Higher Education Academy

TRANSFORMING TEACHING
INSPIRING LEARNING

Issues in retention and attainment in Computer Science

Neil Andrew Gordon, University of Hull

Contents

1. Executive summary

1. Recent research on undergraduate retention and attainment across the disciplines (Woodfield 2014) considered how students perform differently depending on their discipline. Woodfield's report highlighted Computer Science as being atypical in a number of factors related to success, and in particular identified areas of concern.
2. Aspects where Computer Science appears to perform poorly include the relatively low number of upper degrees awarded (joint second worst), and the comparatively low continuation/retention rate (the worst for students leaving with no award, or with a lower award than the original qualification aimed for). Other aspects of the cohort reflect a distinct profile, such as the male bias (second largest).
3. This report investigates such issues of attainment and success for Computer Science students through a synthesis of existing literature, exploring the distinct curricula, culture and practices of Computer Science, and how these affect the experiences and success of students in computing. A key element of the report is to identify areas for further exploration, along with suggestions on how to address some of the problematic ones.
4. The curricula, pedagogy and culture of Computer Science are each explored in detail, with evidence to show there is sometimes a justification for common perceptions e.g. of student syndrome regarding responses to deadlines and attendance, along with approaches to address them.
5. A particular area where Computer Science has been identified as being of concern is that of graduate employability (HEFCE 2015; Shadbolt 2015). As part of the review of success, this report considers positive destination data (being in further study or graduate roles) to show Computer Science from a positive perspective.
6. The report outlines gaps and areas for further research, such as the impact of changes in pre-university computing, how to utilise student data to identify and support students at risk, and how the delivery and style of teaching, learning and assessment may be adapted to reflect the nature of computing students. The context within which students study is also identified as something that needs further analysis, with issues around how to support minority groups and how to effectively develop learning communities.
7. The report concludes with a set of recommendations, for example, how to prepare and manage the expectations of students prior to and upon entering computer science at university and how to make effective use of technology and pedagogy that can tailored to the support the broad community within a typical computer science cohort.

2. Introduction

Woodfield's (2014) report on retention and attainment in higher education (HE) reviewed the performance – in terms of attainment and retention across HE in general, and with a breakdown by disciplines. That report also considered a range of factors that demonstrated differences between disciplines. Woodfield's report considered data for the 2010-11 cohort. Following from that work, the current report is one of a family that focuses on the general issues raised there, and how they play out in relation to the specific discipline of Computer Science.

This report begins by considering those aspects that reflect discipline specific characteristics; examples here are the nature of the content, and of the student and staff communities. The chief issues considered are:

- the popularity of Computer Science as a choice of degree course;
- the perceived gender imbalance within Computer Science in the UK;
- the relatively low level of retention compared to other disciplines;
- the relatively low level of achievement in terms of degree classification;

- the proportion of mature and Black and Minority Ethnic (BME) students;
- the broader issue of achievement, considering the destinations of graduates.

Following the review of the characteristics that distinguish Computer Science, this report will expand on the specific nature of the discipline:

1. the design of curricula – in particular around assessment and student behaviours;
2. issues around student engagement;
3. the potential value and use of big data approaches, with learning analytics and educational data mining;
4. Computer Science as a discipline;
5. issues around the transition into HE computing courses;
6. the nature of Computer Science students – and "student syndrome";
7. the development of communities of students;
8. the gender imbalance in UK Computer Science education;
9. the impact of attendance on performance;
10. customs and practice in the discipline.

The report finishes with a brief review of apparent gaps in the current literature and evidence base, and goes on to explore future work and provide recommendations on how to adopt strategies to improve both student retention and student attainment.

3. Retention and attainment across Computer Science

Woodfield's report (2014) considered a number of issues around attainment and retention across higher education – more commonly known as tertiary education globally. The report identified disciplines that appeared to be outliers in some of the factors that measure or affect student attainment. So firstly, what do we mean by retention and attainment?

In the context of this report, retention is concerned with whether students continue with their chosen course of study at a single institution, or whether they transfer to another course/discipline or withdraw from university altogether. Attainment in Woodfield's report was concerned with final achievement – primarily whether students pass their degree course and what grade (i.e. classification) they achieve. A slightly more general concept of success can include both the degree achieved, along with broader considerations, such as whether students progress to employment, continue on to other study (top-up degrees for foundation degree students, postgraduate taught or research for Bachelor's degrees) or achieve other personal development milestones. In the case of going into the labour market, the nature of the job is considered a measure of success. That is, whether is it a graduate level job that requires a degree, or a role that requires and utilises the subject specific knowledge and skills in the degree, or is it a general role that does not require the specialist discipline skills nor the more general skills of a higher education.

We now consider some of the key points from Woodfield's (2014) Higher Education Academy (HEA) report on retention and achievement with respect to Computer Science (CS) and Science, Technology, Engineering and Mathematics (STEM):

1. Computer Science is a relatively popular discipline, accounting for 4.2% of the student body;
2. while men are under-represented across HE as a whole at 43%, in Computer Science they account for 83%;
3. STEM disciplines generally had a higher rate of withdrawal due to academic failure, at 38% of the leavers; retention in CS, at 91%, was the worst of all disciplines;

4. attainment rates are low compared to the sector, at 56% achieving a upper (good) degree;
5. 32% of students are identified as mature;
6. in terms of ethnicity, Computer Science has a high proportion of BME at 25%.

Computer Science is a popular subject – the eighth most popular in UK higher education, as illustrated in Figure 1 based on UCAS applications, accounting for approximately 4% of all applications. However, its popularity has varied widely over recent years – creating challenges as to how institutions manage to provide the resource to deliver it. One particular issue is how to ensure appropriate provision and support for students, especially within a discipline that changes rapidly, and the corresponding need for updates to curricula and teaching facilities.

While Computer Science is popular in terms of numbers, the gender profile is male dominated. This profile seems to develop as pupils make the choice on qualifications at school and college, being reflected in A-level (and equivalent) choices, and then at university and later in the industry. Exact figures vary, but are typically of the order of 80% or more students being male, while on some university courses men represent over 90% of students. Faculty members may reinforce this and thereby contribute to the continuation of this pattern; for example, university open days that are presented and managed by mainly male staff, with male student guides and predominantly male student cohorts can lead to a less than welcoming environment. A number of studies indicate that male students demonstrate characteristics (Beyer *at al.* 2003; Clarke and Chambers 1989) that can reinforce and may partly explain some of the specific difficulties identified, and there are a number of initiatives to attempt to redress the balance (Barjaktarovic 2014; Binkerd and Moore 2002; Cohoon 2001).

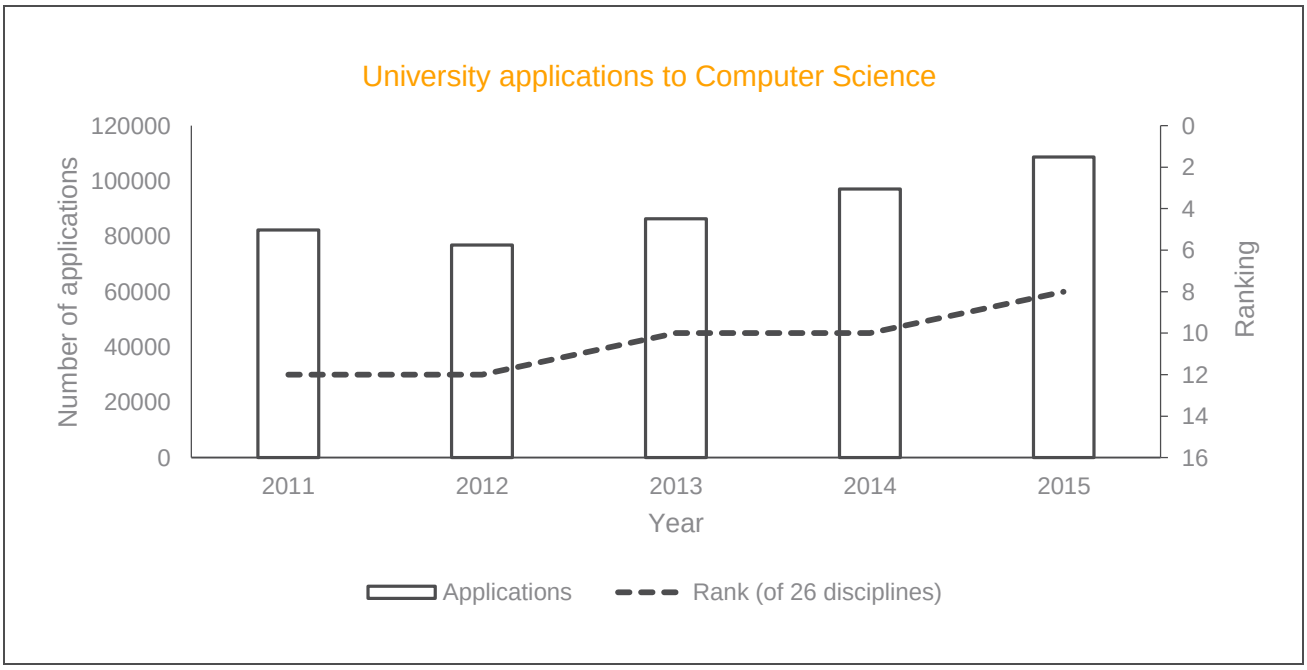

FIGURE 1: APPLICATIONS TO COMPUTER SCIENCES – ALL DOMICILES (SOURCE UCAS 2015)

Another problem for Computer Science is withdrawing due to the nature of the material, potentially due to the issue of expectations of what computing is and what is required. The key point here is that some students embarking on courses do not appreciate what a course involves; they may have experience of a different approach to, and philosophy of, Computer Science as articulated by a specific department, or may have no experience at all. Examples here are the way that some computing courses focus on the use of tools and applications and the creative side – for example, creating computer graphics and models, or designing games. Conversely, other courses target the programming and building of the tools themselves, where the creative elements are subsumed within the process of building solutions and writing software.

A further aspect where computing is identified as performing poorly is in higher attainment levels: the proportion of students achieving an upper degree. This varies widely between institutions, but the general pattern for the discipline is that less than 60% achieve an upper degree. This may reflect a discipline that tends to use the full mark range, and where students who struggle to understand key concepts are not able to complete assignments to achieve a threshold pass. Mark profiles for modules can be more of a twin peak rather than the anticipated bell curve, with a cluster of students with low marks, and another with higher marks. This also connects with the more general STEM issue of student withdrawal, as students who struggle with key concepts are likely to withdraw since they cannot engage with the content or assessments.

While the proportion of mature students is below the sector average, it still seems high at just under one-third of the overall cohort. This can create issues of cohesion and identity. Mature students may have family, work or other commitments that affect their ability to integrate with the cohort – thus reducing the cohesion within the class and their own identification with their classmates. This reflects the number of them failing to finish their studies (retention) or to achieve their potential (attainment) with 10% leaving HE with no award. Similarly, the proportion of BME students (at 25%) seems high for the sector. As with mature students, this group had a high withdrawal rate due to academic failure across the sector, and so may partly explain the higher withdrawal rate for the discipline; the potential problems of cohesion and identity again being potential factors. Proactive approaches to support such students have been identified as a key need (Digital Skills Taskforce 2014), where fewer than half achieve an upper degree, compared to 63% of their non-BME peers. Suggested approaches include considering additional academic support, careers advice and greater focus on ensuring BME students can benefit from placement opportunities. The last point is of growing interest when it comes to improving the employment and positive destinations of computing graduates in general, as data indicates that placements have a positive impact on both measures.

The proportion of students reporting autistic spectrum disorders was 1% – over double that reported in other disciplines: this may reflect the gender imbalance. Regarding entrance qualifications, the proportion with over 340 UCAS points appears low, though again this may reflect the gender imbalance (females tend to outperform males at school) and the broader range of computing qualifications offered at entry, many of which do not qualify for UCAS tariff points.

Woodfield's report also considers aspects of the distance of a course from the students' pre-study home, and the impact of parental experience of higher education. These do not seem to stand out as particular issues for Computer Science, though some of the above characteristics (around the proportion of BME, part-time and mature students) may have some particular relationship with this, since the proportion of BME and mature students studying close to their pre-study home is higher; this would likely reduce the chance of integrating with the broader student body through living in halls of residence and taking part in more extra-curricular activities that can strengthen the identification with their peers.

3.1. Other attainment and retention data for Computer Science

As noted earlier, within the broader context of attainment the destination of graduates becomes relevant. This is another area where Computer Science falls outside the typical range of other disciplines; looking at the number of graduates who are unemployed six months after graduation, Computer Science based courses consistently perform poorly (Shadbolt 2015), with the highest unemployment rates – 13.0% in 2012-13 (HESA 2013) and 11.3% in 2013-14 (HESA 2014). Concerns around employment have led to an investigation into both the role of accreditation and into the different practices within computing courses that may best prepare students for their careers. While the key figure that has caused concern is the consistent proportion of students who graduated from a

Computer Science degree who are classed as unemployed six months after graduation, when considering those in full-time employment, Computer Science is next best behind Architecture, Veterinary Science and Medicine.

This particular perception of attainment seems to be linked to degree outcome – since lower grades are likely to mean that students struggle to get work, whether discipline specific or more general graduate roles, and if the individuals have a relatively low classification compared to other disciplines when competing even for general work.

While the proportion of unemployed Computer Science graduates is the highest of the different disciplines, this should be considered in the context already described, that is, it is one of the most popular disciplines, with high proportions of BME students. As noted above in the context of the Digital Skills Taskforce report (2014), the proportion of computing graduates who are unemployed seems concentrated in the BME group "who tend to achieve lower grades at university and are then more likely to be unemployed" (Digital Skills Taskforce 2014, p. 14). Moreover, as explored below, Computer Science courses typically do not have discipline-specific entry qualifications, and so some students embarking on a course may change their career plans by the end; the impact of this may be that it takes them longer to settle into work. This is consistent with the destination data for three years after graduation, where unemployment rates are much closer – looking at the 2008-09 cohort who graduated in 2011-12, Computer Science is highest for employment of the STEM disciplines, and higher than the sector average (Table 1).

Table 1: Employment rate 3.5 years after graduation (2008-09 cohort). (Source HESA 2014a)

	Unemployed	FT paid work	PT paid work	Paid work (FT or PT)
All subjects	3.4%	78.1%	6.8%	84.9%
All sciences	3.0%	71.0%	5.7%	76.7%
Computer Science	5.4%	72.8%	6.8%	79.6%

A positive distinction is present when we consider the type of destination. Positive destinations are key, that is, the graduates employed in a job that requires a degree, or are in further study. The unemployment data indicates that Computer Science is worse than the average for all sciences and all subjects at 11.3% (see the second column of Table 2). However, when combining the figures for paid employment (full-time or part-time, and excluding the small number in both study and employment), Computer Science is then higher than the science and all subject averages, at 76.2% in work (column 4 of Table 2).

If we combine the data for how many are in work with the proportion of those in professional occupations, then Computer Science is clearly performing better than the general science profile and that of the entire cohort (Table 2). Considering the broader positive destination (in professional role and/or further study), then at 70.3% the total for Computer Science is slightly lower than that for all sciences, but still better than the sector average of 66.2%.

Table 2: Data for six months after graduation for 2013-14 leavers – Employment rate, destination type and overall positive professional destination data for comparisons (Source HESA 2015b[#] and HESA 2015[$])					
	Unemployed[#]	In further study[#]	In paid FT or PT work[#]	Proportion of those in work in professional roles[$]	Overall proportion of cohort in professional roles
All subjects	6.5%	18.1%	70.8%	68.0%	48.1%
All sciences	6.0%	18.4%	71.7%	77.0%	55.2%
Computer Science	11.3%	9.3%	76.2%	80.0%	61.0%

4. Curriculum, culture and custom in Computer Science

When we consider Computer Science in comparison to other disciplines, there are some common issues – especially relating to mathematically based and STEM disciplines, alongside some that reflect the arts and creative elements of computing. However, the characteristics of Computer Science do separate it – the particular set of features give it a unique profile with its own distinct issues as now studied.

4.1. Curriculum

The Computer Science curriculum reflects a discipline that is a mix of vocational knowledge and academic theory. The balance between these depends on the nature of the individual department or institute delivering a course, and may focus more strongly on the theoretical underpinnings of Computer Science, or on the applications, or a mix of the two. As considered later, this reflects the culture and custom of the discipline.

Curriculum is also defined in terms of the Quality Assurance Agency for Higher Education's (QAA's) benchmark for computing (QAA 2007), along with accreditation requirements of the British Computer Society (2010), Association of Computing Machinery (2013) and Creative Skillset (2015). The technical nature depends on the institution, but general awareness and utilisation of computing technologies is a common theme. The demands for Mathematics, prior computing, or other related disciplines naturally impacts on the nature of the material covered within a degree. For Computer Science courses, programming will normally be a key and significant component, though the amount and depth of this varies across courses. As noted elsewhere in this report, prior knowledge of programming and managing a diverse cohort then requires appropriate material to support the new programmer and novice, as well as stretching and interesting an experienced programmer. Alternatives here can involve adapting the curriculum structure – perhaps with streamed pathways, varied assessments with extra credits or marks for extension activities – or extra-curricular initiatives, such as programming competitions (Roberts 2000).

One particular characteristic of Computer Science – considered later in this report – is the variation in intake requirements and the varied nature of students' preparation for the course. Unlike many other STEM disciplines, while there are school-level and college-level courses in Computer Science related areas, these are not a pre-requisite for the majority of university courses. Indeed, where departments do specify a pre-requisite for a Computer Science course, it is usually A-level mathematics. When it

comes to planning and delivering the early stages of the course, the amount of Mathematics for computing, introductory programming and computer systems depends on the nature of the degree, but should also reflect the background and preparation of the students. The use of diagnostic or other tools to identify suitable pathways or material for students can assist here – with students doing some form of assessment to aid in determining which content and level of material is relevant.

Another feature of the curriculum design in Computer Science that may have an effect on attainment (grades) and the uptake of training is the use of four-year integrated Masters programmes. These are quite common across STEM disciplines in general, and in Computer Science in particular. The benefits are numerous – for students, they provide the opportunity to develop level seven skills and knowledge, giving a higher-level qualification. It also has the advantage of falling under the undergraduate fee arrangement in England, thus potentially being more attractive than trying to source funding for a stand-alone MSc (Holly 2014), although the planned introduction of loans for postgraduate study may reduce this over time (THE 2014b). However, the impact on attainment is then more complex – since students who may have done well with a three-year Bachelor's degree may struggle with the level seven material.

Teaching, learning and assessment

In Computer Science – as with many STEM subjects – teaching has traditionally adopted a hybrid approach that pre-empted more recent teaching approaches such as the flipped classroom, something that should be considered when considering the distinct discipline nature of Computer Science. Similarly, the nature of theory and practice is distinct in Computer Science, where the process of applying theory through the development of algorithms, software and building actual solutions is commonly utilised within courses to enable students to demonstrate their intellectual understanding through the application of knowledge.

Another aspect of Computer Science is the long-standing gender imbalance, especially in Western education and industry. One way to address this is considering how to contextualise our teaching to improve engagement and attainment, perhaps by showing the social and broader impact of computing (Margolis 2003).

Assessment itself is another factor that has a significant role when it comes to attainment and retention: as may be expected, Computer Science faculties have adopted technological and other assessment techniques and tools. Current approaches to computer-based assessments commonly utilise multiple choice or numeric values. While these can provide suitable assessment, they can lead to skewed or irregular attainment profiles if they are not designed in a suitable way – they key point here is that if in use, or by design, a system only allows for correct/incorrect answers, then the binary nature of this leads to students having very high or low marks, with no opportunity to recognise approaches to an answer or for partial understanding. Another factor with computer-based assessments is that they can limit the type of feedback, and so reduce the opportunity for the feed-forward process that should help to develop students so that they can improve their attainment over time.

Given the importance of formative assessment in assisting and guiding students to improve, the need for summative assessment to ensure learning outcomes are addressed and that credits correspond to actual outcomes, it becomes critical to consider how assessments are planned, implemented and marked. The threshold (pass) should demonstrate achievement of the outcome, while the overall pattern of formative and summative assessment can provide a framework to guide the student. In practice, the distinction between formative and summative may be quite opaque – with formative work given minimal marks to ensure students give it sufficient focus and do attempt it. Combined with computer-based assessments and directed activities, assessment can be used to guide the student in their learning (Gordon 2009).

Automating assessment of code offers some potential benefits to staff and students – potentially enabling students to try their code as they develop it, rather than await assessment results after a non-recoverable formal submission. Such test-driven approaches (Edwards 2003) are becoming more and more viable as the time taken to build, test and report is now not such a problem for much of software development. Code analysis tools can go further, and provide further feedback and guidance to students on the quality of their code.

A related issue is that of deadlines: the use of technology enhanced learning tools such as electronic submission – already quite common in Computer Science, one of the leading disciplines in adopting such tools over recent years (ICISA 2015) –. can mean that students decide to submit at the last instance. This gives no leeway for technical issues – such as hardware failure or network problems. In combination with the nature of many Computer Science students (see later in this report), this creates a scenario where students may miss deadlines and incur penalties for late submission.

Beyond the formal curriculum, pastoral care through personal supervisors and academic tutors, as well as from the broader range of staff in direct contact with students becomes critical – something that can be hard to balance with the demands and workloads of modern HE staff.

Student engagement

Approaches to try to encourage student engagement may include attendance monitoring –which utilise penalties for non-attendance, such as not allowing reassessment. Such approaches may encourage some students to attend more regularly, but can also mean that others are unable to continue a course and so increase the withdrawal rate.

When considering engagement, the Computer Science discipline offers some specific approaches – such as gamification – that may be applicable and can address some of the shortcomings already considered. We can utilise an existing game or game engine framework to develop activities and/or content with an educational effect. An alternative approach is to design a bespoke game that focuses on an educational outcome – such serious games (Bergeron 2006) being used in a variety of contexts. A final approach utilises game mechanics through a variety of means (Gordon, Brayshaw and Grey 2013) and may not necessarily explicitly use a game itself. For the last example, one tactic is to vary assessment, moving away from the high stakes approach mentioned earlier, to a number of smaller (sub) components that can provide students with feedback more rapidly. Game mechanic concepts such as providing small and clear activities, incremental difficulty, low-risk and rapid feedback, all fit together and offer a different model for assessment. Easy access to self-assessment and routine reward can all be useful in encouraging students to attempt to do work and to repeat it to improve their performance.

Learning analytics

When contemplating how to alleviate some of the negative characteristics of the Computer Science cohort, it is natural to consider some of the technologies that the discipline enables. Learning analytics builds on concepts of data mining, and more specifically educational data mining – where data from potentially disparate sources, and on a large scale (big data) are combined and analysed using a variety of techniques, such as artificial intelligence, statistical analysis, and numerical approaches to enable patterns to be identified.

We can now combine attendance and assessment data – formative and/or summative – with other indicators of engagement. Examples of such indicators include:

- access to a virtual learning environment (VLE), which could be categorised as passive (access to materials) or active (utilising interactive learning materials);

- interaction with team members or team members' ratings of the students within a team activity;
- completion of assessments – diagnostic, formative or summative;
- submission of coursework or sample material;
- interactions with a code repository/code management system for programming activities, or a database server or web server for other practical work;
- measures of attendance – access (log in) logs, lecture or seminar registers.

Through appropriate scoring of the above, it is possible to rank students in terms of a numerical combination. Getting the data together, and finding the appropriate combination is non-trivial, though even a rudimentary approach can be informative, and empirical evidence shows that it can identify students that are having difficulties with their studies (Rountree *et al*. 2004). The results may not always improve retention – from experience, it sometimes means that students recognise they are on the wrong course and withdraw or transfer, but it can improve attainment. In some cases, it can pick up problems early – such as students who have missed the need to do early assessments that contribute towards their course marks.

4.2. Culture: student expectations and preparation

One particular difficulty that Computer Science faces is the prior experience of Computer Science at school or college, and in particular the confusion between IT, ICT, Digital Literacy, Computing and Computer Science. One way to consider this is to recognise that English as a university discipline is quite different to literacy, Mathematics is different to numeracy, and Computer Science is distinct from ICT and digital literacy. Moreover, IT is often associated with business; indeed, many IT courses are run by and based within Business schools/faculties. This confusion is further compounded since IT is often used as the general term that encompasses Computer Science.

Sub-specialisms within Computer Science demonstrate similar issues to those above: Information Systems (IS) can be considered from a technical Computer Science perspective, or may be approached from the business perspective and delivered within a Business school. Computer and video games are increasingly popular as degree specialisms, though many students appear to think of these as game design or game content focused, while many Computer Science approaches are predicated on the technical programming and computing architecture that provides the technical scaffolding for this digital content. While IT, ICT, and digital literacy are important skills for Computer Science students, they are generic skills. The discipline has a broader and deeper content. It is worth noting this can have an impact on data – as some statistics are based on data (e.g. HESA) where courses are grouped by assumed/allocated discipline even if this is as detailed or accurate as it might be. The allocation of many IT courses to Computer Science being particularly problematic.

These problems lead to a gap between the expectations of what a Computer Science related degree will include, and the actuality of degree content and requirements. So many issues around retention relate to students' misunderstanding of the discipline. Some teachers and careers advisors display similar misunderstandings, with the requirements for prior advanced mathematics, and the nature of content of a Computer Science based course being misrepresented, as students are not prepared for the technical programming focus. Conversely, the alternative problem is where students have an extensive programming background – level three (A-level and equivalent) qualifications vary in their approach to content and preparation for Computer Science.

The nature of a Computer Science degree reflects the identity and approach of the department hosting it – some departments focus on digital content and media, other departments on applied computing and engineering, some on Information Technology and Business, while others reflect the science ancestry.

In terms of the culture of Computer Science, another aspect that has an impact on both retention and engagement is that of student autonomy. There is a tension between the freedom and responsibility of independent study – supported by a framework of suitable resources, help and guidance from the host department – and the growing concept of a department as teaching its students, with consequential expectations of monitoring, reviewing (assessing) and acting as a manager of the learner (Smith 2005). A related issue is the relationship and level of interaction between staff and students. The role of the personal (academic) supervisor or tutor is under increasing inspection: perceived as a key element of higher education by many, actual interactions between supervisors and students can be minimal (Lindsay 2011; Luck 2010). The challenge is to determine how this relationship should work: practice varies, with some institutions and departments requiring weekly meetings, that may support a range of modules, while others outline a more erratic and flexible schedule to support the student's development, but much of it left to the individual student to initiate and arrange. This is another area where computer technology – such as automatic meeting arranging and tracking, and the use of computer-mediated meetings is increasingly influential (Samuels 2014).

Student syndrome

Two problematic areas that appear common within the Computer Science community are those of procrastination and organisation. While these issues can affect other disciplines (Manninen and Wadsö-Lecaros 2014), given the nature of curriculum and culture already considered in Computer Science, and thereby among our students and staff, they can be more common (Waite *et al.* 2004). Procrastination and issues of self-organisation combined can cause conflicts that we now consider, in particular under *student project syndrome*.

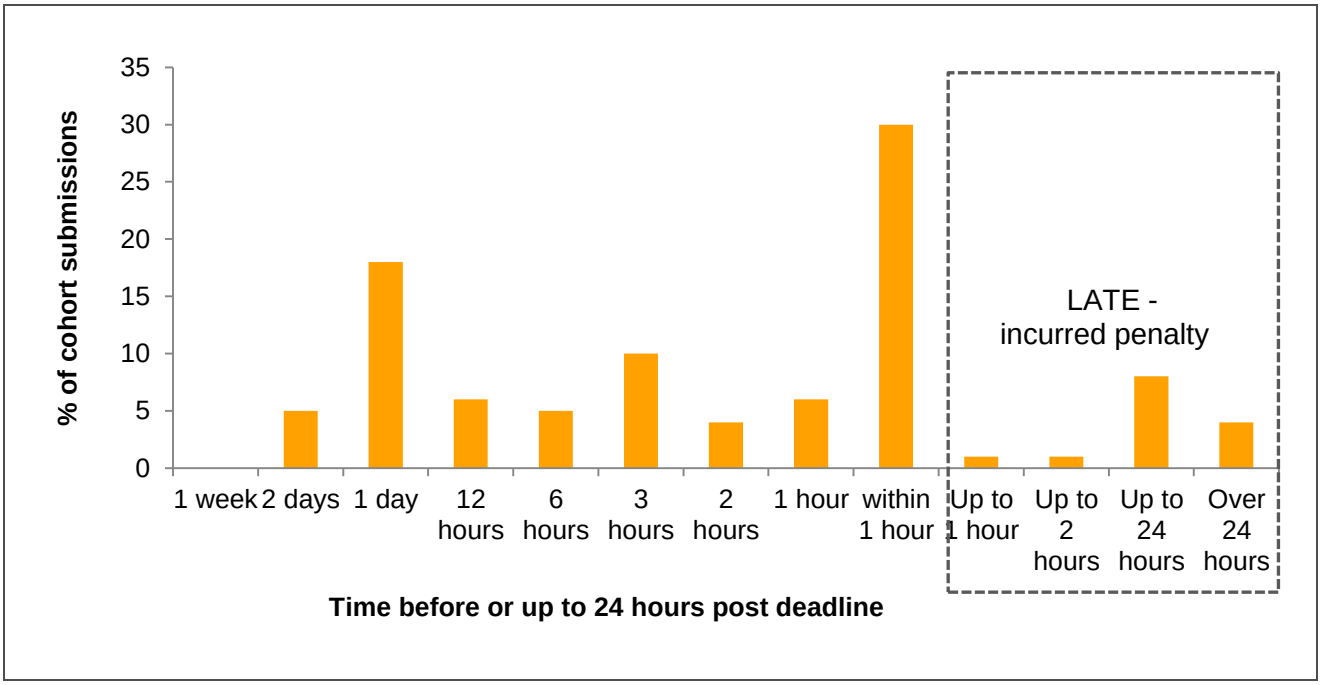

FIGURE 2: DISTRIBUTION OF ELECTRONIC SUBMISSIONS PRIOR TO, AND UP TO 24 HOURS AFTER, THE DEADLINE (GORDON 2014)

A recognised characteristic of Computer Science students, "student syndrome" (Smith 2010), is the tendency to work to deadlines – that is, to leave work until a deadline is imminent, then to work on it at the last minute. In industry, the approach to 'overnighters' and submitting within minutes of the scheduled delivery date is also apparent from experience within Computer Science teaching. A common trait of project management – especially in the context of Information Systems development – this occurs where individuals use the growing pressure of a deadline to encourage them to put in the expected and required effort. The problem with this approach is that it then becomes more likely that a

deadline is missed as it removes the scope for any safety margin and puts the individual under substantial pressure. In academic terms, it may mean that a mark is penalised (reduced) or capped, or causes the individual to suffer adverse effects from the stress and pressure, and so can contribute to student failure.

The tendency towards last minute submission is especially problematic with high stake assessments – typically large single pieces of coursework or end of course exams – where students can fail without an easy recovery pathway. Later we consider some alternative assessment patterns that may alleviate this. As noted earlier, and illustrated in Figure 2, while a piece of work may be available for several weeks, the students focus on the specific day and hour of submission (the example below was set five weeks earlier): The submission date becomes the target for the completion of the work, rather than the final opportunity to submit without penalty.

Tribes and territories

One particular characteristic identified in Computer Science students relates to how well they socialise and integrate with one another, with the aim that the class becomes a supportive environment for learning and encouraging the integration into the learning environment, rather than being overly competitive or isolating; the issues around gender imbalance and prior experience of the discipline can both become negative factors here (Barker *et al.* 2002). There are different potential approaches to this – utilising game or other icebreaker activities, and encouraging or requiring students to co-operate and work together where appropriate. Such group bonding can be problematic with a varied and heterogeneous cohort, but the potential benefits identified in the literature for both retention and attainment are significant. There is evidence that social activities such as games can improve this transition (Talton *et al.* 2006).

Students may be a resource to assist other students: utilising students to assist and support their peers is common practice in Computer Science though the approaches differ, each having different pros and cons, and their own characteristics. Postgraduate students – in particular, research students – are commonly used in demonstration, tutorial or other support roles. The chance to meet a postgraduate and benefit from their different backgrounds is a key advantage, that also strengthens the links between research and teaching, albeit in an implicit way. More advanced (in terms of stage of study) students may be used to assist in formal classes, where they act as paid or unpaid demonstrators and helpers. This is common in programming and other applied laboratory elements of a typical Computer Science course.

Another approach for student-to-student support is through organised peer support systems such as peer-assisted student support (PASS) or peer-assisted learning (PAL) (Keenan 2014). Here, group leaders are typically given training and timetabled sessions with resources, to aid students at an earlier stage in how to approach problems. PASS/PAL approaches are typically focused on study skills and approaches to problem solving, with the leaders' role being to guide students in how to approach and potentially solve a problem.

Informal peer support can also be managed through teamwork, with paired activities or larger team projects encouraging and requiring students to co-operate and work together and can naturally encourage peer support. Team or group activities can be used in a variety of ways to assist in motivating and supporting students. Given the cohort sizes that many institutions have in CS, staff may not have the time to monitor and chase students regularly. However, in the context of a team project, the team members may do a certain amount of chasing and support or at least flag up to the academic manager that there is a problem with a team member.

When considering the culture and student communities, another feature to consider is unfair means and plagiarism. Team and group work are key skills that our graduates will need, and so encouraging the development of these skills through appropriate opportunities and activities is important. However, as we encourage co-operation and collaboration, this can also raise the problem of unfair means and collusion. Perceptions on the acceptability of copying software, understanding the appropriate use of libraries and existing algorithms, and being able to identify when it is appropriate to work together on a solution that is ultimately submitted individually can create a somewhat confusing environment for students to work within. Guides to unfair means and plagiarism of text-based resources do not typically cover the types of content and media that our students are working with. Advice about creative commons and various reuse licenses, along with illustrations of how code, algorithms and libraries can be appropriately referenced can also assist with that material. But the difficulty of when, where and how to co-operate, and how to report that can mean that students either avoid it and lose the potential benefits from working with peers, or do it inappropriately and end up with penalties for unfair means.

Another issue within the cohort is the disproportionately low number of females in Western Computer Science cohorts. As with student syndrome, another feature that is attributed to male students and may disproportionately affect Computer Science is the negative impact of online activity, in particular the implications on social skills through online games and adult content (Zimbardo and Coulombe 2015). There are alternate viewpoints – online communities in gaming, programming and open source may all offer effective alternatives to face-to-face communities and friendships (Wellman and Gulia 1999). However, should such online interaction reduce students' capacity to study and socialise in the context of their studies, it would contribute to the difficulties faced by students and reiterates the need to find the value added in campus-based education. One suggestion to tackle this problem is to adopt the kind of hooks that make games and online communities so attractive, utilising game mechanics and other technologies such as virtual reality and augmented reality to make learning more engaging.

International, BME and mature segments can also be significant minorities depending on the specific institution. Encouraging these groups with specialist get-togethers and support, as well as having activities that ensure that the overall classes are mixed, and working together, would benefit all students. These activities may be optional, extra-curricular social activities, or more structured in-course events such as treasure trails within supervisor groups, or actual module-based activities such as team projects. One point to be aware of in this context is that the continuation rates of these groups within Computer Science can be relatively poor (Jagacinski, Lebold and Salvendy 1988; Katz *et al.* 2006).

Many of the aspects considered so far interact – so prior computing knowledge and gender can affect success (Taylor and Mountfield 1994). As noted earlier, the imbalance in gender can affect the integration and sense of community within the cohort. Similarly, students can feel isolated where they feel most of their peers are already competent programmers as they begin the course. Departments should consider how to manage the combination of factors through appropriate induction processes, such as sessions that allow groups with similar characteristics to get together for mutual support and reinforcement, and other opportunities to encourage integration with the rest of the cohort.

Approaches to encouraging interaction and support among students in computing that have some evidence of success:

1. Paired programming (Salleh *et al.* 2011; Williams 2001): a key challenge for some students as they embark on a Computer Science course is learning how to programme. Paired programming involves getting students to work together in a pair on problems. This may be random pairing or based on students selecting their own partner. However, a more effective approach is to pair students with differing levels of programming experience – typically a novice with a more proficient programmer. As noted elsewhere in this report, the variation in prior programming experience within a cohort can be significant – within a single class, there will be

some proficient and experienced programmers, alongside novice and those for whom it is entirely new. Paired programming activities can provide one solution – that utilises an approach in agile software development and so is practice related. Pairing novices and experts is considered effective for both software development purposes, and for pedagogic purposes.

2. Informal team activities: this typically encompasses team-building activities and may start at induction onwards, with the benefit of helping students settle into their new environment. These may include games and social networking as ways to utilise technology to encourage involvement (Finkelstein *et al.* 2010; Fitz-Walter and Tjondronegoro 2011).
3. Formal teamwork: activities here may be formative or summative, and may be technically focused, or require students to do solve other problems that enable them to develop and demonstrate a range of teamwork and other key skills (Yerion and Rinehart 1995).

Specialised staff support

Taking account of some of the above issues, some institutions have instigated the post of student support advisors with a focus on attainment and retention; such student support, success and transition advisor roles are becoming increasingly common in HE generally, and in STEM and computing departments in particular. The exact focus for the role varies – success, achievement and support being typical key areas for the role to focus on. The role may be to provide direct support, or to signpost potential resources. This can take over some of the pastoral support that personal supervisors would be expected to perform, with the benefit of being separate from those responsible for teaching and assessing students.

Attendance

Attendance is another factor that can affect success: empirical evidence shows that there is a correlation between attendance and success, see for example Figure 3 and Figure 4 – while there is wide variation for a given absence rate, it illustrates the link between attendance and attainment. Of course, this may not be causal – students who opt to regularly attend classes and engage are likely to do better. Nevertheless, this correlation is widely regarded as demonstrating the potential of attendance as both a way to improve performance, and as a way to predict and identify those students likely to benefit from intervention and support. Moreover, many institutions and departments are increasing the requirements, expectations and associated monitoring of attendance and linking this with support, discipline and penalties. There are potentials here for feedback loops within the system that may be counter-productive, since students identified as being relatively poor attenders and with low attainment may then receive penalties and be denied the opportunity to take reassessments, thus increasing the likelihood of withdrawing or being deemed withdrawn from their course.

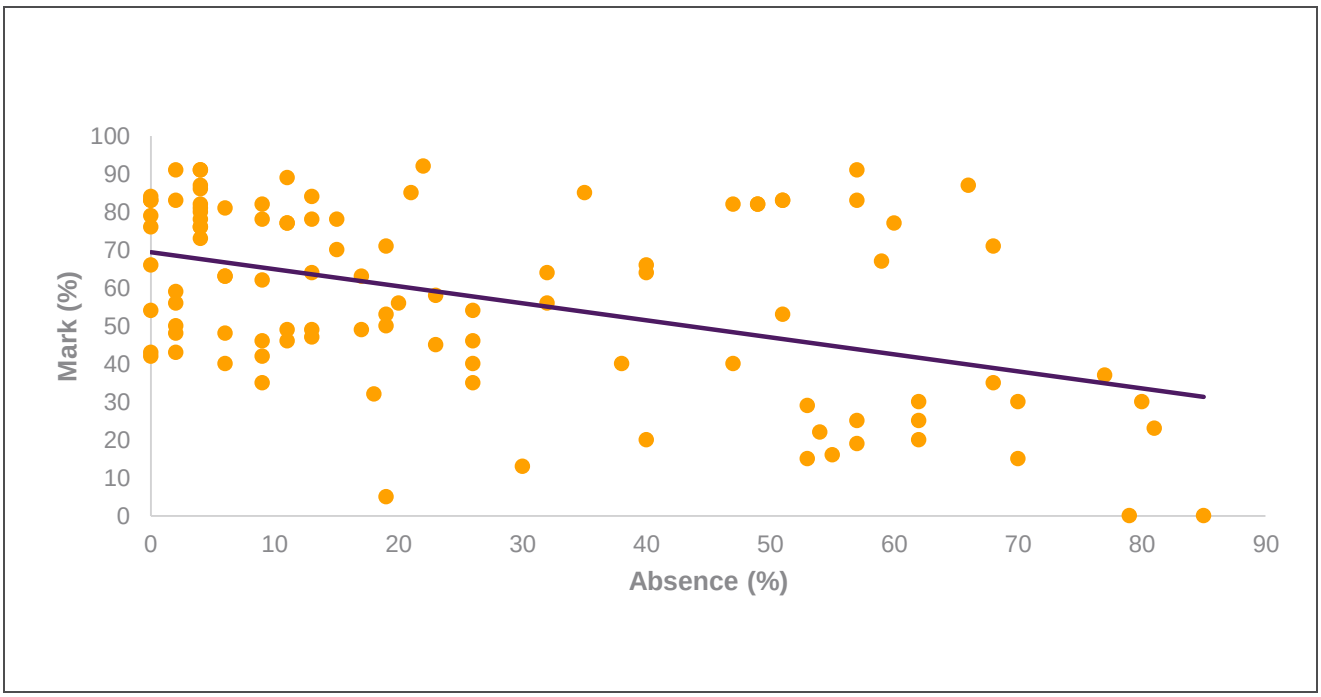

FIGURE 3: ILLUSTRATIVE EXAMPLE OF GRAPH OF MARK (%) AGAINST ABSENCE RATE (%) AND TREND LINE OF MARK AGAINST ABSENCE FOR LECTURE-BASED MONITORING, N=104 (GORDON 2014)

With largely male cohorts – among whom many play online games or do other online activities through the night, apparently preferring to work individually and remotely – there are difficulties in encouraging attendance. Approaches to attendance monitoring vary, from no monitoring, through unannounced audits, regular monitoring of some activities, or monitoring all activities. Collating and managing the data is a non-trivial task – with follow up dependent on resource and the intended remedy. Informal or formal warnings, meetings with supervisors and action plans can help some, but experience shows that many students will continue to be absent. Offering more and more online support – with lecture capture, lecture notes and other asynchronous support we may be aggravating this problem.

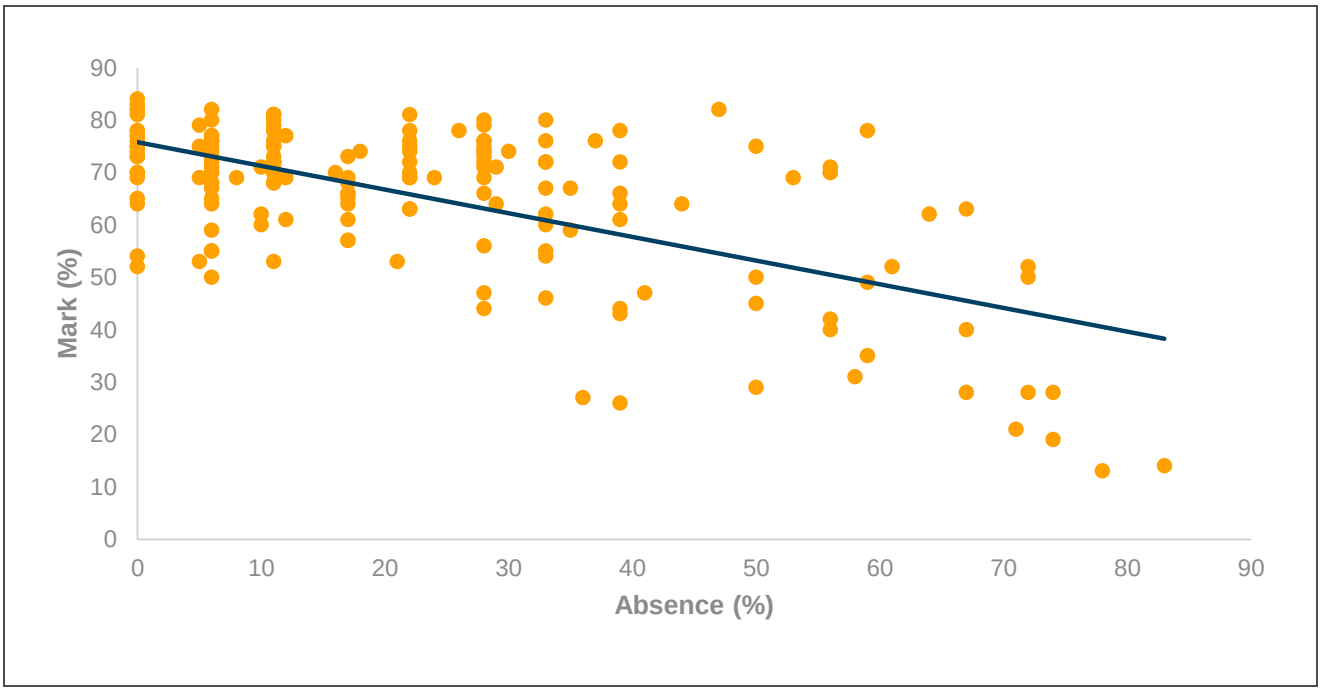

FIGURE 1: ILLUSTRATIVE EXAMPLE OF GRAPH OF MARK (%) AGAINST ABSENCE RATE (%) AND TREND LINE OF MARK AGAINST ABSENCE FOR LAB BASED MONITORING, FROM AUTHORS OWN DATA, N = 178 (GORDON 2014).

Stress and pressure on students in general, and on Computer Science students in particular, can have adverse effects on both attainment and retention. Any problems with time management – especially as a part of more general project management – can greatly accentuate pressure and stress. As a discipline with a strong applied and vocational nature, there can be a tendency towards coursework – at the extreme with entire programmes based on programming and other activities. While an effective way to demonstrate mastery of many aspects of the subject, such assessments, and the corresponding workload, can cause difficulties for students.

The concepts of well-being and mental toughness or resilience can play a role here – though ensuring that the right students know about, and engage with, such support is difficult and returns to the issue of autonomy versus dependence. Some disciplines utilise diagnostic tools to help (self) assess whether individuals are likely to need support – and may have a role in assisting the students at risk of dropping out or doing badly.

Some of the software development and general project management approaches that students are advised, required or choose to adopt may increase stress. Agile approaches to software development are identified as causing stress in the developers (Smith 2010). While a certain level of stress can be positive, too much can lead to failure. Such issues may be amplified where assessments and tasks are being set across several modules at once without an overall view of the workload on students. With large assessed coursework activities, there is a clear potential for students to be overworked.

5. Custom

The changing nature of Computer Science pre-HE is one area where custom in the discipline is changing. Entry requirements and qualifications have long been a cause of confusion, partly reflecting the different practices among institutions earlier in the educational pathway. It is rare for a Computer Science degree to require prior computing qualifications – while some welcome prior programming experience it is not a pre-requisite. Indeed, where there are any specific entry subjects, they are usually mathematics, with further mathematics preferred. Such pre-requisites typically either reflect a particular institutional approach to the Computer Science curriculum, or are being used as an indication of aptitude and to filter applications. This situation will have to be reviewed as the changes to school and college curricula build – with Computer Science now being promoted through initiatives such as the Computing at School (CAS) network, the Computer Science For Fun (CS4FN) initiative, and the move to replacing traditional IT/ICT with a more technical Computer Science approach from primary school through secondary and in further education. There are benefits to students having prior experience (Bergin and Reilly 2005), though until a point is reached where Computer Science is a common level three (A-level) qualification, this is not likely to be adopted due to its impact on potential recruitment.

Innovations such as the Raspberry Pi and the Arduino (e.g. see Campbell *et al.* 2015), the launch of the Micro Bit by the BBC to provide all 11-year olds in the UK with a programmable computer (BBC 2015), and other school or other community coding clubs, are among the positive initiatives contributing to the resurgent popularising of Computer Science. Some activities focus specifically on encouraging females into computing, such as the GirlGeeks (2015) community. It will take some time to see how this affects the uptake of Computer Science at school, college and ultimately at university.

Computer Science has a number of related level-two (GCSE) and level-three (A-level, BTeC and equivalent) qualifications. These vary quite drastically in terms of their focus and content. Related to these are transferrable (key skill) qualifications such as the ‘Use of IT’ or the ‘European Computer Driving License’ (ECDL).

Professional accredited courses: such as those offered by CISCO, BCS or Microsoft add to the complexity of Computer Science related qualifications, as well as potentially giving students a misapprehension of the broader and academic nature of university computing courses.

As full Computer Science A-levels, GCSEs and other qualifications become more popular and developed, the problem of managing student intake with a widely different preparation will grow. This is not a new phenomenon, but the scale is liable to increase: BTeC ICT (Software Development) and existing Computing A-levels mean that many students currently entering HE find they are repeating some elements of introductory programming. However, the nature of these until recently meant that the first year still provided new and challenging material: whether content wise (e.g. Software Engineering) or in terms of learning and assessment (exams would be novel to BTeC students until recent changes; though as BTeC has introduced more formal examination styles, and A-levels have moved to supervised coursework, many universities have been removing exams from year one, and adopting more unsupervised assessed activities).

As we enter an era where students entering HE may have been programming since primary school, have taken full Computer Science based GCSE and/or A-levels, the challenge will be in how to address this: Do university Computer Science departments demand Computer Science level three qualifications, or have separate pathways or courses depending on entry profile?

6. Gaps and areas for future research

- While the literature and evidence referred to in this report identify many issues, some of the emerging issues lack an evidence base. Some areas that would benefit from further work include:
- a review of the impact of the changes in pre-university computing – how effective this is at this stage, with monitoring of the uptake in Computer Science related qualifications;
- establishing effective indicators of risk of failing or dropping out, specifically for students of Computer Science. This may be attendance or other discipline-specific data such as the use of source control systems as considered earlier in this report. These indicators to provide a suitable basis for learning analytics, educational data mining and dashboards of student performance;
- examining how best to support students at risk, and what forms of intervention are appropriate – especially with regards to the atypical profile (gender, BME background, etc.) within Computer Science;
- which forms of teaching, learning and assessment practice within Computer Science education are most effective – enabling students able to develop and demonstrate suitable technical skills, and then able to apply these in industry;
- how to manage the tendency for procrastination with Computer Science students – the 'student syndrome' considered earlier in the culture section. This is especially critical given the proportion of coursework and associated deadlines. What level of assistance and intervention related to time and personal management, enabling students to succeed while avoiding too much negative stress;
- identifying the best approaches to encourage and support female, BME and mature students to both study, and succeed in Computer Science as considered in the culture section;
- to identify suitable strategies for improving engagement within Computer Science – building communities within the student cohort, encouraging good approaches to personal development and planning, while balancing monitoring and intervention with the freedom and autonomy of being a university student.

7. Recommendations

1. Firstly, there is a need to prepare and assist in managing expectations of students. Collaboration and direction from discipline bodies (BCS, CHPC) to provide a source of advice on routes into education and employment would be one approach. In particular, providing advice to careers staff, teachers and those working in schools liaison to ensure that we are managing the expectations of what Computer Science is and how the University experience differs to school and college.
2. Related to this, departments need to work with schools, colleges and the level three curriculum and examination bodies to ensure the articulation route into HE computing is working. This should include updates on the preferred – and potentially required – computing and other discipline qualifications appropriate to an institution's computing provision. This information should inform and define the marketing for a department and its programmes.
3. Within departments, the transition period – pre-induction, and induction (as a process throughout the first year of study) becomes critical. Ensuring that there is appropriate support (perhaps a transition, support or success advisor where these do not already exist) to ensure students have suitable study skills: time and project management, being able to identify and respond to indicators of stress, and developing mental resilience.
4. The transition to university level study should also have activities and processes to encourage students to become part of the community of their cohort. This may include icebreaker activities along with paired programming or other teamwork, and may utilise forums and social media to provide for different student needs. Developing this "culture of belonging" (Thomas 2012) is recognised as a key feature for successful retention and success among higher education students in general. Other features in the Thomas (2012) report worth noting in this context are the need to encourage active learning in the classroom and ensuring that support interventions become part of the mainstream provision within their course.
5. The increasing opportunities for technology to support Computer Science education is a further area for work. For learning analytics, departments should develop appropriate mechanisms to collate and utilise data on students – in particular evidence of engagement, such as attendance, resource access or assessment attempts. Technologies such as diagnostic computer-based assessment, code checking (for both functionality and plagiarism) and encouraging the use of commercial tools (code style, team collaboration and code management).
6. The issues of assessment, development of skills, encouraging behaviours and engaging students may be addressed with gamification approaches: considering the nature of assessments and other activities that may encourage students to work together, and to actively participate in their course.

8. References

Association for Computing Machinery (2013) *Curricula Recommendations*. [Internet]. Available from: www.acm.org/education/curricula-recommendations [Accessed July 10 2015].

Barjaktarovic, M. (2014) Increasing Diversity in Engineering, IT, and Computer Science. In: *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*: Las Vegas, USA, July 2014. pp. 1–6.

Barker, L.J., Garvin-Doxas, K. and Jackson, M. (2002) Defensive Climate in the Computer Science Classroom. *ACM SIGCSE Bulletin,* **34** (1) 43–4.

BBC (2015) *BBC Gives Children Mini-Computers in Make it Digital Scheme* [Internet]. 12 March 2015. Available from: www.bbc.co.uk/news/technology-31834927 [Accessed 20 August 2015].

Bergeron, B. (2006) *Developing Serious Games (Game Development Series)*. MA: Charles River Media.

Bergin, S. and Reilly, R. (2005) Programming: Factors that Influence Success. *ACM SIGCSE Bulletin*, **37** (1) 411–15.

Beyer, S., Rynes, K., Perrault, J., Hay, K. and Haller, S. (2003) Gender Differences in Computer Science Students. *ACM SIGCSE Bulletin*, **35** (1) 49–53.

Binkerd, C.L. and Moore, M.D. (2002) Women/Minorities in Computer Science: Where are They? No Attention No Retention. *Journal of Computing Sciences in Colleges*, **17** (5) 8–12.

British Computer Society (2010) *Guidelines for Academic Accreditation* [Internet]. Available from: www.bcs.org/category/7066 [Accessed July 10 2015].

Campbell, J., Sterritt, R., Moore, G., Bond, R., Fraser, S., Trombino, G., Cleland, B. and Hanna, P. (2015) Inspiring Computer Science Education in a Widening Access Context with Technology. In: *INTED2015 Proceedings (9th International Technology, Education and Development Conference)*: Madrid, Spain, March 2015, pp. 7563–70.

Clarke, V.A. and Chambers, S.M. (1989) Gender-Based Factors in Computing Enrolments and Achievement: Evidence from a Study of Tertiary Students. *Journal of Educational Computing Research*, **5** (4) 409–29.

Cohoon, J.M. (2001) Toward Improving Female Retention in the Computer Science Major. *Communications of the ACM*, **44** (5) 108–14.

Creative Skillset (2015) *Degree Accreditation* [Internet]. Available from: http://creativeskillset.org/who_we_help/training_educators/tick_course_accreditation [Accessed July 10 2015].

Digital Skills Taskforce (2014) *Digital Skills for Tomorrow's World* [Internet]. Available from: www.digitalskillscom [Accessed 1 July 2015].

Edwards, S.H. (2003) Rethinking Computer Science Education from a Test-First Perspective. In: *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*: Anaheim, CA, USA, October 2003, pp. 148–55.

Finkelstein, S.L., Powell, E., Hicks, A., Doran, K., Charugulla, S.R. and Barnes, T. (2010) SNAG: Using Social Networking Games to Increase Student Retention in Computer Science. In: *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education*: Ankara, Turkey, June 2010, pp. 142–6.

Fitz-Walter, Z. and Tjondronegoro, D.W. (2011). Exploring the Opportunities and Challenges of Using Mobile Sensing for Gamification and Achievements. In: *UbiComp 11: Proceedings of the 2011 ACM Conference on Ubiquitous Computing*: Beijing, China, September 2011, pp. 1–5.

GirlGeeks (2015) *About GirlGeeks* [Internet]. Available from: www.girlgeeks.org/about/about.shtml [Accessed 1 July 2015].

Gordon, N. (2009) Enabling Personalised Learning through Formative and Summative Assessment. In: O'Donoghue, J. *Technology-Supported Environments for Personalized Learning: Methods and Case Studies*. Hershey: Information Science Publishing, pp. 268–83.

Gordon, N. (2014) *Using Pedagogy and Learning Analytics to Manage Our Students* [Internet]. Available from: www.heacademy.ac.uk/sites/default/files/resources/gen-108-o.pdf [Accessed 21 August 2015].

Gordon, N., Brayshaw, M. and Grey, S. (2013) Maximising Gain for Minimal Pain: Utilising Natural Game Mechanics. *Innovation in Teaching and Learning in Information and Computer Sciences*, **12** (1) 27–38.

HEFCE (2015) *Graduate employment and accreditation in STEM: Computer science employability and accreditation* [Internet]. Available from: www.hefce.ac.uk/kess/gradstemreview/csreview/ [Accessed 7 January 2016].

HESA (2014a) *Destinations of UK Domiciled Leavers 2008-09 by Level of Qualification Obtained, Mode of Study, Subject Area and Activity* [Internet]. Higher Education Statistics Agency. Available from: www.hesa.ac.uk/dox/dlhe_longitudinal/0809/tables/long_dlhe_0809_all_tables.xls [Accessed 1 July 2015].

HESA (2014b) *Destination and Leavers of Higher Education (DLHE) Introduction 2013-14*. Higher Education Statistics Agency [Internet]. Available from: www.hesa.ac.uk/intros/dlheintro1213 [Accessed 1 July 2015].

HESA (2015) *Destination and Leavers of Higher Education (DLHE) Statistical First Release 217* [Internet]. Higher Education Statistics Agency. Available from: www.hesa.ac.uk/index.php?option=com_content&view=article&id=1899&Itemid=634 [Accessed 1 July 2015].

Holly E. (2014) Surge in Popularity of Two-in-One Degree Courses [Internet]. *Times Higher Education*, 31 July 2014. Available from: www.timeshighereducation.co.uk/news/surge-in-popularity-of-two-in-one-degree-courses/2014853.article [Accessed 21 August 2015].

Jagacinski, C.M., Lebold, W.K. and Salvendy, G. (1988) Gender Differences in Persistence in Computer-Related Fields. *Journal of Educational Computing Research*, **4** (2) 185–202.

Katz, S., Allbritton, D., Aronis, J., Wilson, C. and Soffa, M.L. (2006) Gender, Achievement, and Persistence in an Undergraduate Computer Science Program. *ACM SIGMIS Database,* **37** (4) 42–57.

Keenan, C. (2014) *Mapping Student-Led Peer Learning in the UK* [Internet]. The Higher Education Academy. Available from: www.heacademy.ac.uk/resource/mapping-student-led-peer-learning-uk [Accessed 21 August 2015].

Lindsay, S. (2011) Do Students in UK Higher Education Institutions Need Personal Tutors? *Learning at City Journal*, **1** (1) 40–5.

Luck, C. (2010) Challenges Faced by Tutors in Higher Education. *Psychodynamic Practice,* **16** (3) 273–87.

Manninen, S. and Wadsö-Lecaros, C. (2014) Avoiding Procrastination and the Planning Fallacy: Implementing Study Plans as a Strategy to Increase Student Achievement. In: *Pedagogisk inspirationskonferens för HT-fakulteterna*: Lunds, Sweden, September 2014.

Margolis, J. and Fisher, A. (2003) *Unlocking the Clubhouse: Women in Computing*. Cambridge, MA: MIT Press.

Morgan, J. (2014b) Postgraduate Loans of £10K Announced by George Osborne [Internet]. *Times Higher Education*, 3 December 2014. Available from: www.timeshighereducation.co.uk/news/postgraduate-loans-of-10k-announced-by-george-osborne/2017368.article [Accessed 21 August 2015].

QAA (2007) *Computing Benchmark* [Internet]. Quality Assurance Agency for Higher Education. Available from www.qaa.ac.uk/assuring-standards-and-quality/the-quality-code/subject-benchmark-statements/honours-degree-subjects [Accessed July 10 2015].

Roberts, E. (2000) Strategies for Encouraging Individual Achievement in Introductory Computer Science Courses. *ACM SIGCSE Bulletin*, **32** (1) 295–9.

Rountree, N., Rountree, J., Robins, A. and Hannah, R. (2004) Interacting Factors that Predict Success and Failure in a CS1 Course. *ACM SIGCSE Bulletin*, **36** (4) 101–4.

Salleh, N., Mendes, E. and Grundy, J. (2011) Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, **37** (4) 509–25.

Samuels, J. (2014) NextGen Academic Support: From Face-to-Face Tutoring Online to Intelligent Tutoring Systems. In: *World Conference on Educational Multimedia, Hypermedia and Telecommunications*: Tampere, Finland, June 2014, Issue 1, pp. 539–45.

Shadbolt, N. (2015) *Unemployment among computer science graduates – what does the data say?* [Internet]. Available from: http://blog.hefce.ac.uk/2015/07/08/unemployment-among-computer-science-graduates-what-does-the-data-say/ [Accessed 7 January 2016].

Smith, G.G. and Taveras, M. (2005) The Missing Instructor. *e-Learn,* **2005** (1) 1.

Smith, D. (2010) The Effects of Student Syndrome, Stress, and Slack on Information Systems Development Projects. *Issues in Informing Science and Information Technology*, **7**, 489–494.

Talton, J.O., Peterson, D.L., Kamin, S., Israel, D., and Al-Muhtadi, J. (2006) Scavenger Hunt: Computer Science Retention through Orientation. *ACM SIGCSE Bulletin*, **38** (1) 443–7.

Taylor, H.G. and Mounfield, L.C. (1994) Exploration of the Relationship Between Prior Computing Experience and Gender on Success in College Computer Science. *Journal of Educational Computing Research*, **11** (4) 291–306.

Thomas, L. (2012) What Works? Student Retention and Success [Internet]. York: Higher Education AcademyAvailable from: www.heacademy.ac.uk/sites/default/files/what_works_summary_report_0.pdf [Accessed 17 August 2015].

UCAS (2015*) Applications by Subject at the 15 January Deadline* [Internet]. Available from: www.ucas.com/sites/default/files/subjects_mr_january_150115.pdf [Accessed 17 August 2015].

UCISA (2014) 2014 *Survey of Technology Enhanced Learning for Higher Education in the UK* [Internet]. Available from: www.ucisa.ac.uk/~/media/groups/dsdg/Tel%202014%20Final%2018%20August.ashx [Accessed 21 August 2015].

Waite, W.M., Jackson, M.H., Diwan, A. and Leonardi, P.M. (2004) Student Culture vs Group Work in Computer Science. *ACM SIGCSE Bulletin*, **36** (1) 12–6.

Wellman, B. and Gulia, M. (1999) Virtual Communities as Communities. In: Kollock, P. *Communities in Cyberspace*. London: Routledge. pp. 167-194.

Williams, L. (2001) Integrating Pair Programming into a Software Development Process. In: *14th Conference on Software Engineering Education and Training*: North Carolina, USA, February 2001, pp. 27–36.

Woodfield, R. (2014) *Undergraduate Retention and Attainment across the Disciplines* [Internet]. York: Higher Education Academy. Available from: www.heacademy.ac.uk/node/10293 [Accessed 15 June 2015]

Yerion, K.A. and Rinehart, J.A. (1995) Guidelines for Collaborative Learning in Computer Science. *ACM SIGCSE Bulletin*, **27** (4) 29–34.

Zimbardo, P. and Coulomb, N. (2015) *Man Disconnected: How Technology has Sabotaged What it Means to be Male*. London: Rider.

Contact us

+44 (0)1904 717500 enquiries@heacademy.ac.uk
Innovation Way, York Science Park, Heslington, York, YO10 5BR
Twitter: @HEAcademy www.heacademy.ac.uk

Higher Education Academy (HEA) is the national body for learning and teaching in higher education. We work with universities and other higher education providers to bring about change in learning and teaching. We do this to improve the experience that students have while they are studying, and to support and develop those who teach them. Our activities focus on rewarding and recognising excellence in teaching, bringing together people and resources to research and share best practice, and by helping to influence, shape and implement policy - locally, nationally, and internationally.

HEA has knowledge, experience and expertise in higher education. Our service and product range is broader than any other competitor.

To request copies of this report in large print or in a different format, please contact the communications office at the Higher Education Academy: 01904 717500 or pressoffice@heacademy.ac.uk

Higher Education Academy is a company limited by guarantee registered in England and Wales no. 04931031. Registered as a charity in England and Wales no. 1101607. Registered as a charity in Scotland no. SC043946.