming assignment or a project. Previous research indicates that pairing incompatible students may result in student's dislike the collaborative work [131].

A large-scale empirical meta-study of pair programming conducted by Salleh et al [120] indicates that pairing students with different personalities (as determined using Myers-Briggs Type Indicator) often produced better results compared to pairs of students with similar personalities. This work also indicates that student skill level (whether actual or perceived) also plays a major role in determining the success of a student pair. The actual skill level is determined by the academic background and performance of the student, as well as their programming experience. Salleh et al [120] indicate that the majority of pair programming teams produce better results when the pair has a somewhat similar skill level.

Many instructors prefer allowing students to select their own partners. Out of 41 respondents, 10 indicated that they always allow students to self-select their pair partners, while an additional 21 said that they sometimes allow that. Some instructors choose to make suggestions to students as to how they should select their partners:

*"I allow students to self-select, but I tell them it works better if they pair with someone of their own ability. Also they are not required to continue with a pair, and are even expected to switch."*

*"At the start of the semester, I allow students to select their own pairs for the first lab. Thereafter, I assign pairs randomly, with a change every 2-3 class days (the class meets 4 times per week, so students change partners more often than every week)."*

Many instructors indicated that the rules (or lack thereof) for selecting student pairs depend on the nature and level of the course. For example:

*"It depends on the class. It depends on many things. I don't put students into boxes and I don't do it to myself either... so... it just depends."*

In some circumstances it may be very difficult, or even impossible for the instructor to manage the student pair selection process, which may lead the instructor to abandon the entire idea of pair programming:

*"We use pair programming in our second courses. Our first courses are between 300 and 1200 students per term, which is too difficult to manage pairs in."*

Some instructors are not in favor of allowing students to self-select their partners. In particular, one instructor said: *"I have found self-selected pairs to produce the least effective pairings."*

A small number of instructors use a random selection of student pairs. Out of 41 responses, three indicated that they always select student pairs at random, while 14 said that they do that sometimes. Many instructors indicated that if they do resort to a random pairing, this usually is not the first option for student selection that they exercise. For example, student preference is often taken into account: *"Sometimes random. Sometimes self-select. Sometimes a mixture ("if you have a preference, let me know, otherwise I'll assign")"*

On other occasions, instructors allow students to select their own pairs first, but they may use a random selection later: *"At the start of the semester, I allow students to select their own pairs for the first lab. Thereafter, I assign pairs randomly, with a change every 2-3 class days (the class meets 4 times per week, so students change partners more often than every week)."*

In our survey, a large number of respondents (20 out of 41) indicated that they never use random pair assignment. As illustrated in the quotations above, random pairing is often used as a last resort measure because there are better and more effective ways to maximize the benefits that students receive from pair work. One approach is to pair students based on their abilities, which is often determined using student self-assessment:

*"I assign pairs randomly for the first couple of weeks, then I give a questionnaire and assign based on ability/major and also student requests."*

*"[Students are paired] based on a rough estimate of self-efficacy."*

Current or past academic performance is used by some instructors to pair students together: *"Sometimes I will pair students based on their performance in the class."*

In many cases reported in our survey, students with similar academic performance were assigned to work together:

*"We pair students who have demonstrated similar performance in the course (strong with strong and weak with weak)."*

*"[Students are paired to] balance average in-course achievements of the groups."*

*"[We] try to pair students who aren't both in trouble, but not with huge gaps between them either." "Sometimes due to personality or skill levels I will pair students."*

On the other hand, some instructors deliberately choose to pair students with disparate levels of academic performance, which allows underachieving students to learn from stronger students: *"Sometimes I pair known good students with known strugglers."*

If students had worked together previously, chances are that they will be successful again and this could be used as a good criterion when assigning students to pairs: *"Based on who has not worked together recently, so students get to know new people in the class."*

Katira et al [132] reports that gender is likely to determine the compatibility of a student pair. This work suggests that student pairs of different gender may be incompatible, while pairing female students would likely lead to a compatible pair.

Werner et al [133] indicate that that pair programming is particularly beneficial to female students, especially when they are paired together. They argue that "it addresses factors that potentially limit their participation in CS. The collaborative nature of pair-programming teaches women students that software development is not the competitive, socially isolating activity that they imagined." Pair programming, therefore, is one of the ways to encourage female students to pursue studies and careers in CS.

Katira et al [132] report that when they are allowed to self-select their pair partners, students belonging to a minority group tend to pair up with other minorities, although not necessarily from the same minority group.

Some instructors responding to our survey believe that pairing up students from underrepresented groups can help them achieve higher academic results:

*"I always pair women with women."*

*"[We] keep minorities in the same group."*

In general education courses where there are many non-CS majors, student major and their current level are the most straightforward criteria for pairing students: *"Also, I pair majors with majors and try to put non majors with others in their own major. Within THAT I pair by year in school (do not put a senior with a freshman)."*

A number of instructors, however, may be opposed to pair programming because of added responsibilities for managing student pairs and because of perceived increased risks of student failure. Based on their work, Jacobson and Schefer [134] offer a number of good suggestions how such risks can be alleviated .

From a cross-section of the survey responses we received, it appears that using paired student work plays a prominent role in many CS courses, although a small number of instructors seem to be completely alien to this idea. There may be many factors affecting this: staff workloads; local circumstances; nature of the students.

Some instructors use random pairing, but often only as the last resort, when other criteria, such as pairing based on the level of student achievement, schedule, and/or interests, do not work well enough to produce student pairs. The following quote can be used to summarize the prevailing opinion of instructors with regards to pairing students: *"Sometimes I will pair students based on their performance in the class. Other times, they are "randomly" selected or self-selected. I have found self-selected pairs to produce the least effective pairings."*

## 4.6 Getting to Know our Students

Just over half the respondents routinely collect previous programming experience data from their incoming students (question 10). (22/41) Of these, approximately one third use the collected data to help plan their teaching, ahead of the course.

Such planning and preparation might take the form of streaming the new students - guiding them into particular, more appropriate courses based on their level of experience, tailoring the course material, adjusting existing material to help student orientation to the course, general administration purposes and establishing instructor-student rapport. Half of the respondents collecting data on level of current experience indicated that this was primarily for streaming their students, though whether this was to produce an intentional mix of abilities or align those with a similar level of experience was not evident from the responses. Typical responses included: *"*

*We counsel students into [X] or [Y] depending on whether they have experience programming recursive functions", "In placement into an appropriate [X]"* (mentioned by three respondents)

*"...to enlist them into slightly more advanced projects, etc."*

The collected data informed instructors in several other ways, including identifying those at risk;

*"I take note ... to be sure that some students who do not have previous programming background are able to excel in my class." "My primary concern is the students taking the course more than once - how many there are, and is their fraction still decreasing as during the past 4 years."*

The ability to offer a "heads-up" for changing patterns of student experience was another motivation for the exercise, as evidenced by the following responses:

*"90% of our students have no previous programming background. If that changes, we would adjust our instruction"*

*"To watch for significant changes in student preparation, which we have not seen recently."*

Those wanting to tailor the course material to suit the declared experience levels said:

*[were we safe in] "assuming a lot of mathematical background or [should] I ... review some algebra and geometry concepts."*

*"In the unlikely event that an advanced student comes in I make a point to establish a relationship with them."*

*"The main audience is taken into account in lectures, level of presentation."*

Tuning and tailoring instruction to help orientate the student, making them more comfortable with the material, was another useful outcome reported from the collection process reported on within the survey responses. The purported benefits of this are many, from reducing anxiety and the fear of the unknown, which might be barriers to full engagement, to ensuring good progression and continuity.

*"This helps me in how I structure the progression of the course..."*

*"To customize explanations: if a student has already seen some Java or C++, I'll use different terminology and analogies in explaining things to that student."*

In contrast, one respondent made no such use, asserting,

*"We only use it to get an idea of who is taking the course, and do not use it to structure their learning."*

Several responses indicated either only relatively cursory use of the data, or else labeled it for "administration purposes" only. Responses in this category included:

*"qualitatively, to understand student background"*

*"First day of class I collect information on 3x5 cards ... [including] ... Why taking this class?"*

*"For initial identification"*

*"I collect it informally and the result is always the same for my school" "[previously] we had a formal entrance test; now we just remind them of the prerequisite and let them self-select."*

*"For interest" "We only use it to get an idea of who is taking the course ..."*

*"It's not retained, but students are asked at the beginning of the course."*

Finally, two respondents, though currently not making use of the collected data, had plans to use it in the future:

[The data will be] *"Incorporated as baseline into longitudinal outcomes study."*

*"I administer a survey to [X]. In Spring 2011, I hope to analyze four years' data."*

Traditionally, effort invested in enhancing student motivation has gone into the under-achievers. Students who "cope" and meet "satisfactory" levels of achievement have tended to find their way through the tertiary learning experience because or in spite of the efforts of instructors. High achievers have tended not to attract extra attention, partly because their progress poses no threat to pass/completion rates.

Perhaps also, their particular development requires a different kind of extra effort. Producing additional material for students working at the "satisfactory" level is relatively simple; however, when trying to create additional material at an appropriately high level in order to stretch high achievers, the task becomes much more demanding. This opportunity, to provide some assistance to those who find themselves with exceptional students by pointing in the direction of effective resources, approaches or practices should:

1. Initially, make the instructor's life a little easier
2. Help the high achievers realize more of their potential.

## 4.7 Collaboration with Colleagues

From the survey (question 9) it is surprising how few teachers of CS collaborate with other colleagues, just over 34% reported that they did. Yet many CS teachers voice that they considered it most important to provide help for both strugglers and over achievers. The importance of providing help for strugglers was given a score of 8 or higher by 85% of introductory CS teachers, and almost 50% gave the same score for high achievers. It would seem that many teachers in CS use their own imagination and to some degree 'think on their feet' when it comes to coming up with ideas on how to deal with their range of students. Others have quite a range of options/facilities in place such as providing a homework club, extra support session groups and extra challenge session groups, upper-level students acting as mentors and for over achievers the availability of interesting assignments with challenging extra parts. The range of students' ability can vary from year to year depending on the intake and therefore the onus is on the teacher to possibly adapt his assignments or to have contingency plans in place to provide extra projects or tasks should they be required. From the 14 respondents who reported that they did collaborate 50% said they got their ideas from attending workshops at conferences such as SIGCSE, CCSC,

ITCSE and other institutional conferences. 36% said they embarked on informal work with another colleague and only 7% took part in a collaborative project specifically aimed at addressing the needs of high achievers such as an organised inter collegiate competition. Collaboration took the form of simply enjoying an exchange of ideas with friends on the topic by 7% of the respondents who responded positively. The benefits of collaborative programming are described very aptly as, 'To work over time and distance..adds the dimensions of collaborative technologies , language, and culture' [137].

It would seem that depending on the personality and experience of teachers that there is a wealth of ideas and approaches in dealing with the problem of motivating both our struggling and over-performing students and that it would be a welcome development to provide a repository of these ideas for the benefit of others as collaboration in this area would be considered helpful. "Educators can share requirements, ideas for features, and experiences with a particular project or technology" [138]. Like the open-source movement itself, they would collaborate not only ideas, but on the actual artefacts themselves' [139].

It would then be possible for a teacher to choose an approach, a set of assignments or simply make use of a colleague's pertinent quote which had previously got through a message of motivation to their students. For example one CS teacher uses an analogy of 'The Karate Kid' to motivate his experienced students and another anticipates his being comfortably over confident and warns them 'that often their experience ends about 2 weeks before they realize it does.'

International collaboration can prove to be a very useful method of approaching an international problem. The Runestone project was an example of collaborative work carried out which involved two international universities. 'The projects' primary goal was to provide international collaboration into Undergraduate Computer Science Education in away that has value for all participants' [123].

## 4.8 Competitions

Typically all first year classes will have students with varying degrees of subject knowledge. Universities computing programs rarely ask for previous experience of programming however we invariably find that a small proportion of our first year students are more than proficient in coding. These students can very quickly become extremely bored in class whilst the novices start from scratch. Programming competitions aimed at first year students can be used as a tool to motivate these students (see section 2.4.1).

There can be little doubt that access to suitable programming competitions can motivate and inspire some of these students (and not just the over achievers). It can be an opportunity for students to test their ability in designing, understanding and implementing code. Competition can also be the spur that pushes a very ordinary student to achieve much more and it is a fact that a competition win will greatly enhance a new graduates' CV.

Both academics and students opinions of the competitions on offer vary as does the type of students who volunteer themselves for competitions – typically a very small proportion of a class. Of those surveyed (question 8) one academic noted:

*"I run one [competition] myself in project week. However it is voluntary and only a very small number of students participate (about 10 out of 160)"*

another said

*"... only a limited set of students tends to participate"*

Pastor et al [129] looked at an international robot contest as a way to develop professional skills in engineering Students "...with the aim of strengthening a set of basic skills that would be useful for the future professional lives of the participants..." Importantly they asked the students what motivated them to "participate in the competition, what they gained in their personal and professional lives for having participated as well as positive and negative aspects of the experience". The students cited social and personal reasons. 19% wanted to have a good time against 13% being interested in the competition. However we can include another 11% who were interested in the personal challenge, a further 16% attracted by the desire to participate and the fact that they had passed previous national competitions as a reason for going. Another 7% of the participants participated in the competition in order to learn and to gain experience. Clearly the students who compete gain a great deal from the experience.

There is also concern from some that competition leads to bad habits:

*"... we have serious concerns that programming competitions reward quick and dirty coding that is hard to maintain in the long term ..."*

O'Leary [130] discusses using poster competition to motivate students (rather than programming competitions) and claims that:

*"... that learning and interaction can be accelerated, through the introduction of an additional incentive (for example prizes for best entries and peer recognition)"*

however the aim is for an

*"effective method of communication with a group in a nonthreatening and informal way".*

This is a different objective and aims for inclusivity rather than intense competition. The British Computer Society also runs an annual one-day event (BCSWomen Lovelace Colloquium) that is open to all undergraduate and taught postgraduate women in computing and related disciplines across the UK, and beyond. Again the competition is a in the form of a poster, do poster competitions appeal more to female students and programming ones to male students?

Results of our survey cite the ACM International Collegiate Programming Contest (ICPC)as the most popular programming competition however this could be influenced by the fact that of the forty one responses thirty were from US academics. The ACM competition started in 1970 and they note that "the idea quickly gained popularity within the United States and Canada as an innovative initiative to raise the aspirations, performance, and opportunity of the top students in the emerging field of computer science". The competition is now a global network of universities hosting regional competitions that advance teams to the ACM-ICPC World Finals. At the other end of the scale many individual institutions run in-house and intercollegiate competitions. Carter et al, established TOPS (Teaching our Over-Performing Students) competition in 2005, a competition run between four UK universities. The competition was one of the components of the project and was to

*"... be fun, but also had to address our departmental learning objectives"*

Carter [26, 27] shows that the students enjoyed the day and that their comments were overwhelmingly positive:

*"Working together was great. Everyone worked amazingly well in teams"*

*"I liked that we were supposed to work at our own natural pace and that we had to think"*

On a smaller scale again Rosenbloom [92] suggests "Take [ing] a break from the ordinary lecture, test, assignment routine and run an in-class competition to motivate, challenge and boost student self confidence." Rosenbloom concludes that "Students enjoyed this exercise, debated the efficiency of their solutions and were engaged in the followup lectures. The competition was a great, motivational, educational and engaging break from their usual routine."

Undoubtedly the students who choose or are selected to compete in programming competitions enjoy the process and are enthused by the challenge.

# 5. DISCUSSION

The evidence which we have drawn from the survey is based on current practice. It is important to be conscious that this evidence, and the evidence which we have drawn from the literature is the product of individual or institutional compromises which must balance workload represented by staff student ratio and individual teaching commitments. It will be mediated by the prior experience of the teaching faculty member and the availability and additional support such as experience of graduate teaching assistants. Individual departments or institutions may have local cultural traditions and practices. The data reflects the balance of the available evidence, but does not necessarily guarantee that any of the recounted methods will provide a perfect solution for a particular problem of introductory teaching of programming at any given institution. Key findings which are discussed in the subsequent sections cover the following key strategies

- Streamed teaching
- Meeting student expectations
- Research experiences
- Maximizing individual potential
- Interdisciplinary connections

## 5.1 Streaming

It is becoming increasingly apparent that we need to consider offering students with different skill sets a variety of approaches to learning the fundamental nuts and bolts of computer science. Davis et al [38] notes "... we must find a way of enabling complete beginners to learn the basics, while providing enough interesting subject matter to keep the experienced programmers enthused". Some universities use aptitude tests to categorize students and at Leeds University students are classified as either Rocket Scientists, Averages or Strugglers [135]. At Southampton, where they use a student self-evaluation survey to calibrate prior experience, Davis and his team embarked on a project to:

- find out whether student satisfaction would be improved by providing differentiated experiences for the groups of students at either extreme of the initial experience continuum;
- find out whether students were capable of correctly deciding for themselves which group they belonged in;

Davis's team discovered that the student experience was improved by allowing students to study at their own pace.

There is much research on differentiated learning in higher education and many approaches have been tried and written up [12, 38, 39, 51, 100, 122]. It is also accepted that in order to motivate our more gifted students we need to engage them in a different way from the rest of the cohort, it seems the success of these approaches is entirely dependant on the skill and enthusiasm of the academic leading the initiative.

## 5.2 Meeting Student Expectations

Students' expectations and what they actually encounter at the beginning when they arrive at university is very important. If this differs a lot it can lead to de-motivation. This tends to be the case for both top students and less able students [25]. If we, the teachers, get this right it can have a very positive affect on the students' success. Peer learning, active learning and collaborative learning are beneficial for a whole range of students [87]. Some institutions will give tests to students on entry to find out what they know and what they don't know [47]. Streaming and letting students set pace of what they are aiming for can help keep everyone on board. Industry wants colleges to produce 'international life long learners' and students who can apply their learning to the real world [15]. The challenge is to design an introductory programming course that addresses the fact that there is diversity in the group of entry students who have very possibly varied expectations [51].

## 5.3 Research experiences

One of the ways top students can be kept motivated and engaged in their academic programs is to involve them into research projects [85, 104]. A number of research works indicate that such research experiences can be very effective to increase student retention and encourage undergraduates to continue their studies and enter graduate school. In the US, National Science Foundation sponsors the Research Experiences for Undergraduates (REU) program, which enables universities to host small cohorts of undergraduate working on faculty-led research projects during summer months [37, 71, 89]. Projects conducted at REU sites hosted at each university typically are centered on a particular unifying theme, e.g. visualization, information security, or bioinformatics. These projects often lead to collaborations between faculty and students extending well beyond the summer projects. REUs frequently result in student-authored or co-authored research papers, posters and presentations. Finally, REU projects frequently serve as a springboard for top students to gain useful research experience prior to entering graduate school. Many REU programs are very competitive with many top students across the country competing for a spot in each program.

Although REU programs provide an excellent support for faculty and students, they are not the only avenue for undergraduate research [83, 88]. Many faculty often involve top-performing undergraduate students into their ongoing research projects in the form research assistantships, independent studies (in which students receive academic credit), or by offering such students more challenging course projects in the framework of a regular course.

Successful outcome of a research project, such as attendance and presentation at a research conference or a student research competition, is always an exciting event for a student because it

not only provides them with an opportunity to showcase their work, but also to compare it with the work of their peers. Finally, engaging students research projects have been shown to be especially successful to attract female students to study CS and keep them motivated to continue their studies [4].

## 5.4 Maximising Students' Potential

One of the tasks of the instructor is to deliver a course to meet the appropriate educational needs and expectations of the student, for the duration of the whole course. This requires supplying material that will allow the average (coping) student to pass the course and also to assist and recover strugglers wherever possible. Helping along those not coping with the course has long been a requirement of academic staff (e.g., [27, 60]); the situation has been exacerbated in recent years, by the need to comply with the institution's student pass/completion rates – allowing students to fail or drop out invariably leads to financial penalty in countries such as the UK and New Zealand. It is argued here that instructors should also provide intellectual stimulation sufficient to retain the motivational levels of our high achievers – not only is it good for their own sense of achievement, it is likely to repay the effort with a reduction in drop-out rates that might in turn produce financial penalties. A student lost from the roll costs the same whether or not they are low-achievers or potentially high ones. Catering for the high achievers, then, we maintain, occupies far less of our time than it probably ought to do. Courses are understandably built around the abilities of the average student. If we are prepared to invest extra effort in the strugglers, why not the high-flyers? We owe it to them to meet there needs too.

## 5.5 Interdisciplinary connections

One way to enthuse top-performing students who may have already explored many areas of CS, is to expose them to other disciplines by showing them how CS can be applied to solve practical/research problems in these disciplines. Establishing such interdisciplinary connections has been successfully used as a technique to increase enrollments in CS programs and to attract and retain female students [73]. In recent decades, it has become evident that CS is having a profound and pervasive impact on a range of other scientific disciplines, paving a way to interdisciplinary courses offered within CS programs and research projects, from which many CS students can reap tremendous benefits [93, 111]. Students who have experienced such first-hand connections between CS and other disciplines become more aware of the breadth and richness of career and study opportunities, which can be a significant factor in increasing their motivation and interest in the discipline. Interdisciplinary research projects have been shown to be especially successful and are very popular among REU programs discussed above [5].

# 6. RECOMMENDATIONS

## 6.1 The Good

### *6.1.1 Challenge Tasks*

One common theme that emerges from literature and survey responses is that of setting graduated assessments. 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. This may motivate the students who want to prove (to themselves or others) that they do indeed understand the subject matter thoroughly. Whilst challenge tasks may not motivate all, they are easy to administer and may well help some.

### *6.1.2 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 that test 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. Yet other students stop attending classes because they are bored and this leads to disengagement. Streaming can help to alleviate this. Students can be explicitly excused certain lectures about basics, or even all lectures. Classes are streamed by ability and different materials should be provided to students in classes at different levels. Some suggest different assessments, but others like the self-confessed top students to prove that they do indeed have the knowledge they claim by attempting the same assessments as others.

## 6.2 The Bad

### *6.2.1 Doing nothing*

It is easy to treat everyone in the class the same, but it isn't fair to anybody – even the instructor suffers when students fail and drop out.

### *6.2.2 Humiliating students*

Don't do it – EVER!

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 / hungover / suffering a bereavement) is not the way to motivate anybody.

# 7. WHAT NEXT?

The outcomes from this work may not address all the goals identified at the outset, but they do form a solid basis for future work in the area. We have identified recurrent themes and linked current practice with current literature.

The working group initially aspired to investigate methods for motivating our top students following on from work which established an inter-university programming competition created for that specific purpose in the UK.

In reviewing the literature and conducting the survey we have 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. In reviewing our analysis we have identified places where our survey failed to establish evidence, although we know it to exist (for example there was no mention of the Imagine Cup or Lovelace events, even though it is the experience of the authors that these events are used as motivators by some academic colleagues). Further work could usefully be established to make a more thorough classification of available interventions, perhaps in some dynamic form such as an information wiki. An associated area of potentially useful investigation might be the consideration of programming competitions and poster competitions. It would be useful to derive some evidence of the impact and outcomes of such events, and to gather associated attitudinal data from faculty and students in order to evaluate their perceived use and value. Are their impacts similar or different, are their effects which vary

by gender or other variables such as mode of study or prior experience?

We have not been able to provide any indicative descriptions of what constitutes a struggler or high achiever beyond the bald calibration of such students against their performance/potential in programming. It might be interesting to gather some ethnographic data which documented the range of backgrounds and experiences (prior and post their introductory computing course).

Our interest in competitions as a device or intervention for enhancing motivation might usefully be explored; what is the difference for such events between the group or individual experience; is it the competition that drives the students or the opportunity to work in a group with others of equal ability? Are their any patterns which can be discerned across the students to participate in competitions, does that vary according to the focus of the competition?

We contend that students involved in competitions enjoy them and greatly benefit from that experience – it might be useful therefore to investigate those over-achieving students not involved in competition.

These questions point to this area as being one with considerable potential for future research in computer science education. We hope that our readership find this paper a useful contribution to this area and are motivated to join us in future research.

## 8. ACKNOWLEDGMENTS

Thanks are due to Tony Jenkins from the University of Leeds, UK who has helped with the project but was unable to attend the ITiCSE 2010 conference.

Also, many thanks to everybody who responded to the survey.

## 9. REFERENCES

[1] Ala-Mutka K, Uimonen T and Jarvinen HM, *Supporting Students in C++ Programming Courses with Automatic Program Style Assessment*, Journal of Information Technology Education, Volume 3, pp245-262, 2004

[2] Alice – Learn to Program Interactive 3D Graphics, http://www.alice.org/

[3] Aluísio S, Tomas de Aquino V, Pizzirani R and de Oliveira Jr ON, *Assessing High-Order Skills with Partial Knowledge Evaluation: Lessons Learned from Using a Computer-based Proficiency Test of English for Academic Purposes*, Journal of Information Technology Education, Volume 2, pp185-202, 2003

[4] Alvarado C and Dodds Z, *Women in CS: an evaluation of three promising practices*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[5] Amoussou G, Boylan M and Peckham J, *Interdisciplinary computing education for the challenges of the future*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

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

[7] Barker L, McDowell C and Kalahar K, *Exploring factors that influence computer science introductory course students to persist in the major*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[8] Beck LL and Chizhik AW, *An experimental study of cooperative learning in cs1*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[9] Benaya T and Zur E, *Advanced Programming in Java Workshop-Teaching Methodology*, proceedings of ITiCSE'05, Lisbon, 2005

[10] 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

[11] Biggs J, *Teaching for Quality Learning at University*, Society for Research into Higher Education, 1999

[12] 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

[13] BlueJ – the interactive Java environment, http://www.bluej.org/

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

[15] Bouslama F, Lansari A, Al-Rawi A and Abonamah A, *A Novel Outcome-Based Educational Model and its Effect on Student Learning*, Curriculum Development, and Assessment, Journal of Information Technology Education, Volume 2, pp203-214, 2003

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

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

[18] Boyer KE, Dwight RS, Miller CS, Raubenheimer CD, Stallmann MF and Vouk MA, *A case for smaller class size with integrated lab for introductory computer science*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[19] Boyer KE, Phillips R, Wallis MD, Vouk MA, and Lester JC, *The impact of instructor initiative on student learning: a tutoring study*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[20] 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

[21] Boyle R, Carter J and Clark M, *What Makes Them Succeed? Entry, progression and graduation in Computer Science*, Journal of Further and Higher Education, 26(1), 2002

[22] Braband C and Dahl B, *Analyzing CS Competencies using The Solo Taxonomy*, proceedings of ITiCSE'09, Paris, 2009

[23] Burd L and Hodgson B, *Attendance and Attainment Revisited*, proceedings of 6th Annual HEA-ICS conference, York, August 2005

[24] Carter J, *The Value of Guided Revision*, proceedings of 5th Annual HEA-ICS conference, Ulster, August 2004

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

[26] 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

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

[28] Carter J, English J, Ala-Mutka K, Dick M, Fone W, Fuller U, and Sheard J, *How Shall We Assess This?* ACM SIGCSE Bulletin, vol. 35, pp. 107-123, 2003

[29] Chalk P, Boyle T, Pickard P, Bradley C, Jones R and Fisher K, *Improving Pass Rates in Introductory Programming*, proceedings of 4th Annual HEA-ICS conference, Galway, August 2003

[30] 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

[31] Chen TY, Lewandowski G, McCartney R, Sanders K and Simon B, *What do Beginning Students Know, and What can they Do?*, proceedings of ITiCSE'06, Bologna, 2006

[32] Chinn D, Martin K and Spencer C, *Treisman workshops and student performance in CS*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[33] Clark E, *Hacking as a Form of "Self-Improvement"*, proceedings of ITiCSE'05, Lisbon, 2005

[34] Colby J, *Attendance and Attainment - A Comparative Study*, proceedings of 5th Annual HEA-ICS conference, Ulster, August 2004

[35] Cowan J, *On Becoming an Innovative University Teacher*, Society for Research into Higher Education, 1998

[36] Cutts Q, Cutts E, Draper S, O'Donnell P, and Saffrey P, *Manipulating mindset to positively influence introductory programming performance*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[37] Dahlberg T, Barnes T, Rorrer A, Powell E and Cairco L, *Improving retention and graduate recruitment through immersive research experiences for undergraduates*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[38] Davis HC, Carr LA, Cooke EC and White SA, *Managing Diversity: Experiences Teaching Programming Principles*, presented at the 2nd LTSN-ICS Annual Conference, London, 2001

[39] Davy JR and Jenkins T, *Research-led innovation in teaching and learning programming*, presented at ITiCSE'99, Krakow, Poland, pages 5–8. ACM, 1999

[40] Dawson R and Newman I, *Empowerment in IT Education*, Journal of Information Technology Education, Volume 1, pp125-142, 2002

[41] Djordjevic M, *Java Projects Motivated by Student Interests*, proceedings of ITiCSE'08, Madrid, 2008

[42] Eidelman L and Hazzan O, *Eccles' model of achievement-related choices: the case of computer science studies in Israeli high schools*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[43] Eisikovits RA, *Coping with high-achieving transnationalist immigrant students: The experience of Israeli teachers*, Teaching and Teacher Education 24(2), February 2008

[44] Entwistle N, *Motivation and approaches to learning: Motivating and conceptions of teaching*, in Brown S, Armstrong S and Thompson G, editors, Motivating Students, pages 15–23, Kogan Page, 1998

[45] Fallows S and Ahmet K, *Inspiring Students: Case Studies in Motivating the Learner,* Kogan Page, 1999

[46] Fincher S, Barnes DJ, Bibby P, Bown J, Bush, V, Campbell P, Cutts Q, Jamieson S, Jenkins T, Jones M, Kazakov D, Lancaster T, Ratcliffe M, Seisenberger M, Shinners-Kennedy D, Wagstaff C, White L, and Whyley C, *Some Good Ideas from the Disciplinary Commons*, presented at the 7th Annual Conference of the HE Academy Subject Centre for Information and Computer Science, Dublin, 2006

[47] Ford M and Venema, *Assessing the Success of an Introductory Programming Course*, Journal of Information Technology Education, Volume 9, pp133-146, 2010

[48] Freudenthal EA, Roy MK, Ogrey AN, Magoc T and Siegel A, *MPCT: media propelled computational thinking*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[49] Fuller U, Johnson C, Ahoniemi T, Cukierman D, Hernan-Losada I, Jackova J, Lahtinen E, Lewis T, McGee Thompson D, Riedesel C and Thompson E, *Developing a Computer Science-specific Learning Taxonomy*, INROADS 39(4), pp.152-170, December 2007

[50] Gehringer EF and Miller CS, *Student-generated active-learning exercises*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[51] Grandon Gill T and Holton C, *A Self-Paced Introductory Programming Course*, Journal of Information Technology Education, Volume 5, pp95-106, 2006

[52] Greenfoot, http://www.greenfoot.org

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

[54] Hanks B, Murphy L, Simon B, McCauley R and Zander C, *CS1 students speak: advice for students by students*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[55] Hansen S and Eddy E, *Engagement and frustration in programming projects*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[56] Hendrix D, Myneni L, Narayanan H and Ross M, *Implementing studio-based learning in CS2*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[57] Higher Education Academy: Information and Computer Sciences, (HEA ICS), http://www.ics.heacademy.ac.uk/

[58] Hill C, Vijayakumer and Miteva M, *Agents Help Students in ProgrammingLand*, proceedings of ITiCSE'06, Bologna, 2006

[59] 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

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

[61] Hundhausen C, Agrawal A, Fairbrother D and Trevisan M, *Does studio-based instruction work in CS 1?: an empirical comparison with a traditional approach*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[62] Huss-Lederman S, Chinn D and Skrentny J, *Serious fun: peer-led team learning in CS*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[63] Jenkins T, *A participative approach to teaching programming*, presented at ITiCSE'98, Dublin, 1998

[64] Jenkins T, *The motivation of students of programming*, Master's thesis, University of Kent, 2001

[65] Jenkins T, *How do they think they're doing?* proceedings of 6th Annual HEA-ICS conference, York, August 2005

[66] Johnston K, Anderson B, Davidge-Pitts J and Ostensen-Saunders M, *Identifying Student Potential for ICT Entrepreneurship using Myers-Briggs Personality Type Indicators*, Journal of Information Technology Education, Volume 8, pp29-44, 2009

[67] Katz S, Allbritton D, Aronis J, Wilson C and Soffa ML, *Gender, achievement, and persistence in an undergraduate computer science program*, SIGMIS Database 37(4), November 2006

[68] 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

[69] Keenan F and Coleman G, *Extreme Programming: Results of an Educational Experiment*, proceedings of 4th Annual HEA-ICS conference, Galway, August 2003

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

[71] 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

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

[73] LeBlanc MD, Armstrong T and Gousie MB, *Connecting across campus*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[74] Lewandowski G, Johnson E and Goldweber M, *Fostering a creative interest in computer science*, proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, 2005

[75] Lewis TL, Chase JD, Pérez-Quiñones MA and Rosson MB, *The effects of individual differences on CS2 course performance across universities*, proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, 2005

[76] Loftus C and Ratcliffe M, *Extreme Programming Promotes Extreme Learning?*, proceedings of ITiCSE'05, Lisbon, 2005

[77] 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

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

[79] Mead J, Gray S, Hamer J, James R, Sorva J, St.Clair C and Thomas L, *A Cognitive Approach to Identifying Measurable Milestones for Programming Skill Acquisition*, proceedings of ITiCSE'06, Bologna, 2006

[80] Meneely A, Williams L and Gehringer GF, *ROSE A Repository of Education-Friendly Open-Source Projects*, proceedings of ITiCSE'08, Madrid, 2008

[81] Mogharreban N, *Approximate Degrees of Similarity between a User's Knowledge and the Tutorial Systems' Knowledge Base*, Journal of Information Technology Education, p219-226, 2004

[82] Murphy L and Tenenberg J, *Do Computer Science Students Know What they Know?: A Calibration Study of Data Structure Knowledge*, proceedings of ITiCSE'05, Lisbon, 2005

[83] Musicant D, Kumar A, Baldwin D and Walker E, *Mechanics of undergraduate research at liberal arts colleges: lessons learned*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

[84] Nosek JT, *The case for collaborative programming*, Communications of the ACM, 41:105– 108, 1998

[85] 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

[86] Perrenet JC, *Levels of thinking in computer science: Development in bachelor students' conceptualization of algorithm*, Journal of Education and Information Technologies, 2009

[87] Poindexter S, *Assessing Active Alternatives for Teaching Programming*, Journal of Information Technology Education, Volume 2, pp257-266, 2003

[88] Polack-Wahl JA and Anewalt K, *Learning strategies and undergraduate research*, proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, 2006

[89] Raicu DS and Furst JD, *Enhancing undergraduate education: a REU model for interdisciplinary research*, proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, 2009

[90] Riordan B and Traxler J, *Supporting Computing Students at Risk Using Blended Technologies*, proceedings of 4th Annual HEA-ICS conference, Galway, August 2003

[91] Rogerson C and Scott E, *The Fear Factor: How It Affects Students Learning to Program in a Tertiary Environment*, Journal of Information Technology Education, Volume 9, pp147-171, 2010

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

[93] Sahami M, Aiken A and Zelenski J, *Expanding the frontiers of computer science: designing a curriculum to reflect a diverse field*, proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, 2010

[94] Soh L, Samal A, Person S, Nugent G and Lang J, *Closed laboratories with embedded instructional research design for CS1*, proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, 2005

[95] Stamouli I, Begum M and Mancy R, *ExploreCSEd: Exploring Skills and Difficulties in Programming Education*, proceedings of ITiCSE'05, Lisbon, 2005

[96] Stubbings R, Franklin G, Boden D, Powis C and Bent M, *The SirLearnaLot Project*, proceedings of 10th Annual HEA-ICS conference, Canterbury, August 2009

[97] Sudol L, *Forging connections between life and class using reading assignments: a case study*, proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, 2008

[98] SurveyMonkey http://www.surveymonkey.com/

[99] Talton JO, Peterson DL, Kamin S, Israel D and Al-Muhtadi J, *Scavenger hunt: computer science retention through orientation*, proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, 2006

[100] Tomlinson CA, *The Differentiated Classroom: Responding to the Needs of All Learners*, Alexandria, Virginia, Association for Supervision and Curriculum Development, 1999

[101] Valenti S, *Information Technology for Assessing Student Learning*, Journal of Information Technology Education, Volume 2, pp181-184, 2003

[102] Van Der Vyver, *The Search for the Adaptable ICT Student*, Journal of Information Technology Education, Volume 8, pp19-28, 2009

[103] Venables A and Tan G, *Thinking and Behaving Scientifically in Computer Science: When Failure is an Option!* Journal of Information Technology Education, Volume 5, pp121-132, 2006

[104] Way TP, *A Virtual Laboratory Model for Encouraging Undergraduate Research*, proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, 2006

[105] Weir G, Vilner T, Mendes AJ and Nordstrom M, *Difficulties Teaching Java in CS1 and How We aim to Solve them*, proceedings of ITiCSE'05, Lisbon, 2005

[106] Whaley H and Grice S, *Do Students Know Best? Experiences of allowing students to become course designers*, proceedings of ITiCSE'07, Dundee, 2007

[107] White S, Carter J, Jamieson S, Efford N, and Jenkins T, *Tops - Collaboration and Competition to Stretch our Most Able Programming Novices*, presented at 37th Frontiers in education Conference, Milwaukee, WI, October 2007

[108] Whitworth D, *'Who wants to learn Web-Design Anyway?': Course Design for Student Diversity in an ICT Sub-Discipline*, proceedings of 5th Annual HEA-ICS conference, Ulster, August 2004

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

[110] Wills G, Davis H and Cooke E, *Paired Programming for Non-Computing Students*, proceedings of 5th Annual HEA-ICS conference, Ulster, August 2004

[111] 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

[112] 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

[113] McDowell C and Werner L, *The Effects of Pair-Programming on Performance in an Introductory Programming Course*, proceedings of 33rd SIGCSE Technical Symposium on Computer Science Education, Northern Kentucky, 2002

[114] Williams L, Wiebe E, Yang K, Ferzli M and Miller C, *In Support of Pair Programming in the Introductory Computer Science Course*, Computer Science Education 12(3), September 2002

[115] Nagappan N, Williams L, Ferzli M, Wiebe E, Yang K, Miller C and Balik S, *Improving the CS1 experience with pair programming*, SIGCSE Bulletin 35(1), January 2003

[116] McDowell C, Werner L, Bullock H and Fernald J, *The Impact of Pair Programming on Student Performance*, Perception and Persistence, proceedings of 25th International Conference on Software Engineering (ICSE'03), Portland, 2003

[117] 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

[118] McDowell C, Werner L, Bullock HE and Fernald J, *Pair programming improves student retention, confidence, and program quality*, Communications of the ACM 49(8) August 2006

[119] Han J and Beheshti M, *Enhancement of computer science introductory courses with Mentored Pair Programming*, Journal of Computing in Small Colleges 25(4), April 2010

[120] Salleh N, Mendes E and Grundy J, *Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review*, IEEE Transactions on Software Engineering 99, 2010

[121] Ragonis N and Ben-Ari M, *On understanding the statics and dynamics of object-oriented programs*, proceedings of the 36th SIGCSE technical symposium on Computer Science Education, St Louis, 2005

[122] Jenkins T and Davy J, *Dealing With Diversity in Introductory Programming*, proceedings of 1st annual LTSN conference, Edinburgh, 2000

[123] Last M, Daniels M, Almstrum V, Erickson C and Klein B, An International student/faculty collaboration: the Runestone project, ACM SIGCSE Bulletin 32(3), September 2000

[124] Lawhead P, Bland C, Barnes D, Duncan M, Goldweber M, Hollingsworth R and Schep M, *A Road Map for Teaching Introductory Programming Using LEGO Mindstorms Robots*, ACM SIGCSE Bulletin, 35(2), 2003

[125] Jadud M, *Toys + Motivation = Cool Stuff in Computer Science*, proceedings of 5th annual HEA ICS 1-day conference on the teaching of programming, Oxford, 2005

[126] Hamer J, *An Approach to Teaching Design Patterns using Musical Composition*, proceedings of 9th annual ITiCSE conference, Leeds, 2004

[127] Jamieson S, *Introductory Programming Meets The Martial Arts*, proceedings of 3rd Annual LTSN 1-day conference on the teaching of programming, Huddersfield, 2003

[128] Ben-Ari M, *Constructivism in Computer Science Education*, Journal of Computers in Mathematics and Science Teaching 20(1), 2001

[129] 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

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

[131] Thomas L, Ratcliffe M and Robertson A, *Code Warriors and Code-a-phobes: a Study in Attitude and Pair Programming*," ACM SIGCSE Bulletin 35(1), 2003.

[132] Katira N, Williams L and Osborne J, *Towards Increasing the Compatibility of Student Pair Programmers*, proceedings of 27th International Conference on Software Engineering, ICSE'05, St Louis, 2005

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

[134] Jacobson N and Schaefer SK, *Pair programming in CS1: overcoming objections to its adoption*, SIGCSE Bulletin 40(2), June 2008

[135] Jenkins T and Davy T, *Dealing With Diversity in Introductory Programming*, proceedings of 1st annual LTSN-ICS conference, Edinburgh, 2000

[136] Jenkins T and Efford N, *And Now for Something Completely Different: Learning Programming with Python,* proceedings of 9th annual HEA ICS conference, Liverpool, 2008

[137] Last M, Daniels M, Hause M and Woodroffe M, *Learning from students: continuous improvement in international collaboration,* proceedings of ITiCSE '02, Aarhus, 2002

[138] Layman L, Williams L and Slaten K, *Note to self: make assignments meaningful*, proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, 2007

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

[140] Wenger E, *Communities of Practice: Learning, Meaning and Identity*, Cambridge University Press, 1998

[141] White S and Irons A, *Informatics in the UK: Current Perspectives*, proceedings of Innovation in Teaching And Learning in Information and Computer Sciences, 2007

## 10. APPENDIX

This appendix comprises the survey, which was advertised to academics via mailing lists that working group participants subscribe to. It was administered through SurveyMonkey [98]. The free survey service allows for a maximum of ten questions with question types chosen from a small selection of predetermined styles. It allows a maximum of 100 responses to be stored and analyzed.

1. In which country is your institution?
   If you are willing to be contacted about your responses then please provide your email address.

   Country: _____

   Email Address: _____

2. About your institution.
   Level of course taught:

   ○ sub-degree

   ○ undergraduate

   ○ postgraduate

   Type of institution (e.g. community college, university) _____

3. At the start of the academic year how do you support / motivate the students who are:
   a. Inexperienced and having difficulty mastering the basics _____
   b. Inexperienced yet quick to learn _____
   c. Experienced? _____

4. As the course progresses and differentiation within the classroom becomes more apparent, how do you support:
   a. Strugglers _____
   b. Over-achievers? _____

5. How important is it to provide help for:
   (1-not important – 10-very important)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Strugglers | | | | | | | | | | |
| Over-achievers | | | | | | | | | | |
| Rest of the class? | | | | | | | | | | |

6. Do you employ students as tutors?
   ○ yes

   ○ no

7. Do you use pair programming with your students?

| | always | sometimes | never |
|---|---|---|---|
| Do you allow pair programming | | | |
| Do students self-select pairs | | | |
| Are pairs randomly assigned | | | |

   If pairs are not randomly assigned how do you allocate them? _____

8. Do your students enter competitions?

| | yes | no |
|---|---|---|
| At your institution | | |
| Nationally | | |
| Internationally | | |

If your students do enter competitions, which ones? _____

9. Have you collaborated with colleagues from other institutions to provide tasks to better motivate your students?
   - ○ yes
   - ○ no

   If so, please provide brief details _____

10. Do you routinely collect data relating to the previous programming background of your students?
    - ○ yes
    - ○ no

    If yes, how do you use it? _____