How to Manage Student Software Projects with Industrial and Commercial Clients

by

Mike Holcombe[1] & Helen Parker[2]
with contributions from Alex Bell and Stan Price

University of Sheffield,

Sheffield, UK.

1. Department of Computer Science, University of Sheffield
2. Staff Development Unit, University of Sheffield

Contents.

Chapter 1.

Introduction and context:

The current teaching of computer science and software engineering and the skills gap

1.1 The software engineering process

The software engineering process is typically an uncertain and subtle affair, which commercial engineers attempt to control in environments that are subject to continuous change and other risks of failure. It is a process that we attempt to teach software engineering students to understand, mostly by reference to text book examples and rarely with the richness of human and business purpose that industrial and commercial software systems development involves.

The purpose of this handbook is to reflect on the nature of practical software engineering for university students: which processes and techniques are we equipped to teach in the higher education curriculum and which are valuable to teach? How do we introduce industrial relevance into our teaching? Is what we have to teach our students still an arcane art rather than an engineering discipline?

Software engineering: a potted history

Nearly forty years ago, the computing industry found itself in crisis, in the absence of any generally agreed and accepted process for building and delivering large scale industrial and commercial computer systems. During the 1960s, this crisis came to be resolved with the introduction of structured methods for software development and with the introduction of the concept of a software lifecycle that consisted in a sequence of phased activities, beginning with analysis of requirements, progressing through design, code and test to delivery and maintenance of a complete system [1]. Fundamentally, these phases of activity have remained core to

the software engineering process over the last three decades, although there has been a continuing and shifting debate about the nature of the software lifecycle, whether it is fundamentally sequential, cyclical or evolutionary, and about those stages in software development that are critical to a software project's success.

Techniques for structured software development became enshrined in lengthy and complex methodologies for system development (such as SSADM in the UK, IE in the US and Merise in the European continent). Universities teaching software engineering attempted to introduce structured methodologies to their students with reference to small fragments of case study examples. This at least meant that software engineering graduates would be familiar with the terminology and diagrammatic conventions adopted by a particular methodology, but of course it did not equip them to apply that methodology in a real business context. More than twenty five years after the introduction of structured techniques into the university curriculum, we still wrangle with the same problem of giving our students a flavour of software development as it is practiced in the large in an industrial environment whilst only being able to give them a first-hand experience of small-scale, partial and fragmentary system delivery. These difficulties in delivering a computing curriculum that is industrially relevant are exacerbated with the rapidity of technological change and obsolescence within the computing and software sector.

In the academic computing community, in parallel with the introduction of structured development methods and techniques, formal methods and notations were developed that promised a more rigorous approach to software construction, with which consistent, accurate and correct systems could be built [2]. However, this was not the first time, nor the last, that the academic view of good software engineering practice was at odds with the actual practice in mainstream software development houses.

Despite ample evidence to indicate that the use of formal methods is not more costly nor more time consuming than the use of structured or object oriented methods, this has been the perception across much of industry and this view that they are not commercially viable nor necessary has prevented their take-up on a large scale. So, a formal software engineering process still

forms a part of the software engineering curriculum at many university institutions, but formal development techniques are increasingly offered as a part of specialist courses related either to theoretical aspects of computer science education or to practical development of safety critical systems.

By the mid 1980s, the introduction of 4GL programming languages opened the possibility of much more rapid software development and heralded the use of prototyping techniques for exploring the feasibility and acceptability of system design alternatives. New tools for software engineers, CASE development environments, meant that the design, re-design and perfection of features at the graphical user interface played a much greater and more time-consuming part in the software engineering process. Academics duly reflected these developments in their curricula, introducing courses in human-computer interaction and interface design and, when they could afford it, buying in sample CASE tools with which students could practice system modelling techniques, usually on fragments of system descriptions.

The last ten years have seen at least two technological developments that have completely dominated and shaped the software industry: the proliferation of networked systems, the Internet and the Web, together with the almost total acceptance of an object-oriented paradigm for software development practice. These changes have brought others in their wake: the rise in the use of Java and other programming languages and environments that specifically support internet application development; the adoption of Unified Modelling Language, [3], and its object oriented development techniques as an industry standard. These developments now form the core of current software engineering education. Most universities will teach courses in this area accompanied by simple, constrained, assignments set by the instructor which purport to illustrate some of the concepts discussed. Textbooks describing OO software engineering and UML design abound but few address the process of building a real and complete software solution for a business client. In the context of a semester course it is very difficult to go through the whole bureaucratic design process and produce something useful.

Many academic computing departments have adopted Java as the first and main programming

language for students. A number, ourselves included, have struggled to incorporate meaningful and useful parts of the UML toolkit into our analysis and design courses. Most of us recognize the importance of exposing our software engineering students to the Rapid Application Development (RAD) techniques introduced in industry in the 1990s, but the only way to do so is through practical projects. The adoption of object-oriented software development leaves us without a fully worked out and coherent development methodology to guide practical student projects.

With the advent of agile methodologies such as eXtreme Programming [4] and Feature Driven Development, FDD, [5] it is now possible to run a complete development project in a semester, building a useful application for a client and adapting to some of the clients changing requirements in a successful way. The introduction of these new software methods which allow for rapid system development without an expensive and time consuming design phase is an exciting opportunity for teaching software engineering. These notes accompany a book on eXtreme Programming [6] and have been successfully used with real projects for a number of years.

We are facing many challenges in devising a relevant and cohesive software engineering curriculum for our students, not least the challenge of placing students' project work in a satisfactory methodological framework. Can the ideas of "eXtreme Programming", with its reliance on pair programming and on development driven by test specification, work on students' projects?

Managing risks, managing resources.

All projects involve risks, risks of failing to deliver, risks of going over budget, risks of delivering something of poor quality, risks of damaging the client's business or users and so on. Since many projects involve technology and components that may be poorly understood it is always difficult to eliminate these risks. Projects need to be kept under review continually and the assessment of the important and critical risks is then on-going. Factor in the added dimension of the projects being carried out by inexperienced and potentially unreliable students seems to be another risk just not worth taking. However, if the issue of risk analysis is consid-

ered up front; and students are required to carry out risk analysis as part of their planning process; and they review this at key stages of the project there is a great deal to be learnt and a lot of benefit to be had. Despite all the ways that projects can go wrong we have never let down a client and this must show that it can be done, if one takes some care.

Communications, negotiation and agreement.

We all know that poor communications, in any aspect of life, can have disastrous consequences. It is particularly true of software projects. We have a more difficult problem in those cases where the project teams do not have a stable place to work or if the times available to carry out their projects have to compete with many other activities. It is very important to help the students plan their activities out carefully and consistently. We emphasise the importance of having formal meetings and of taking full minutes and notes of these meetings. These have to be submitted weekly to us, the managers. These minutes must include information about who was present, what was discussed, what tasks were agreed, who was to carry them out and by what date. Without this elementary information chaos can ensue!

Dealing with a real client also provides many students with a new challenge, both trying to adjust to the right level of technical discussion as well as being as professional as they can about the exercise. The process of negotiation, of trying to establish what the requirements of the system to be delivered are and agreeing the details will, in many cases, be the first time that they have had to do this sort of thing. Our experience is that, with a little guidance from us, they can cope very well and gain a lot of personal skills in the process.

1.2Industrial skills, what they are and how we achieve them

What are industrial skills?

We can define industrial skills as those skills that we expect our students to use and rely on when they take up employment in the software engineering industry. Alternatively, we can define them as the skills that employers expect, demand and prefer in new computing and software engineering graduates. These two definitions are by no means equivalent.

Academics, particularly those who teach on degree programmes that are accredited by the BCS, or that comply with UK Accreditation (SARTOR[1]) regulations governing Chartered Engineer status in the UK, are committed to a curriculum that delivers knowledge, skills and experience prescribed by these accrediting bodies. Both bodies demand a vocationally relevant component to undergraduate computer science education and recognize the importance of realistic industrial experience.

So, it is important, for example, that we teach a selection of techniques for systems analysis and design, problem solving, algorithm design and basic competence in programming. It is also important that our students do project work, for a number of reasons, including their gaining experience in the management of projects, people, resources and time; demonstrating an ability to work in a team and completing a substantial piece of work that ranges across stages in the software lifecycle. In the context of accreditation, all these skills are considered industrially relevant.

What does industry want?

However, these skills and experiences are not necessarily those that graduate employers say they want. Surveys conducted by UK Government departments, by the computing press and by software services organizations present a picture of diverse employer requirements. Some employers are seeking quite specific technical skills in new graduate recruits. For example, many small companies, organizations with only a small number of IT staff or companies within particular market sectors such as real time and instrumentation software may have a strong preference for graduates who can program in a particular language, such as Java, Visual Basic, C or C++.

Generic skills

Other employers place greater value on generic skills, two of which feature prominently in their list of demands: the ability to communicate clearly and effectively with customers, team members and managers and the ability to understand the business contexts in which software

1. SARTOR (Standards and Routes to Registration).

systems are commissioned, developed and deployed. With regard to these two skills, some major employer representatives and senior graduate recruitment specialists have been openly critical of the calibre of graduates from computer science and computing- related departments, preferring instead to recruit non-technical graduates who can communicate confidently and who possess awareness of business issues.

So, for a good proportion of employers, particularly software houses and national and international corporations with large IT divisions, the key industrial skills demanded of graduate entrants are not technical ones: they are social, business and managerial skills. Successful graduate applicants will be able to demonstrate the ability or potential to listen to clients; to identify the client's business problems; to suggest appropriate business solutions; to evaluate the cost effectiveness of these solutions; to advise when user requirements or selected solutions are not in the client's interest and to offer a software development package that forms an acceptable compromise between what the client wants and what the client is willing to pay. Furthermore, successful graduate applicants will show some awareness of the importance of effective management of software development resources, of delivering software within planned time-scales and budgets and to agreed quality standards.

Specific skills

Having said that, the value of computing graduates with specific technical skills fluctuates in response to technological changes and fortunes in the computing market. The computing industry is currently suffering a skills gap, with a demand for internet and Web related skills that cannot be met. "Cutting edge" skills in Java, HTML and XML programming, in web site design and testing are very much in demand, so new graduates with a broad understanding of both Internet server technology and front-end Web application skills are likely to find themselves in the comfortable position of receiving multiple job offers, with high starting salaries.

Entrepreneurial skills

The explosion of internet based applications and the enormous opportunity for e-commerce activity that internet and wireless technologies afford also open the way for many graduates to

get involved in start-up companies and in some case to use the entrepreneurial skills that they can develop in university to start their own companies.

1.3 An evaluation of current practice

Across the UK's universities and their 100+ computer science and computing-related departments, software engineering curriculum content and modes of delivery show marked variation. Indeed, beyond a common core of course topics in software engineering, it is generally to be welcomed that such diversity exists, that there is room for individual creativity and room to respond to the needs of any particular local market.

We have not performed an exhaustive survey of industrially relevant experience in the UK's university computer science departments, but we have collated data about industrial involvement in the curriculum from a self-selecting sample of 35 institutions and we have discussed current project provision with representatives from approximately 30 other institutions, as a part of our work on the UK Government (HEFCE) funded Industrial Software Project Support Network, between 1997 and 2000.

Taking our experiences together with the results from the HEFCE funded EPCOS project and the HEFCE report, "Industry-Academic Links in the UK", we have a fairly representative view of the roles that industry plays in current educational practice and of the types of exposure to industrial software development that students receive.

By far the most prevalent and longest established form of industrially based education is that provided by placement of individual students with a company for a sandwich year or for a shorter period, often during the summer vacation. Good quality industrial placements are demonstrably beneficial to students, both in terms of the quality of students' final year work and degree classification and in terms of the students' attractiveness to employers.

However, the sandwich degree model of provision is not suitable for every university, perhaps

because it does not fit with an institution's teaching ethos, or because the university's geographical and economic setting militates against it forging the necessary links with large companies in the IT sector, or simply because the demand for good quality placements at national level may be too great to satisfy. There is little point, then, in increasing software engineering placements in industry, if the quality of those placements cannot be maintained.

A well managed and challenging placement can expose a student to a company's business processes and managerial practices: it is likely to involve team working and should engage the student in activities related to one or more stages of the software engineering lifecycle. A poorly managed and ill thought out placement may do little or nothing to develop students' technical, business or managerial skills.

Whilst a student on placement at an industrial or commercial site gains experience and insight into the workings of a company that could never be recreated in the lecture theatre or the lab, there are some industrial skills that placement students are quite unlikely to experience, but which they can begin to get to grips with when an industrial or commercial organization becomes involved with the university as an external project client. For example, the responsibility for handling customers and their requirements and the responsibility for driving and managing a software project from start to finish are rarely trusted to placement students, but they are the most critical aspects of what we call our "client-led" student projects.

1.4 The need for a fresh approach

There are industrial skills that cannot be taught successfully in a lecture theatre, that can only be acquired through practice and that are best practiced and experienced in an industrially relevant environment.

Traditional placements do not suit all universities. The supply of placements is finite and it may be difficult to maintain and monitor placement quality. For example, it is rare that a placement student has any contact with a client. In some placements the students is merely put into a

corner and given some defined work to do. It is hard, under those circumstances, to develop the sort of skills that dealing directly with a client can give you. Developing your own skills in estimating, planning and managing a project may also be difficult to do in such an environment. This is because you have to follow the *in house* procedures, these can be a useful experience but so much of the software industry is poorly organised and managed that the experience might not be that enlightening.

We are convinced that the use of projects which have a real external client with a business need or problem to solve can provide an enormously stimulating and rich experience for almost every student. The fact that we can arrange for the working environment to reflect best practice in software engineering is an important dimension.

Some academic worry that the idea of client-based projects raises all sorts of quality issues and may be risky in those countries where governments review the quality of university courses in a bureaucratic manner (such as the UK where the Quality Assessment Agency (QAA) carries out visits and assessments which may not favour innovative approaches to teaching). In our opinion this is far from the case and the traditional placement would create many more concerns about the quality of the experience which is largely beyond the control of the academic staff.

The purpose of this handbook is to encourage practitioners to review the range of industrially relevant experience that their own curriculum offers and to be creative, if not daring, in the way they involve "enterprise skills" into their teaching.

Chapter 2.

How to introduce industrial/ client-led projects into the computing curriculum

2.1Identifying your project objectives

The first issue to be addressed is the nature of the objectives that client-based projects involve.

A key aspect will be the introduction of new skills – interpersonal, organizational, managerial; that can sometimes be achieved through the medium of game type activities but the fact that the context is not real seems to reduce their impact amongst many students.
Further more, these projects will provide practice and help to extend recently acquired skills, in a realistic context.

Don't forget that the teachers will also benefit by also developing many new skills and insights into software engineering, project management and related areas.

In the final analysis the objectives must sit comfortably with the department's own, those departments that desire engineering accreditation should note that many of the practical skills required can be found in such projects.

2.2Gaining the support of colleagues

Find a champion

Any new initiative is more easily established in a university department, when it is championed by one or more members of staff who understand clearly the benefits to the department and its students of the initiative, in this case, the advantages of building working relationships with industrial collaborators.
In many cases of which we are aware, an industrial project initiative has been introduced by

staff at senior level, by an enlightened Professor or Head of Department. Alternatively, a member of staff with specific responsibility for teaching and teaching quality may have the degree of influence and respect necessary to drive the initiative successfully. If the department already employs a co-ordinator to organize student placements, this person is likely to have valuable personal contacts and experience of negotiation with industrial partners that will increase the chances of project success in its introductory stages.

An open minded culture

Whoever the champion of industrial projects, they will be more easily embedded into the curriculum where the departmental culture prizes the quality of teaching and the authenticity of students' learning experiences highly. As we remark throughout this handbook, industrial project organization and outcomes are less easily controlled than activities in the lecture theatre. Defining and assessing success for the students can be less straightforward than it is when marking traditional assignments.

We sense that a number of university departments have resisted the introduction of industrial partners in their teaching for fear of an adverse effect upon their teaching quality assessment. Conversely, we feel that with appropriate contingency planning for projects that might otherwise fail and with appropriate student and staff review mechanisms in place, this fear is groundless.

Some persuasive arguments

Where resistance to the idea of industrial projects is anticipated or met, it is worth bearing in mind that a number of external demands on university teaching can be satisfied by the introduction of industrial project work. Both the Dearing review of UK Universities and the UK Committee of Vice Chancellors and Principals envisaged a substantial increase in work experience and industrial placements for students, that can partly be met by our model of “client-led” projects run at the university site.

As mentioned earlier, our model of industrial projects finds favour with accreditors at the Brit-

ish Computer Society and Institute of Electrical Engineers.

There are a number of initiatives that seek to introduce entrepreneurship into the curriculum. One way in which to do this is to introduce a course similar to our fourth year “Set up and run your own software company” course.

There are also a number of initiatives that seek to increase universities' role in the local and regional economy. We feel we have been very successful in raising our department’s profile in the local small and medium enterprise (SME) sector, as a result of advertising for companies to become clients for our students’ software development and consultancy projects. The end result is that the University benefits from excellent public relations. The local business community sees that the "ivory tower" can provide a practical and effective source of expertise for many local organizations, both large and small.

Side effects for other courses

If you are about to embark on an industrial project course then your lecturing colleagues should be warned of some anticipated side-effects, not all of which are desirable!

When we surveyed our second year computing and software engineering undergraduates about their motivations to study, the approval of a real industrial client was judged to be the most important motivating factor of six alternatives. We have to face it: students prefer to impress company representatives than us, their lecturers. As a result, they will devote hours of their energy to completing a project for an industrial collaborator, often exceeding our recommendations for man hours and sometimes to the detriment of effort expended on other courses.

In fact, it is common for students to ask for extensions to deadlines for other course work because they have failed to contain their dedication and enthusiasm for pleasing their industrial project clients. Our colleagues who are committed to students learning by doing a job in a realistic setting are by and large tolerant of these requests for course extensions. Other colleagues are not at all tolerant. After all, they argue, the ability to plan ones time to accommodate the

demands of all courses, is itself a skill that students must learn. What's more, they are right. Some students have learned this lesson the hard way, by succeeding in the industrial project and failing another course.

Another side-effect is the tendency of students to seek out lecturers who are not involved with the project course, but whose subject expertise is related to the project's demands, for help and advice. Again, some colleagues may have the inclination and interest to help, but this tendency clearly needs to be controlled. We would advise careful prescription of project rules at the start of the project course, rules which clearly identify permissible sources of help and advice.

2.3 Identifying suitable student cohorts

Throughout this handbook, we are keen to acknowledge that every university computing department and curriculum is different. Each department will tailor its teaching content to the strengths of its staff. We know that what works for us is likely to need some adaptation to fit with the curriculum and resources available in other institutions. That said, there are some pre-requisites for running successful industrial projects that have general application.

Industrial projects usually provide a very challenging experience to students, but it is our duty as educators to ensure that students are armed with some, if not all, of the technical skills required for their projects to succeed. So, first year undergraduate cohorts are not suitable for industrial projects that require an understanding of the software lifecycle and a rudimentary ability to analyze, design, code and test a working software system.

We do, however, subject our conversion Masters students to an industrial project, virtually from week one. This is partly a pragmatic decision, determined by the fact that they spend only a total of eight months on lecture courses. We justify throwing these students in at the deep end, on the grounds that, whilst their technical skills are weak or non-existent, their life and work experience will already have exposed them to some of the communications and managerial skills that industrial projects demand. Additionally, our Masters projects extend over the better part of two semesters, with very rigorous management by an experienced commercial

project manager.

In common with a number of other institutions, we let our second year students loose on industrial group projects, having prepared them with a dummy, internal group project at the end of their first year. By the second semester of the second year we feel they have sufficient appreciation of lifecycle issues, competence in at least one programming language and the experience of courses in human-computer interaction and professional issues, to develop these skills further through practical application to a real problem. At one institution we know of, the industrial project is arranged for the summer vacation at the end of the students' second year.

In courses with a predominantly business and management orientation it is still possible to carry out realistic projects. Even if in those student groups, where programming skills may be very weak it is possible for the groups to carry out business process analysis and perhaps develop simple web pages and databases for small organizations.

In the United States and Australia, there are a number of examples of industrial group project courses run as "capstone" courses in students' final year, courses that are the crowning glory of a student's training as a software engineer. In the UK, we know of one institution that runs a third year undergraduate group project course focusing on software maintenance. We, however, have not run group projects in the third year, for two reasons. Our industrial projects tend to have specific and practical requirements: they do not pose open-ended nor theoretical topics for investigation This undoubtedly distinguishes computer science education in the UK from software engineering training in the US.

Secondly, there remains the thorny issue of fair assessment of individual students on group projects. In the second year, when project work comprises roughly 5% of the marks that contribute to a student's eventual degree classification, some margin of error in the assessment of an individual's contribution to a group effort is tolerable. In the third year, when project work comprises roughly 15% of a student's degree mark, it is less defensible to set group work and to involve an external industrial organization. Should the group fail or the industrial contact

prove unreliable, there is too much at stake.

Increasingly, universities are seeking to offer four year degree courses leading to the award of a masters degree. This offers a further opportunity for students to do novel project work to industrially specified briefs. Indeed, when we have interviewed our fourth year M.Eng and MSc students about their motivation for staying on another year, when quite lucrative employment prospects beckon, they cite the opportunity to involve themselves in our "Run your own software company course" as the greatest determining factor.

2.4 Assessing and obtaining resources

In seeking to introduce external industrial and commercial clients into software engineering project work, the need for additional resources must be recognized and met. Project courses have a reputation for being expensive in terms of staff time. When we surveyed other institutions already running projects with an industrial flavour, a number of respondents asked "Is there any way of doing what we do more cheaply?"

Estimating staffing requirements

Recruitment, selection and support of clients; negotiation of potential project outcomes with clients; assessment of large volumes of software documentation are all time-consuming activities that tend to fall outside the traditional expectations of lecturing staff. So, how much staff time is required, for each student group, for each project client? Who is best suited to these roles? Do they have to be full-time members of academic staff and can teaching assistants or post-graduate students be involved?

It is not always possible for university departments to find staff suited to running such projects, nor for them to allocate the extra time required for project administration. Staff at both Durham and Salford, for example, have echoed these sentiments quite forcibly.

At Sheffield, we have operated a ratio of one project supervisor to approximately twenty five students on our second year projects, where students work in teams of five. We find it is not too

onerous for each supervisor to monitor five development teams, all competing to build a system for the same client, with a total of three hours' student contact time per week, (30-36 hours' contact time per supervisor per project course). A similar time is required for assessment, marking and smoothing potential project problems, per supervisor per project course.

For the past 12 years the Department has employed a very experienced software consultant to manage the conversion MSc Maxi project. This has been a very successful exercise. These students are on an accelerated conversion degree and it is important that they get used to the formalities of software project management as quickly as possible. The manager establishes a routine for them and has the credibility to do this since he is a professional manager and not an academic. If funds were available it would be nice to do this in the other areas as well but, to be realistic, it is unlikely to be possible. However, academics can play the role with a little preparation and thought.

On the more intensive fourth year course, "Run your own software company", it becomes difficult for a project supervisor to monitor more than about a dozen students. Typically, a dozen students will form three teams and will work for between one and three clients during the course of an academic year. This may mean that a team works for all three clients simultaneously at some points in the year.

The ratios we suggest for project staff to students are quite demanding. They can be fulfilled when there are under 100 students on a second year undergraduate course and when there are only 25 or so fourth year M.Eng students. Project organizers in institutions where undergraduate class sizes are in the range of 150-200 students will probably have to delegate project supervision to teaching assistants or post-graduate students. We have not been keen to do this but we recognize it is a pragmatic solution to increasing student numbers.

What space is required to run projects?

Need a space in which student groups can meet together and with their client to plan and to review project progress. In effect, you need access to many small rooms which are conducive

to small group activities.

What are the likely hardware and software demands?

This question is, of course, highly specific to an individual project and a particular student cohort. However, we would advise against taking on a project that utilizes software or hardware that the department does not already possess. Even if the industrial collaborator has offered to sponsor new hardware or software to support a project, lack of familiarity with an environment may prove a headache for technicians and may prove an insurmountable obstacle to effective management of project progress, as we have found to our cost.

A recent fourth year project that we undertook with some hesitation, was required to use complex transaction and database management software with which we as staff were unfamiliar. The project began to founder and time-scales began to slip unacceptably when tested in a network environment. As managers of the project, we were unable to identify the root of the problem, which was eventually traced, with the aid of a friendly commercial IT consultant, to an inefficient and inappropriate strategy for managing transactions.

Are external resources available?

Software development for external clients may require “state of the art” technical resources that are in short supply in university departments. This is particularly the case in the current climate where industrial clients’ foremost demand is for internet based applications.

Chapter 3.

How to identify potential projects and clients

3.1 Advertising to external organizations

Using personal contacts

For many years, when student numbers were lower and we did not require so many industrial clients for our projects, our Department tended to rely on informal contacts for client recruitment: friends of Departmental staff or lecturers in other University departments.

In fact, from time to time, we still use clients who work for another part of the University, whether in an academic or administration department, in the Students' Union or in a spin-off company. Our University's many health and medically related departments and organizations have proved to be a particularly fruitful source of IT projects.

Some, but not all, of these clients have the advantage that they understand well the educational motives behind our projects and they may, as a consequence, be more tolerant and forgiving of student groups who fail to meet project requirements. Another institution we know of operates industrial projects for Higher National Diploma (equivalent to the 1st 2 years of a bachelors degree) level students, using exclusively this type of client, who is internal to the university, but external to the department, because the project organizers find that they can negotiate project outcomes with these clients more easily.

Clients who are internal to the University can still provide the students with a real need for a software system and with a real challenge. However, we have some evidence that, when given a choice between University-based clients and organizations totally unconnected with the academic world, our students prefer to work with the latter. Although we perceive the University to be a big business environment, projects provided by entirely external organizations seem to

be accorded greater credibility or authenticity!

Some academics may have personal contacts with big IT companies and these might be a useful source of projects. There may be some dangers, however, if the projects turn out to require methods of tools that are not part of the general academic curriculum. However, it is worth considering.

Some clients become personal contacts because they come back to us with further project ideas, and this is an excellent way to get further projects.

Cold Calling

About three years ago, our demand for project clients increased significantly, in part because of the introduction of our fourth year software company course. We needed to recruit upwards of a dozen clients each academic year, whereas previously, we might only have recruited two or three. As a result, we embarked on an exercise in cold calling, contacting almost fifty local companies across business and industry sectors, identified from the Electronic Yellow Pages.

Roughly a quarter of the companies asked for more information to be supplied about our requirements for project clients, from which three came forward and visited our Department. Eventually, one of these companies, a china and glassware retailer, became a client for our second year project that year.

The exercise did not prove effective or efficient as a means of finding suitable clients. However, we learnt a lot from it: a number of organizations will drop out of the client selection process, either because they are too busy, or because the need for a software system has changed. Hence, when selecting clients, always keep one or two potential clients in reserve, so that the student project can go ahead even if a company pulls out at the very last minute. This does happen. You need to be prepared for it!

Furthermore, you may find that you have not been speaking to the right person in a potential

client organization. More than one company representative we negotiated with during this exercise was very enthusiastic about a computing project, but was too junior in the organization to sanction the project and authorize the payment of project prize money.

We concluded that cold calling as a method of recruitment placed no particular onus or responsibility on potential clients to commit to successful project completion: clients recruited this way may have insufficient stake in the project's outcome. Better to advertise in such a way that client organizations have to take the first step in contacting us.

Successful advertising channels

Broadly speaking, we have found two different successful channels for recruiting industrial project clients. The first is via existing publicity and networks in the local business community, both through a paper newsletter that is produced on a quarterly basis and circulated to local businesses by Sheffield's Chamber of Commerce and through an electronic mailing list maintained by Sheffield's Business Club. When we use either of these channels, we tend to receive roughly half a dozen serious enquiries from potential project partners. Using these channels, we can phrase our advert so that recipients know the time-scales within which we operate and have an idea of the scope of the project we are looking for.

We would wholeheartedly recommend that you explore contacts with the Chamber of Commerce and Business Link networks in your area. We have always found them obliging and supportive. The main caveat here is to be prepared and plan ahead! When a newsletter is only produced on a quarterly basis, you may need to supply copy four to six months in advance of your requirement to start a project, to allow for both newsletter circulation and the assessment of client and project suitability.

The second avenue that we exploit is a very good working relationship with our University's Careers Service, which operates a brokerage for student projects. The Careers Service employs a member of staff whose job it is to raise projects from local organizations, for students of all disciplines to pursue, either as an extra-curricula activity or as a part of an accredited learning

programme. This member of staff is well briefed on our industrial project requirements and refers to us, on average, about six project clients per year. On a recent occasion, a client referred by this route has supplied three usable project briefs.

We are uncertain how many other universities' careers services organize a similar student project brokerage facility, but we would encourage anyone involved in industrial project work to make contacts and develop links with their respective careers office. At our institution, they are very supportive and also make a contribution to our group project curriculum in the areas of developing teamwork skills and developing negotiation skills.

3.2 Assessing client suitability

Assessing client motivation

There are a range of motives that an external organization may have in approaching us as potential project clients. These motives vary, depending on the size of a client's business, the business or industry sector and whether the business is in the private, public or voluntary sector.

Low cost development: frequently, our clients are small private enterprises or voluntary and public sector organizations for whom the development of software products by commercial developers would be prohibitively costly.

Low risk development: sometimes, clients want feasibility studies and prototypes for software products which are initially too risky to implement on a commercial basis. This has usually been the motivation of small, private client companies who already utilize a considerable amount of software and communications technology, or whose main business is within the IT sector.

Some potential clients may be attracted by the high quality of development methods, or by increased access to the latest technologies. Potential skills transfer between commercial and

academic organizations has also been the motivation of small, private client companies who already utilize a considerable amount of software and communications technology, or whose main business is within the IT sector.

In only a minority of cases have our clients been motivated by the opportunity to assess the calibre of our future graduates. Few of our clients are of the size or business type to recruit computing graduates. However, we would expect this factor to be of paramount importance in the case of large national or international IT companies acting as industrial project clients.

Whilst low cost software development is often the main motivation for a client, if this is their only motivation and if they show very little interest in the educational rewards that students might derive from the project, then reject the project. Do it tactfully, but reject it. Furthermore, on no account accept low cost development of a possible commercial product, intended for sale to a third party, without making the copyright and licensing issues crystal clear, (see below on negotiating copyright).

In general, be guided by your gut instincts in selecting a client. If the client has both demonstrable enthusiasm and realistic expectations for the project then there is a good chance the project will succeed.

Assessing clients' time commitments

Of course, an enthusiastic client who has no free time will not be suitable! So, at the start, invite the potential client to meet you in the Department. If he or she does not show, or has to reorganize the appointment because of pressure of work, this is nearly always an accurate indication that this client does not have the time that you and your students need. Similarly, if the client wants you or other lecturers to visit them, then they probably do not have enough time to get involved. After all, our clients are expected to make regular visits to meet student groups at the University site, sometimes on a weekly basis, when the project is underway.

Is client location an issue?

The location of the client can have a bearing on project suitability. It is useful to offer students the opportunity to visit the client site, perhaps as a part of the requirements elicitation process, - to see the business context in which the system will operate, or later, to obtain system validation from a range of potential users. So, for this reason, it has helped us to recruit clients who are located within a short car, bus or tram ride of our campus.

However, we recognize that this mode of operation will not suit every institution. Some universities utilize Web-based communications to particularly good effect in the management of student project progress, for example Durham's SEG World structure, (http://www.durham.ac.uk/~dcs8s00/)

We are one of a number of institutions that has experimented with teleconferencing as a vehicle for communication between our fourth year students and clients located in our region, twenty to forty miles away. However, in the main, we feel some face-to-face communication between client and students is vital to a successful project. There is a danger in over-reliance upon e-mail to support client – student communications. There's a real risk that the requirements elicitation process will be superficial and incomplete and that it will not yield the level of mutual understanding of a project's objectives necessary for the project to work well.

Assessing the suitability of the client's representative

Before going any further, you need to ensure that you are dealing with the right person in the client company. In this case, the right person is someone with sufficient authority to approve project and design decisions and who is also sufficiently representative of the intended user group of the planned software system to influence matters relating to system usability. If the person with authority will not also be a representative system user, then other user representatives ought to be encouraged to have a regular involvement with the project.

Assessing the client's knowledge of computing

Previously, we have said that an ideal client representative should also have little or no experi-

ence of computers, in order that students are afforded the fullest opportunity to practise eliciting customer requirements. We felt that a computer literate user might provide too great a part of the software specification themselves and might unduly prejudice design decisions from the start. However, as computers become more ubiquitous and as the recent expansion of internet usage continues apace, it is now very rare to meet a potential client who has no experience of technology and no conception of the system or, at least, the kind of user interface that they want.

There are still clients for whom a little knowledge is a dangerous thing and, if possible, these clients are to be avoided. A case in point is a client who wanted a database system to store information on his suppliers and customers that would also deal with aspects of contract expiry and renewal. This client had some specific ideas about relational databases and a very particular semi - relational model of his system in his head. Compliance with the model in his head seemed to become the key acceptance criteria for all our student teams' systems. For him, the project was a failure, in large part because all the students' systems relied on more normalized structures than he had conceived or understood. For the students, project work became dispiriting because they could not conform to the client's convictions about the data model.

3.3 Assessing project suitability

Assessing a project's suitability is one of the most difficult aspects of organizing industrial projects and it is one on which we have frequently been asked to advise by colleagues considering the introduction of industrial clients and real-world project briefs.

Suitability relates to scope and technical difficulty of a project's requirements and to the degree of familiarity with required hardware and software on the part of students, project supervisors and technical support staff. First and foremost, suitability should be assessed with respect to the level of risk of project failure.

Assessing severity of risk

Our most fundamental rule is this. A project is not suitable if it is mission critical to the client organization. In this case, the consequences of failure are too high. Project supervisors and students should not have to carry this level of responsibility.

With regard to project resources, we would advise against a project that required software or hardware that the department does not already possess. Bringing in new hardware or software has many attendant risks: the new kit may not be delivered in time; the technicians may not be able to install it successfully; staff may not be trained in time to offer support to students. The nature of our project clients – typically SME organizations – has meant that most software development takes place on a PC platform, with Visual Basic, (VB), and PHP as the students' most popular programming languages.

Not every university department will want or be able to host projects on a Microsoft platform. Some universities may prefer to develop industrial project software under Unix, or for an Oracle platform, in which case larger client organisations will be needed, who have this level of technical architecture in place.

Despite the prevalence of VB, and PHP, for project implementation in our department, we wait to see what the impact will be of Java having been adopted as the department's main programming language. We anticipate some resistance from small client companies, who in the future may need to find Java programmers to maintain their system. Yet, we also anticipate projects having slightly less risk. Previously, our students had to teach themselves VB during the course of the project and a few really struggled with this. Now, all project students should be properly prepared for the programming tasks demanded of them by the industrial project.

Assessing scope and technical difficulty

The scope of a project and the amount of effort it will require cannot be determined or quantified with any accuracy at the time the project is first discussed between potential client and project lecturers and supervisors. We ask potential clients to bring a description of their project

idea to the initial meeting with us, outlined on a single side of A4 paper. It is hard enough to make useful estimates of effort even when a project's requirements have been fully specified by the students, let alone when we have only a high-level sketch and some broad brush statements of business objectives.

So, although we know, for example, that our second year projects typically allow a team of five students to devote up to five hundred hours of effort to system development, we do not make judgments of suitability on this quantitative basis. Instead, we have tended to make predictions on past experience of the capabilities of students in a particular year, using a particular type of technology to solve a particular category of problem.

Whilst every computing department will have its own ideas about the capability of its students and the particular competencies that it expects of them at a given stage in their university education, here are some guidelines that we tend to use. Of course, these guidelines change, particularly in response to changes in technologies and development platforms. Whereas once we preferred clients who wanted a computer system to replace an existing manual system, we increasingly encounter clients who have an existing computer system, want it replaced and require our students to help with the migration of data. Doubtless, the guidelines below will be out of date within the next three to five years!

The design of a simple web site is not sufficiently challenging for an industrial group project in the second or subsequent years of undergraduate study, although it could provide a useful dummy run in the first year.

A single user database system or a small networked system for a PC platform, constructed using a 4GL tool, is typically feasible for our second year and conversion Masters students to develop to the point where the client can install and use it.

A small to medium sized database rarely offers students much scope to practise interesting system modelling, nor to engage in serious problem solving and algorithm design. So, a database

system that has some additional, more “meaty” requirement is preferred. This gives the best students the opportunity to shine.

A multi-user system, a database which is to be editable across the Web, or a Web site that requires considerable scripting to support e-commerce transactions are all considered too risky for development by students in their first two years of study, but can be suitable for fourth year undergraduates and advanced Masters students.

A project is eminently suitable if it has a few core mandatory requirements (functional or non-functional), that most student teams have a reasonable chance of implementing, together with a number of optional, highly desirable requirements that not all student teams will elicit or satisfy.

3.4 Allocating students to projects

On our second year and conversion Masters industrial group projects, we have more than one client. Usually we have three clients, so student teams are offered a choice, based on the outline A4 description of requirements that the client has discussed with supervisors.

How we have managed this choice and allocation process is open to debate, but we firmly believe a choice must be offered, in order to maintain students’ feelings of ownership of the project and their continued commitment to it. All right, in the real world software employees may not have the luxury of choosing the projects on which they work! Didn’t we say claim real world authenticity for our projects? Well, in order to minimize risks of project failure and to encourage best results from all participants, we allow them more control over the project selection process than might be the case in an industrial setting.

Latterly, student teams have been asked to nominate their first and second choice of client, up to six weeks before the project starts. Clients are allocated to teams on a first come first served

policy, so a team will not get its first choice if another four or five teams have already expressed this client preference. It has been argued that the first come first served policy encourages a quick rather than a considered response. On the other hand, those teams who organize themselves quickly are to be encouraged and should be able to do background preparation prior to the project's start date, if they so wish.

How has this allocation policy worked in practice? Over the past four or five years, it happens to have worked well, with teams' preferences for clients being sufficiently well distributed that most have worked with their client of first choice and none has had to work with their third choice.

However, care must be taken in preparing the outline project descriptions for student consideration, to try to ensure that no one project sounds much more or less attractive than the others. The easy and difficult aspects of each project should be stated. The status of each client organization should be made clear.

Is the organization a charity? (This will appeal to the more altruistic students.)

Is the organization a private company? (This will appeal to the more business minded students.)

Does the organization have direct links with the University? (This may deter students who do not equate realistic problems with an academic setting.)

Chapter 4.

How to secure commitment from clients

4.1 Negotiating the level of client involvement

It is vitally important before a project starts to set realistic expectations of the amount of time the client should devote to the project and of the role or roles the client will play.

About the single most difficult aspect of industrial project work is the fitting and squeezing of the client's timetable into the straightjacket of the university calendar.

For our second year and conversion Masters team projects, we ask our clients to be available on our premises, for a minimum of one hour per week, at the same time each week, for the first four or five weeks of the project. Every week, the client will spend approximately fifteen minutes with representatives of each student team that is competing to build the required system. This is quite a commitment and it is quite exhausting work.

Indeed, in a semester-long project, during which there are probably twelve weeks when student teams are working seriously, it can be helpful for the client to attend for an hour on each of the twelve weeks. In practice, at the half-way stage, personal client-student meetings may not be so necessary, if the objective is to pass on test data, or to assess the latest version of the students' prototype. Some of this can be done remotely, using e-mail, file transfer or the Internet.

4.2 Negotiating potential project outcomes

It is vitally important before a project starts to set realistic expectations of project outcomes.

Never be tempted to guarantee that your students will build a working system, even if the vast majority of previous projects have delivered this.

We negotiate on the basis that our students will do their best, that they are very well motivated towards building a real system and that we will manage them to the best of our ability. Where our projects involve a number of teams competing for the same client, we can reassure the client that it is very likely that two or three teams will build a useful product, even if one or more teams fails to get to grips with the project brief.

Where our projects involve fourth year students in the student company, we can reassure the client that these students are our very best students, who already have experience of project work in each of the previous three years at our institution.

Always leave open the possibility that the client may not get a finished software product, but discuss this possibility in a positive light. If the product is nearly finished, it may be possible to employ students over the vacation, to add those finishing touches. If the product is no more than a prototype,

A common problem, well known amongst all those who have been involved in real software projects, is that of the requirements changing during the project. We call this *requirements creep* and it can be a major contributor to project failure. This problem needs to be discussed with any prospective client. In the light of the short timescale for many university-based projects it is recommended that the requirements should be more or less fixed after an initial exploratory period. We have found 5 weeks to be an appropriate period. If eXtreme Programming is used then this could be relaxed. One observation that we have made is that many of our clients, who are often first time clients and not always technically aware, need to have a period where they can be helped to think about what is possible and what is desirable for their business need. EXtreme Programming should be an ideal way of managing client expectation and

the intense communication between the clients and the teams is an important factor in the development of ideas for both sides.

Finally we need to address the issue of maintenance. For many years we did not offer any maintenance service and clients took their winning solution on sight and at risk. Because they are so cheap this was always acceptable. It is possible to broker maintenance arrangements through introducing clients to students who were interested in doing some work over a vacation period for payment. This is often all that is required. If a student company is introduced then this could also be a mechanism for paid maintenance contracts with former clients.

4 3 Negotiating contractual and copyright issues

At a number of academic events, - conferences and workshops – we have attended or organized over the past three years, some participants have expressed the view that attempts to involve industrial clients in projects at their institutions had foundered over legal and contractual issues. Either the university or the client had insisted on a contract that the other party was unwilling to enter into. Although there may be instances in which both project partners' interests are best protected by a formal contract, our view is that client-led projects should be negotiated on the basis of trust, with the benefits to the students' education being a central shared commitment and motivating factor. If the management culture in a university is heavily reliant on formal contractual arrangements, it may not be possible to organize successful projects using our model.

Refer again to low cost development of a possible commercial product. The danger here is that students will feel that they are being exploited. At some point during the project, the students will probably glean enough information to make an informed guess at the amount of money that the client hopes to make if their software is sold. When they compare this potential profit

to the amount of labour they are donating free or supplying at a nominal charge, they will understandably feel aggrieved. On one occasion, we had to terminate a project after only a few weeks' activity, when the client's real motive, to maximize profit, became apparent.

This is probably the strongest argument we can make in favour of negotiating copyright issues at the outset of a project, if there is a realistic chance that the client will seek to sell the students' work to a third party.

4.4 Negotiating prizes and fees

Prizes

Educational industrial software development projects are little different from software development projects conducted entirely in a commercial arena. All participants are stakeholders and, from time to time, they may need some encouragement to remain committed to the project.

A modest prize offered to the winning student team provides some incentive to the students, to "go that extra mile" or put an extra bit of polish on their work. For our second year team projects, each client puts up a prize of £250, - £50 for each member of the winning team. With the possible exception of small charitable bodies, who may have absolutely no budget for software, you should be very cautious about proceeding with a client who questions whether this level of prize is really necessary.

The prize is also valuable as a symbol of the client's commitment to the project. It can bolster the prestige of a client company, if that company's sponsorship of the project is publicized in the university's newsletters and in the local press. At Durham University, a number of prizes are awarded by more than one of the major international retailing companies for the best software engineering project work. These companies do not necessarily supply project briefs for students to work on, but they do seek to endorse students' software engineering achievement,

partly to ensure that the company's name remains in the students' minds when it comes to their graduation and their search for a new graduate job.

A prize is one way in which clients and companies can demonstrate the value they place on students' work. Payment of a fee is another, but this financial mechanism may not suit all computing departments contemplating an industrial client's involvement in their teaching.

Fees

Asking a client to pay a fee is something that is only done by the fourth year members of the student software company. They typically negotiate a notional fee, dependent on the client's business circumstances. This fee has ranged from a payment of £50 or £100 by a local charity up to payments of £3000 by commercial organizations. The student company states in its charging policy that it may cost a job at a rate of £10 per hour, but in practice, this fee level has virtually never been applied.

Chapter 5.

How to support/ prepare students for work with clients

5.1 Supporting/ facilitating team working

Team Selection

There are many different strategies for arranging a student cohort into project teams, a number of which we have experimented with over the past ten to fifteen years. In part, the strategy is determined by how much we know about the students and how much they know about each other. Creating teams in a cohort of new conversion Masters students is, of necessity, an act of social engineering: we barely know the students and they have never met each other before. In such a context it is only feasible to be firm and allocate teams at random.it is necessary to help break the ice and some sort of social event can do this. We have found that most students will quickly get on with the job, their life is so full of new activities and the pace of the courses so swift that they have no time to think about potential problems with the members of their group. Within a few days they have met their client and they have to produce something for the meeting with the manager that week. Student who have problems at this stage tend to have problems throughout and may well be unsuited to such a demanding course.

It is rather different with our second year students, who will already have worked in a tutorial group on a weekly basis from the start of their degree courses and who will have worked together on an internal team project, the Crossover project, in the first year. We could impose the composition of each team, with a view to distributing the most able students amongst teams, or the female students or whatever selection criterion we felt was most significant. However, we don't do this. We let the students choose their teams themselves.

This selection strategy surprises some people. They do not feel it fits real world practices. In

industry, a manager would decide who to put in a team, based on some mixture of demonstrable skill and assessment of personalities.

We justify the students having a choice of team members in much the same way that we justify them having a choice of project client. We need some, if not all, of our teams to succeed, for the students' sake, for our sake and for our clients. Hence, we need to minimize the risk of project failure through failure of teamwork. Allowing friends to work together does seem to reduce the incidence of teams and team communication breaking down completely. Furthermore, the element of team choice may give students a greater degree of ownership and sense of stake in the project, which is also likely to increase team commitment.

It is not hard to anticipate that there are problems with giving students choice. When friends in a team fall out, over differences of opinions about designs, about coding techniques, about team roles and effort required, this can be a lot more acrimonious and personally damaging than when team members who do not know each other come to blows. We don't see this happen very often, but when it does, project supervisors may have to display considerable tact and diplomacy in handling the fall-out.

Perhaps a greater difficulty is knowing how to handle the students who do not want a choice about joining a team, who want to be told whom to work with. Usually, about 10% of the second year student cohort fall into this category. If we were to generalize about these students, we could identify some of them as the weaker students (rather like those of us who were never picked for the football or the rounders team, because people knew we could not control the ball); some of them are bright students whose preference is to work on their own and some of them are unmotivated students who have not bothered to make any effort to seek out a potential team. Most commonly, one or two teams will comprise students who have been placed together by the project organizer. We have not analyzed the performance of these teams quantitatively, but the strong feeling is that they struggle to get reasonable marks. It is possible that they do worse than if we had socially engineered team membership for the whole student group.

Team roles

Even when we give students the choice of whom to work with, we still advise them that they should choose team members judiciously and that they should look for a mixture of skills and strengths. We send an E-mail to all the students a few days before the course starts with the following suggestions: “Bear in mind, when organizing yourselves into teams, that most teams require a blend of skills, good programming knowledge, good organization and planning skills, good documentation skills, good skills at talking to the client etc. try to balance your team so that you can cover all these attributes. You will have to meet outside the lecture times so that is another thing to bear in mind.“

Both on our second year "Software Hut" and conversion Masters "Maxi" project, we expect a team to appoint a leader and a team secretary. We further expect that these two positions will rotate around the team, so that each member has the experience of managing the group for a stage of the project and each has the experience of note taking and group document management. Our industrial project manager for the "Maxi" project, Stan Price, places a great deal of responsibility on the student as project stage manager. He or she is expected to collate all the team statistics and present figures for effort and progress at the end of the stage.

Of course, asking each student to take a turn at team leadership and management is helpful in assessing individual student performance (see section 7.2). It can presuppose that a project sticks to a waterfall lifecycle pattern, which in reality is not likely to be the case. Furthermore, in a short, semester-long project, the project stages are themselves so brief that the influence of the leader on team performance may not be discernible. Even in a two semester project like "Maxi", there may be a whole range of personality and inter-personal factors that determine the effectiveness of each individual as team manager, so assessment of performance of team roles must be conducted very carefully using evidence from a wide range of sources.

Team building and communications skills

So many times in the press we read criticisms of technology graduates and their paucity of

team and communications skills. To a great extent, these skills can only be acquired through practice. So, what better justification for running some team activities on a computing curriculum! Many of the students, however, are uncertain about the prospect. Whilst they tell us that they value the real-life aspect of working for an industrial client, the majority also say that they would prefer to work alone.

Amongst our second years, we see many groups of students getting on with their industrial project as friends, supporting each other and sharing a joke, but we also see teams who do not know each other and who need some immediate support to start communicating together. After all, within a week or two of getting together, they will have to function as a team in front of an external client. So, to help ensure that teams who do not know each other well are not placed at a great disadvantage, we introduced specific tutorial sessions on team working at the start of the second year "Software Hut". We hope that these sessions are also helpful for the teams that are comprised of friends, that they too will learn from reflecting on the strengths they bring to a team and the areas in which they are reliant on other team members.

You do not necessarily have to run sessions on teambuilding and communications skills yourself. Few computing lecturers will have the experience of teaching such generic and personal skills. We enlisted help from our university Careers Service, who run and repeat an introductory two hour session for our second year project students, working with them in sets of approximately thirty students. If your Careers Service cannot help then your Staff Development Unit might be able to put you in touch with a suitable tutor.

We are at the stage now where we need to assess whether these teambuilding sessions are genuinely helpful. We also feel the need to do more to support students with extra practice in interviewing skills. Undergraduate students with limited work experience may have little experience of planning and conducting a structured interview. It is worth considering running some role playing sessions in which students experience the roles of both interviewer and interviewee and have the opportunity to reflect on comfortable interview techniques and successful information gathering.

Our fourth year students in the student software company, "Genesys Solutions", have specifically asked for help and practice in negotiation skills, for which an experienced careers trainer was brought in. These students are expected to negotiate a fee for their work and find that difficult to do. However, even without a fee, the process of specifying, designing and building any software system is one in which compromises have to be negotiated between client and developers, so this is a useful skill that could be practised on team project courses at any level of study.

5.2 Setting expectations of students' work and attitudes

It is always easier to do something a second time round. After you have been running industrial projects for a few years, they begin to gain a reputation amongst the students: anecdotes about spectacular project successes and failures become part of the department's folklore. Indeed, your reputation for industrial projects might become such that students use this feature as a criterion in selecting your institution's degree courses. Once a course has a reputation, students will already have a firm expectation of it before they start. With industrial project work, it certainly helps if that reputation is a tough one! There's no excuse for being late! You sweat blood on this project! There's no excuse for not doing the work!

Expectations of professionalism

On both the second year undergraduate "Software Hut" project and the conversion Masters "Maxi" project, although the first is managed by lecturers and the second by an external industrial project manager, both tend to take a tough line in the first few weeks, but ease up later on, particularly with students who take our demands very seriously and who may find the pressure of project work very stressful.

In the main, we do not need to set high expectations to motivate the students. The clients' expectations are usually sufficient motivation. We do, however, have to reinforce expectations

about professional behaviour, about time keeping, courtesy and confidentiality. Put simply, whilst non-attendance and late attendance at other lectures may be tolerated, lateness or absence from team project activities will be noted and marked down. It is made clear to students from day one of the project that each one is responsible for his or her own efforts and for the success of the team. Things will go wrong with the project and part of the learning curve is coming to terms with problems as they arise and working out alternative solutions, rather than apportioning blame and giving up. The students are already aware of the concept of the "personal software process" (ref) and it is useful to remind them that the project is an opportunity to put this process into practice and find out what it means for real.

Expectations of software quality

It also helps students to set standards for the production of software documentation if they can look at and discuss concrete examples. This can prove difficult, because most textbooks only include small fragments of system documentation, contributing to a shortage of realistic case-study software development materials. This is one area in which a good working relationship between a department and a large software company might be an asset. You may be able to persuade the company to supply example documents, suitably anonymized. That is, if you are happy that the company's methods and quality standards complement those used throughout your degree and discussed in the lecture theatre.

One advantage of having run industrial projects for a number of years is that we can select some of the best examples of previous requirement specifications, test specifications, user manuals and other documentation and pass them round as illustrations of both the quality and quantity we are looking for.... say something more about changes in methods, difficulties students have faced in choosing appropriate modelling techniques...and the way our views have changed on the amount and nature of documents.

5.3 Setting time-scales for students' deliverables

The biggest risk to industrially focused student projects is that they will not deliver a successful outcome within the time-scales of the academic semester or year. In sharp contrast with real world commercial projects that frequently run late, albeit with the payment of penalties, educational projects must deliver work that can be assessed in time for examination boards, as well as work that constitutes a reasonably useful product for the client. There can be no exceeding the end date for the project.

This pressure to deliver is very obvious to us as project supervisors, but may not be immediately apparent to the project students, at second year or conversion Masters level. Left without guidance on deadlines for deliverables and project stage completion, most teams would get off to too slow a start and find it very difficult to make up lost ground in the last weeks of the project. Whilst we do want our students to learn about time management, resource planning in a practical environment, except at the level of the fourth year student company, it is too risky to leave all the decisions to them. We provide a week by week framework that suggests where we expect them to be in terms of project activity and percentage of total project effort completed. This framework is also useful, because we use it with our external clients when negotiating their involvement. This establishes a common agenda and a set of common expectations that students and clients may have of one another.

Chapter 6.

How to run projects on a weekly basis

6.1 Role of the project lecturer/ supervisor

The role of the industrial project lecturer is very closely related to the objectives that have been set for a project course. If an objective of the project course is to introduce a substantial element of new technical skill then it has to be done with some care. It may be that some of the students already have that skill if it is knowledge of some programming language and could teach the others. Or we could find a more senior student to do it. In the main these technical skills do not present a serious problem. What is more likely to happen is that students' management or negotiation skills are weak and it is then up to the lecturers to intervene in an appropriate way, perhaps with a short course or exercise or, and this is surprisingly effective, by holding a discussion on the issues.

First and foremost, the project lecturer is a facilitator. He or she sets up a project environment and experience that will challenge students to learn by doing and by making mistakes. The good project lecturer is able to build a supportive atmosphere in which it is acceptable to make mistakes. The good project lecturer knows, also by experience, when to let students work and make decisions unaided and when to intervene, offer support, advice and guidance. He or she should be an enthusiast and show interest in and express praise about the students' work.

Last and least, the project lecturer is a fount of wisdom about software management practice.

6.2 Curriculum content and modes of delivery

There are two basic approaches to providing the environment for students to learn about a par-

ticular technical subject - a specialized programming language, a design tool or a new technology. One way is to provide detailed lectures on the material, these could either be by the lecturer or a guest speaker, depending on the available resources and the context. This was the way that we used to deal with this issue but there were a number of problems that have convinced us to revise our thoughts. Firstly, the lectures tend to be introductory and general and do not really address the specific problem area that the students need to understand for their projects. This led to numerous complaints that the lectures were of little value in the context of the project. In recent years we have been much more insistent that the students do their own research and we provide only general advice and no technical backup. The consequences of this is that the students become much more self reliant and develop their own lifelong learning skills. The motivation for building a software solution is so strong that the teams actually relish delving into the technicalities of the new subject area and are perfectly capable of doing so. This is so important for their future professional careers that we now make it an open assumption throughout all our project work. As an aside, it is quite remarkable how this produces a highly capable and mature professional, we see the difference between the Sheffield educated 4th year students in Genesys alongside students on the MSc from other universities with high quality degrees. Initially these other students want us to teach them the material and it takes them a semester or more to adjust to the more enterprising approach that we promulgate.

The best use of the formal lectures in the curriculum is to help students to manage the stages of their projects and to reinforce specific issues such as planning, risk analysis, preparation of user manuals and maintenance documents, and so on. We recommend that these relate directly to the projects and ask students to reflect on their work, both problems and successes throughout the course. It should also be said that lecturing for the sake of it is counterproductive in our opinion, the students want to get on with their project and only need reinforcement that relates

directly to their principle endeavours.

6.3 Monitoring project progress

Real projects are not a cheap way to teach. In order to ensure that the projects work well it is vital that lecturers meet with every team at least weekly. This is so that specific advice can be given, offer encouragement in the face of what seems to be a daunting task initially, monitoring the progress of the teams and of individuals within them. Are there some students who seem to be drifting, disengaged from the group in some way, lacking in motivation or confidence. The lecturer *must* intervene as soon as this is apparent. These meetings provide an impression of who is doing what in a team and will be useful when calculating the final assessment of the module.

6.4 Managing client involvement

It is also vital that the lecturer talks frequently with the client. Usually our clients have never been clients before and need some advice and managing in order to make the project successful. Sometimes their knowledge of their own business and the facilities available to them needs to be checked out if they are to receive a viable and relevant system.

Ultimately the systems will have to be installed in the client's business and this may be by the client themselves, in which case an installation guide and as far as possible a very simple installation process is needed. Sometimes, however, it is necessary to deal with a specialist administrator in the client company and this needs to be done with care. Sometimes these individuals resent what they consider to be unqualified amateurs becoming involved with *their* system. We try to emphasize that the quality of many of the student's systems are of equivalent standard to a professional system because of the care that we take with the management of the

project.

We also involve the clients in the assessment process. We believe this to be vital. In our Software Hut module the client awards 50% of the total marks for the module. This is achieved by providing a simple marking sheet which is discussed with the client to ensure that it meets their aspirations. Different categories are identified, typically these include:

ease of installation;

quality of user manual;

usability of the system;

reliability of the system;

functionality of the system;

overall impressions.

Each category is marked on a 1 (poor)... 5 (excellent) scale and room is provided for comments.

For the software hut the system with the best mark from the client is then the system that the client can use in their business - they provide a prize for the winning team as their part of the bargain.

In Genesys, of course, things are different and upon acceptance of the solution the client's receive an invoice from the company.

Chapter 7.

How to assess client-led group projects

The issue of assessment procedures and criteria for client-led group projects was raised at almost every event we attended. Two potential difficulties exist: making an accurate assessment of individuals' contributions to a team effort and making a justifiable comparative assessment of students working on different client briefs. In forming these judgments, assessors are frequently reliant on statements made by students, supported by their own observations. There is always the possibility that students will make false claims and that observations are inaccurate, however careful the assessment process is. If the assessment procedure in a university is particularly rigid and does not admit students' self assessment as legitimate evidence of achievement, it may not be possible to run projects such as ours at that institution.

7.1 Who makes the assessments and when

There are three stakeholders in the system, the students, the clients and the academic staff. All have a role in the assessment of the projects.

Firstly the client is giving up some time and money and needs to be satisfied with what they receive. In our system they formally assess the quality of the products delivered and provide marks broken down into categories for such things as, ease of use, functionality, robustness, quality of user manuals and so on. This is done at the end of the project.

The students are asked to assess the project both as a team an individually. They can describe the problems that they experienced and what they have learnt from it all. we ask them to identify the contributions of all the team members and to sign the report. This can help us to iden-

tify both a team mark and an individual mark.

The lecturers also assess the quality of the processes and the products generated throughout the project. We look at the timesheets and the minutes of the meetings as well as the designs, documentation, code, test strategies, etc. This aspect will provide the other 50% of the module mark.

7.2 What are the assessment criteria

For the client the assessment is as follows.

The criteria are negotiated with the clients. We have found that a simple mark sheet with about 7-8 criteria works well. The clients are asked to allocate a mark from 1 (poor) to 5 (excellent) against each criteria for all the systems they evaluate. In the Software Hut this then counts for half of all the assessment and is a good means of motivating the students to satisfy the client, where this can be achieved!

The students are asked to assess the module in terms of its interest, its relevance and enjoyability. They also state how difficult it was, how time-consuming and whether they were satisfied with the support provided by the lecturers.

The lecturers look at the quality of the products and also the success of the management process and the team work.

7.3 What evidence of performance and achievement is assessed

We collect almost everything that we can. The students submit, electronically, weekly timesheets, minutes of meetings and all key deliverables such as requirements documents and

designs, the code and test results. All of this information is useful in the assessment process.

7.4 The assessment process in detail.

Software hut.

Deliverables

Submission of assignment work will be in two stages:

1) Requirements Documentation (end of week 5)

1.1 A statement of requirements, signed off by the client 5 marks

1.2 A detailed specification of the proposed system, using an appropriate language, signed off by the client 10 marks

2) Final deliverable (end of week 12)

2.1 A working system together with appropriate system documentation for the client.

The software should be capable of running on a PC or network of PCs (with an appropriate technical specification - as defined by the client).

2.2 The total documentation should contain:

2.2.1 A copy of the Requirements documentation in 1) above

2.2.2 Design documents for the proposed system using a suitable methodology 10 marks

2.2.3 A sample of appropriate (commented) source code for the delivered system, screen lay-

outs, etc. (maximum 20 sides).

2.2.4 A completed test plan and acceptance test results 10 marks

2.2.5 User documentation containing: 5 marks

installation instructions

user guide/ manual

maintenance guide/ manual

2.2.6 A commentary on the project, (maximum 10 sides) containing: 10 marks

A log of the project describing important milestones

A description of the group structure, the roles of individuals and mechanisms for communication used between group members.

A description of the quality control strategy, who did what, when and how the acceptance criteria were defined.

A list of any references to the literature used during the project.

An evaluation of the group performance including an allocation of the proportional effort contributed to the project by individuals in the group. This statement must be signed by all members of the group.

In addition to the commentary, please supply copies of all team meeting minutes, showing responsibilities for actions, progress made and revisions to the project plan.

Assessment and Marking Scheme

The breakdown of marks for this particular assignment will be:

Evaluation of the processes used and the engineering quality of the product, based on the sys-

tem documentation and the commentary = 50%

See above for detailed allocation of these 50 marks

Suitability of the delivered system, based on client satisfaction = 50%

Documentatio

Presentation 5 marks

User Manual 5 marks

Installation guide 5 marks

Software system -

Ease of use 5 marks

Error handling 5 marks

Understandability (use of appropriate language, etc.) 5 marks

Base Functionality (completeness) 5 marks

Innovation (extra features) 5 marks

Robustness (correctness - doesn't crash) 5 marks

Happiness with product 5 marks

Genesys.

COM 4010/6900 Assessment procedures for Genesys.

The assessment of Genesys is entirely by coursework. Since the course is group based it is important that regular observations and collection of data and other information.

The observations are carried out at the weekly Board meeting when general company business and progress is discussed (always chaired by a different student) and then followed by individual project meetings at which a tutor discusses the project with the project team.

The data is:

Minutes of meeting, these are collected weekly.
Time sheets, these are collected weekly.
Monthly reports dealing with the individual projects and a system administration report and occasional Research & Development reports.
Information and questionnaires from clients together with client interviews, where possible.
Individual self assessments carried out by students at the end of each Semester – these are moderated by the academic staff.

There are two stages to the assessment.

1. Formative assessment carried out in February, this does not count towards the final assessment.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

Mark sheets are produced, one is based on the group's performance and the other incorporates the group mark together with individual marks into an interim individual mark.

2. Summative assessement carried out in June. This provides the final mark for the module.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

The students also each write an individual critical assessment of their time in Genesys, this is also used to structure the discussion.

Two mark sheets are produced, one is based on the group's performance, (Appendix A), and the other incorporates the group mark together with individual marks into a final individual mark, Appendix B).

Appendix A

Genesys Solutions' Students 2003-4: Preliminary Assessment, February 2004

Team _____

Team Qualities and Achievements:

Team Grade:

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement	1	2	3	4	5

(relative to abilities at start of course)

Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	1	2	3	4	5

Individual Grade:

Appendix B

Genesys Solutions' Students 2003-4: Final Assessment, June 2004

Team __

Team Qualities and Achievements:

Planning Effectiveness	1	2	3	4	5
Requirements Elicitation	1	2	3	4	5
Specification and Analysis	1	2	3	4	5
Design and Implementation	1	2	3	4	5
Testing and Quality Strategy	1	2	3	4	5

Comments:

Team Grade: __/25 (__%)

Team Member :

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement (relative to abilities at start of course)	1	2	3	4	5
Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	12	3	4	5	

Comments:

Individual Grade: __ / 35 + Team Grade: __ / 25 Total: __ /60 = __

Equivalent to __ %

7.5 Justifying the assessment process

The assessment methodology has evolved over a number of years. We are happy with both the process and the results it produces. We have had very few problems or complaints from the students - or from colleagues or external examiners!

A key factor is that we provide a great deal of information to the examiners, all the documentation from the projects. All this is archived.

Chapter 8.

Benefits of industrial group projects

8.1 The students' views

"We feel that we worked effectively together in achieving the aims of our project. Every member brought a different element of his or her personality and character to create a productive working environment. The diverse mix of skills meant that the workload could be shared.

"In completing the project, every member of the team has had a thorough grounding in the area of project management. The skills learnt and experience gained will prove invaluable when applied in industry in the future. The rigid time schedule meant that the team had to be organized and have the ability to make critical decisions about the direction the project was moving in. On most occasions, a democratic approach was taken to resolve any disputes."

Students in Software Hut team 8, 1999, the winning team working for Sheffield Volunteer Bureau, (Jane Berry, Shabana Mushtaq, Ajay Mistry, Matthew Wood and Darren Mothersele).

"The whole team visited the client at his site to demonstrate an Access-based prototype to him. This was very useful in highlighting what changes needed to be made. The visit was also very successful in that it helped to bridge the "semantic gap" (between us and the clients) and it provided us with a better understanding of the company and its computing facilities. As a result of the meeting we were able to finalize the requirements. This was a major milestone.

"Each member of the group was fully committed to his duties on the project from the start. Each member was trying to be competitive in his allocated duties. What is important though is

the full understanding, cooperation and trust between the group members as a whole."

Students in Software Hut team 4, 1999, the winning team working for Don Valley Bearings Ltd.,
(Jason Chow, Ali Al-Khalifah, Chris McCarthy, Tak Sing Lam and William Dowie).

"We think the industrial projects are an extremely good idea for the
following reasons:

- Customer facing - this is becoming more important for Software Engineers
- Exposure to real life problems - makes a refreshing change to artificial problems designed by Universities
- Potential for working relationships with customers after completion of MSc.
- Confidence boost from solving real-life problem by applying skills learnt at University
- Requires interpersonal skills development - this may be missed with individual projects
- Confirms importance of requirements capture - the customer actually receives the end product so it must meet their spec.

Our views have not changed a great deal in hindsight, and if anything we are now more convinced of the usefulness of such courses."
Tariq Hussain and Paul Lyon, Members of Genesys Solutions student company, 1998.

"While working as a member of the Genesys company, I have learnt a number of things which I believe will provide invaluable experience in my future. The administration of a small company with weekly board meetings and monthly reports will be useful as I now have an idea of the infrastructure of a small company.

"I have learnt a lot about team work, how to work together in a team to achieve a goal and how to deal with problems, such as disagreements, that occur when working in teams. As team leader, I had to delegate work to other team members, which was sometimes difficult to do.

“One of the most useful experiences was that of working with clients to build bespoke software. This highlighted many problems that I had not been previously aware of. Problems arise, for example, because no one member of the client’s team is entirely aware of the full scope of their organisation’s business.

“My experiences at Genesys lead me to believe that the requirements capture and specification of the system are much more difficult than actual design and implementation.”

Trevor Harris, Member of Genesys Solutions student company, 2000.

8.2 The industrial clients’ views

Software Hut - Year 2.

Uniplex, a company marketing medical equipment:
Client: Giles, Chief Executive.

Financial costs were negligible - "it was cheap"

The company obtained three good quality prototypes, each of which had bits that could be used in a final version of the system, "which was brilliant for us". "The company can get so much out of a project if they approach it in the right way." The project came "at the right time" for Uniplex, as they were already attempting to build software to handle sales data.

“The overall ability of the students seemed very high - I was pleased with what I saw.”

The Leadmill, a leading local nightclub and concert venue.
Client: Julie, Chief Executive

“I understand my business (process) so much better talking to the students who asked deep questions made me think about things in a way I would not otherwise have done. This has improved our business process.”

A scientific instruments company.
Dr David Payne

The project was beneficial in that it opened up contact between the company and computing students, who are potential graduate recruits.

“As a general comment, the students’ work and ability to keep us informed of progress was very good. The majority of students seemed well motivated and enthusiastic.”

GENESYS, Student software company - Year 4.

A company specialising in designing hospital facilities.
Client: Ken, Research Director.

The students were asked to investigate constraints-based CAD design with VR simulation. This was a feasibility study involving the development of a prototype solution.

“Very impressed with the quality of work and the short time taken to achieve a great deal. Have got very close to what the company wanted. Could be very significant to the company’s future business. Some problems caused by lack of liaison between members of the company.

Very good technical knowledge, very keen.”

A company specializing in consultancy in the field of metalurgical processing.
Client: Barry, Managing Director.

The students produced a self-paced web-based learning package for the company who were in the process of introducing new technology.

“This was highly successful, popular with staff and is still used 12 months later for new staff.”

Barry was very impressed with the students’ professionalism, their attitude to dealing with changes and problems caused by delays at the company end in the installation of the systems and said that the quality of their product was excellent.

A major medical charity.
Client: Sharon, Chief Field Officer.

Project: To provide a database for field officers to record all contacts and donations details and to generate reports, receipts etc. - currently all paper based. Needs to be available on laptops and the central database has to be updated every day by each officer. Significant replication and integrity issues. CRC HQ (London) want to expand the system out into all regions and to interface it to their central records. Major issues relating to vast quantities of data and interfacing Access to professional databases as well as differences of opinion between regional officers and HQ IT directors.

Involved two Genesys teams, both very professional and enthusiastic.

“Very helpful and dealt with a lot of difficult and changing problems superbly. Very, very pleased.”

8.3 The project lecturers’ views

Having been involved in the running of second and fourth year industrial projects for a number of years, we have become increasingly committed to the value of project-based learning in

software engineering. Learning by application of engineering concepts and practices, that have been introduced in the lecture theatre, to real business problems exposes students to the sometimes messy and rarely straightforward human, business and social elements of software development.

We feel our projects are enormously valuable to our students, even if they only come to realize how valuable when they graduate, look for jobs and get started in their careers. In the majority of cases, our students respond exceptionally well to the real world challenges we organize for them.

For lecturers and tutors who like small group work, focused on practical and immediate problem solving, the experience of supervising industrial projects with second and fourth year undergraduate students can be quite exhilarating. Every student team and every client's brief pose different challenges and problems. You have to be both a tough manager and stickler for detail, as well as a sympathetic sounding board for the students' ideas. You are frequently asked for opinions and guidance in project situations where there is no obviously right answer, nor correct way to make progress. You have to live with this, sometimes having helpful insights into students' problems and dilemmas, sometimes making mistakes.

Our personal feelings about the Software Hut is that it usually runs very well. The proposed changes, to extend the course to 20 credits and to incorporate initial teaching material about human-computer interaction and software testing, could be very successful. Both the human-computer interaction and software testing syllabi can be made more lively with reference to problems in real industrial project practice. Providing students' motivation does not wane over the two semesters, this extension of the duration of the project, together with its corresponding increase in credit rating, should enable students to smooth some rough edges that are often still present in the software at the end of most students' one semester efforts.

APPENDICES. Software Hut minutes template.

COM2070 Software Hut. Minutes of group meetings.

Group no. …….

Date of meeting [dd/mm]…….Time of meeting [hh:mm]…….. Place of meeting…….

Present…….

Absent (reason)……...

Agenda item	**Details:**	**Action by:**	**Deadline :**

Minutes submitted by:…….. Date……

Software hut timesheets.

COM2070 Software Hut. WEEKLY TIMESHEET.

Group no……..

Week beginning [dd/mm]…….…….

Name	**Requirements**	**Specification/ design**	**Coding**	**Testing**	**Review**	**Other**

Timesheet submitted by:…….. Date……

COM2070 Software Hut Assignment

Deliverables (XP groups)

Submission of assignment work will be in two stages:

1) Requirements Documentation (end of week 5)

1.1 A statement of requirements, signed off by the client *5 marks*

1.2 A detailed specification of test cases for the proposed system, using an appropriate language, signed off by the client *10 marks*

2) Final deliverable (end of week 12)

2.1 A working system together with appropriate system documentation for the client.

The software should be capable of running on a PC or network of PCs (with an appropriate technical specification - as defined by the client).

2.2 The total documentation should contain:

2.2.1 A copy of the Requirements documentation in 1) above

2.2.2 Test management process, how you develop and apply the tests and any scripts etc. that you need to develop to automate the process. *10 marks*

2.2.3. Completed test results and acceptance test report. *10 marks*

2.2.4 A sample of appropriate (commented) source code for the delivered system, screen layouts, etc. (maximum 20 sides).

2.2.5 User documentation containing:

installation instructions

user guide/ manual

maintenance guide/ manual

5 marks

2.2.6 A commentary on the project, (maximum 10 sides) containing: *10 marks*

A log of the project describing important milestones

A description of the group structure, the roles of individuals and mechanisms for communication used between group members.

A description of the quality control strategy, who did what, when and how the acceptance criteria were defined.

A list of any references to the literature used during the project.

An evaluation of the group performance including an allocation of the proportional effort contributed to the project by individuals in the group. This statement must be signed by all members of the group.

In addition to the commentary, please supply copies of all team meeting minutes, showing responsibilities for actions, progress made and revisions to the project plan.

Assessment and Marking Scheme

The breakdown of marks for this particular assignment will be:

Evaluation of the processes used and the engineering quality of the product, based on the system documentation and the commentary = 50%

See above for detailed allocation of these 50 marks

Suitability of the delivered system, based on client satisfaction = 50%
(Marking scheme subject to negotiation and agreement with clients.)

Documentation -

Presentation	*5 marks*
User Manual	*5 marks*
Installation guide	*5 marks*

Software system -

Ease of use	*5 marks*
Error handling	*5 marks*
Understandability (use of appropriate language, etc.)	*5 marks*
Base Functionality (completeness)	*5 marks*
Innovation (extra features)	*5 marks*
Robustness (correctness - doesn't crash)	*5 marks*
Happiness with product	*5 marks*

Software Hut Clients' sheet.

COM2070 Software Hut.

Information for clients.

Background.

The course is organised around teams of students, usually 5 in a team, competing to produce software solutions for their clients. Typically a client will work with 5 or 6 teams. The course takes place in Semester 2.

The students need to meet with the client on a regular basis for about 5 weeks until they have established the *Requirements* for the project.

It is very helpful if the client brings, to the initial meeting, as much information as they can about the system that they would like. This should include, information about:

the intended operating environment - type of computers,
their configuration (where known),
the software available (this might be Microsoft Office - including Access, Excel, Word etc.)
any paper copies of data and information relating to the proposed system (for example, if it is to replace a manual system)
any special circumstances that will affect the project.

Then each team will produce a draft *Requirements Document* for the client to amend and eventually agree and sign off over the next 5 weeks. This document is a description of the software that the teams propose to build for their client. It is important that this is correct - as far as possible. Although the students have had some teaching related to communication with their clients, the construction of a requirements document and working in teams this is possibly the first time that they have done it *for real*. It is an excellent experience for them and highly valued both by the students and their ultimate employers. It may also be the first time that the client has ex-

perienced being a client for a software project! The students may ask you questions about what you do which may enable you to look at your operations in a new light. It might mean that your initial thoughts about what you actually want will change during the discussions you have with the students. This can be very valuable but it can also result in you wanting to change your mind at a late stage in the process. Please exercise care in this regard as late changes invariably threaten projects. If you feel that there is still a lot of uncertainty or confusion 3 weeks or so into the project then please speak to the lecturers. We will endeavour to discuss with you progress each week in the initial stages of the project. We also monitor the students carefully, throughout. Each team has to produce minutes of team meetings, project plans as well as a considerable amount of technical design documentation which you are welcome to see but may not be easily understood.

It is strongly urged that changes to the requirements document should not be made after it is finalised, unless absolutely necessary, since this could seriously affect the likelihood of a team being able to deliver software of high quality. Naturally it is possible to make "cosmetic" changes at a later stage but whatever is decided should be considered very carefully.

The students will then spend the next few weeks building and testing the software. During this period it may be necessary for the students to contact the client to clarify some aspect of the project, an e-mail address or fax number is useful in this regard.

Around weeks 9 and 10 the students may invite the client to view a prototype in the Department's teaching labs (Lewin Laboratory, Regent Court). This is an opportunity to make minor adjustments to the system, perhaps improvements to the interface or some prioritization of the functionality of the system. It is often the case that original plans turn out to be rather optimistic and that the difficulties of programming can result in it all taking longer than was expected. At this point it is better to agree on the delivery of a high quality basic solution rather than one with lots of features and lots of bugs!

By week 12 all teams have to deliver their solution to the client, usually this will be on a CD and include a simple to use installation program. There should be a User Manual and an Installation Guide (unless this is automatic). Some teams may offer to install the software on the cli-

ent's computer. If the target operating environment is a computer network then it is advisable to seek advice from the local system administrator concerning installation as the network configuration may pose particular problems that the students cannot deal with.

Once there are some solutions to evaluate (hopefully all the teams will produce something) then the client is asked to assess the different solutions with the aid of a simple mark sheet. This involves identifying a number of attributes of the software and awarding a mark from 1 (low) to 5 (high) for each one. The total mark awarded by the client counts for 50% of the assessment for the course. These marks are needed by the time the exam board meets, (it's helpful if they are available a few days beforehand).

Once the client's marks are available a presentation is held and the teams that receive the top marks from the client are awarded a prize (£50 per person in the winning teams). Thus, the cost of the software to the client is £250.

The client is then free to use the winning software in their business. However, clients are asked not to sell on the software in any form without the prior agreement of the Department of Computer Science and the University's Research Consultancy Office.

We hope that the software chosen will be trouble-free and productive, however, we realise that problems may occur after the students have finished the course. It should be possible to contract someone to carry out maintenance if that is necessary, we will help if we can and there are occasional mechanisms for our senior students to take on this work, at a nominal cost, usually. In the past, however, our clients have been very pleased with their software and problems have been few and far between. It is difficult to estimate the commercial value of the software which is developed, suffice it to say that a typical application of the type we have delivered to many of our clients would cost tens of thousands of pounds if produced by consultants!

We might like to follow up the client's experience at some time, to help us assess the professionalism of the students and to identify any improvements we can make to the project arrangements.

COM 4010/6900 Assessment procedures for Genesys.

The assessment of Genesys is entirely by coursework. Since the course is group based it is important that regular observations and collection of data and other information.

The observations are carried out at the weekly Board meeting when general company business and progress is discussed (always chaired by a different student) and then followed by individual project meetings at which a tutor discusses the project with the project team.

The data is:

Minutes of meeting, these are collected weekly.
Time sheets, these are collected weekly.
Monthly reports dealing with the individual projects and a system administration report and occasional Research & Development reports.
Information and questionnaires from clients together with client interviews, where possible.
Individual self assessments carried out by students at the end of each Semester – these are moderated by the academic staff.

There are two stages to the assessment.

1. Formative assessment carried out in February, this does not count towards the final assessment.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

Mark sheets are produced, one is based on the group's performance and the other incorporates the group mark together with individual marks into an interim individual mark.

2. Summative assessement carried out in June. This provides the final mark for the module.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

The students also each write an individual critical assessment of their time in Genesys, this is also used to structure the discussion.

Two mark sheets are produced, one is based on the group's performance, (Appendix A), and the other incorporates the group mark together with individual marks into a final individual mark, Appendix B).

Appendix A

Genesys Solutions' Students 2003-4: Preliminary Assessment, February 2004

Team _____

Team Qualities and Achievements:

Team Grade:

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement (relative to abilities at start of course)	1	2	3	4	5
Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	1	2	3	4	5

Individual Grade:

Appendix B

Genesys Solutions' Students 2003-4: Final Assessment, June 2004

Team __

Team Qualities and Achievements:

Planning Effectiveness	1	2	3	4	5
Requirements Elicitation	1	2	3	4	5
Specification and Analysis	1	2	3	4	5
Design and Implementation	1	2	3	4	5
Testing and Quality Strategy	1	2	3	4	5

Comments:

Team Grade: __/25 (__%)

Team Member :

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement (relative to abilities at start of course)	1	2	3	4	5
Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	12	3	4	5	

Comments:

Individual Grade: __ / 35 + Team Grade: __ / 25 Total: __ /60 = __

Equivalent to __ %

COM 4010/6900 Assessment procedures for Genesys.

The assessment of Genesys is entirely by coursework. Since the course is group based it is important that regular observations and collection of data and other information.

The observations are carried out at the weekly Board meeting when general company business and progress is discussed (always chaired by a different student) and then followed by individual project meetings at which a tutor discusses the project with the project team.

The data is:

Minutes of meeting, these are collected weekly.
Time sheets, these are collected weekly.
Monthly reports dealing with the individual projects and a system administration report and occasional Research & Development reports.
Information and questionnaires from clients together with client interviews, where possible.
Individual self assessments carried out by students at the end of each Semester – these are moderated by the academic staff.

There are two stages to the assessment.

1. Formative assessment carried out in February, this does not count towards the final assessment.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

Mark sheets are produced, one is based on the group's performance and the other incorporates the group mark together with individual marks into an interim individual mark.

2. Summative assessement carried out in June. This provides the final mark for the module.

Each student is interviewed for 30 minutes and the discussion orients around what the student has done, their contribution, problems they have faced, what they have learnt, etc.

The students also each write an individual critical assessment of their time in Genesys, this is also used to structure the discussion.

Two mark sheets are produced, one is based on the group's performance, (Appendix A), and the other incorporates the group mark together with individual marks into a final individual mark, Appendix B).

Appendix A

Genesys Solutions' Students 2003-4: Preliminary Assessment, February 2004

Team _____

Team Qualities and Achievements:

Team Grade:

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement (relative to abilities at start of course)	1	2	3	4	5
Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	1	2	3	4	5

Individual Grade:

Appendix B

Genesys Solutions' Students 2003-4: Final Assessment, June 2004

Team __

Team Qualities and Achievements:

Planning Effectiveness	1	2	3	4	5
Requirements Elicitation	1	2	3	4	5
Specification and Analysis	1	2	3	4	5
Design and Implementation	1	2	3	4	5
Testing and Quality Strategy	1	2	3	4	5

Comments:

Team Grade: __/25 (__%)

Team Member :

Individual Qualities and Achievements:

Evidence of leadership skills	1	2	3	4	5
Evidence of working towards team and company goals	1	2	3	4	5
Evidence of planning and organisational ability	1	2	3	4	5
Evidence of involvement in quality control activities	1	2	3	4	5
Evidence of technical achievement (relative to abilities at start of course)	1	2	3	4	5
Evidence of skills in customer liaison	1	2	3	4	5
Evidence of skills in document production and management	12	3	4	5	

Comments:

Individual Grade: __ / 35 + Team Grade: __ / 25 Total: __ /60 = __

Equivalent to __ %

Some important dates.

Thursday 8th February, 11.00-12.00 Hicks Building, clients meet with all their project teams together to explain the background to their business and their basic requirements.

Thursday 15th February, 11.00-12.00 Hicks Building, clients meet each team separately for about 10 minutes.

Thursday 22nd February, 11.00-12.00 Hicks Building, clients meet each team separately for about 10 minutes.

Thursday 15th February, 11.00-12.00 Hicks Building, clients meet each team separately for about 10 minutes.

Thursday 1st March, 11.00-12.00 Hicks Building, clients meet each team separately for about 10 minutes.

Thursday 8th March, 11.00-12.00 Hicks Building, deadline for agreement on the Requirements Document.

Weeks 9-11 prototype demonstrations, Lewin lab, Regent Court. These may take place on Wednesdays from 12.00-1.00 or on Thursdays from 11.00-12.00 or at other times to be arranged in consultation with clients and teams.

Tuesday 19th June, undergraduate examination board.

Mike Holcombe.

References.

[1]. I. Sommerville, *"Software Engineering*", Addison-Wesley, 2000.

[2]. J. M. Spivey, "*The Z notation: A reference manual*", (2nd. Edition). Prentice Hall, 1992

[3]. M. Fowler, *UML distilled*, Addison Wesley, 1999.

[4]. K. Beck, *"Extreme Programming Explained"*, Addison-Wesley, 1999.

[5]. P. Coad, J. de Luca & E. Lefebre, "*Java modelling in color*", Prentice Hall, 1999.

[6]. M. Holcombe, *"Extreme Programming for Real",* to appear.