Teaching Delivery Issues - Lessons from Computer Science

Janet Carter
University of Kent at Canterbury
J.E.Carter@ukc.ac.uk

Roger Boyle
University of Leeds
roger@comp.leeds.ac.uk

Executive Summary

Information Technology (IT) is a subject that is distinct from Computer Science (CS), but is often taught by CS faculty; there is a large overlap between the content of curricula for the two subjects. In this paper, we discuss some of the issues and problems experienced within CS that are also of relevance to the IT educator. We discuss the effects of student and faculty expectations along with curricular issues, and we conclude that setting student expectations and aligning them with our own at as early a stage as possible is crucial to success.

Expectations

Students: Students arrive at University from a much broader range of backgrounds than previously; they may be mature or have experienced a non-traditional education. When they draw upon their past experiences to help interpret this new environment they are drawing upon scenarios that many faculty have not experienced. Many students also arrive with different expectations regarding higher education. In some cases, the benefit of a degree with the correct title far outweighs any thirst for particular knowledge.

Faculty: Many faculty members previously qualified in a different subject; they want to teach CS and don't want to teach IT; they find it difficult to recognize students' lack of domain understanding which is a required underpinning; they expect their implicit expectations to be noted and acted upon by students without making them explicit.

Curriculum

The breadth of the IT curriculum requires a variety of approaches and delivery styles, dependent upon context, in order to cater for the differing learning styles and backgrounds of the students. The successful use of 'Theatre' within the lecture room is dependent upon the individual faculty member's personality. It is not the case that something that works in one area will necessarily successfully transfer to another, and there is therefore a need to allocate suitable solutions to problems.

Programming: Programming (not necessarily coding) is a skill that aids advanced understanding and we present some practical suggestions for helping students to improve.

Formality: Material must be explained within context and not just presented as abstract theories. Progressive disembedding coupled with surface

learning is an excellent way to boost confidence before we can expect any deep learning to occur.

Transferable skills: Many useful transferable skills can be employed within the IT curriculum. Here we discuss some of the more profitable and successful along with guidelines for replication.

Keywords: expectation, past experience, delivery, formality, programming, transferable skills.

Introduction

'IT', in its broadest sense, is of overwhelming importance to modern societies and their educational systems. The collapse of traditional industries has been mirrored by the growth of technology based systems, and the ability for people to be able to function fluently within them will determine the success or otherwise of the parent society.

People need not only to know how to 'use' IT (in the sense of getting money from an ATM at their bank) but also increasingly will need to 'understand' it. Anecdotally, numerous employees can use version *N* of a word-processor, but are floored by the appearance of version *N+1*, since they have only a superficial, button-pressing comprehension. Such people are suffering from inadequate conceptual understanding - a shortfall in IT education.

There is thus a significant problem in education, which stretches from pre-school through adulthood. There is an argument to be held over the 'depth' of IT education necessary for the broader population, but it is clear that there is a need to go beyond the rote learning of which button makes what happen. The need for specialists - that is, advanced education - grows at the same time. The demand for graduates and Ph.D.s in 'Computer Science' has grown year on year for the last decade or more, and is unlikely to stop.

Within many educational systems, there is a blurring of the distinction between Information Technology (IT) and Computer Science (CS) since it is not clear where one stops and the other starts. Many high-school teachers are not even aware that the subjects are different. The awareness and distinction do, however, become apparent once University level is reached. This is despite CS and IT both usually being taught by the same members of faculty, and IT often being taught under the guise of 'Applied Computing'. In academic circles, the distinction is important. There is a stated Government desire (in the UK, the Dearing report, (NCIHE, 1997), in the US, the Snyder report (Snyder, 1999)) for all graduates to be 'IT fluent', and it is common for CS departments to take on some or all of this responsibility. Meanwhile, the number of specialist degree Programs is enormous (in the UK, over 2000 different Programs exist). The curricula of these Programs are reasonably well specified (ACM, 2001; QAA, 2001), and it is possible to summarize the obvious differences and overlaps (figure 1).

There is much overlap between the 'softer' end of CS and the 'harder' end of IT, and this suggests that many of the issues that have recently been raised within CS are also relevant to IT. CS education is widely documented to be 'difficult' for a number of reasons and much work has gone into solving the problems it raises. Many of these issues are equally applicable to IT education as well. We present here some work on the teaching of CS that is of relevance to the wider IT education problem.

The Problem

There is a widespread recognition that traditional University teaching delivery is flawed. It is fair to assume that this was always the case, but this truth is made more apparent in a context of broadening intake that comes with new forms of expectation and motivational problems. Solving this problem is particularly difficult unless faculty appreciates its existence, but even then, most University Programs deploy a range of techniques in syllabi that contain a wide range of material. Even when faculty do appreciate the need for change, it is unlikely that all staff will all have the skills required to present ideal de-

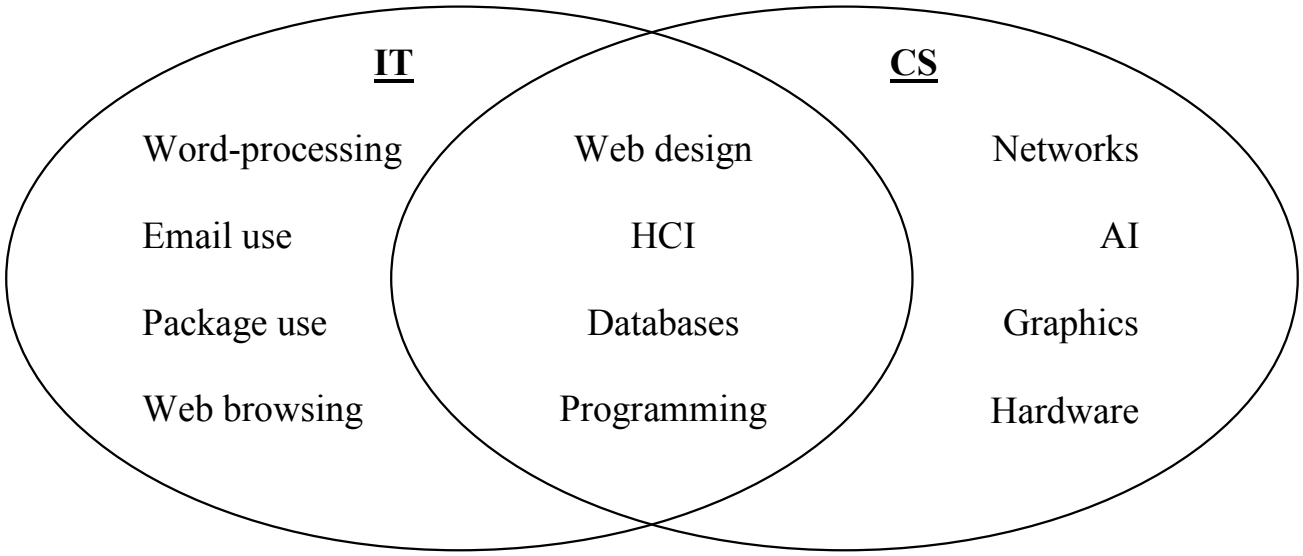

Figure 1: Common perceptions of IT and CS 'content'

livery. We can thus identify a problem of matching solutions to problems given a range of constraints in the standard department

Here, we present this problem within the context of CS, which we argue is an extreme case of the issue; we identify some options and possible solutions, and note that the conclusions we draw are applicable to a range of disciplines.

Background

Computing as a University discipline is young and in few institutions firmly settled into the academic fabric. Despite the publication of curricula (ACM, 2001) there is a lack of discipline definition that, coupled with the nature of students and staff, makes it an extreme example of the problems facing teachers in Higher Education (HE) in several ways simultaneously;

Curriculum

Even if agreement is reached on what Computing 'is', it has a breadth of curriculum that is surprising in most disciplines; curricula vary from theoretical through practical, from hardware through software, from scientific through commercial in application. It overlaps with a range of other disciplines, with increasing attention being devoted to the 'professional issues' now seen as essential (Townsend, 1999). Coupled with this is the rate of change that requires the regular rewriting of most syllabi and the cascading down the curriculum of material that is seen as the hallmark of a healthy Program.

Students

The clear benefits of a degree involving Computing attract many students, but often without the thirst for University learning that faculty like to see. In pressured economic climates, students often exhibit advanced 'strategic' (Kneale, 1996) approaches that focus them on grades and rapid completion and that make them a serious teaching challenge. Programs are accessible to wide ranges of students, meaning that audiences are as heterogeneous as the curriculum, often with unusual numbers of mature and other non-traditional entrants. This is often exacerbated by gender imbalances, with Program intakes display-

ing a gender bias that also presents challenges to the unwary teacher (Carter & Jenkins 1999, Haller & Fossum, 1998).

Faculty

CS departments usually have a disproportionate number of staff members who have moved discipline - historically because they had no opportunity to qualify in the discipline before it existed. When coupled with curriculum breadth, this can present significant problems in matching faculty to material, especially in the areas in which recruitment is difficult given the industrial and commercial competition for personnel - networks, systems and web-design are good examples of this.

Thus, there is an ill defined and changing discipline, with a growing population that produces particular problems to staff, who may well come from a different background academically. Aspects of these issues may be observed in all disciplines, and all can be directly applied to IT. Thus any solutions to the problems raised within CS are of relevance to the IT community.

Delivery Issues

Delivery of material in education is clearly important. It is apparent that students do not learn by simply being present in a room whilst someone stands at the front and attempts to transmit information to them, particularly with a practical subject such as IT. They need to be active in the process: interacting with the concepts, testing out theories, being allowed the space to make mistakes (Ben Ari, 1998; Donaldson, 1978; Kolb, 1984). If we are to accept this constructivist model of education, and if students are to construct meaningful and relevant mental models of the issues and the material presented to them, then a supportive environment is clearly required. This can be problematic for teaching staff, as students will demonstrate a variety of learning styles that need to be accommodated. The issue is exacerbated if we admit that a generic supportive environment, applicable across the entire discipline, cannot exist.

Coupled to this is the issue of expectation. For most, the University experience is new, whatever their prior experience. A very limited window of opportunity exists in which to provide new students with an expectation of how the system is expected to work, and the majority of this expectation will be derived from experience in formal delivery (‘lectures’). Faculty exhibits a range of reactions to this issue, some structured and some personal. Examples of this are:

- Generally sound pedagogic approaches of maintaining attention and variety can be seen in most disciplines, particularly when faculty members have formal training in teaching.
- Likewise, student participation is usually encouraged, although the degree of this is variable, being dictated by class sizes (often very large), ergonomics of the teaching space, staff pre-conceptions, the nature of material coupled with available resources.
- In Mathematics, and similar formal ‘service’ material, it has long been acknowledged (Donaldson, 1978) that the ‘slow and steady’ approach is important to reducing the intimidation felt by students hostile to the subject matter.
- In Programming (a discipline specific issue), there is a wide acceptance of the problem of communicating understanding (Davy & Jenkins, 1999; Hagan & Sheard, 1998) that leads to specially tailored regimes.
- Highly personalized approaches are also evident (Astrachan, 1998; Siegel, 1999) that seek to maintain and, within the context of learning, entertain.

There is thus a range of weapons in the faculty armory. Some of these are reliable and based in known good technique, and others speculative. Matching them to curricular matter, and seeing them used by

staff, remains a problem. Additionally, there are issues connected to the audience that need to be considered:

- New students will draw upon past experiences to help them understand what is going on; this shapes their expectations and their reactions to new experiences. Students from other cultures, even single-sex versus co-educational schools, will be attempting to make sense of the HE scenario in the light of differing previous experiences. Often this will manifest itself as competition.
- Students have natural learning styles, which they prefer. How does particular staff behavior affect activists, reflectors, theorists, and pragmatists? This can be addressed by incorporating a variety of learning experiences into each module (not just the overall curriculum) (Kolb, 1984; Lewandowski, 1998).
- The cohort is changing constantly; increasing numbers of mature students, and students with non-traditional entry qualifications suggest that student expectations of HE must also be changing.
- Teaching staff also bring past experiences to bear upon the classroom situation. Whether it is a misguided belief that all females have the same learning style because most adopt a particular learned working style (Ben Ari, 1998), or an expectation that the present cohort will behave similarly to a previous one. The projection of these beliefs will necessarily affect the ways in which the students respond.

Problems and Techniques

The delivery issues raised here are both important in their own right and of relevance to the IT educator. Student expectations upon arrival at University and their level of preparedness for formal and logical reasoning cannot be ignored; also their ability to connect the practical skills of the subject to the abstract theory behind it must be developed.

Expectation

When students arrive at University for the first time they have no past experiences of HE upon which to draw; many expect the experience to be similar to that of the school or college that they have just left. As school league tables become more important in the UK, the school experience becomes a less useful model on which to base such assumptions. Some schoolteachers finding high grades the only metric of importance increasingly teach to the exams rather than encouraging good study skills or exploring any interesting but non-examinable aspects of their subject - there simply is not time. After an experience such as this, students understandably arrive at University with little notion of what independent study actually entails.

It is also the case that many members of faculty believe that, despite the recent changes in the school system, the past experiences of the new students are similar to their own (often rose-tinted) memories of school. This mismatch in expectations, if not addressed, can lead to problems.

The range of potential problems is enormous and includes both discipline specific and general educational issues. Some of the more specific issues for IT often include an assumption that students will automatically be able to use institution-specific set-ups. For example, it is often not the case that a student taking ‘advanced word-processing’ cannot use Microsoft Word, but rather they do not know how to access it under NT when they are used to a Windows 98 environment. There is much more scope for such problems to arise within IT and CS than most other disciplines.

It is essential that staff make their expectations explicit at the outset, rather than implicitly expecting students to simply know. It is obviously essential that this is done at the very start of the year, and that all members of staff agree on what is expected.

Formality

Many disciplines have aspects for which formal reasoning is required, and this is an area in which the differences between deep and surface learning can be best exploited. For example, the formal reasoning (Mathematical) content is usually the most problematic area of a CS degree Program, and CS students are frequently unconfident in their Mathematical ability. Explaining why or how a particular method works and is relevant is more important to them than an algebraic proof or the formal reasoning behind the result. When students think they can already follow the reasoning they are much more likely to engage with the (often alien) symbols. One simple and effective way to engender confidence is easy to achieve in any standard lecture room. After completing an example of a particular problem, ask the students to attempt one that is similar. This provides an almost instant feedback to the instructor about aspects that require revision, but it is also beneficial to the students. This surface learning provides a welcome confidence boost, which many students require before applying the theory to solving problems within a different context. On the other hand, this is not an approach that is suited to the more discursive areas of the curriculum.

Practical Issues

Especially in science and engineering, there is an explicit practical component to curricula. It is in HE that the craft skills of the professional are developed, and the teaching of these skills is often challenging and critical. In Computing, the problems center on the teaching of Programming, and problem solving in understanding and using complex software packages.

It is important to realize that the skills of understanding how Programs work, if not the actual skill of writing code, is one that applies to all IT education, not solely the specialist degree courses. Phrasing a database enquiry, for example, is a much simpler task with a rudimentary understanding of logical constructions, and an appreciation that precision is non-negotiable in our dealings with computers. While it will not always be necessary or desirable to ‘teach Programming’ at the coding level, an understanding of how to communicate coding skills is an important part of the IT educator’s armory.

The challenges of teaching Programming are widely recorded. A technique that has been shown to be successful, and which has been successfully transferred, involves the use of ‘discussion classes’ (Davy & Jenkins, 1999; Hagan & Sheard, 1998), held deliberately remote from a computer, which engage students in participative discussion of (usually) technical coding material. This is an interesting approach because not only has it been seen to improve student learning, but is tuned to this particular segment of the curriculum. It is of course possible to use it within other sections, but it has been found particularly easy to install in the elementary software engineering area. Students are split into small groups and timetabled to be in a seminar room rather than a computer room for the session. It does not make specific demands on the staff custodian of the module, although does require a population of supporting tutors.

It is well known that the teaching of software systems is non-trivial. This is an area of direct relevance since it notes the need to consider the background of the learners - for example; novices do not share the domain understanding of software metaphors that often underpin such systems (Mantovani, 1996). This is of relevance in all disciplines as the use of computers pervades HE.

Techniques from HCI make it possible to look at the behavior of learners, and to inspect what they have learned and when, and thereby improve delivery (Thomas, 1998). This is an area where improvement in learning can be measured directly in performance rather than though the filter of ‘grades’. This is an il-

lustration perhaps of the obvious, but in most curricula, and especially IT, some critical parts will rest upon students coming to grips quickly with suites of software to enable them to learn more abstract or advanced material. This phase is often underestimated in difficulty and often leads directly to a student's failure to comprehend the material that follows. Closer attention to it would ease the larger learning tasks.

Interleaving Delivery, Assessment and 'Transferable Skills'

Increasingly, HE is being asked to develop and rehearse 'transferable skills,' and it is important to embed these in core curricular matter to reinforce their relevance. It is possible to do this, as part of standard delivery - a particular exercise, devised to rehearse student readiness to address a large audience at little notice, is the 'spot talk'. Students are given a list of topics and told to be ready to talk authoritatively at zero notice about one of them for five minutes. This technique depends to some degree on the staff using it having the force of personality to make it work, but this is not an enormous obstacle, and this may be applied in a wide range of subject areas.

There are many ways to move away from examination as the major form of assessment, but opportunities exist to interleave it explicitly with delivery. One mechanism is the multi-choice questionnaire (MCQ), completed by students on pre-printed forms that are then assessed by some cheaply available OMR package. This simple approach has interesting features apart from its extreme economy. It is useful in subject areas that are assessed predominantly by coursework in pseudo-examination conditions; it can identify absence of understanding that may be being concealed by plagiarism in submitted work. Anecdotally, students have also professed it to be an easy and fair means of assessment, especially when it is used as a basis for tailoring the remaining delivery to their particular cohort: It is thus a mechanism that permits easy adaptability of teaching. Most interestingly, it is especially useful for moving students' understanding from surface to deep since it is straightforward to introduce questions that mimic material recently presented, and make them progressively different and more difficult.

The use of 'theatrical' and similar techniques is now well known (Astrachan, 1998; Siegel, 1999). Anecdotal evidence suggests that this sort of approach is popular among many students, particularly the younger members of the cohort, but firm evidence that it enhances learning is not available. Such an idea is usual entirely dependent on its originating personality and dies with the disappearance or reallocation that individual - this has been witnessed several times. Staff who engage in this are thus a resource of a particular kind that management can view as an asset or not as they see fit, but not a model *pour encourager les autres*.

Specific Teaching Areas

Two areas that stand up to a more in-depth investigation are Programming and Mathematics. Programming, as mentioned earlier, is an essential skill that is important throughout IT. Mathematics, despite its lack of relevance to most of the IT curriculum does share with IT the dubious honor of being a 'service' discipline - it can be studied for its own sake, but is often viewed simply as a necessary evil to aid comprehension of the students' chosen discipline.

Programming

The skill of Programming is essential to CS students. Having said that, it is not in any sense intellectually advanced; it is something that all novice students will be required to do in order to acquit any advanced work. Students of more general IT courses may well also have formal Programming to study. Where this is not the case, we contend that skills in this area (algorithmic representation, formal syntax

etc.) are important and worthwhile to these more general areas. The teaching of Programming is therefore of broad relevance.

The teaching of Programming in our home institutions has gone through a number of phases in attempts to meet the many problems it raises. Frequently, one discovers little correlation between a talent for Programming and obvious indicators such as High School grades or Mathematics qualifications. There is a perennial problem in teaching this essential skill to people who find it genuinely confusing and difficult, whatever their intelligence.

Certainly the traditional University approach of formal lectures supported by out-of-class exercises fails in this domain. Students without innate talent and motivation simply do not learn. One obvious amendment to the regime is to have many scheduled laboratory classes with a high ratio of demonstrators or tutors standing by, and this improves matters but is costly and still imperfect.

Using assessment as a driver can see an improvement. A common format for such practical modules is to set significant quantities of assessed work, perhaps one piece per week. Normally this will be to the exclusion of formal examination. Some aspects of this move are good and some less so; certainly, formal examinations in skills such as Programming are of limited value, since the exam room bears no resemblance to the environment of the Programmer in which it is fair to take reference manuals and a computer for granted. On the other hand, the superficially attractive way of 'making' students work by dint of demanding assessed submissions fails when the work turns into a treadmill rather than a learning experience. Experience suggests that requiring more than, say, four pieces of submitted work in a semester of twelve weeks may have this precise effect. We have seen students pass with entirely adequate grades as a result of handing in many pieces of satisfactory work, only to turn out to be inadequately skilled Programmers when things get 'real'.

One aspect of this is certainly plagiarism, on which there is an extensive literature (Dick, 2001; Plagiarism.org, 2001; Culwin, 2000). The strategically minded student will often resort to more and more evasive forms of plagiarism in an effort to 'get the work done', whilst bypassing any form of learning whatever. We have found one antidote to this to use short, but closed, in-class tests based on earlier practical work. Such tests are simple for those who have genuinely done the work, but impossible otherwise.

What is important is actually to communicate with the students. A satisfactory solution, but beyond any institution's resources, is for each individual to have a single tutor. We have found an intermediate solution by breaking a large class (perhaps 300 students) into seminar groups of twenty or so, and requiring them to meet in a workshop mode once per week with a tutor, that brings the resources within reasonable bound. These workshops are held away from a computer laboratory, so all the work is verbal, or pen-and-paper or whiteboard. The group size is small enough to require everyone to do something, and the environment is non-threatening and encourages discussion of the problems at hand.

Staffing it with the most suitable tutors of all - the students, can enhance this regime yet further. Students who have recently studied the same material are peculiarly well placed to understand the problems that novices experience - with practical issues such as programming, this is truer than with strictly academic material. This idea has been used elsewhere (Roberts, 1995), and we have re-used it with great success; replacing the tutors with student tutors costs no more, and very significantly enhances the quality of supervision. The student tutors also provide role models for the novices, who see the job as an opportunity for themselves.

We have thus seen an incremental shift in our teaching of Programming away form 'chalk and talk', through (over-) assessment based ideas, to what we regard as imperfect, but better than all predecessors. This is a good example of the tailoring of delivery to material that illustrates the special purpose nature of parts of the curriculum well.

Mathematics

IT, like Mathematics, is often thought of / taught as a set of disembedded skills which are divorced from the real world; one needs to know how to solve equations such as $7x+3=17$ or how to emphasize text within a word processor because they are skills which may 'come in useful' some day. Learning abstract disembedded skills that may be transferable, of benefit and applicable to many different contexts may be the ultimate goal of educators, but is a daunting task. "People know what they do, they frequently know why they do what they do, but they don't know what what they do does." (Foucault, 1991) is an apt description of the situation. In keeping with the link between CS and IT, Mathematics is a service subject for CS rather like IT is seen to be a service subject for many disciplines. Much has been written about Mathematics education in recent years (Klein, 1998; Boylan, 2000), especially since the introduction of the National Curriculum within UK schools.

The current philosophy behind Mathematics education is based upon the works of Piaget and Vygotsky; this is rooted within the discipline of psychology and is characterized by levels of attainment although it does make a small contribution towards our understanding of social processes. The Piagetian model conjures up images of a lone student puzzling over increasingly decontextualized problems and, if successful, passing through from one level of understanding to another until eventually formal abstract reasoning is attained (Piaget, 2000). Vygotsky argued that 'it is not what we can do alone but with assistance which is important' (Vygotsky, 1993) but still there remains the concept of levels. Hence it is often supposed that Mathematics is something one can do or one cannot - there is no middle ground. IT has the same reputation; people are labeled as either IT literate or technophobes. Gates (2000) argues that education as a social practice plays a part in sustaining such beliefs. In order to understand this better both the social context and the practice must be investigated. Teachers play a key part in this so the characteristics they adopt and their orientations cannot be ignored. The level of disparity between experiences and backgrounds of members of CS faculty may appear problematic on the surface but, if harnessed correctly, could be of great benefit to the students.

A further issue that cannot be ignored at the HE level is the fact that students have previously been exposed to both Math and IT (sometimes, due to lack of training and school teacher shortages, by teachers who are less than well qualified in the subject) in schools and have already formed a mental model of the disciplines before arriving at University. They bring this conceptual model to bear when exposed to the subjects within their new environment. So it is not simply a case of 'filling the blank slate' when we start to educate them. Gates (2000) states that the Mathematics classroom is not merely a neutral arena for the exercise of pedagogy and the ways in which teachers organize activity in the classroom has wider implications for the nature of society and the opportunities of learners within it. This can also be applied to the IT classroom because of the similarity between the roles of the two subjects.

Recommendations

We suggest that a combination of disparate curriculum, student audience and staff base demand particular thought in choosing appropriate *Horses for Courses* in IT delivery. We consider that the reasoning is applicable at least in part to many or most other disciplines, and that we are looking at extreme case of a widespread problem. The range of approaches open is wide, with some being much more specific than others are. It is not the case that something that works in one area will necessarily work elsewhere. This is especially true in the case of 'charismatic' approaches in which the personality of the deliverer is key to the event and mimicry would fail badly.

For students, it is important to set expectations early but this is a challenge when they arrive with a wide variety of backgrounds. Critically, it is important to move them away from the high-school mode of working, and encourage a sense of enquiry and self-motivation, and techniques to achieve this across the

cohort are elusive. Motivation is a particular problem among many modern high-school graduates (Kneale, 1996) and its absence quickly invalidates many approaches to teaching. Meanwhile a significant proportion of the intake does find certain parts of the curriculum straightforward. This is especially true of teaching Programming and systems such as word-processors, spreadsheets and the WWW to large mixed classes (Davy & Jenkins, 1999; Hagan & Sheard, 1998), but it is also a challenge 'teaching' issues of professionalism and project management to mature recruits who have been working in commerce for 5 or 10 years. Charismatic techniques can prove popular with young students but counter-productive with mature audiences. Further problems with diversity arise in gender imbalance and the different attitudes to learning of men and women, which are particular evident in technical material (Carter, 1999).

For faculty, unless blessed with very favorable staff-student ratios, there is often a need to deliver in areas outside one's own specialty, and to maintain up-to-date knowledge in them. This is a managerial challenge even before proceeding to change staff habits in their attitude to new delivery techniques, and makes staff development an acute problem. Often, where possible, students will select their Programs of study on the basis of the identity of teachers, but this is compounded when specific strands of the subject are seen to be at the mercy of this mode of thinking. Conversely, when staff seen by the students to be 'successful' move to new areas of the subject, there is a dual issue of whether techniques are portable, and the preconceptions and experience of the audience in the next situation.

Thus there exists an acute problem for teaching management. To balance mismatched skills in domains that stretch from the highly technical through to the imprecise, in which a range of techniques, some inbred, are evident, is a significant challenge. Change management is well known to be difficult and painful and altering the practice of academic staff may be seen to lie at the extreme of this. 'Staff development' is not a new concept (Boyer, 1990), and the idea of a profile of skills in faculty is well established. It remains a problem to find a suitable mechanism to match supply to demand with respect to the range of constraints we have outlined.

This represents a systematic threat to the teaching of the discipline, but the optimistic view is that it also an opportunity. Success in delivery should lead the way to improved teaching in other areas as well that do not experience our particular combination of challenges; this is predicated firstly on recognition that a range of techniques is on offer and that not all of them work all of the time, and secondly on communicating this fact to the teaching staff. While students are implicitly aware of the needs for and virtues of variety, it assists to guide their expectation to what is actually on offer. We see the following practical measures that may be taken:

- Student expectation is key, and is set very early - often in the first week of University experience On arrival, they often don't actually know what a 'lecture' is, so use opportunities to provoke questions and create variety as early as possible.
- Provide mechanisms in which staff can talk to students. Tutorials are one obvious way, but small-group seminars offer another suitable from of opportunity. Personalities can be an issue if students find some staff 'unapproachable'.
- Spread the stronger teaching staff over the whole Program so that expectation is maintained. The addresses the question of whether strong staff are allocated to teach big classes (usually introductory) or hard material (final year) - the answer should be 'both'.
- Experience suggests that altering staff behavior is difficult - 'telling' people things does not work but demonstration often does. Thus it is necessary to lead from the front, but equally necessary to penetrate the broad community beyond 'innovators' (Rogers, 1995) that will influence the behavior of the broader population.

- Use teaching teams. This allows grouping of staff across the curriculum, and of different talents and experience within the same module. In this way, it is possible to encompass a wider range delivery styles, which in turn will benefit students with different learning styles.

Conclusions

We note that the evergreen problem of teaching delivery is at an extreme in both IT and CS because of the nature of the curriculum and the students. This difficulty is not eased by the highly disparate nature of faculty to be found in many CS departments.

The fact that CS and IT are similar, overlapping, and often taught by the same people necessarily implies that they are perceived by many as the same subject. Indeed some of the problems and issues that arise within CS are directly applicable and relevant to IT educators. We have presented the problems identified within CS and some of the solutions and approaches to coping in the hope that their presentation here will be of relevance to IT educators who are currently going through similar experiences as they come to terms with the emergence of IT's dual status as both a discipline in its own right and a 'service' subject for many other disciplines.

A wide range of delivery techniques exists; many of these are established good practice, while others are the result of recent enthusiasms for innovative practice within HE. It is mistaken to believe that these can be used without due regard to the material and audience, and there is therefore a need to allocate suitable solutions to problems. Orthogonally, some techniques make demands on the experience or personality of teaching staff, meaning that the repertoire of individuals may be restricted, or that faculty must engage in suitable staff development Programs.

A primary influencing factor is the prior experience of students, which moulds their expectations; these can be hard if not impossible to generalize. Faculty expectations too can be an important factor; these may be inaccurate expectations of student ability or behavior, or may be colored by experience in significantly different areas of the curriculum, which do not transfer.

Departments need to be clear that a diverse curriculum requires a range of approaches, and must seek ways to allocate them as best as is possible; this is likely to imply changes in the behavior of staff, which can be difficult to effect. A necessary first step is to ensure staff are aware of the (less than obvious) point that their discipline is indeed heterogeneous and that 'internal transfer' within it may be as hard as transfer across disciplines.

Traditional constructivist literature suggests that the problems encountered within different classroom cultures will necessarily be different. Experience, however, appears to show that the same problems are being encountered everywhere and that similar innovations are being successfully employed to aid student understanding (Astrachan, 1998). Through trial and error, Computing educators are adopting teaching methods that have a sound educational basis. This needs to be brought to the fore in order to promote confidence; recognize efforts; encourage continued innovation and to encourage others to become more inventive in their own teaching arenas. 'Reviewing our roles as teachers is not simply an interesting philosophical question; it is an imperative thrust upon us by new realities in the world' (Denning, 1999).

Acknowledgement

This work is derived from a paper delivered to the Improving University Teaching conference held in July 2000, Frankfurt (Carter & Boyle, 2000).

References

ACM, (2001). Retrieved June 30, 200l from the World Wide Web. ACM curricula recommendations, http://www.acm.org/education/curricula.html

Astrachan, O., (1998). Concrete Teaching: hooks and props as instructional technology, *Proceedings of 3rd annual ITiCSE conference*, Dublin.

Ben-Ari, M., (1998). Constructivism in Computer Science Education, *Proceedings of 29th SIGCSE symposium*, Atlanta.

Boyer, E., (1990). *Scholarship revisited*, Carnegie.

Boylan, M., (2000). Numeracy, numeracy, numeracy and ideology, ideology, ideology *in Proceedings 2nd Mathematics Education and Society conference*, Portugal.

Carter, J., (1999). Peer Scaffolding: what really happens when students work together*? Proceedings of 4th annual ITiCSE conference*, Krakow.

Carter, J., & Boyle, R., (2000). Teaching Delivery Issues - Lessons from Computer Science, *Proceedings of 25th Improving University Teaching Conference*, Frankfurt.

Carter, J., & Jenkins, T., (1999). Gender and Programming: what's going on*?, Proceedings of 4th annual ITiCSE conference*, Krakow.

Culwin, F., & Lancaster, T., (2000). A Review of Electronic Services for Plagiarism Detection in Student Submissions, *Proceedings of 1st annual LTSN-ICS conference*, Edinburgh.

Davy, J., & Jenkins, T., (1999). Research-led Innovation in Teaching and Learning*, Proceedings of 4th annual ITiCSE conference*, Krakow.

Denning, P., (1999). Teaching as a Social Process, *Educom Review*, Vol. **34**, No 3.

Dick, M., (2001). Is it OK to cheat? - the views of postgraduate students*, Proceedings of 6th annual ITiCSE conference*, Canterbury, 2001.

Donaldson, M., (1978). *Children's Minds*, Fontana Press, London.

Foucault, M., quoted in Dreyfus, Rabinow, & Dowling, (1991). *Gender, Class and Subjectivity*, FLM.

Gates, P., (2000). Retrieved June 30, 200l from the World Wide Web. *A study of the structure of the Professional orientation of two teachers of mathematics: a sociological approach, Volume 1 – Theoretical and Conceptual Frameworks*, http://www.nottingham.ac.uk/~tezpga/thesis/

Hagan, D., & Sheard, J., (1998). The Value of Discussion Classes for Teaching Introductory Programming, *Proceedings of 3rd annual ITiCSE conference*, Dublin.

Haller, S., & Fossum, T., (1998). Retaining Women in CS with Accessible Role Models*, Proceedings of 29th SIGCSE symposium*, Atlanta.

Klein, M., (1998). How Teacher Subjectivity In Teaching Mathematics-As-Usual Disenfranchises Students, *Proceedings of 1st Mathematics Education and Society conference*, Nottingham.

Kneale, P., (1996). The rise of the strategic student, how can we adapt to cope? in Armstrong, S., & Thompson, G., (eds.), *Facing up to radical change in colleges and Universities*, SEDA/Kogan Page.

Kolb, D., (1984). *Experiential Learning: experience as the source of learning and development*, Prentice Hall.

Lewandowski, G., & Morehead, A., (1998). Computer Science Through the Eyes of Dead Monkeys: learning styles and interaction in CS1, *Proceedings of 29th SIGCSE symposium*, Atlanta.

Mantovani, G., (1996). Social context and HCI: a new framework for mental models, cooperation, and communication, *Cognitive Science*, Volume 20.

NCIHE, (1997). Retrieved July 4, 2001 from the World Wide Web. The National Committee of Investigation into Higher Education (The "Dearing Report"), http://www.leeds.ac.uk/educol/ncihe/

Piaget, J., (2000). *Studies in reflecting Abstraction,* edited and translated by Campbell, R., Psychology Press.

Plagiarism.org (2001) Retrieved June 30, 200l from the World Wide Web, http://www.plagiarism.org

QAA, (2000). Retrieved June 30, 200l from the World Wide Web. *Computing Subject benchmark*, http://www.qaa.ac.uk/crntwork/benchmark/benchmarking.htm

Roberts, E., Lilly, J., & Rollins, B., (1995). Using undergraduate teaching assistants in introductory Programming courses: an update on the Stanford experience, *Proceedings of 26th SIGCSE symposium*.

Rogers, E., (1995). *Diffusion of innovations*, New York: Free Press.

Siegel, E., (1999). Why do Fools Fall into Infinite Loops: singing to your Computer Science class*, Proceedings of 4th annual ITiCSE conference*, Krakow.

Snyder, L., (1999). *Being Fluent with Information Technology,* NRC

Thomas, R., (1998). *Long Term Human-Computer Interaction : An Exploratory Perspective*, Springer.

Townsend, G., (1999). TheTenthStrand==EthicalDebates+Solution*, Proceedings of 30th SIGCSE symposium*, New Orleans.

Vygotsky, L., (1993). Genesis of the higher mental functions, in *Learning to think*, Routledge, London, (Abridged translation of the original Russian, published in 1930).

Biographies

Janet Carter is Director of First Year Computer Science at the University of Kent at Canterbury, UK, specialising in teaching Mathematics to Computer Scientists. She has previously taught mathematics and IT to the 11-18 year old age group. Her research background is in educational theory and her current research interests are focused upon the student perspective; how students manage the transition from the relatively sheltered school environment to the university one, gender issues, the informal symmetrical working relationships formed by students.

Roger Boyle is a Senior Lecturer in Artificial Intelligence at the University of Leeds, UK, specialising in computer vision. He has had a strong interest in teaching and learning issues for many years, particularly in the areas, which mark computer science out from other disciplines. He has published widely in research topics in computer vision (notably automated tracking) and in issues related to higher education. He has been teaching manager in a large computer science school, and is currently Head of Staff Development.