Computational Science Technical Note **CSTN-011**

Teaching students how to be Computer Scientists through student projects

H. A. James and K. A. Hawick and C. J. James

2005

Student project work is a vital educational component of a Computer Science degree. We have enjoyed supervising student computing projects in six different universities around the world. We attempt to distill these experiences into a "formula" for the time-pressured academic supervisor. We discuss project themes of recent interest to computing students and various strategies for managing portfolios of student projects for the greatest benefit of both the students concerned and the supervisors. We expect many colleagues will share our views and will have had similar experiences but we note a surprising "silence" in the Computing Science Education literature.

Keywords: Final year student projects; computing culture; research training

BiBTeX reference:

```
@INPROCEEDINGS{CSTN-011,
        author = {H. A. James and K. A. Hawick and C. J. James},
        title = {Teaching students how to be Computer Scientists through student projects},
        booktitle = {Proc. Seventh Australasian Computing Education Conference (ACE2005)},
        year = {2005},
        editor = {Alison Young and Denise Tolhurst},
        series = {CSTN-011},
        pages = {259-267},
        address = {Newcastle, Australia},
        month = {30 January - 3 February},
        organization = {Conferences in Research and Practice in Information Technology},
        publisher = {Australian Computer Science Communications, Vol 42, Number 5},
        keywords = {Final year student projects; computing culture; research training}
}
```

Teaching students how to be Computer Scientists through student projects

H. A. James   K. A. Hawick   C. J. James

Computer Science
Institute of Information and Mathematical Sciences
Massey University, Albany
North Shore 102-904, Auckland, New Zealand
Email: {h.a.james,k.a.hawick}@massey.ac.nz

Abstract

Student project work is a vital educational component of a Computer Science degree. We have enjoyed supervising student computing projects in six different universities around the world. We attempt to distill these experiences into a "formula" for the time-pressured academic supervisor. We discuss project themes of recent interest to computing students and various strategies for managing portfolios of student projects for the greatest benefit of both the students concerned and the supervisors. We expect many colleagues will share our views and will have had similar experiences but we note a surprising "silence" in the Computing Science Education literature.

Keywords: Final year student projects; computing culture; research training

1 Introduction

One of our major expected outcomes of a degree in Computer Science is the ability to write computer programmes to test ideas, or hypotheses. Some would argue the ultimate test of the Computer Science education is the final-year project – where students must put their collected knowledge into effect.

When we counted the number of student projects we have supervised, we realised that there are over forty individual projects in Scotland, Wales, the US, Australia and New Zealand (Hawick and James 2004). Depending on the education system, student projects have either been carried out in the final year of a four-year degree-with-Honours (Scottish and US systems), or three-year degree-with-Honours (English-Welsh system), or as part of an extra year Honours degree (Australian and New Zealand systems). Student projects from each of these years are meant to be of comparable quality and scope. In this paper we restrict our discussion of student projects to final-year and Honours level projects.

While there exists a wealth of literature on general Computer Science curricula (ACM/IEEE-CS 2001, ACM/AIS/IEEE-CS 2004, Shackelford et al. 2004, Hawick and James 2003) and more general educational (Bransford et al. 2000) and pedagogical research (ACM SIGCSE 2004, Simpson et al. 2003),

there does not seem to be much written about Computer Science final year student projects from an academic point of view. Guides such as (Dawson 2000) have been written for students in England and (Philips and Pugh 2000) describes earning a PhD from a US view point.

In this paper we relate our experiences with creating and supervising student projects in a variety of educational institutions. We describe the motivations and expected outcomes of student projects and the physical requirements that must be available in section 2. Section 3 describes the less tangible aspects of a department and degree structure that could affect the chances of a project's success. We discuss our use of project themes and threads in section 5 which leads on to our informal project classification scheme described in section 6. We summarise what we consider makes a good project in section 7. Finally we discuss other topical issues in section 8 and conclude the paper in section 9.

2 Project Motivations & Outcomes

In this section we describe and discuss the fundamentals of student projects: the purpose and motivations behind student projects. We also discuss some of the more general student requirements for computing projects.

We believe that motivations and expected outcomes do and should vary somewhat between degree programmes in the UK, USA, Australia and New Zealand. In particular, in the UK system it is more common for undergraduates to undertake project work that is not expected to necessarily lead them to a research-oriented career. In Australasia, it is our impression that individual undergraduate project work is far less common; such project work is most often found only in an Honours year. Our belief is that project work is vital component of *any* computing-based education both undergraduate and Honours.

What is the difference between a project report and a thesis? Many undergraduates do not seem to understand that a thesis is meant to make a conjecture (a hypothesis) and seek to either prove or disprove the hypothesis. At the worst an answer stating that the hypothesis could neither be proved nor disproved might be returned.

We see the aim of a computing project is to answer some question while at the same time having the side-effect of teaching the student some extra skills or theories that they may have not been exposed to during their undergraduate education. The project

is meant to be of greater complexity and scope than an assignment distributed during the continual assessment of an undergraduate paper. The student is expected to put some effort into understanding the problem, reading the relevant literature to prepare a plan of attack, to proceed through the work methodically, taking notes and making observations at each stage, before making any conclusions. It may eventuate that the question the student ends up answering is not precisely the same question as they originally posed, in which case they will either have to revisit their original hypothesis or their experimental apparatus.

As such, it is important to consider the motivation for all parties involved in a student project. Some of the points that follow may seem obvious to some of us but we feel the issue needs emphasising that both the student and the supervisor need to take a critical look at why student projects are such an important part of any curriculum.

A relevant and topical project can prove an enormous motivator for a student. Often the project will finally make it clear why all the obscure issues covered in compulsory courses/lectures are actually quite useful after all. Project work often clarifies for a student what they are interested in and what sort of work they would like to pursue. Even at a level separate from computing *per se*, a project can help a student decide whether they enjoy and/or are good at reading; analysing; compiling bibliographies and reviews; formulating methodologies; programming; running simulations; writing about experiments; reporting on findings; plotting graphs and numerically analysing data; and all the other wonderful aspects of carrying out scientific research and development that a good project can entail. We believe that exposure to research endeavours is of enormous value to students, whether they will become researchers themselves or not.

In some disciplines it is possible for a project student to work alongside if not actually part of a research group. The project acts as a common vehicle for discussion. An appropriate project in a research context will help a student to appreciate what research actually is and what it involves, and how its practitioners actually work and behave. This can be a source of inspiration to students and may help them decide on important career and indeed life choices.

We have found from experience and discussion with past students that a project is a valuable discussion point for students at job interviews. Interviewers are often disinterested in bland details of student transcripts but discussion of a relevant project can be a useful lead into understanding a student's prospective interests and match with a company and its operations. Indeed to one of the authors the attraction of their honours project was so they had something novel and interesting to talk to prospective employers about at job interviews. For the other author it was seen mainly as a proving ground to see whether they enjoyed the somewhat boundless landscape of research. As academics we must remember that students have different reasons for choosing and committing to their projects – not all of which may be academically related.

Many students in our experience have used their project experience as a spring-board for further post graduate work or decisions related to it. In this respect the project is often a turning point in an undergraduate student's life.

Project supervision from the supervisor's viewpoint has many advantageous features too. Projects can be seen from many perspectives, including:

- a mechanism for seeding activity on a supervisor's pet project;

- a way of assessing a student's abilities and potential, also their interest in joining a research group or for progressing to postgraduate work;

- a way of identifying what students find difficult or indeed interesting about a particular sub topic or discipline. We have found the insights gleaned from supervising a large sample of project students helpful in identifying weakness, gaps and inconsistencies in our whole computer science programme. This information can be fed back into teaching programmes in useful ways.

These aspects are important and go considerably beyond what are sadly sometimes the "official" reasons for supervising projects - "because the accrediting body said you had to" (British Computer Society 2004) and "because everyone has to supervise their quota of project students".

We have also found project supervision an important way of tutoring students and helping provide a personalised front-end to what are increasingly becoming degree "sausage factories". This aspect can be time consuming and it is unfortunate that it is not always recognised by university management structures as a critical factor for success in genuine higher education.

In short, projects can be an incredibly interesting and worthwhile experience for both student and supervisor. This is true in general. There are some specifics that relate to computing projects.

We have found that project work is very useful in helping students understand the different aspects of computing and in particular to help them appreciate the differences between "IT", Computer Science, Computer Engineering and Information Systems. Sadly, some students arrive in first year with the impression that Computer Science involves just being able to drive a word-processor! The European Computer Driving License certification process (European Computer Driving License Foundation 1997) run by the British Computer Society (BCS) is an excellent IT qualification but our impression has been that it and similar efforts have perpetuated this mis-belief about Computer Science amongst the lay-public. Furthermore, it is an effect of broader participation in higher education, amongst other societal factors no doubt, that many students coming to university to do a Computer Science degree do not actually appreciate the difference between a "degree" and a "vendor certification course". Project work and close interaction with a supervisor in a mentoring role can help this problem.

Departments are frequently being driven to modular approaches to teaching computing subjects. One problem we observe due to this approach is that students in a given class can have very diverse backgrounds. It is often difficult to customise teaching to any one cadre of students in a class and therefore inevitably a subject is in danger of being dumbed-down to the lowest common denominator. There are strategies for tackling this problem, but the mentoring and guidance of a tutor/project supervisor can go a long way to guiding the smart students who genuinely want and need a higher education experience to appropriate reading and study.

In times of resource pressure tutoring is not feasible but project supervision support may be. In which case we argue that it can and should be used to the best effect possible to enhance the higher educational experience.

An approach we have regularly taken in the past few years is to employ dual supervisors for our students. In typical supervisory arrangements, the

principal supervisor is responsible for the student's progress and the 'secondary' supervisor only plays the part of the thesis second marker. We find this most beneficial especially when dealing with cross-disciplinary projects. Having two supervisors allows us to use staff from different disciplines who are able to have an equal say in the direction and conduct of the student; we have usually found that at the end of the project both supervisors feel the cross-disciplinary project was worth the time and effort. Due to both supervisors being responsible, no one supervisor feels undervalued. It is, of course, an open question whether a supervisor should, in fact, be the principal 'marker' of the project. Staff resourcing issues often dictate a somewhat sub-optimal approach to such matters.

We also find that having dual supervisors is beneficial even if they are both from the same discipline. We are to virtually guarantee continual supervisory availability for the students. This is helpful when staff go abroad on conference leave, sabbatical or vacation. Many students feel reassured in the knowledge that at least one of their supervisors will be available nearly all the time. This is also a useful way of passing on "supervision lore" to less experienced colleagues.

The question "Who designs the project?" must be addressed quite early in the project cycle. Common wisdom has been for potential academic supervisors to consider what projects they would like to offer during the next teaching session and publish some sort of synopsis for students to peruse.

This technique has the benefit of allowing other academic members of the same department to survey other projects on offer and to maintain some sort of informal quality control over project offerings. Some departments have formal meetings to discuss the possible student project offerings, while other departments rely on a more informal feedback mechanism. This also ensures the department as a whole protects the University's brand in terms of quality of student learning and projects. Our experience of the external examiner system in the UK, both as the reviewer and the reviewed, was that external scrutiny and a more objective perspective on student project work was very beneficial. We believe that, resources permitting, the Australasian system would benefit greatly from impartial external reviewer mechanisms.

Having student projects initially specified by academic staff also allows them or their groups to achieve some sort of continuity in student projects to allow, for example, further work to build upon past student work. This is especially useful when a group or academic has a strategic vision with definite milestones along the way to achieving it.

However, it has been our experience that in the last few years students seem to be turning away from the more traditional academic forms of student project to what is perceived at the moment to be "cool" and "exciting", be this graphics or peer-to-peer file sharing (see section 5). Thus, when consulted, students are more likely to suggest the topics that excite them and we see it as up to the academic staff member to negotiate a project with the student that is motivating to them but also of enough academic merit to justify inclusion in an Honours programme. It is a matter for the staff in the Computer Science department to collectively agree what levels of academic merit are appropriate in their environment.

Unsurprisingly, we have found that the more a student feels "ownership" of their project, the more effort they are likely to put into the work and it will quite possibly be of higher quality.

Commonly accepted wisdom states that unless you are very familiar with the abilities of the student doing your project, it is better to structure the project design in terms of achievable goals. Ostensibly this is done to give the student shorter-term milestones against which they can measure their progress. Cynically some academics might argue that having multi-stage projects makes it easier for the weaker students to achieve the minimum work, thus equating to a minimum grade and acting as a convenient exit point for the project.

It is important for both the academic supervisor and student to agree upon Pass, Credit and Distinction levels for the project work. Typically, a department or University will have available a body of past student theses. This historical archive is generally instrumental in setting a benchmark for quality levels in the institution. Our experience in a department where no such historical archive of Computer Science theses existed was that both supervised students and our staff colleagues experienced difficulties in appreciating acceptable levels.

Expected project outcomes and revised outcomes are sometimes quite different. Sometimes what you get out of the project isn't what you (or the student) specified. Is this because the project was too difficult? Too easy? Did it lack a clear purpose or specification? Did the student simply change their focus part-way through the project or did they get bored?

In our experience, many successful projects are quite open-ended. As such, they could be extended by an interested and able student to make out of the project whatever they are willing to put into it. And we find it is better if the project is such that it would suit the student were they interested in proceeding on with a higher degree. Or perhaps another student may be able to pick up the work that the student has completed thus far, to extend.

There is an important, but mostly unanswered question of who owns the work completed by the project student? In this paper we do not consider the issue of commercialisation of projects, but simply the right of the staff member to retain a copy of any code/design produced by the student. What if the work was completed by a student on the specification of an academic? What if the academic and the student designed the project together? This is a very grey area.

Most academics would argue for the right to expect a copy of the student's code or design with the thesis - even if only to ensure they did not copy or plagiarise the work. Most students are only too happy to oblige; many have taken to supplying a CD-ROM with all resources as an appendix to their thesis.

In general, Computer Science students have quite different requirements when it comes to computing services than many different academic disciples. It is a difference that is only now being recognised by some universities with centralised information technology service (ITS) divisions. Computer Science students require the ability to actually programme the computers they are working on, which is quite different from the more traditional ITS models, which simply provide a stable platform upon which the majority of staff and students can do their day-to-day work and report-writing. Some universities, such as Massey University, maintain a standard laboratory image across all campuses and departments. While an excellent resource for non-Computer Science students, we find this model overly restrictive in our day-to-day work and the institutional inertia in making modifications to this image is quite significant.

We conclude that the only realistic solution to the provision of specialised computing services is to establish and maintain separate laboratories for the Computer Science departments. While supported, in principle from ITS, the idea seems to gain little favour from the university administration in the current

trend of centralising all computing resources. Staffing and equipment procurement is obviously a vexing issue. Computer Science students require effectively a *sand-boxed* environment, which allows them the required flexibility to experiment with programmes and code which, if incorrect, could conceivably crash a number of computers.

At a more advanced level, Computer Science project students require an even more flexible environment, in which they can write or inspect device drivers and implement different protocols for communication. They need the ability to change the settings on their machines without being hampered by a "locked-down" centralised laboratory image.

3 Educational Environmental Factors

In this section we discuss the non-academic factors that we feel greatly influence the success of student projects. Possibly one of the most over-looked influencing factors is the existence of an existing staff/student culture.

In order to foster good project students then it is desirable they have good role models. This can be in the form of a pre-existing research group into which the student's work fits, or at the very least research students in the same department that the student can model their work (culture, ethics, etc.) on.

Furthermore we believe the existence of a "programming culture", where students are encouraged to write programs that solve problems and to experiment with different systems, is vital for a thriving Computer Science department. It is very important for undergraduates to be encouraged to practice programming and to follow their interests outside of the classroom. The authors remember being taken under the wings of more experienced undergraduates and even friendly postgraduates to gain considerable extra experience in programming and systems understanding.

We feel the idea of a culture is especially important when it comes to the contentious issue of which operating systems and programming languages – and even document preparation system – should be used for both teaching undergraduates and student projects. While we would like to be unbiased, we do believe that students should be exposed to as many different operating systems, programming languages and the like, as possible. Care must be taken not to sacrifice depth for breadth, but university is, after all, meant to be about life-long education rather than vocational training.

There is also the matter of the culture that exists within the department. Are the academic staff good programmers (we are constantly reminded that ability to teach programming doesn't necessarily mean the teacher can program, or program well). This is particularly important due to the current phenomenon of departments changing their teaching languages. Many staff are quite comfortable with the programming languages they are used to teaching; and while not showing any outwards resistance to change, may take some time to fully embrace the new teaching language – causing a negative reaction in student performance.

4 International Perspectives

Computer Science is fortunate in having active international bodies (ACM, BCS, IEEE-CS amongst others) (Association for Computing Machinery 2004, British Computer Society 2004, Institute of Electrical and Electronics Engineers Computer Society 2004) and indeed an international community that helps define just what computer science is. This is less so for some of the emerging computing disciplines. The recent Curriculum 2004 draft document acknowledges Information Systems; Computer Engineering; Software Engineering; and most recently Information Technology as computing disciplines in their own right. It has been our experience that outside the discipline there is still considerable misunderstanding of what Computer Science actually is. Sometimes this is genuine, sometimes wilful misunderstanding - sometimes an organisational entity in a university has to play several roles and as the Strawman document (ACM/AIS/IEEE-CS 2004) observes, this marketing labelling game is particularly prevalent in the UK. The consequence is that a given department may be handling students from many different backgrounds and levels of mathematical and theoretical maturities. This makes it particularly important to formulate projects that can be carried out at different levels of student ability.

Our experiences in teaching Computer Science in the English speaking world [1] convince us that as far as project work is concerned the most important difference is whether an undergraduate degree is a four year one or only three. Educational systems such as those in Australia and New Zealand seem to draw heavily on the Scottish system of four year honours degrees. In consequence we found it much easier to handle quality "final year projects" there. The English/Welsh three-year integrated honours system places considerable strain on curriculum space – meaning the project is allocated less student time, and has less assessment worth through necessity. Also, students have surprisingly less ability and maturity in third year than they do in fourth year. We came to this conclusion after working with relatively large numbers of student projects - forty - a number we believe large enough to justify our making this sweeping claim.

The USA system is different again, but seems to mix the two highlighted issues above. Namely the project supervisor must be prepared for mixed student experiences and abilities and project work at different levels. We believe that the same basic formula can be applied to projects under all these systems, but a prospective supervisor must have thought through the staging issues we discuss in section 7 to cope with the unknown qualities of a project student.

5 Project Themes and Threads

Over several years of experiences of having to come up with quite large numbers of project topics, often at short notice, and not always knowing what student would be allocated to a particular project in advance, we have found deliberately grouping projects into themes useful. Theming projects (or grouping them) so that, for example, several students can work on different aspects of a broader area at once is useful. We have also found that good areas remain popular with successive student intakes across subsequent years so that time-lines or threads of development are also useful. Students can be motivated by having them build upon a prior student's work - and it teaches them a vital aspect of real research and the scientific method. Students often see a project as an artificial assessment instrument. Showing them that they are expected to develop and not just reproduce a prior project is an important learning experience and can be a great motivator.

Students often do not have opportunities for team work in packed curricula and with all the resourcing

---

[1]Including the USA

implications. A portfolio of themed projects with a common supervisor (or supervisors) can be a feasible substitute for providing this experience. Students working on related projects can be encouraged to collaborate and cooperate and with appropriate supervisors' advice and guidelines can make this a very positive experience. In this way we avoid the effects of students' reluctance to group work as described in (Waite et al. 2004).

Some specific thematic areas and time-threads we have employed are:

- Distributed Middleware - this proved popular in the pre-grid era and attracted honours level and postgraduate students to work on sub-topics including: scheduling; namespace management; server-server (aka P2P) protocols; wide-area data management; secure protocols.

- Lego Robotics - this area continues to be popular in many departments world-wide for engineers and scientists alike. We had many projects including sub-topics such as control algorithms; software architecture design; artificial intelligence algorithm experimentation for vision, path-following and sensor management amongst others. This is still a rich area and continues to appeal to students.

- Random numbers - this is an area that continues to fascinate students and is a rich picking ground for projects in simulation and algorithm development. Recent work involves performance optimisation and software engineering for 64-bit random number generators.

- Graph and Network Visualisation - recent new capabilities in portable packages such as Java3D continue to inspire students to work in visualisation. Our particular interest has been in graph and network visualisation which helps reinforce and explore data structures and algorithms ideas that are sometimes otherwise seen as dull by students.

- Discrete Event Simulation - originally inspired by object serialisation and a persistence "spin" we have had students work on various algorithmic developments in this area.

- Security and Biometrics - inspired by recent publicity including James Bond movies and other media, we have had systems integration and algorithm exploration projects in various biometric and secure protocol issues.

- Artificial Life and Simulation - this is a new area for us, but is exciting students and provides a way of expounding and motivating ideas in complexity and computational science.

Students are often motivated by a particular project or project area because of some specific keyword or idea or practical skill that is involved and which they have latched onto for cultural or job prospect reasons. No doubt many of us are familiar with the "Java" craze of recent years where students wanted to have a reason/opportunity to learn about Java generally or just about specific Java technologies (Sun Microsystems Inc. 2004). Recent sub-crazes we have experienced have been: Java/Swing/GUI; Java3D; Java/XML; and Java/P2P/JXTA.

Other fads in student interest and hence project theme popularity include: sound files and encoding and MP3 format related technologies; Internet and network management; OS and Linux-developer related topics; grid development topics.

Fads and fashions are no bad thing if properly considered. We discuss what we believe makes a "good" project elsewhere in section 7, but one noteworthy danger of a fad-based project is that it is too closely tied to a product or technology and does not really provide a research-lead learning experience. We note with some despair the existence of "evaluate some trendy software package my supervisor had not time to learn" projects that some time-pressured colleagues have offered to students. We believe projects along these lines condemn the student to mediocrity and should be avoided.

We do believe that almost any technical area can be used as basis for a project providing appropriate goals are set.

6 Our Project Classification Scheme

Through our experiences, we have broadly classified the styles of student projects into four main classifications: building/using software, building hardware, theoretically-based projects, and re-writing projects. These are shown in table 1 and are described in the subsequent paragraphs. Of course, not every student project fits neatly into only one category, but in our experience the majority of student projects fall mostly within one category.

Project type	General Characteristics
build demo software / use software package	student spends bulk of time building software or perfecting use of a package or toolkit tends not to address a hypothesis
build hardware	student spends bulk of time building hardware from components to investigate some phenomenon often scientific question gets lost along the way
theoretical study	student builds software/uses package to investigate a hypothesis hypothesis gets addressed properly
re-writing	reading, summarising and re-writing existing literature student focuses on background, less on completing project

Table 1: A simple taxonomy of types of student project with their broad characteristics and some comments drawn through observations

In our opinion, one of the factors contributing to the wide mix in project classifications is due to their differing abilities and interests which is brought about by their academic backgrounds. Figure 1 shows the broad mix of student disciplines of two different universities we have worked at (in different countries). The left-hand diagram shows a mix of Computer Science and Business, Mathematics and Psychology variants to the degree. Students from both Computer Science and its variants had to do a Computer Science-based project. In this example the Computer Science degree structure is the lynch-pin off which the variants sit. The right-hand diagram shows the more traditional case in which students from Computer Science, Information Systems and Information/Computer Systems Engineering backgrounds have to select an appropriate project. In this example each of the degree structures is equal in contribution to not only undergraduate theoretical papers but also student project offerings.
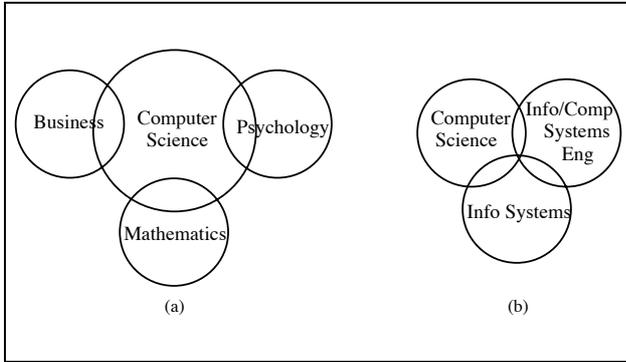
Figure 1: Intersection between Computer Science and related disciplines in two different universities from a project-supervision perspective. Depending on the political make-up of the educational institution, student projects must be tailored to fit whatever particular background/interests the student has. (a) The case in which a single Computer Science department offers Business, Mathematics and Psychology variants of its degree (and hence projects) to undergraduates. (b) The more traditional case in which students are more constrained in their selection of final-year project. Projects on offer do not have to be tailored to the particular degree background of students.

Software building projects tend to involve the student investigating the usefulness of a software package or system for some purpose. In our experience many of the students whose projects fit into this category do not end up answering any scientific questions, but instead are able to produce excellent 'tutorials' or introductory sections in their thesis, at the expense of results. Some of the more recent projects we have supervised that fall into this category are: P2P architectures in resource sharing; robust network management and cluster computing; algorithms for audio encoding and algorithms and rendering for scene visualisation. We do not necessarily condone the students producing this type of project but sometimes despite how well-posed the project specification is, the student is able to move in the direction they find most comfortable.

We classify software building projects as distinct from theoretical projects, as we have found a growing number of students who are not really interested in investigating a particular problem or phenomenon, but are simply interested in getting to know a particular software system well (perhaps in the effort to make themselves more attractive to employers). Our more theoretical projects all feature a (narrow) scientific question that is to be answered which requires some software to be built. Projects we have supervised in this category include: particle engines; discrete event simulations; distributed algorithms for feature analysis; graphical learning tools for petri-nets and designing and building neural networks for pattern and feature recognition.

Hardware building projects involve the construction of some physical piece of equipment that can be used to experiment with some ideas. Due to the expensive nature of most physical components most of our hardware-based projects have focused on construction apparatus out of Lego (Lego 2004a). The simplistic nature of Lego components allows some quite complex apparatus to be constructed. We have had success using Lego bricks as a basis in projects such as: image capture and feature recognition with Lego robots; and motion tracking through stereo vision. Other simple componentry includes kit-based robots such as the Real Robots (Ultimate

Real Robots 2004). Hardware-oriented students have had considerable success in replacing the hard-wired robot brain with slightly-more-sophisticated Lego bricks (Lego 2004b). We recognise that our Computer Engineering colleagues are also fond of hardware simulation projects, where a more complex hardware system than a student could practically or economically build, can be investigated as a simulation using appropriate simulation packages.

In the final category of student projects, re-writing projects, we describe those projects in which the student has not done any particular software programming work or construction of a model to solve the problem posed in the project. However we find that the students who produce these types of projects tend to do an excellent job of understanding the literature and taxonomising the different aspects of the work. We find that many of the theses produced at the end of these projects would be worthy of a literature review section in a PhD thesis. Not surprisingly, we find, in general, that the students less confident in their software or hardware creation skills take up these type of projects and then self-fulfil the prophecy by concentrating far more on the literature review than the programmatical aspects of the project. Some recent examples of this type of project we have supervised include: investigations into a computational grid economy; integrating an artificial intelligence reasoning engine with Java; designing the household grid; and a discussion of the Internet and its influence on society.

Of course, there are outliers to this system of categorisation. We have also supervised projects which could only be described as philosophical in nature. Such projects, not featuring any software or hardware componentry, contain the best aspects of our theoretical classification with an extended discussion on the project specific topics. One such project, directions in AI, might be held up as a paragon of this type.

In a sense, the broad classifications of student project might even be used to classify the students in their interests and predilections. We find that the student project experience gives students a great insight as to the types of jobs they might wish to pursue – or avoid.

7 What Makes A "Good" Project

Many of us will no doubt have our own prejudices about what makes a "good" student project. We believe that a useful guiding principle is as follows. A good project should have at least three separate phases or stages or partitions in the supervisor's mind when the project is designed or formulated. Firstly there should be some known and quantifiable task or part of the work that an average student can work on and be expected to complete. Secondly there might be the expected stage or partition of the project that a "good" student ought to be able to tackle, make sense of, and do a "good" job of. Finally there should be some aspect of the project that a "star" student should be able to "shine" in. This third stage is generally open ended and may not be anticipated in its entirety by the supervisor/project designer.

We believe it is vital that a supervisor go through this mental checklist before giving out a project. We regret having seen good students condemned to mediocrity by a project that is missing the second and/or third stages. Unfortunately a less-able student can become completely lost by being given a project with only the second and third parts.

Our experience has been that this formula works well and that the student can be told explicitly what is going on. Often a student is relieved to be told

what is and is *not* expected of them. We have not found it has discouraged good students from tackling something they know is beyond what is expected of them. We have also found that "mediocre" students – and let us be blunt and admit there are some – can be inspired to greater effort and achievement by being given clear guidelines on part one. The lowered stress and feeling of growing confidence once they have achieved something specific from a well laid out part one task can lead them onto surprising accomplishments on the more open ended parts of the project. We therefore advocate transparency with the students about how the assessment system works and what is expected of them.

We have experienced the flurry of activity in departments that surrounds quality assurance (Quality Assurance Agency 2004) inspections. Although it may have its own merits we do not believe that a wad of documentation about the tenth decimal place of assessment percentage that is to be allocated to each aspect of the project thesis necessarily helps improve the student's project experience. Instead we believe that the supervisor having some empathy with students of low, medium and high abilities and making appropriate provision in project specifications does.

8 Discussion

One quite contentious issue is the length of the student project (and the supervisor's expectation of input effort) in comparison to the comparative worth of the project in terms of percentage to their final degree classification. This is quite a major issue in the degree-with-Honours courses where all students complete a project – which could contribute as little as one sixth to their final degree classification. We feel this may have a bearing on some students' lack of effort spent on their project. At some Australian universities, with extra honours years, the project is worth 40% of their final degree classification; it is little wonder we have seen far more effort go into these projects.

Also if the students are to prepare public talks and/or posters on their research, what is the allocation of marks and percentages for each? Is it really necessary to have marks allocated for the so-called "soft skills" such as presentation and poster-making abilities? We certainly understand the need for graduates to be able to talk intelligently about their research in both the formal setting of a presentation and the less formal setting of a conference poster session. We also sympathise with the view that some students are averse to spending any significant amount of time on an activity that is not worth a meaningful percentage of their final grade.

We also stress the importance of having an adequate administrative system to manage and provide support for student projects. For example, should the accreditation requirements for the degree include the necessity to create a poster on the research, we stress the need to have the necessary software and a competent staff member who can help the students perform this 'off-mission' task.

We have found poster sessions to be quite a useful indication of a student's achievements. Through a casual conversation with a student about their poster (and work) an academic staff member is able to get a good indication of the student's understanding of their project. However, we also caution our fellow academics against using any 'poster judging competition' to give indications as to the final merit of the project work when there are many cross-disciplinary projects being displayed. We have seen quite a few posters that are quite well-produced but hide the fact that the student has done very little work. And sometimes our over-worked colleagues have awarded these projects the 'best poster' awards!

It is not always easy to have an honours course – or an honours project – that is being shared between students belonging to different departments, with different marking requirements. One example, from Britain, was where the School was teaching both an Engineering degree and a Science degree. Students could choose to do a project either with engineering staff or Computer Science/Maths staff. The Engineering course accreditation requirements stipulated that the marking of projects must include an assessment of the student's ability to create a poster and also make a presentation of their work (Institute of Electrical Engineers 2004). We would argue that while these soft skills are valuable, in order to motivate students to spend any time in their preparation marks must be allocated towards the activities, thus diluting the contribution of their thesis, as discussed above. We also caution supervisors of the potential conflicts between satisfying a cross-disciplinary degree's accreditation requirements, a student from that degree who wants to do a science project, and the supervisor who would like to supervise a science project.

One avenue we have trialled to improve the quality of our honours students is through the use of Summer Schools. We have collectively run summer schools in the US, Australia and also Britain. We have run the summer schools with a small number of interested and gifted third-, second- and even sometimes first-year students over the summer holidays. Undergraduate students worked on project of our own devising and were mentored by us and our small team of postgraduate students for eight weeks. At our last summer school in Britain we provided a small monetary incentive to attract the best students (who would have otherwise gone to well-paying summer contracts); at the end of the summer school we held a poster session open to interested public and also a conference-style retreat. Those project students who participated in the summer schools performed significantly better, on average, than the non-participants we supervised.

As rewarding as the summer schools are, we recognise that there is a significant difference in the types of projects achievable during a summer school and year-long projects. There is also a vast difference in the quantity and quality of contact an academic supervisor will have with their students on a summer school. We recommend that in order to keep one's sanity a summer school not be run every year! We also stress the vital importance of having excellent administrative support in any summer school venture.

We firmly believe that projects are an important way of passing on our technical professional and cultural values. It is important we academics do this in a way that does not prejudice our students against one particular technology or product, but opens their minds to consider using the best tool for the job.

9 Conclusions

So we come to the unsurprising conclusion that project work is a vital part of the higher educational experience in Computer Science, if only teaching students how to perform 'science' and also to pass on the Computer Science culture. We also conclude that it can be immensely rewarding for student and supervisor. Again, unsurprisingly the more effort goes in from both student and supervisor the more both parties with get out of it. Our experiences from different universities in different countries suggest that the quality of projects in four-year honours degrees

is converging rapidly, while three-year honours programmes are, in general, perhaps a little too short.

The typical project supervisor in Australia and New Zealand often fulfils the role of the traditional personal tutor (including responsibility for students' pastoral care) found in the British higher educational system. We have found that project supervision actually presents a better forum for providing support than the traditional personal tutor due to the shared nature of the project work leading to a shared element of trust and confidence, which is much better than an arbitrary assignment of tutees to a staff member for the duration of their degree. Pastoral care is an important part of project supervision. It's not just an academic relationship – this relationship is often quite enduring. We have found that often we form better relationships with our project students during the one academic year of their project than perhaps three years of ordinary personal tutoring.

We expect it will be continue to be a challenge to come up with resourcing strategies for Computer Science project supervision, but we believe there is scope for justifying commitment of staff effort on grounds of: teaching experience quality improvement; research-driven curricula; and simply motivation for staff involvement in higher educational teaching.

10   Acknowledgements

References

Association for Computing Machinery (ACM)(2004), *ACM Homepage* Available at http://www.acm.org last visited 26 August, 2004.

Association for Computing Machinery (ACM)(2004), *ACM Special Interest Group on Computer Science Education*, Available from http://www.sigcse.org/topics/ last visited 26 August, 2004.

ACM/IEEE-CS Joint Curriculum Task Force. (2001), *Computing Curricula 2001*. IEEE Computer Society.

ACM/AIS/IEEE-CS Joint Curriculum Task Force. (2004), *Computing Curricula 2004, Strawman Draft*. IEEE Computer Society.

Bransford, J. D. & Brown, A. L. & Cocking, R. R. & Donovan, M. S. & Pellegrino, J. W. (2000), *How people learn brain, mind, experience, and school expanded edition*, National Academy Press.

British Computer Society (BCS) (2004), *BCS Homepage*, Available from http://www.bcs.org/ last visited 26 August, 2004.

British Computer Society (BCS) (2004), *British Computer Society Higher Education Accreditation*, Available from http://www.bcs.org/BCS-Products/HEAccreditation/ last visited 26 August, 2004.

Dawson, C. W. (2000), *The essence of computing projects a student's guide*, Pearson Education Limited.

European Computer Driving License Foundation (1997), *European Computer Driving License*, Available from http://www.ecdl.com/main/-index.php last visited 26 August, 2004.

Hawick, K. A. &James, H. A. (2003), Bootstrapping Computer Science in Old North Wales *in* Proc. Fifth Australasian Computing Conference (ACE2003), Conferences in Research and Practice in Information Technology, 20. Greening, T. and Lister, R., Eds., Australasian Computer Society (ACS), Adelaide, 2003, pp. 25-34.

Hawick, K. A. &James, H. A. (2004), *Student Project Webpages* Available at http://www.-massey.ac.nz/~kahawick/student-projects.html and http://www.massey.ac.nz/~hajames/-undergrad-projects.html last visited 26 August, 2004.

Institute of Electrical Engineers (2004), *Institute of Electrical Engineers Homepage* Available at http://www.iee.org/ last visited 26 August, 2004.

Institute of Electrical and Electronics Engineers (IEEE) Computer Society (2004), *IEEE-CS Homepage* Available at http://www.computer.org last visited 26 August, 2004.

The Lego Group (2004), *Lego.com Homepage*, Available from http://www.lego.com/eng/ last visited 26 August, 2004.

The Lego Group (2004), *Lego Mindstorms*, Available from http://mindstorms.lego.com/eng/-products/ris/index.asp last visited 26 August, 2004.

Phillips, E. M. & Pugh, D. S. (2000), *How to get a PhD A handbook for students and their supervisors, 3rd ed*, Open University Press.

Quality Assurance Agency for Higher Education (2004), *Quality Assurance Agency for Higher Education Homepage*, Available at http://www.qaa.ac.uk/ last visited 26 August, 2004.

Shackelford, R. & Cassel, L. & Cross, J. & Impagliazzo, J. & Lawson, E. & LeBlanc, R. & McGettrick, A. & Sloan, R. & Topi, H. (2004), *Computing Curricula 2004: The Overview Project*, in Proc. SIGCSE'04, March 3–7, 2004. pp 501.

Simpson, M. & Burmeister, J. & Boykiw, A. & Zhu, J. (2003)in Greening, T. & Lister, R. eds., Proc. Computing Education 2003 Fifth Australasian Computing Education Conference, Vol. 20, Australian Computer Science Communications **20**(5), pp. 41–51.

Sun Microsystems Inc. (2004), *Java Technology Products Homepage*, Available from http://java.sun.com/ last visited 26 August, 2004.

Ultimate Real Robots (2004), *Real Robots*, Available from http://www.realrobots.co.uk/ last visited 26 August, 2004.

Waite, W. M. & Jackson, M. H. & Diwan, A. & Leonardi, P. M. (2004), *Student Culture vs Group Work in Computer Science*, in Proc. SIGCSE'04, March 3–7, 2004. pp 12 – 16.